Code Listings

Hour 2

Listing 2.1 HTML to Display a Single Image

```
1:  <!DOCTYPE html>
2:  <html>
3:  <body>
4:  <img src="https://s3.amazonaws.com/bffmedia.examples/delessio_family_pool.jpg"
5:            alt="family photo"  />
6:  <body>
7:  </html>
```

Listing 2.2 HTML to Display a Single Image with Autozoom Off

```
1:  <!DOCTYPE html>
2:  <html>
3:  <head>
4:  <meta name="gtv-autozoom" content="off" />
5:  </head>
6:  <body>
7:  <img src="https://s3.amazonaws.com/bffmedia.examples/delessio_family_pool.jpg"
8:            alt="family photo" height=”720”  />
8:  <body>
9:  </html>
```

Listing 2.3 Adjusting Image Height for 720p or 1080p

```
1:  <!DOCTYPE html>
2:  <html>
3:  <head>
4:  <meta name="gtv-autozoom" content="off" />
5:  </head>
6:  <body>
7:  <SCRIPT language="JavaScript">
8:      <!--
9:      alert(screen.height);
10:      if ( (screen.height>=720))
11:      {
12:      document.write(
13:  '<img src="https://s3.amazonaws.com/bffmedia.examples/delessio_family_pool.jpg"  14:
height="972" alt="screenshot"  />');
15:      }
16:      else
17:      {
18:      document.write(
19:  '<img src="https://s3.amazonaws.com/bffmedia.examples/delessio_family_pool.jpg"
20:  height="648" alt="screenshot"  />');
21:      }
22:  //-->
23:  </SCRIPT>
24:  </body>
25:  </html>
```

Listing 2.4 Adjusting the Autozoom Ratio

```
1:  <!DOCTYPE html>
2:  <html>
3:  <head>
4:  <script language="JavaScript">
5:  function init() {
6:    var w = screen.width;
7:    var h = screen.height;
8:
8:    var bw = window.innerWidth;
10:    var bh = window.innerHeight;
```

```
11:
12:    var wRatio = bw/w;
13:    var hRatio = bh/h;
14:    var ratio = (wRatio + hRatio) / 2;
15:
16:    document.getElementsByTagName('body')[0].style.zoom = ratio;
17:    alert(ratio);
18:  }
19:  window.onload=init;
20:  </script>
21:  </head>
22:  <body  >
23:      <img  src="https://s3.amazonaws.com/bffmedia.examples/delessio_family_pool.jpg" 24:
height="648"  alt="screenshot"  />
25:  </body>
26: </html>
```

Listing 2.5 HTML Video Tag

```
1:  <video controls="controls" >
2:    <source src= "http://www.bffmedia.com/trailer_1080p.ogg" type="video/mp4" />
3:  </video>
```

Listing 2.6 Adjusting Video Height for 720p or 1080p

```
1:  <!DOCTYPE html>
2:  <html>
3:  <head>
4:  <meta name="gtv-autozoom" content="off" />
5:  </head>
6:  <body>
7:  <SCRIPT language="JavaScript">
8:      <!--
9:      alert(screen.height);
10:      if ( (screen.height>=720))
11:      {
12:      document.write('<video controls="controls"
13:              autoplay="autoplay" height=972 >
14:              <source src= "http:// http://www.bffmedia.com/trailer_1080p.ogg"
15:  type="video/mp4" /></video>');
16:      }
17:      else
18:      {
19:      document.write(
20:            '<video controls="controls" autoplay="autoplay" height=648 >
21:             <source src= "http://www.bffmedia.com/trailer_1080p.ogg"
22:             type="video/mp4" /></video>');16:     </video>');
23:      }
24:  //-->
25:  </SCRIPT>
26:  </body>
27:  </html>
```

Listing 2.7 Text Size and Color

```
1:  <p style=font-size:21;color:F1F1F1;background:black;">
2:     &nbsp21 Point text on black
3:  </p>
```

Listing 2.8 Defining a Font-Face

```
1:  @font-face
2:  {
3:      font-family: Moonstar;
4:      src: url("http://gtvdemos.appspot.com/static/fonts/bonzai.ttf");
5:  }
```

Listing 2.9 Adding Unique Fonts

```
<!DOCTYPE html>
<html>
<body >
<link rel="stylesheet" type="text/css"
   href="http://fonts.googleapis.com/css?family=Tangerine">
<style>
.font-demo-google {
  font-family: 'Tangerine', serif;
  font-size: 36px;
  text-shadow: 4px 4px 4px #aaa;
 }
 @font-face
 {
  font-family: Moonstar;
  src: url("http://gtvdemos.appspot.com/static/fonts/bonzai.ttf");
 }
 .font-demo-css {
  font-family: Moonstar, sans-serif;
  font-size: 36px;
 }
 </style>
 <p class="font-demo-css">Font test using CSS3's @font-face</p>
 <p class="font-demo-google">Font test using Google's Font API</p>
 </body>
 </html>
```

Listing 2.10 Hovering on Links and Buttons

```
<style type="text/css">
a:hover {font-size:24; font-weight:bold; color: red;}
button:hover {font-size:24; font-weight:bold; color: red;}
</style>
<a href="#">Hover Link</a>
<button>Simple</button>
```

Listing 2.11 Supporting Play and Pause on a Remote

```
<!DOCTYPE html>
<html>
<body>
<button onclick="playPause()">Play/Pause</button>
<br />
<video id="video1" controls="controls" autoplay="autoplay" height=648 >
  <source src= "http://www.w3schools.com/html5/movie.mp4" type="video/mp4" />
</video>
<script type="text/javascript">
 var myVideo=document.getElementById("video1");
 function playPause()
 {
   myVideo.play();
   myVideo.pause();
 }
 window.onload = function() {
       document.onkeydown=function(e){
         if (!e) e=window.event;
         switch(e.keyCode) {
           case 179:
             playPause();
             break;
         }
       }
     }
 </script>
 </body>
 </html>
```

Hour 3

Listing 3.1 Index.js Starting the Template Page

```
1:   **
2:   * Starts the template page.
3:   */
4:  gtv.jq.TemplatePage.prototype.start = function() {
5:    var templatePage = this;
6:    templatePage.keyController = new gtv.jq.KeyController();
7:    templatePage.preloadImages();
8:    templatePage.doPageZoom();1:
9:    templatePage.instanciateData();
10:   templatePage.makeTooltip();
11:   templatePage.makeProgressBar();
12:   templatePage.makeVideoControl();
13:   templatePage.makeTabs();
14:   templatePage.startFadeTimeput();
15    $(document.body).css('visibility', '');
16:   templatePage.keyController.start(null,
17:                          false,
18:                          gtv.jq.VideoControl.fullScreenLayer);
19:  };
20:  var templatePage = new gtv.jq.TemplatePage();
21:  templatePage.start();
```

Listing 3.2 Index.js Reading from Data Provider

```
1:   /**
2:    * Instanciates data from the data provider.
3:    */
4:     var templatePage = this;
5:     templatePage.dataProvider = new gtv.jq.DataProvider();
6:     templatePage.data = templatePage.dataProvider.getData();
7:   };
```

Listing 3.3 Data.html to Dump Data from dataprovider.js

```
1:   /**
2:    * Instantiates data from the data provider.
3:    */
4:     var templatePage = this;
5:     templatePage.dataProvider = new gtv.jq.DataProvider();
6:     templatePage.data = templatePage.dataProvider.getData();
7:   };
```

Listing 3.4 Creating Video Objects

```
1:   gtv.jq.DataProvider.prototype.getData = function() {
2:    var buck_videos = [
3:      {
4:         sources:
5:        ['http://bffmedia.com/trailer_400p.ogg'],
6:         title: 'Big Buck 400p Video Trailer',
7:         thumb: 'http://www.bffmedia.com/buck1.png;,
8:         description: ['Common Creative Project Movie'],
9:         subtitle: ' Smaller Version'
10:       },
11:       {
12:         sources:['http://bffmedia.com/trailer_1080p.ogg'],
13:         title: 'Big Buck 1080p Video Trailer',
14:         thumb: 'http://www.bffmedia.com/buck2.png',
15:         description: ['Common Creative Project Movie'],
16:         subtitle: 'Big Buck is a Rabbit'
17:       }
18:   ];
```

Listing 3.5 Creating a Data Object with Categories and Videos

```
1:   var data = {
2:       categories: [
3:       {
4:        name: 'Dev Events',
5:        videos: event_videos
6:      },
7:      {
8:        name: 'Big Buck',
9:         videos: buck_videos
10:      }
11:      ]
12:    };
13:     return data;
```

Hour 4

Listing 4.1 Interacting with Video

```
1:   <!DOCTYPE html>
2:   <html>
3:   <body>
4:   <button onclick="playPause()">Play/Pause</button>
5:   <button onclick="myVideo.currentTime+=10">Skip Ahead</button>
6:   <button onclick="myVideo.currentTime-=10">Skip Back</button>
8:   <button onclick="myVideo.currentTime=50">Jump to</button>
8:   <br>
9:   <button onclick="myVideo.playbackRate++">Speed up</button>
10:  <button onclick="myVideo.playbackRate--">Slow Down--</button>
11:  <button onclick="myVideo.volume+=1">Volume up</button>
12:  <button onclick="myVideo.volume-=1">Volume Down</button>
13:  <button onclick="mute()">Mute On/Off</button>
14:  <video id="video1" controls="controls" autoplay="autoplay"
15:      poster="http://www.bffmedia.com/poster.png" height="648"
16:      src="http://www.bffmedia.com/trailer_1080p.ogg"> />
17:  </video>
18:  <script type="text/javascript">
19:  var myVideo=document.getElementById("video1");
20:  function playPause()
21:  {
22:  if (myVideo.paused)
23:    myVideo.play();
24:  else
25:    myVideo.pause();
26:  }
27:
28:  function mute()
29:  {
30:   if (myVideo.muted)
31:     myVideo.muted = false;
32:   else
33:     myVideo.muted=true;
34:  }
35:  window.onload = function() {
36:      document.onkeydown=function(e){
37:      if (!e) e=window.event;
38:      switch(e.keyCode) {
39:         case 179:
40:           playPause();
41:           break;
42:       }
43:    }
44:  }
45: </script>
46: </body>
47: </html>
```

Listing 4.2 Implementing Rewind

```
1:   /**
2:   var timer;
3:
4:   function playPause()
5:   {
6:      clearTimeout(timer);
7:      myVideo.playbackRate=1.0;
8:      if (myVideo.paused)
9:        myVideo.play();
10:      else
11:       myVideo.pause();
12:   }
13:
14:   function rewind(){
15:      myVideo.currentTime-=1
16:      if (myVideo.currentTime>0){
17:              clearTimeout(timer);
18:              timer = setTimeout("rewind()", 100);
19:      }else{
20:              clearTimeout(timer);
21:              myVideo.pause();
22:      }
23:    }
```

Listing 4.3 Progress Bar

```
1:   var setup=false;
2:
3:   var progressBar = document.getElementById('progressbar');
4:   function initProgress() {
5:     if (!setup){
6:       progressBar.min = 0;
7:       progressBar.max = myVideo.duration;
8:       setup=true;
9:     }
10:
11:   progressBar.onchange = resetProgress;
12:   function resetProgress() {
13:     myVideo.currentTime = progressBar.value;
14:   }
15:
16:   function updateProgress() {
17:      progressBar.value = myVideo.currentTime;
18:   }
19:
20:   myVideo.addEventListener('durationchange', initProgress);
21:   myVideo.addEventListener('timeupdate', updateProgress);
```

Listing 4.4 Stylesheet for Buttons: video-styles.css

```
1:   body {
2:       background: #f1f1f1;
3:       overflow: hidden;
4:   }
5:
6:   .video-button {
7:      width: 92px;
8:      height: 37px;
9:      margin: 0 6px;
10:     border: 2px solid #454545;
11:     border-radius: 5px 5px / 5px 5px;
12:   }
13:
14:   .video-button:hover {
15:     color: #FFF;
16:     background: #900;
17:   }
18:
```

```
19:   .video-seek {
20:      width: 860px;
21:      height: 37px;
21:   }
```

Listing 4.5 Applying the Style

```
1:   <div>
2:     <button class="video-button" onclick="playPause()">Play Pause</button>
3:     <button class="video-button" onclick="myVideo.currentTime+=10">
4:   Skip Ahead</button>
5:     <button class="video-button" onclick="myVideo.currentTime-=10">Skip Back</button>
6:     <button class="video-button" onclick="myVideo.playbackRate++">
7:   Fast Forward</button>
8:     <button class="video-button" onclick="rewind()">   Rewind  <<   </button>
9:     <button class="video-button" onclick="myVideo.volume+=1">Volume +</button>
10:     <button class="video-button" onclick="myVideo.volume-=1">Volume -</button>
11:     <button class="video-button" onclick="mute()">Mute</button>
12:   </div>
13:   <div>
14:     <input class="video-seek" type="range" step="any" id="progressbar">
15:   </div>
```

Listing 4.6 Styles for Row Control

```
1:   <link rel="stylesheet" href="../source/css/controls.css" />
2:
3:   <style>
4:   .scroll-row-style {
5:     overflow: hidden;
6:     width: 100%;
7:     background-color: red;
8:      padding: 10px;
9:     position: relative;
10:  }
11:  .scroll-div-style {
12:    margin: 5px;
13:     padding: 10px;
14:    background-color: green;
15:  }
16:  .scroll-items-div-style {
17:  }
18:  .scroll-item-style {
19:    background-color: blue;
20:    border-radius: 10px;
21:    color: #ddd;
22:    padding: 10px;
23:    font-size: 20pt;
24:    border: #58b solid 6px;
25:  }
26:  .item-hover {
27:    border: yellow solid 6px !important;
28:  }
29:  .scrolling-page {
30:    position: absolute;
31:    width: 95%;
32:  }
33:  </style>
```

Listing 4.7 JavaScript for Row Control

```
1: <script type="text/javascript"
2:   src=
3:   "http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
4:   </script>
5:   <script type="text/javascript" src="../source/js/scrollrow.js"></script>
6:   <script type="text/javascript" src="../source/js/gtvcore.js"></script>
7:   <script type="text/javascript">
8:   function makePage(topParent) {
```

```
9:     var keyController = new gtv.jq.KeyController();
10:    keyController.start();
11:    var container = $('<div></div>').addClass('scrolling-page');
12:    topParent.append(container);
13:    var numItems = 10;
14:    var itemArray = new Array(numItems);
15:    for (var i = 0; i < numItems; i++) {
16:      itemArray[i] ='<button> button ' + (i+1) +'</button>';
17:
18:    }
19:    var styles = {
20:    row: 'scroll-row-style',
21:    itemsDiv: 'scroll-items-div-style',
22:    itemDiv: 'scroll-div-style',
23:    item: 'scroll-item-style',
24:    hover: 'item-hover'
25:  };
26:
27:      var createParams = {
28:        containerId: 'row-container',
29:        styles: styles,
30:        keyController: keyController
31:      };
32:      rowControl = new gtv.jq.RowControl(createParams);
33:      var controlParams = {
34:        topParent: container,
35:        contents: {
36:          items: itemArray
37:        }
38:     };
39:      rowControl.showControl(controlParams);
40:
41:      rowControl.enableNavigation();
42:    };
43:    $(document).ready(function() {
44:        makePage($("body"));
45:     });
46:    </script>
```

Hour 5

Listing 5.1 Hello-tv.html File Using Google TV Closure UI Library

```
1:  <!DOCTYPE html>
2:  <html>
3:    <head>
4:      <!-- Custom paths to Closure Library and GTV Closure UI Library files -->
5:      <script src='../closure-library/closure/goog/base.js'></script>
6:      <script src='../source/deps.js'></script>
7:      <!-- End custom script includes -->
8:      <script type='text/javascript'>
9:        goog.require('tv.ui');
10:       goog.require('tv.ui.Component');
11:       goog.require('tv.ui.Container');
12:       goog.require('tv.ui.Document');
13:     </script>
14:     <style type='text/css'>
15:       .tv-component.tv-component-focused {
16:         color: #fff;
17:         background-color: #000;
18:       }
19:     </style>
20:   </head>
21:   <body>
22:     <div class='tv-container-horizontal'>
23:       <span class='tv-component' id='hello'>Hello</span>
24:       <span class='tv-component' id='world'>World</span>
25:     </div>
26:     <script type='text/javascript'>
```

```
27:       tv.ui.decorate(document.body);
28:       var elementToFocus = goog.dom.getElement('hello');
29:       var componentToFocus = tv.ui.getComponentByElement(elementToFocus);
30:       tv.ui.Document.getInstance().setFocusedComponent(componentToFocus);
31:     </script>
32:   </body>
33: </html>
```

Listing 5.2 The onFocus Function

```
1:  /**
2:   * Handles focus event.
3:   * @param {goog.events.Event} event Focus event.
4:   * @protected
5:   */
6:  tv.ui.Component.prototype.onFocus = function(event) {
7:    // Add focused class to element.
8:    goog.dom.classes.add(this.element_, tv.ui.Component.Class.FOCUSED);
9:    event.stopPropagation();
10  };
```

Listing 5.3 Button Includes

```
1:  <!DOCTYPE html>
2:  <html>
3:    <head>
4:      <script src='../closure-library/closure/goog/base.js'></script>
5:      <script src='../source/deps.js'></script>
6:      <script type='text/javascript'>
7:        goog.require('tv.ui');
8:        goog.require('tv.ui.Component');
9:        goog.require('tv.ui.Container');
10:       goog.require('tv.ui.Document');
11:       goog.require('tv.ui.Button');
12:       goog.require('tv.ui.DecorateHandler');
13:     </script>
```

Listing 5.4 Button Styles

```
1:  <style>
2:    .button-demos {
3:     margin: 20px auto;
4:     width: 430px;
5:   }
6:   .button-demo {
7:     border: 1px solid white;
8:     display: inline-block;
9:     padding: 1px;
10:    margin: 3px;
11:  }
12:  .button-demo-inner {
13:    color: white;
14:    padding: 2px 6px;
15:    text-decoration: none;
16:  }
17:  .button-demo:hover,
18:  .button-demo.tv-component-focused {
19:    outline: none;
20:    -webkit-box-shadow: 0 0 30px #6391de, 0 0 15px #f2f2f2, 0 0 5px #FFF;
21:  }
22:  .button-demo, .button-demo-inner {
23:    border-radius: 5px;
24:    cursor: pointer;
25:    display: inline-block;
26:    -moz-border-radius: 5px;
27:    -webkit-border-radius: 5px;
28:  }
29:  #button-action-output {
30:   color: red;
```

```
31:    font-weight: bold;
32:    margin: 10px auto;
33:    padding: 20px;
34:    text-align: center;
35:    text-shadow: none;
36:    width: 400px;
37:    -webkit-border-radius: 5px;
38:  }
39:  body {
40:    background-color: #000;
41:    background-repeat: no-repeat;
42:    color: #FCFCFC;
43:    font-family: "Droid Sans TV", "Driod Sans", Verdana, sans-serif;
44:    font-size: 18px;
45:    margin: 0;
46:    overflow: hidden;
47:    padding: 0;
48:    text-shadow: rgba(0, 0, 0, 0.75) 0 -1px 1px;
49:  }
50:  </style>
```

Listing 5.5 Button Components

```
1:  </head>
2:    <body >
3:      <div class="tv-container-horizontal button-demos">
4:        <div class="tv-button button-demo">
5:          <div class="button-demo-inner">
6:          Button One
7:          </div>
8:         </div>
9:         <div class="tv-button button-demo">
10:         <div class="button-demo-inner">
11:           Button Two
12:         </div>
13:        </div>
14:      <div   id="button-action-output"></div>
```

Listing 5.6 Button Actions

```
1:  <script type="text/javascript">
2:          var decorateHandler = new tv.ui.DecorateHandler();
3:          function updateText(text) {
4:            var actionOutputEl = goog.dom.getElement('button-action-output');
5:            actionOutputEl.innerHTML = text;
6:          };
7:          decorateHandler.addClassHandler('button-demo', function(button) {
8:            goog.events.listen(button, tv.ui.Button.EventType.ACTION, function(e) {
9:              updateText('You clicked ' + e.target.getElement().innerText);
10:           });
11:         });
12:         tv.ui.postponeRender(function() {
13:           tv.ui.decorate(document.body, decorateHandler.getHandler());
14:         });
15:     </script>
16:   </body>
17: </html
```

Listing 5.7 Defining Component Styles

```
1:   .component-demo {
2:     border: 1px solid white;
3:     padding: 5px;
4:     margin: 20px;
5:     -webkit-border-radius: 5px;
6:   }
7:   .different-size {
8:     height: 50px;
9:     width: 200px;
```

```
10:    margin: 0 auto;
11:  }
12:  .component-demo.tv-component-focused {
13:    background-color: rgba(255, 255, 255, 0.3);
14:  }
15:  .different-highlight.tv-component-focused {
16:    background: none;
17:    -webkit-box-shadow: 0 0 20px #6391de, 0 0 7px #f2f2f2, 0 0 2px #FFF;
18:  }
19:  .different-zoom {
20:    margin: 0 auto;
21:    text-align: center;
22:    width: 200px;
23:  }
24:  .different-zoom.tv-component-focused {
25:    zoom: 2;
26:    -webkit-transition: all 1s ease;
27:  }
```

Listing 5.8 Using Component Styles

```
1:   <div class="tv-container-vertical">
2:   <div class="tv-component component-demo">
3:   <p>Use any HTML here, like text or an image:</p>
4:    <img src="http://www.google.com/tv/static/images/logo_146x38.png"/>
5:   </div>
6:   <div class="tv-component component-demo different-size">Different sizes.</div>
7:   <div class="tv-component component-demo different-highlight">And different
8:   highlight.
9:   </div>
10:  <div class="tv-component component-demo different-highlight different-zoom">
11:    Zoom it
12:  </div>
13:  </div>
```

Listing 5.9 Grid Styles

```
1:   tv-grid {
2:     height: 580px;
3:   }
4:   .photo-grid {
5:     overflow: hidden;
6:     margin: 10px;
7:   }
8:   .photo-grid .photo-grid-row {
9:     display: block;
10:    white-space: nowrap;
11:  }
12:  .photo-grid .photo {
13:    border: 1px solid #626262;
14:    cursor: pointer;
15:    display: inline-block;
16:    height: 133px;
17:    margin: 7px;
18:    overflow: hidden;
19:    width: 200px;
20:    box-shadow: 0 0 10px #000;
21:    vertical-align: top;
22:    -webkit-box-shadow: 0 0 10px #000;
23:  }
24:  .photo-grid .photo:hover,
25:  .photo-grid .photo.tv-component-focused {
26:    -webkit-box-shadow: 0 0 30px #6391de, 0 0 15px #f2f2f2, 0 0 5px #FFF;
27:  }
28:  .photo img {
29:    width: 200px;
30:  }
31:  .photo.portrait img {
32:    margin-top: -60px;
33:  }
```

Listing 5.10 Grid Components

```
1:  <div class="tv-grid photo-grid tv-container-vertical">
2:      <div class="tv-container-horizontal photo-grid-row">
3:        <div class="tv-button photo portrait">
4:  <img src="http://www.bffmedia.com/bffphotopic.png"></div>
5:        <div class="tv-button photo landscape">
6:  <img src="http://www.bffmedia.com/delessio_family_pool_1080.jpg"></div>
7:      </div>
8:      <div class="tv-container-horizontal photo-grid-row">
9:        <div class="tv-button photo landscape">
10: <img src="http://www.bffmedia.com/delessio_family_pool_1080.jpg"></div>
11:       <div class="tv-button photo portrait">
12: <img src="http://www.bffmedia.com/bffphotopic.png"></div>
13:     </div>
14: </div>
```

Listing 5.11 Lightbox Style Components

```
1:  .tv-lightbox-background {
2:    opacity: 0.9;
3:  }
4:  .tv-lightbox .tv-container-start-scroll,
5:  .tv-lightbox .tv-component img {
6:    -webkit-transition: all 1s ease;
7:  }
8:  .tv-lightbox .tv-component.tv-container-selected-child img {
9:    -webkit-transform: scale(1);
10: }
```

Listing 5.12 Showing the Lightbox Using JavaScript

```
1:  <script type="text/javascript">
2:          var decorateHandler = new tv.ui.DecorateHandler();
1:          var photos = [
3:            'http://www.bffmedia.com/bffphotoalbums.png',
4:            'http://www.bffmedia.com/bffphotopic.png',
5:            'http://www.bffmedia.com/bffphotoaviary1.png',
6:            'http://www.bffmedia.com/bffphotoaviary2.png'
7:          ];
8:          tv.ui.Lightbox.show(photos, 0 );
9:          tv.ui.postponeRender(function() {
10:             tv.ui.decorate(document.body, decorateHandler.getHandler());
11:           });
12:       </script>
```

Listing 5.13 TabContainer HTML for Tabs

```
1:  <div class="tv-tab-container tv-container-vertical">
2:    <div class="tv-tab-container-bar tv-container-horizontal demo-tab-bar">
3:      <div class="demo-tab tv-component">Info</div>
4:      <div class="demo-tab tv-component">Image</div>
5:      <div class="demo-tab tv-component">Video</div>
6:    </div>
```

Listing 5.14 TabContainer HTML for Tab Content

```
1:    <div class="tv-tab-container-content tv-tab-container-focus-attractor
2:  tv-container-horizontal demo-tab-content-container">
3:      <div class="tab-demo-content tv-component">Some info about Google TV</div>
4:      <div class="tab-demo-content tv-component">
5:  <img src = "https://developers.google.com/tv/images/remote-logo.png"></div>
6:      <div class="tab-demo-content tv-component">Some video</div>
7:    </div>
8:  </div>
```

Listing 5.15 TabContainer Styles

```
1:    .demo-tab-bar {
2:    height: 46px;
3:    text-align: center;
4:    white-space: nowrap;
5:    width: 400px;
6:   }
7:   .demo-tab {
8:     display: inline-block;
9:     line-height: 46px;
10:    text-align: center;
11:    width: 100px;
12:    vertical-align: bottom;
13:    -webkit-border-top-left-radius: 5px;
14:    -webkit-border-top-right-radius: 5px;
15:  }
16:  .demo-tab.tv-container-selected-child {
17:    background-color: #4c6ea7;
18:    font-weight: bold;
19:  }
20:  .demo-tab-content-container {
21:    background: #4c6ea7;
22:    height: 200px;
23:    position: relative;
24:    width: 400px;
25:    -webkit-border-radius: 5px;
26:  }
27:  .tab-demo-content {
28:    opacity: 0;
29:    padding: 20px;
30:    position: absolute;
31:  }
32:  .tab-demo-content.tv-container-selected-child {
33:    opacity: 1;
34:  }
```

Hour 6

Listing 6.1 Basic Web Sitemap for a Single URL

```
1:  <?xml version="1.0" encoding="UTF-8"?>
2:  <urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
3:     <url>
4:        <loc>http://www.bffmedia.com/</loc>
5:        <lastmod>2012-01-01</lastmod>
6:        <changefreq>monthly</changefreq>
7:        <priority>0.8</priority>
8:     </url>
9:  </urlset>
```

Listing 6.2 Video Sitemap with Required Elements

```
1:  <urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
2:          xmlns:video="http://www.google.com/schemas/sitemap-video/1.1">
3:     <url>
4:       <loc>http://www.bffmedia.com</loc>
5:       <video:video>
6:         <video:thumbnail_loc>
7:          http://i3.ytimg.com/vi/NGc7gW-jcVA/1.jpg
8:         </video:thumbnail_loc>
9:         <video:title>Kindle Fire Demo</video:title>
10:         <video:description>BFFPhoto App on Kindle Fire</video:description>
11:         <video:content_loc>
12:           http://www.youtube.com/embed/NGc7gW-jcVA
13:         </video:content_loc>
14:   </video:video>
15:     </url>
16:  </urlset>
```

Listing 6.3 Price and Subscription Options

```
1:  <video:price currency="USD" type="rent" resolution="hd">5.99</video:price>
2:  <video:price currency="USD" type="rent" resolution="sd">3.99</video:price>
3:  <video:price currency="USD" type="purchase" resolution="hd">11.99</video:price>
4:  <video:price currency="USD" type="purchase" resolution="sd">8.99</video:price>
5:  <video: requires_subscription>no</video:price>
```

Listing 6.4 Indicating Episodes in a Video Sitemap

```
1:  < <video:tvshow>
2:        <video:show_title>Star Trek</video:show_title>
3:        <video:episode_title>The Trouble with Tribbles</video:episode_title>
4:        <video:season_number>2</video:season_number>
5:        <video:episode_number>15</video:episode_number>
6:        <video:premier_date>: 1967-12-29 </premier_date>
7:  </video:tvshow>
```

Hour 7

Listing 7.1 Hour7Activity.java

```
1: package com.bffmedia.hour7;
2: import android.app.Activity;
3: import android.os.Bundle;
4: public class Hour7Activity extends Activity {
5:     /** Called when the activity is first created. */
6:     @Override
7:     public void onCreate(Bundle savedInstanceState) {
8:         super.onCreate(savedInstanceState);
9:         setContentView(R.layout.main);
10:    }
11:}
```

Listing 7.2 Main.xml

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="fill_parent"
4:     android:layout_height="fill_parent"
5:     android:orientation="vertical" >
6:     <TextView
7:         android:layout_width="fill_parent"
8:         android:layout_height="wrap_content"
9:         android:text="@string/hello" />
10:</LinearLayout>
```

Listing 7.3 Strings.xml

```
1: strings.xml
2: <?xml version="1.0" encoding="utf-8"?>
3: <resources>
4:     <string name="hello">Hello World, Hour7Activity!</string>
5:     <string name="app_name">Hour7</string>
6: </resources>
```

Hour 8

Listing 8.1 LinearLayout in XML

```
1:   <?xml version="1.0" encoding="utf-8"?>
2:   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:      android:orientation="vertical"
4:      android:layout_width="fill_parent"
5:      android:layout_height="fill_parent">
```

```
6:      <TextView android:layout_width="fill_parent"
7:         android:layout_height="wrap_content"
8:         android:text="Hello Google TV"
9:         android:id="@+id/greeting"/>
10:     <Button android:text="Button"
11:        android:id="@+id/button1"
12:        android:layout_width="wrap_content"
13:        android:layout_height="wrap_content" />
14:   </LinearLayout>
```

Listing 8.2 Coding an Activity Using the Layout

```
1:   public class Hour8Activity extends Activity {
2:       Button mButton;
3:       TextView mText;
4:       @Override
5:       public void onCreate(Bundle savedInstanceState) {
6:           super.onCreate(savedInstanceState);
7:           setContentView(R.layout.main);
8:           mText = (TextView)findViewById(R.id.greeting);
9:           mButton= (Button)findViewById(R.id.button1);
11:          mButton.setOnClickListener(new OnClickListener() {
12:            public void onClick(View v) {
13:               mText.setText("You clicked it!");
14:            }
15:         });
16:     }
```

Listing 8.3 Updating the LinearLayout XML file

```
1:   <?xml version="1.0" encoding="utf-8"?>
2:   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:      android:orientation="horizontal"
4:      android:background="#777777"
5:      android:layout_width="fill_parent"
6:      android:layout_height="fill_parent">
7:      <TextView android:layout_width="fill_parent"
8:         android:layout_height="wrap_content"
9:         android:text="Hello Google TV"
10:         android:id="@+id/greeting"/>
11:     <Button android:text="Button"
12:        android:id="@+id/button1"
13:        android:layout_width="wrap_content"
14:        android:layout_height="wrap_content" />
15:   </LinearLayout>
```

Listing 8.4 Setting Button Padding

```
1:   <?xml version="1.0" encoding="utf-8"?>
2:   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:      android:orientation="vertical"
4:      android:background="#777777"
5:      android:layout_width="fill_parent"
6:       <Button android:text="Button"
7:        android:id="@+id/button1"
8:        android:layout_width="wrap_content"
9:        android:layout_height="wrap_content" />
10:      <Button android:text="Button"
11:       android:layout_height="wrap_content"
12:       android:layout_width="wrap_content"
13:       android:id="@+id/ButtonPadding"
14:       android:padding="20dp">
15:     </Button>
16:     <Button android:text="Button"
17:       android:layout_height="wrap_content"
18:       android:layout_width="wrap_content"
19:       android:id="@+id/ButtonSidePadding"
20:       android:paddingRight="30dp"
21:       android:paddingLeft="30dp">
```

```
22:     </Button>
23:  </LinearLayout>
```

Listing 8.5 Setting Layout Margins

```
1:  <Button android:text="Button"
2:    android:id="@+id/button1"
3:    android:layout_width="wrap_content"
4:    android:layout_height="wrap_content"
5:    android:layout_marginTop="40dp"
6:    android:layout_marginLeft="120dp"/>
```

Listing 8.6 Fixing the FrameLayout

```
1:   <?xml version="1.0" encoding="utf-8"?>
2:   <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="fill_parent"
4:     android:layout_height="fill_parent"
5:     android:orientation="vertical">
6:   <ImageView
7:     android:id="@+id/imageView1"
8:     android:layout_width="wrap_content"
9:     android:layout_height="wrap_content"
10:    android:src="@drawable/delessio_family_pool_1080" />
11:  <TextView android:layout_width="wrap_content"
12:    android:layout_height="wrap_content"
13:    android:text="Hello Google TV"
14:    android:id="@+id/greeting"/>
15:  <Button android:text="Button"
16:    android:id="@+id/button1"
17:    android:layout_width="wrap_content"
18:    android:layout_height="wrap_content"
19:    android:layout_marginTop="40dp"
20:    android:layout_marginLeft="120dp"/>
21:  <Button android:text="Button"
22:    android:layout_height="wrap_content"
23:    android:layout_width="wrap_content"
24:    android:id="@+id/ButtonPadding"
25:    android:padding="20dp" android:layout_marginTop="20dp"/>
26:  <Button android:text="Button"
27:    android:layout_height="wrap_content"
28:    android:layout_width="wrap_content"
29:    android:id="@+id/ButtonSidePadding"
30:    android:paddingRight="30dp" android:layout_marginTop="80dp"/>
31:  </FrameLayout>
```

Listing 8.7 Placing Buttons with RelativeLayout

```
1:    <Button android:text="Button"
2:        android:id="@+id/button1"
3:        android:layout_width="wrap_content"
4:        android:layout_height="wrap_content"
5:        android:layout_marginTop="40dp"
6:        android:layout_marginLeft="120dp"/>
7:    <Button android:text="Button"
8:        android:layout_height="wrap_content"
9:        android:layout_width="wrap_content"
10:       android:id="@+id/ButtonPadding"
11:       android:padding="20dp"
12:       android:layout_toRightOf="@+id/button1">
13:   </Button>
14:   <Button android:text="Button"
15:       android:layout_height="wrap_content"
16:       android:layout_width="wrap_content"
17:       android:id="@+id/ButtonSidePadding"
18:       android:paddingRight="30dp"
19:       android:paddingLeft="30dp"
20:       android:layout_below="@+id/ButtonPadding"
21:       android:layout_toRightOf="@+id/ButtonPadding">
```

```
22:    </Button>
```

Hour 9

Listing 9.1 Adding Style to a TextView

```
1:  <TextView
2:    android:textAppearance="?android:attr/textAppearanceLarge"
3:    android:layout_height="wrap_content"
4:    android:layout_width="wrap_content"
5:    android:id="@+id/textView1"
6:    android:text="Large">
7:  </TextView>
8:  <TextView
9:    android:layout_height="wrap_content"
10:   android:layout_width="wrap_content"
11:   android:id="@+id/textView4"
12:   android:textSize="32sp"
13:   android:textStyle="bold"
14:   android:textColor="#ff0000"
15:   android:text="RED with 32dp Size Bold">
16:  </TextView>
```

Listing 9.2 Adding Drawables to a Button

```
1: <Button android:id="@+id/button1"
2:   android:layout_width="wrap_content"
3:   android:layout_height="wrap_content" android:text="Button 1"
4:   android:textColor="#ff0000"
5:   android:textSize="32sp"
6:   android:textStyle="bold">
7: </Button>
8: <Button android:text="Button"
9:   android:id="@+id/button2"
10:  android:layout_width="wrap_content"
11:  android:layout_height="wrap_content"
12:  android:drawableLeft="@drawable/icon">
13: </Button>
```

Listing 9.3 Updating Button Attributes Within Code

```
1: mButton2.setOnClickListener(new OnClickListener() {
2:    public void onClick(View v) {
3:    if (mButton1.getVisibility() == View.VISIBLE){
4:      mButton1.setVisibility(View.INVISIBLE);
5:      mButton2.setText("Invisible");
6:      mButton2.setCompoundDrawablesWithIntrinsicBounds(mX, null, null, null);
7:    }else{
8:     mButton1.setVisibility(View.VISIBLE);
9:     mButton2.setText("Visible");
10:    mButton2.setCompoundDrawablesWithIntrinsicBounds(mCheck, null, null, null);
11:   }
12:  }
13: });
```

Listing 9.4 EditView Layout

```
1: <EditText
2:   android:hint="Enter some text"
3:   android:id="@+id/editText1"
4:   android:inputType="text"
5:   android:layout_marginTop="10dp"
6:   android:layout_height="wrap_content"
7:   android:layout_width="match_parent">
8: </EditText>
```

Listing 9.5 AutoCompleteView Layout

```
1: <AutoCompleteTextView
2:   android:hint="This is an auto complete view"
3:   android:layout_height="wrap_content"
4:   android:layout_width="match_parent"
5:   android:id="@+id/autoCompleteTextView1"
6:   android:layout_marginTop="10dp">
7: </AutoCompleteTextView>
```

Listing 9.6 AutoCompleteTextView App Code

```
1: public class Hour9EditViewActivity extends Activity {
2: AutoCompleteTextView mAuto;
3: EditText mEdit;
4: String[] values = {"one", "two", "three", "one hundred", "one thousand" };
5: @Override
6:   public void onCreate(Bundle savedInstanceState) {
7:       super.onCreate(savedInstanceState)
8:       setContentView(R.layout.edit_layout);
9:       mEdit= (EditText)findViewById(R.id.editText1);
10:      mAuto= (AutoCompleteTextView)findViewById(R.id.autoCompleteTextView1);
11:      ArrayAdapter autoAdapter = new ArrayAdapter<String>
12:          (this,android.R.layout.simple_dropdown_item_1line,values);
13:      mAuto.setAdapter(autoAdapter);
14:  }
15:}
```

Listing 9.7 Defining a ProgressBar

```
1: <ProgressBar android:id="@+id/progressBar1"
2:    android:layout_height="wrap_content"
3:    android:layout_width="wrap_content"
4: </ProgressBar>
```

Listing 9.8 Setting ScaleType to fitXY for an ImageView

```
1: <ImageView
2:  android:layout_width="200dp"
3:  android:scaleType="fitXY"
4:  android:layout_height="200dp"
5:  android:src="@drawable/bbbsplash"
6:  android:id="@+id/bbb">
7: </ImageView>android:layout_alignParentLeft="true"
```

Listing 9.9 Setting ScaleType to centerInside for an ImageView

```
1: <ImageView
2:  android:layout_width="wrap_content"
3:  android:scaleType="centerInside"
4:  android:layout_height="wrap_content"
5:  android:src="@drawable/bbbsplash"
6:  android:id="@+id/bbb">
7: </ImageView>android:layout_alignParentLeft="true"
```

Listing 9.10 Setting Alpha Value Through Java Code

```
1: mAlpha= (Button)findViewById(R.id.changeAlpha);
2: mAlpha.setOnClickListener(new OnClickListener() {
3: public void onClick(View v) {
4:    mCheck1.setAlpha(60);
5: }
6:);
```

Listing 9.11 Creating a Bitmap Through Java Code

```
1: mAlpha= (Button)findViewById(R.id.changeAlpha);
```

```
2: mAlpha.setOnClickListener(new OnClickListener() {
3: public void onClick(View v) {
4:    mCheck1.setAlpha(60);
5:    Bitmap imageBitmap = Bitmap.createBitmap(120, 120, Bitmap.Config.ARGB_8888);
6:    Canvas canvas = new Canvas(imageBitmap);
7:    Paint p = new Paint();
8:    p.setTextSize(24);
9:    canvas.drawText("hello", 10, 20, p);
10:   mCheck2.setImageBitmap(imageBitmap);
11: }
12:);
```

Listing 9.12 VideoView Layout

```
1: <VideoView android:id="@+id/VideoView01"
2:   android:layout_height="fill_parent"
3:   android:layout_width="fill_parent">
4: </VideoView>
```

Listing 9.13 Assigning a Video to a VideoView

```
1: String videoToPlay = "http://bffmedia.com/bigbunny.mp4";
2: Uri videoUri = Uri.parse(videoToPlay);
3: mVideoView.setVideoURI(videoUri);
```

Listing 9.14 Incrementing the SeekBar with Video Progress

```
1: private Runnable UpdateProgress=new Runnable() {
2:   public void run() {
3:     if (mVideoView!=null) {
4:       seekTimeline.setProgress(mVideoView.getCurrentPosition());
5:       frame.postDelayed(UpdateProgress, 200);
6:     }
7:   }
8: };
9: @Override
10:protected void onResume() {
11:  super.onResume();
12:  frame.postDelayed(UpdateProgress, 200);
13:}
14:@Override
15:public void onPause() {
16:  frame.removeCallbacks(UpdateProgress);
17:  super.onPause();
18:}
```

Listing 9.15 Changing Video Position from SeekBar

```
1: seekTimeline.setOnSeekBarChangeListener(new   OnSeekBarChangeListener(){
2:  public void onProgressChanged(SeekBar seekBar, int progress,
3:    boolean fromUser) {
4:    if (!fromUser) return;
5:  }
6:  public void onStartTrackingTouch(SeekBar seekBar) {}
7:  public void onStopTrackingTouch(SeekBar seekBar) {
8:    mVideoView.seekTo(seekTimeline.getProgress());
9:  }
10:});
```

Listing 9.16 Rewinding a Video

```
1: private Runnable Rewind=new Runnable() {
2:   public void run() {
3:     if (mVideoView!=null && mRewind==true) {
4:       int current = mVideoView.getCurrentPosition();
5:       if (current <=0 ){
6:         mVideoView.seekTo(0);
7:         seekTimeline.setProgress(0);
```

```
8:       }else{
9:         mVideoView.seekTo(current-200);
10:        seekTimeline.setProgress(mVideoView.getCurrentPosition());
11:        frame.postDelayed(Rewind, 100 );
12:      }
13:    }
14:  }
15:};
```

Hour 10

Listing 10.1 Creating the ActionBar

```
1:   import android.app.ActionBar;
2:   import android.app.Activity;
3:   public class GTVActionBar extends Activity {
4:     @Override
5:       public void onCreate(Bundle savedInstanceState) {
6:         super.onCreate(savedInstanceState);
7:         setContentView(R.layout.gtvdemo);
8:         final ActionBar actionBar = getActionBar();
9:         actionBar.setDisplayShowTitleEnabled(false);
10:        actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
11:      }
12:  }
```

Listing 10.2 Adding Tabs to the ActionBar

```
1:    import android.app.ActionBar;
2:    import android.app.Activity;
3:    import android.app.ActionBar.Tab;
4:    public class GTVActionBar extends Activity {
5:       Tab mTab1;
6:       @Override
7:       public void onCreate(Bundle savedInstanceState) {
8:         super.onCreate(savedInstanceState);
9:         setContentView(R.layout.gtvdemo);
10:        final ActionBar actionBar = getActionBar();
11:        actionBar.setDisplayShowTitleEnabled(false);
12:        actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
13:        mTab1= actionBar.newTab().setText("Tab 1")
14:                    .setTabListener(new DemoTabListener());
15:                actionBar.addTab(mTab1);
16:      }
17:      private class DemoTabListener implements ActionBar.TabListener {
18:        public DemoTabListener() {
19:        }
20:        @Override
21:        public void onTabReselected(Tab tab, FragmentTransaction ft) {
22:        }
23:        @Override
24:        public void onTabSelected(Tab tab, FragmentTransaction ft) {
25:        }
26:        @Override
27:        public void onTabUnselected(Tab tab, FragmentTransaction ft) {
28:        }
29:      }
30:  }
```

Listing 10.3 Declaring the MenuItems

```
1:   import android.view.Menu;
2:   import android.view.MenuItem;
3:   import android.app.Activity;
4:   public class GTVActionBar extends Activity {
5:     MenuItem mItem1;
```

Listing 10.4 Defining MenuItems

```
1:   public boolean onCreateOptionsMenu(Menu menu) {
2:     super.onCreateOptionsMenu(menu);
3:     mItem1= menu.add("Menu Option 1");
4:     mItem1.setShowAsAction(MenuItem.SHOW_AS_ACTION_NEVER);
5:     return true;
6:   }
7:   public boolean onOptionsItemSelected(MenuItem item){
8:     if (item == mItem1){
9:     }
10:    return true;
11:  }
```

Listing 10.5 Defining an ActionItem

```
1:   public boolean onCreateOptionsMenu(Menu menu) {
2:     super.onCreateOptionsMenu(menu);
3:     mItem3 = menu.add("Menu Option 3");
4:     mItem3.setIcon(android.R.drawable.ic_menu_help);
5:     mItem3.setShowAsAction(MenuItem.SHOW_AS_ACTION_ALWAYS);
```

Listing 10.6 Creating an Activity with LeftNavBar, MenuItems, and an ActionItem

```
1:   package com.bffmedia.gtv.leftnav;
2:   import com.bffmedia.gtv.leftnav.R;
3:   import com.example.google.tv.leftnavbar.LeftNavBar;
4:   import com.example.google.tv.leftnavbar.LeftNavBarService;
5:   import android.app.ActionBar;
6:   import android.app.ActionBar.Tab;
7:   import android.app.Activity;
8:   import android.app.FragmentTransaction;
9:   import android.os.Bundle;
10:  import android.view.Menu;
11:  import android.view.MenuItem;
12:  import android.view.View;
13:  import android.widget.Toast;
14:
15:  public class GTVLeftActionBar extends Activity {
16:   Tab mTab1;
17:   Tab mTab2;
18:   Tab mTab3;
19:   MenuItem mItem1;
21    MenuItem mItem2;
22:   MenuItem mItem3;
23:   private LeftNavBar mLeftNavBar;
24:
25:   @Override
26    public void onCreate(Bundle savedInstanceState) {
27:     super.onCreate(savedInstanceState);
28:     setContentView(R.layout.gtvdemo);
29:     final LeftNavBar actionBar = (LeftNavBarService.instance()).
30:     getLeftNavBar((Activity) this);
31:     mTab1=actionBar.newTab().setText("Tab 1").setTabListener(new DemoTabListener());
32:     mTab2=actionBar.newTab().setText("Tab 2").setTabListener(new DemoTabListener());
33:     mTab3=actionBar.newTab().setText("Tab 3").setTabListener(new DemoTabListener());
34:     actionBar.setDisplayShowTitleEnabled(false);
35:     actionBar.setDisplayShowHomeEnabled(false);
36:     actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
37:     actionBar.addTab(mTab1);
38:     actionBar.addTab(mTab2);
39:     actionBar.addTab(mTab3);
40:  }
```

Listing 10.7 Adding the MenuItems

```
1:   public boolean onCreateOptionsMenu(Menu menu) {
2:     super.onCreateOptionsMenu(menu);
3:     mItem1= menu.add("Menu Option 1");
```

```
4:     mItem1.setShowAsAction(MenuItem.SHOW_AS_ACTION_NEVER);
5:     mItem2 = menu.add("Menu Option 2");
6:     mItem2.setShowAsAction(MenuItem.SHOW_AS_ACTION_NEVER);
7:     mItem3 = menu.add("Menu Option 3");
8:     mItem3.setIcon(android.R.drawable.ic_menu_help);
9:     mItem3.setShowAsAction(MenuItem.SHOW_AS_ACTION_ALWAYS);
10:    return true;
11:  }

11:  public boolean onOptionsItemSelected(MenuItem item){
12:    if (item == mItem1){
13:      Toast.makeText(getBaseContext(),  "Menu 1",
14:      Toast.LENGTH_LONG).show();
15:    }
16:    if (item == mItem2){
17:      Toast.makeText(getBaseContext(),  "Menu 2",
18:      Toast.LENGTH_LONG).show();
19:    }
20:    if (item == mItem3){
21:      Toast.makeText(getBaseContext(),  "Menu 3",
22:      Toast.LENGTH_LONG).show();
23:    }
24:    if (item.getItemId()== android.R.id.home) {
25:      Toast.makeText(getBaseContext(),  "Home Icon",
26:      Toast.LENGTH_LONG).show();
27:    }
28:    return true;
29:  }
```

Listing 10.8 Adding the TabListener

```
1:     private class DemoTabListener implements ActionBar.TabListener {
2:       public DemoTabListener() {
3:       }
4:       @Override
5:       public void onTabSelected(Tab tab, FragmentTransaction ft) {
6:         if (tab ==mTab1){
7:           Toast.makeText(getBaseContext(),  "Tab 1", Toast.LENGTH_LONG).show();
8:         }
9:         if (tab ==mTab2){
10:          Toast.makeText(getBaseContext(),  "Tab 2", Toast.LENGTH_LONG).show();
11:        }

12:       if (tab ==mTab3){
13:         Toast.makeText(getBaseContext(),  "Tab 3", Toast.LENGTH_LONG).show();
14:       }
15:     }
16:     @Override
17:     public void onTabUnselected(Tab tab, FragmentTransaction ft) {}
18:     @Override
19:     public void onTabReselected(Tab tab, FragmentTransaction ft) {}
21:   }
20: }
```

Hour 11

Listing 11.1 Showing Activity Life Cycle

```
1:  package com.bffmedia.hour11;
2:  import android.app.Activity;
3:  import android.os.Bundle;
4:  import android.util.Log;
5:  public class Hour11Activity extends Activity {
6:      private static final String TAG = "Hour11Activity";
7:      /** Called when the activity is first created. */
8:      @Override
9:      public void onCreate(Bundle savedInstanceState) {
10:     Log.v(TAG, "onCreate");
11:         super.onCreate(savedInstanceState);
```

```
12:         setContentView(R.layout.main);
13:     }
14:     @Override
15:     public void onStart() {
16:     Log.v(TAG, "onStart");
17:         super.onStart();
18:     }
19:     @Override
20:     public void onResume() {
21:     Log.v(TAG, "onResume");
22:         super.onResume();
23:     }
24:     @Override
25:     public void onPause() {
26:     Log.v(TAG, "onPause");
27:         super.onPause();
28:     }
29:     @Override
30:     public void onDestroy() {
31:     Log.v(TAG, "onDestroy");
32:         super.onDestroy();
33:     }
34: }
```

Listing 11.2 Layout for Fragment (one_fragment.xml)

```
1:  <?xml version="1.0" encoding="utf-8"?>
2:  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:      android:layout_width="fill_parent"
4:      android:layout_height="fill_parent"
5:      android:orientation="vertical">
6:  <TextView
7:      android:textAppearance="?android:attr/textAppearanceLarge"
8:      android:layout_height="wrap_content"
9:      android:layout_width="wrap_content"
10      android:id="@+id/textView1" android:text="Layout for fragment 1">
11: </TextView>
12: </LinearLayout>
```

Listing 11.3 Inflating a View (FragmentOne.java)

```
1: import android.app.Fragment;
2: import android.os.Bundle;
3: import android.view.LayoutInflater;
4: import android.view.View;
5: import android.view.ViewGroup;
6: public class TwoFragment extends Fragment   {
7:   @Override
8:   public View onCreateView(LayoutInflater
9:          inflater, ViewGroup container, Bundle savedInstanceState) {
10:    View v = inflater.inflate(R.layout.one_fragment, container, false);
11:    return v;
12:  }
13: }
```

Listing 11.4 Activity Code for Swapping Fragments

```
1: public class Hour11Activity extends Activity {
2:   Button mButton1;
3:   @Override
4:   public void onCreate(Bundle savedInstanceState) {
6:     super.onCreate(savedInstanceState);
7:     setContentView(R.layout.main);
8:     FragmentOne of = new FragmentOne();
9:     FragmentTransaction ft = getFragmentManager().beginTransaction();
10:    ft.add(R.id.linearLayout2, of, "one");
11:    ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
12:    ft.commit();
13:    mButton1= (Button)findViewById(R.id.button1);
```

```
14:    mButton1.setOnClickListener(new OnClickListener() {
15:    public void onClick(View v) {
16:      Fragment showing = getFragmentManager().findFragmentById(R.id.linearLayout2);
17:      if (showing.getTag().equalsIgnoreCase("one")){
18:        FragmentTwo tf = new FragmentTwo ();
19:        FragmentTransaction ft = getFragmentManager().beginTransaction();
20:        ft.replace(R.id.linearLayout2, tf, "two");
21:        ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
22:        ft.commit();
23:      }else{
24:        FragmentOne of = new FragmentOne ();
25         FragmentTransaction ft = getFragmentManager().beginTransaction();
26:        ft.replace(R.id.linearLayout2, of, "one");
27:        ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
28:        ft.commit();
29:      }
30:    }
31:  });
32: }
```

Listing 11.5 Adding Buttons with Actions to a Fragment View

```
1: @Override
2: public View onCreateView(LayoutInflater inflater,
3:             ViewGroup container, Bundle savedInstanceState) {
4:   View v = inflater.inflate(R.layout.one_fragment, container, false);
5:   mButton1= (Button)v.findViewById(R.id.button1);
6:   mButton1.setOnClickListener(new OnClickListener() {
7:   public void onClick(View v) {
8:     FragmentTwo tf = new FragmentTwo();
9:     FragmentTransaction ft = getFragmentManager().beginTransaction();
10:    ft.replace(R.id.linearLayout2, tf, "two");
11:    ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
12:    ft.addToBackStack("fragment two");
13:    ft.commit();
14:  }
15: });
16: return v;
17:}
```

Listing 11.6 New Methods for the Activity

```
1: public class Hour14ActivityConsolidated extends Activity {
2: Button mButton1;
3: @Override
4: public void onCreate(Bundle savedInstanceState) {
5:   super.onCreate(savedInstanceState);
6:   setContentView(R.layout.main);
7:   FragmentOne of = new FragmentOne();
8:   FragmentTransaction ft = getFragmentManager().beginTransaction();
9:   ft.add(R.id.linearLayout2, of, "one");
10:  ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
11:  ft.commit();
12:  mButton1= (Button)findViewById(R.id.button1);
13:  mButton1.setOnClickListener(new OnClickListener() {
14:  public void onClick(View v) {
15:    Fragment showing = getFragmentManager().findFragmentById(R.id.linearLayout2);
16:    if (showing.getTag().equalsIgnoreCase("one")){
17:      showFragmentTwo();
18:    }else{
19:      showFragmentOne();
20:    }
21:  }
22: });
23:}
```

Listing 11.7 Activity Method to Show a Fragment

```
1: public void showFragmentTwo(){
```

```
2:   FragmentTwo tf = new FragmentTwo();
3:   FragmentTransaction ft = getFragmentManager().beginTransaction();
4:   ft.replace(R.id.linearLayout2, tf, "two");
5:   ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
6:   ft.addToBackStack("fragment two");
7:   ft.commit();
8: }
```

Listing 11.8 Calling an Activity Method from a Fragment

```
1: @Override
2: public View onCreateView(LayoutInflater inflater,
3:           ViewGroup container, Bundle savedInstanceState) {
4:   View v = inflater.inflate(R.layout.one_fragment, container, false);
5:   mButton1= (Button)v.findViewById(R.id.button1);
6:   mButton1.setOnClickListener(new OnClickListener() {
7:   public void onClick(View v) {
8:     ((Hour11ActivityConsolidated) FragmentOne.this.getActivity()).showFragmentTwo();
9:   }
10: });
11: return v;
12:}
```

Listing 11.9 Creating an ActionBar with Two Tabs

```
1:  public void onCreate(Bundle savedInstanceState) {
2:  super.onCreate(savedInstanceState);
3:  setContentView(R.layout.main);
4:  final ActionBar actionBar = getActionBar();
5:  mTab1= actionBar.newTab().setText("Show Fragment 1")
6:                           .setTabListener(new DemoTabListener());
7:  mTab2= actionBar.newTab().setText("Show Fragment 2")
8:                           .setTabListener(new DemoTabListener());
9: };
```

Listing 11.10 Implementing ActionBar.TabListener

```
1:  @Override
2:  public void onTabSelected(Tab tab, FragmentTransaction ft) {
3:   if (tab ==mTab1){
4:     showFragmentOne();
5:   }
6:   if (tab ==mTab2){
7:     showFragmentTwo();
8:   }
9:  }
10: @Override
11: public void onTabUnselected(Tab tab, FragmentTransaction ft) {
12:  if (tab ==mTab1)
13:    if (mFragmentOne != null) {
14:       ft.hide(mFragmentOne);
15:    }
16:    if (tab ==mTab2)
17:     if (mFragmentTwo != null) {
18:      ft.hide(mFragmentTwo);
19:     }
20: }
21: public void onTabReselected(Tab tab, FragmentTransaction ft) {
22: }
```

Listing 11.11 TabListener onTabSelected Method

```
1: private class DemoTabListener implements ActionBar.TabListener {
2: FragmentOne mFragmentOne;
3: FragmentTwo mFragmentTwo;
4: @Override
5: public void onTabSelected(Tab tab, FragmentTransaction ft) {
6: if (tab ==mTab1){
```

```
7:   if (mFragmentOne == null) {
8:     mFragmentOne = new FragmentOne();
9:     ft.add(R.id.actionBarLayout, mFragmentOne, "one");
10:  } else {
11:    ft.show(mFragmentOne);
12:  }
13:}
14: if (tab ==mTab1){
15:    …
```

Listing 11.12 TabListener onTabUnselected Method

```
1: @Override
2: public void onTabUnselected(Tab tab, FragmentTransaction ft) {
3:  if (tab ==mTab1)
4:    if (mFragmentOne != null) {
5:      ft.hide(mFragmentOne);
6:    }
7:  if (tab ==mTab2)
8:    if (mFragmentTwo != null) {
9:      ft.hide(mFragmentTwo);
10:   }
11: }
12:}
```

Listing 11.13 Defining a Click Method

```
1: public void clickTabTwo(){
2:   mTab2.select();
3: }
```

Listing 11.14 Using the Click Method Within Activity

```
1: @Override
2: public View onCreateView(LayoutInflater inflater,
3:        ViewGroup container, Bundle savedInstanceState) {
4:   View v = inflater.inflate(R.layout.one_fragment, container, false);
5:   mButton1= (Button)v.findViewById(R.id.button1);
6:   mButton1.setOnClickListener(new OnClickListener() {
7:   public void onClick(View v) {
8:     ((Hour11ActivityActionBarFT) FragmentOne.this.getActivity()).clickTabTwo();
9:   }
10: });
11: return v;
12:}
```

Hour 12

Listing 12.1 Creating a DialogFragment Class

```
1: public class YesNoDialogFragment extends DialogFragment  {
2:   Button mYes;
3:   Button mNo;
4:   @Override
5:   public View onCreateView(LayoutInflater inflater,
6:             ViewGroup container, Bundle savedInstanceState) {
7:     View v = inflater.inflate(R.layout.yes_no_fragment, container, false);
8:     getDialog().setTitle("Quick Question");
9:     mYes = (Button)v.findViewById(R.id.yes);
10:    mYes.setOnClickListener(new OnClickListener() {
11:      public void onClick(View v) {
12:        YesNoDialogFragment.this.dismiss();
13:      }
14:    });
15:    mNo = (Button)v.findViewById(R.id.no);
16:    mNo.setOnClickListener(new OnClickListener() {
17:      public void onClick(View v) {
```

```
18:        YesNoDialogFragment.this.dismiss();
19:      }
20:    });
21:    return v;
22:  }
23:}
```

Listing 12.2 Opening a DialogFragment

```
1: mButton1= (Button)findViewById(R.id.button1);
2: mButton1.setOnClickListener(new OnClickListener() {
3:   public void onClick(View v) {
4:     YesNoDialogFragment yesNoDialog = new YesNoDialogFragment();
5:     yesNoDialog.show(getFragmentManager(), "yes_no_dialog");
6:   }
7: });
```

Listing 12.3 Adding a Listener Interface to a DialogFragment

```
1: public class YesNoDialogFragment extends DialogFragment  {
2:   Button mYes;
3:   Button mNo;
4:   YesNoListener mListener;
5:   public interface YesNoListener {
6:     public void ready(String answer);
7:   }
8:   public YesNoDialogFragment( YesNoListener yesNoListener) {
9:     super();
10:    this.mListener = yesNoListener;
11:  }
```

Listing 12.4 Calling the Listener Ready Method

```
1: mYes = (Button)v.findViewById(R.id.yes);
2: mYes.setOnClickListener(new OnClickListener() {
3:   public void onClick(View v) {
4:     mListener.ready("yes");
5:     YesNoDialogFragment.this.dismiss();
6:   }
7: });
```

Listing 12.5 Creating a Listener in the Activity

```
1: private final class PostDialogListener implements YesNoListener {
2:   @Override
3:   public void ready(String answer) {
4:      mTextView.setText(answer);
5:   }
6: }
```

Listing 12.6 Passing Data in the Activity

```
1:public void onClick(View v) {
2:  YesNoDialogFragment yesNoDialog = new YesNoDialogFragment(new PostDialogListener());
3:  Bundle args = new Bundle();
4:  args.putString("QUESTION", "Do you like Fragments?");
5:  yesNoDialog.setArguments(args);
6:  yesNoDialog.show(getFragmentManager(), "yes_no_dialog");
7:}
```

Listing 12.7 Read Data in the Fragment

```
1: public void onActivityCreated(Bundle savedInstanceState) {
2:   super.onActivityCreated(savedInstanceState);
3:   Bundle b = this.getArguments();;
4:   String question=b.getString("QUESTION");
5:   mQuestionText.setText(question);
```

```
6: }
```

Listing 12.8 Defining a ListFragment Class

```
1: import android.app.ListFragment;
2: import android.os.Bundle;
3: import android.view.View;
4: import android.widget.ListView;
5: import android.widget.Toast;
6: import android.widget.ArrayAdapter;
7: public class ArrayListFragment extends ListFragment  {
8:   private String[] mInfo = {"one", "two", "three", "four", "five"};
9:   @Override
10:   public void onActivityCreated(Bundle savedInstanceState) {
11:     super.onActivityCreated(savedInstanceState);
12:     setListAdapter(new ArrayAdapter<String>(getActivity(),
13:     android.R.layout.simple_list_item_1, mInfo));
14:  }
15:  @Override
16:  public void onListItemClick(ListView l, View v, int position, long id) {
17:   Toast.makeText(this.getActivity(), mInfo[position], Toast.LENGTH_SHORT).show();
18:  }
19:}
```

Listing 12.9 Displaying a ListFragment Class

```
1: public void onClick(View v) {
2:   ArrayListFragment aListFragment = new ArrayListFragment();
3:   FragmentTransaction ft = getFragmentManager().beginTransaction();
4:   ft.replace(R.id.linearLayout1, aListFragment, "list");
5:   ft.commit();
6: }
```

Listing 12.10 Layout for a GridView (grid_fragment.xml)

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <GridView
3:   xmlns:android="http://schemas.android.com/apk/res/android"
4:   android:layout_width="fill_parent"
5:   android:numColumns="4"
6:   android:layout_height="fill_parent" >
7: </GridView>
```

Listing 12.11 Creating a GridView Fragment

```
1: import android.app.Fragment;
2: import android.os.Bundle;
3: import android.view.LayoutInflater;
4: import android.view.View;
5: import android.view.ViewGroup;
6: import android.widget.ArrayAdapter;
7: import android.widget.GridView;
8:
9: public class GridFragment extends Fragment   {
10:  GridView mGrid;
11:  private String[] mInfo = {"one", "two", "three", "four", "five"};
12:  @Override
13:  public void onActivityCreated(Bundle savedInstanceState) {
14:    super.onActivityCreated(savedInstanceState);
15:    mGrid.setAdapter(new ArrayAdapter<String>(getActivity(),
16:    android.R.layout.simple_list_item_1, mInfo));
17:  }
18:  @Override
19:  public View onCreateView(LayoutInflater inflater,
20:    ViewGroup container, Bundle savedInstanceState) {
21:    mGrid  = (GridView) inflater.inflate(R.layout.grid_fragment, container, false);
22:    return mGrid;
23:  }
```

```
24:}
```

Listing 12.12 Layout for a Gallery (gallery_fragment.xml)

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <Gallery
3: xmlns:android="http://schemas.android.com/apk/res/android"
4: android:layout_width="fill_parent"
5: android:layout_height="fill_parent" >
6: </Gallery>
```

Listing 12.13 Creating a Gallery Fragment

```
1:  public class GalleryFragment extends Fragment   {
2:    Gallery mGallery;
3:    private String[] mInfo = {"one", "two", "three", "four", "five"};
4:    @Override
5:    public void onActivityCreated(Bundle savedInstanceState) {
6:      super.onActivityCreated(savedInstanceState);
7:      mGallery.setAdapter(new ArrayAdapter<String>(getActivity(),
8:      android.R.layout.simple_list_item_1, mInfo));
9:    }
10:   @Override
11:   public View onCreateView(LayoutInflater inflater,
12:     ViewGroup container, Bundle savedInstanceState) {
13:     mGallery  = (Gallery) inflater.inflate(R.layout.gallery_fragment,
14:        container, false);
15:     return mGallery;
16:   }
17: }
```

Hour 13

Listing 13.1 Setting Navigation Focus in an XML Layout

```
1: <Button
2:   android:id="@+id/button1"
3:   android:layout_width="wrap_content"
4:   android:layout_height="wrap_content"
5:   android:layout_alignParentLeft="true"
6:   android:layout_below="@+id/textView1"
7:   android:text="Button 1"
8:   android:nextFocusLeft="@+id/button6"
9:   android:nextFocusDown="@+id/button2"
10:  android:nextFocusRight="@+id/button2"
11:
12: />
13:
14: <Button
15:   android:id="@+id/button2"
16:   android:layout_width="wrap_content"
17:   android:layout_height="wrap_content"
18:   android:layout_below="@+id/textView1"
19:   android:layout_toRightOf="@+id/button1"
20:   android:text="Button 2"
21:   android:nextFocusDown="@+id/button3"
22:   android:nextFocusRight="@+id/button3"
23:   android:nextFocusUp="@+id/button1"
24:   android:nextFocusLeft="@+id/button1"
25: />
```

Listing 13.2 Switch Based on KeyCode

```
1: @Override
2: public boolean onKeyDown(int keyCode, KeyEvent event) {
3: switch (keyCode) {
4:         case KeyEvent.KEYCODE_MEDIA_REWIND:
```

```
5:          case KeyEvent.KEYCODE_DPAD_LEFT: {
6:              System.out.println ("LEFT CLICK");
7:              break;
8:          }
```

Listing 13.3 Extending ImageView

```
1:  public class DpadImageView extends ImageView {
2:    int x = 0;
3:    int y = 0;
4:    int newX = 0;
5:    int newY = 0;
6:    static int INTERVAL = 8;
7:    ArrayList<Coordinate> drawing = new ArrayList<Coordinate>();
8:    public DpadImageView(Context context) {
9:      super(context);
10:     drawing.add(new Coordinate(0,0));
11:   }
12:  public void right(){
13:    newX =x+INTERVAL;
14:    newY = y;
15:    drawing.add(new Coordinate(newX, newY));
16:  }
17:  public void left(){
18:    newX =x-INTERVAL;
19:    newY = y;
20:    drawing.add(new Coordinate(newX, newY));
21:  }
22:  public void up(){
23:    newX =x;
24:    newY = y-INTERVAL;
25:    drawing.add(new Coordinate(newX, newY));
26:  }
27:   public void down(){
28:     newX =x;
29:     newY = y+INTERVAL;
30:     drawing.add(new Coordinate(newX, newY));
31:   }
32:   @Override
33:   public void onDraw(Canvas canvas) {
34:     Paint paint = new Paint();
35:     paint.setStrokeWidth(INTERVAL);
36:     for (int i=0; i < drawing.size()-1; i++){
37:         canvas.drawLine(
38:                         drawing.get(i).x,
39:                         drawing.get(i).y,
40:                         drawing.get(i+1).x,
41:                           drawing.get(i+1).y,
42:                           paint);
43:     }
44:     x= newX;
45:     y= newY;
46:   }
47:   private class Coordinate {
48:     public int x;
49:     public int y;
50:     public Coordinate(int newX, int newY) {
51:     x = newX;
52:    y = newY;
53:   }
54:  }
55:}
```

Listing 13.4 Activity with OnKeyDown Implemented

```
1:   public class Hour13ActivityDpad extends Activity {
2:     DpadImageView mDpadView;
3:     @Override
4:     public void onCreate(Bundle savedInstanceState) {
5:       super.onCreate(savedInstanceState);
```

```
6:       setContentView(R.layout.dpad);
7:       mDpadView = (DpadImageView)findViewById(R.id.imageView1);
8:     }
9:     @Override
10:     public boolean onKeyDown(int keyCode, KeyEvent event) {
11:       switch (keyCode) {
12:         case KeyEvent.KEYCODE_DPAD_LEFT: {
13:           mDpadView.left();
14:           break;
15:         }
16:         case KeyEvent.KEYCODE_DPAD_RIGHT: {
17:           mDpadView.right();
18:           break;
19:         }
20:         case KeyEvent.KEYCODE_DPAD_DOWN: {
21:           mDpadView.down();
22:           break;
23:         }
24:         case KeyEvent.KEYCODE_DPAD_UP: {
25:           mDpadView.up();
26:           break;
27:         }
28:     }
29:     mDpadView.invalidate();
30:     return true;
31:    }
32:  }
```

Listing 13.5 Layout Including DpadImageView

```
1:   <com.bffmedia.hour13.DpadImageView
2:      android:id="@+id/imageView1"
3:      android:layout_width="match_parent"
4:      android:layout_height="match_parent"
5:      android:src="@drawable/icon" />
```

Hour 14

Listing 14.1 Creating a Bitmap from a URL

```
1: URL imageUrl = new URL(mImageString);
2: URLConnection connection = imageUrl.openConnection();
3: connection.connect();
4: InputStream is = connection.getInputStream();
5: mBitmap = BitmapFactory.decodeStream(is);
```

Listing 14.2 Downloading an Image Using an AsyncTask

```
1:  private class LoadImage extends AsyncTask<String , String , Long > {
2:  ImageView mImageView;
3:  String  mImageString;
4:  Bitmap mBitmap;
5:  public LoadImage (ImageView v, String imageString){
6:    mImageView =v;
7:    mImageString = imageString;
8:  }
9:  @Override
10: protected void onPreExecute() {
11:  }
12: @Override
13: protected void onPostExecute(Long result) {
14:   if (result==0){
15:     mImageView.setImageBitmap(mBitmap);
16:   }
17: }
18: @Override
19: protected Long doInBackground(String... params) {
20:   try {
```

```
21:     URL imageUrl = new URL(mImageString);
22:     URLConnection connection = imageUrl.openConnection();
23:     connection.connect();
24:     InputStream is = connection.getInputStream();
25:     mBitmap = BitmapFactory.decodeStream(is);
26:     return (0l);
27:   } catch (MalformedURLException e) {
28:     e.printStackTrace();
29:     return (1l);
30:   } catch (IOException e) {
31:     e.printStackTrace();
32:     return (1l);
33:   }
34:  }
35: }
```

Listing 14.3 Calling the AsyncTask for Background Image Loading

```
1:  public void onActivityCreated(Bundle savedInstanceState) {
2:    super.onActivityCreated(savedInstanceState);
3:    LoadImage li = new LoadImage(
4:         mImageView,"http://www.bffmedia.com/delessio_family_pool_1080.jpg");
5:    li.execute();
6:  }
```

Listing 14.4 Caching an Image for Performance

```
1:  protected Long doInBackground(String... params) {
2:    Date now = new Date();
3:    System.out.println("time start: " + now.getTime() );
4:    String imageFileName = mImageString.replace(":", "");
5:    imageFileName =imageFileName.replace("/", "");
6:    imageFileName =imageFileName.replace(".", "");
7:    File imageFile = new File(getActivity().getCacheDir(), imageFileName);
8:    OutputStream imageOS;
9:    if (imageFile.exists()){
10:      mBitmap = BitmapFactory.decodeFile(imageFile.getAbsolutePath());
11:      System.out.println("Getting file from Cache");
12:      return 0l;
13:   }
14:   try {
15:     imageFile.createNewFile();
16:     URL imageUrl = new URL(mImageString);
17:     URLConnection connection = imageUrl.openConnection();
18:     connection.connect();
19:     InputStream is = connection.getInputStream();
20:     mBitmap = BitmapFactory.decodeStream(is);
21:     imageOS = new BufferedOutputStream(new FileOutputStream(imageFile));
22:     mBitmap.compress(Bitmap.CompressFormat.JPEG, 100, imageOS);
23:     imageOS.flush();
24:     imageOS.close();
25:     return (0l);
26:   } catch (MalformedURLException e) { . . .
```

Listing 14.5 Make an HttpUrlConnection Request and Read the Response

```
1:  HttpURLConnection connection = null;
2:  try {
3:    URL dataUrl = new URL("https://graph.facebook.com/" + mPageId + "/photos");
4:    connection = (HttpURLConnection) dataUrl.openConnection();
5:    connection.connect();
6:    int status = connection.getResponseCode();
7:    System.out.println("Response: " + status);
8:    InputStream is = connection.getInputStream();
9:    BufferedReader reader = new BufferedReader(new InputStreamReader(is));
10:   String responseString;
11:   StringBuilder sb = new StringBuilder();
12:   while ((responseString = reader.readLine()) != null) {
13:     sb = sb.append(responseString);
```

```
14:   }
15:   System.out.println("Data: \n" + sb);
16:   String photoData = sb.toString();
17: } catch (MalformedURLException e) { . . .
```

Listing 14.6 JSON Example

```
1:  {
2:  id: "10150861198333306",
3:  from: {
4:    name: "Coca-Cola",
5:    category: "Food/beverages",
6:    id: "40796308305"
7:  },
8:  name: "April showers bringing you May flowers!",
9:  picture: "http://photos-c.ak.fbcdn.net/hphotos-ak-
10:  ash3/546114_10150861198333306_40796308305_9759003_909307049_s.jpg",
11:  }
```

Listing 14.7 JSONArray Structure

```
1:  {
2:  data: [
3:    {
4:      id: "1"
5:    }
6:    {
7:     id: "2"
8:    }
9:   ]
10: }
```

Listing 14.8 Loading and Reading a JSONArray

```
1:  JSONObject data = new JSONObject(photoData);
2:  JSONArray photoArray = data.optJSONArray("data");
3:  for(int i = 0; i < photoArray.length(); i++) {
4:    JSONObject photo= (JSONObject) photoArray.get(i);
5:    System.out.println (photo.optString("id);
6:  }
```

Listing 14.9 Making a Photo Object from a JSONObject

```
1:  public class Photo extends Object{
2:  String id;
3:  String name;
4:  String source;
5:  String picture;
6:  public Photo(JSONObject jsonPhoto) throws JSONException{
7:    this.id=(String) jsonPhoto.optString("id");
8:    this.source=(String) jsonPhoto.optString("source");
9:    this.name=(String) jsonPhoto.optString("name");
10:   this.picture=(String) jsonPhoto.optString("picture");
11  }
```

Listing 14.10 Making a Photo Array from a String

```
1:  public static ArrayList <Photo> makePhotoList (
2:           String photoData ) throws JSONException, NullPointerException {
1:    ArrayList <Photo> photos = new ArrayList<Photo>();
3:    JSONObject data = new JSONObject(photoData);
4:    JSONArray photoArray = data.optJSONArray("data");
5:    for(int i = 0; i < photoArray.length(); i++) {
6:      JSONObject photo=(JSONObject) photoArray.get(i);
7:      Photo currentPhoto = new Photo (photo);
8:      photos.add(currentPhoto);
9:    }
```

```
10:   return photos;
11: }
```

Listing 14.11 AsyncTask in Activity

```
1:   private class LoadPhotos extends AsyncTask<String , String , Long > {
2:   String  mPageId;
3:   Bitmap mBitmap;
4:   public LoadPhotos (String pageId){
5:     mPageId = pageId;
6:   }
7:   @Override
8:   protected void onPostExecute(Long result) {
9:     if (result==0){
10:      mButton2.setEnabled(true);
11:    }
12:  }
```

Listing 14.12 Retrieving Data in Background to Populate Photo Array

```
1:  @Override
2:  protected Long doInBackground(String... params) {
3:    HttpURLConnection connection = null;
4:    try {
5:      URL dataUrl = new URL("https://graph.facebook.com/" + mPageId + "/photos");
6:      connection = (HttpURLConnection) dataUrl.openConnection();
7:      connection.connect();
8:      int status = connection.getResponseCode();
9:      if (response!=200) return 1l;
10:     InputStream is = connection.getInputStream();
11:     BufferedReader reader = new BufferedReader(new InputStreamReader(is));
12:     String responseString;
13:     StringBuilder sb = new StringBuilder();
14:     while ((responseString = reader.readLine()) != null) {
15:       sb = sb.append(responseString);
16:     }
17:     String photoData = sb.toString();
18:     mPagePhotos = Photo.makePhotoList(photoData);
19:     return (0l);
20:   } catch (Exception e) {
21:      e.printStackTrace();
22:     return (1l);
23:   } finally {
24:    connection.disconnect();
25:   }
```

Listing 14.13 Button onClickListener for Random Photo

```
1:  mButton2.setOnClickListener(new OnClickListener() {
2:    public void onClick(View v) {
3:      ImageViewFragment fbImg = new ImageViewFragment();
4:      Bundle args = new Bundle();
5:      int random = (int)(Math.random() * mPagePhotos.size());
6:      args.putString("URL", mPagePhotos.get(random).source);
7:      fbImg.setArguments(args);
8:      FragmentTransaction ft = getFragmentManager().beginTransaction();
9:      ft.replace(R.id.linearLayout1, fbImg, "Image from Facebook");
10:     ft.commit();
11:   }
12: });
```

Listing 14.14 ImageViewFragment

```
1:  @Override
2:  public void onActivityCreated(Bundle savedInstanceState) {
3:    super.onActivityCreated(savedInstanceState);
4:    Bundle b = this.getArguments();
5:    String url = null;
```

```
6:    if (b!=null){
7:      url=b.getString("URL");
8:    }
9:    if (url==null){
10:     url = "http://www.bffmedia.com/delessio_family_pool_1080.jpg";
11:   }
12:   LoadImage li = new LoadImage(mImageView,url );
13:   li.execute();
14: }
```

Hour 15

Listing 15.1 Using SQLiteOpenHelper

```
1:  private static final String DATABASE_NAME = "FB_PHOTOS";
2:  private static final String DATABASE_TABLE = "photo";
3:  private static final int DATABASE_VERSION = 3;
4:
5:  private static class DatabaseHelper extends SQLiteOpenHelper {
6:    DatabaseHelper(Context context) {
7:      super(context, DATABASE_NAME, null, DATABASE_VERSION);
7:    }
9:    @Override
10:   public void onCreate(SQLiteDatabase db) {
11:     db.execSQL(DATABASE_CREATE);
12:   }
13:   @Override
14:   public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
15:     db.execSQL("DROP TABLE IF EXISTS " + DATABASE_TABLE );
16:     onCreate(db);
17:   }
18: }
```

Listing 15.2 Creating the Photo Database

```
1: private static final String DATABASE_CREATE =
2:  "create table photo (_id INTEGER PRIMARY KEY AUTOINCREMENT,"
3:  + "photo_id not null,"
4:  + "page_id not null,"
5:  + "name text,"
6:  + "source text,"
7:  + "picture text,"
8:  + "createdDate INTEGER,"
9:  + "modifiedDate INTEGER"
10: +");";
```

Listing 15.3 Open, Close, and Update

```
1:  public PhotoDBHelper open() throws SQLException {
2:    mDbHelper = new DatabaseHelper(mCtx);
3:    mDb = mDbHelper.getWritableDatabase();
4:    return this;
5:  }
6:  public void close() {
7:    if(mDbHelper!=null){
8:      mDbHelper.close();
9:    }
10: }
11: public void upgrade() throws SQLException {
12:   mDbHelper = new DatabaseHelper(mCtx); //open
13:   mDb = mDbHelper.getWritableDatabase();
14:   mDbHelper.onUpgrade(mDb, 1, 0);
15: }
```

Listing 15.4 Inserting a Photo in the Database

```
1: public long createPhoto(Photo photoToCreate) {
```

```
2:   ContentValues initialValues = new ContentValues();
3:   if (photoToCreate.id!=null)
4:      InitialValues.put("photo_id", photoToCreate.id);
5:   if (photoToCreate.pageId!=null)
6:      initialValues.put("page_id", photoToCreate.pageId);
7:   if (photoToCreate.name!=null)
8:      initialValues.put("name", photoToCreate.name);
9:   if (photoToCreate.source!=null)
10:     initialValues.put("source", photoToCreate.source);
11:  if (photoToCreate.picture!=null)
12:     initialValues.put("picture", photoToCreate.picture);
13:  if (photoToCreate.createdDate!=0)
14:     initialValues.put("createdDate", photoToCreate.createdDate);
15:  if (photoToCreate.modifiedDate!=0)
16:     initialValues.put("modifiedDate", photoToCreate.modifiedDate);
17:  return mDb.insert(DATABASE_TABLE, null, initialValues);
18:}
```

Listing 15.5 Updating a Photo in the Database

```
1: public boolean updatePhoto(String id, Photo photoToUpdate) {
2:  ContentValues updateValues = new ContentValues();
3:  if (photoToUpdate.id!=null)
4:    updateValues.put("photo_id", photoToUpdate.id);
5:  if (photoToUpdate.pageId!=null)
6:    updateValues.put("page_id", photoToUpdate.pageId);
7:  if (photoToUpdate.name!=null)
8:  ...
9:  if (photoToUpdate.modifiedDate!=0)
10:   updateValues.put("modifiedDate", photoToUpdate.modifiedDate);
11: return mDb.update(DATABASE_TABLE, updateValues, "photo_id" + "=" + id, null) > 0;
12: }
```

Listing 15.6 Deleting a Photo from the Database

```
1: public boolean deletePhoto(String photo_id) {
2:    return mDb.delete(DATABASE_TABLE, "photo_id" + "=" + photo_id, null) > 0;
3: }
```

Listing 15.7 Getting Photos from the Database

```
1: public Cursor fetchByPageId(String id) throws SQLException {
2:   Cursor mCursor =
3:     mDb.query(true, DATABASE_TABLE, PHOTO_FIELDS, "page_id" + "='" + id+"'", null,
4:     null, null, null, null);
5:   if (mCursor != null) {
6:      mCursor.moveToFirst();
7:   }
8:   return mCursor;
9: }
```

Listing 15.8 Defining the Photo Columns

```
1:  public static final String KEY_ROWID = "_id";
2:  public static final String[] PHOTO_FIELDS = new String[] {
3:    KEY_ROWID,
4:    "photo_id", // id of the photo
5:    "page_id",
6:    "name",
7:    "source",
8:    "picture",
9:    "createdDate",
10:   "modifiedDate"
11: };
```

Listing 15.9 Getting Photos from a Database in Specified Order

```
1: public Cursor fetchByPageIdDateOrder(String id) throws SQLException {
```

```
2:   Cursor mCursor =
3:     mDb.query(true, DATABASE_TABLE, PHOTO_FIELDS, "page_id" + "='" + id+"'", null,
4:     null, null, "photo_id DESC", null
```

Listing 15.10 Returning a Photo Object from a Cursor

```
1:  public static Photo getPhotoFromCursor(Cursor cursor){
2:    Photo photo = new Photo();
3:    photo.dbId = cursor.getLong(cursor.getColumnIndex("_id"));
4:    photo.id = cursor.getString(cursor.getColumnIndex("photo_id"));
5:    photo.name = cursor.getString(cursor.getColumnIndex("name"));
6:    photo.source = cursor.getString(cursor.getColumnIndex("source"));
7:    photo.picture = cursor.getString(cursor.getColumnIndex("picture"));
8:    photo.createdDate = cursor.getLong(cursor.getColumnIndex("createdDate"));
9:    photo.modifiedDate = cursor.getLong(cursor.getColumnIndex("modifiedDate"));
10:   return(photo);
11: }
```

Listing 15.11 Updating the Database from an ArrayList of Photo Objects

```
1:   String photoData = sb.toString();
2:   mPagePhotos = Photo.makePhotoList(photoData);
3:   long dbId;
4:   for (Photo currentPhoto : mPagePhotos) {
5:     currentPhoto.pageId = mPageId;
6:     Photo exists = photoDbHelper.fetchPhotoById(currentPhoto.id);
7:     if (exists==null){
8:       dbId = photoDbHelper.createPhoto(currentPhoto);
9:     }  else{
10:      photoDbHelper.updatePhoto(exists.id, currentPhoto);
11:    }
12:  }
```

Listing 15.12 Checking the Database for Photos

```
1: protected void onPreExecute() {
2:   PhotoDBHelper photoDbHelper = new PhotoDBHelper(Hour15Activity.this);
3:   photoDbHelper.open();
4:   Cursor photoCursor = photoDbHelper.fetchByPageId("99394368305");
5:   if (photoCursor.getCount() >0){
6:     mButton2.setEnabled(true);
7:   }
8:   photoDbHelper.close();
9: }
```

Listing 15.13 Using a Cursor to Select a Random Photo

```
1: mButton2.setOnClickListener(new OnClickListener() {
2:  public void onClick(View v) {
3:    ImageViewFragment fbImg = new ImageViewFragment();
4:    Bundle args = new Bundle();
5:    PhotoDBHelper photoDbHelper = new PhotoDBHelper(Hour15Activity.this);
6:    photoDbHelper.open();
7:    Cursor photoCursor = photoDbHelper.fetchByPageId("99394368305");
8:    int random = (int)(Math.random() * photoCursor.getCount()-1);
9:    photoCursor.moveToPosition(random);
10:   Photo randomPhoto =  PhotoDBHelper.getPhotoFromCursor(photoCursor);
11:   photoDbHelper.close();
12:   args.putString("URL", randomPhoto.source);
13:   fbImg.setArguments(args);
14:   FragmentTransaction ft = getFragmentManager().beginTransaction();
15:   ft.replace(R.id.linearLayout1, fbImg, "Image from Facebook");
16:   ft.commit();
17: }
18:});
```

Hour 16

Listing 16.1 The Shell of a ContentProvider

```
1: package com.bffmedia.content;
2: import android.content.ContentProvider;
3: import android.content.ContentValues;
4: import android.database.Cursor;
5: import android.net.Uri;
6: public class EmptyProvider extends ContentProvider {
7:   @Override
8:   public int delete(Uri uri, String selection, String[] selectionArgs) {
9:     return 0;
10:  }
11:  @Override
12:  public String getType(Uri uri) {
13:    return null;
14:  }
15:  @Override
16:  public Uri insert(Uri uri, ContentValues values) {
17:   return null;
18:  }
19:  @Override
20:    public boolean onCreate() {
21:   return false;
22:  }
23:  @Override
24:  public Cursor query(Uri uri, String[] projection, String selection,
25:                      String[] selectionArgs, String sortOrder) {
26:    return null;
27:  }
28:  @Override
29:  public int update(Uri uri, ContentValues values,
30:                    String selection,String[] selectionArgs) {
31:    return 0;
32:  }
33: }
```

Listing 16.2 PhotoProvider Declarations

```
1:  package com.bffmedia.content;
2:  import android.content.ContentProvider;
3:  import android.content.ContentResolver;
4:  import android.content.ContentUris;
5:  import android.content.ContentValues;
6:  import android.content.UriMatcher;
7:  import android.database.Cursor;
8:  import android.database.SQLException;
9:  import android.database.sqlite.SQLiteQueryBuilder;
10: import android.net.Uri;
11: public class PhotoProvider extends ContentProvider {
12: private PhotoDBHelper mPhotoDbHelper;
13: private static final String AUTHORITY = "com.bffmedia.content.PhotoProvider";
14: private static final String PHOTOS_BASE_PATH = "photos";
15: public static final Uri CONTENT_URI =
16:                  Uri.parse("content://" + AUTHORITY+ "/" + PHOTOS_BASE_PATH);
17: //content://com.bffmedia.content.PhotoProvider/photos
18: public static final Uri PHOTOS_URI =
19:                Uri.parse(CONTENT_URI +"/page");
20: //content://com.bffmedia.content.PhotoProvider/page
21: public static final String CONTENT_ITEM_TYPE =
22:                ContentResolver.CURSOR_ITEM_BASE_TYPE + "/com.bffmedia.photo";
23: public static final String CONTENT_TYPE =
24:                ContentResolver.CURSOR_DIR_BASE_TYPE + "/ com.bffmedia.photo ";
25: public static final int PHOTOS = 10;
26: public static final int PHOTO_ID = 20;
27: private static final UriMatcher sURIMatcher =
28:                new UriMatcher(UriMatcher.NO_MATCH);
29: static {
30:   sURIMatcher.addURI(AUTHORITY, PHOTOS_BASE_PATH + "/page/#", PHOTOS);
```

```
31:   sURIMatcher.addURI(AUTHORITY, PHOTOS_BASE_PATH + "/#", PHOTO_ID);
32: }
```

Listing 16.3 PhotoProvider Query Method

```
1:  @Override
2:  public Cursor query(Uri uri, String[] projection, String selection,
3:    String[] selectionArgs, String sortOrder) {
4:    Cursor cursor;
5:    int uriType = sURIMatcher.match(uri);
6:    switch (uriType) {
7:      case PHOTO_ID:
8:        cursor =  mPhotoDbHelper.fetchByPhotoId(uri.getLastPathSegment());
9:        cursor.setNotificationUri(getContext().getContentResolver(), uri);
10:     break;
11:     case PHOTOS:
12:       cursor =  mPhotoDbHelper.fetchByPageId(uri.getLastPathSegment());
13:       cursor.setNotificationUri(getContext().getContentResolver(), uri);
14:     break;
15:     default:
16:       throw new IllegalArgumentException("Unknown URI");
17:   }
18:   return cursor;
19: }
```

Listing 16.4 Supporting Another URI in Query

```
1: case PHOTO_QUERY:
2:   cursor =   mPhotoDbHelper.mDb.query(true, PhotoDBHelper.DATABASE_TABLE,
3:            projection, selection, selectionArgs, null,null,sortOrder, null);
4:   cursor.setNotificationUri(getContext().getContentResolver(), uri);
5:  break;
```

Listing 16.5 Return Type of Data in GetType

```
1:  public String getType(Uri uri) {
2:    int uriType = sURIMatcher.match(uri);
3:    switch (uriType) {
4:      case PHOTOS:
5:        return CONTENT_TYPE;
6:      case PHOTO_QUERY:
7:        return CONTENT_TYPE;
8:      case PHOTO_ID:
9:        return CONTENT_ITEM_TYPE;
10:     default:
11:      return null;
12:    }
13: }
```

Listing 16.6 Insert Method

```
1: @Override
2: public Uri insert(Uri uri, ContentValues values) {
3:   long newID = mPhotoDbHelper.mDb.insert(PhotoDBHelper.DATABASE_TABLE, null, values);
4:   if (newID > 0) {
5:     Uri newUri = ContentUris.withAppendedId(uri, newID);
6:     getContext().getContentResolver().notifyChange(uri, null);
7:     return newUri;
8:   } else {
9:     throw new SQLException("Failed to insert row into " + uri);
10:  }
11: }
```

Listing 16.7 Update Method

```
1: @Override
2: public int update(Uri uri, ContentValues values, String selection,
3:                   String[] selectionArgs) {
```

```
4: return mPhotoDbHelper.mDb.update(PhotoDBHelper.DATABASE_TABLE,
5:         values, selection, selectionArgs);
6: }
```

Listing 16.8 Delete Method

```
1: @Override
2: public int delete(Uri uri, String selection, String[] selectionArgs) {
3:    return mPhotoDbHelper.mDb.delete(PhotoDBHelper.DATABASE_TABLE,
4:           selection, selectionArgs);
5: }
```

Listing 16.9 OnPreExecute

```
1: @Override
2: protected void onPreExecute() {
3:   //Cursor photoCursor = managedQuery(Uri.withAppendedPath(PhotoProvider.PHOTOS_URI 4:
,"99394368305"), null, null, null, null);
5:   Cursor photoCursor = managedQuery(PhotoProvider.CONTENT_URI, null,
6:                        "page_id='99394368305'", null, null);
7:   if (photoCursor.getCount() >0){
8:       mButton2.setEnabled(true);
9:   }
10:}
```

Listing 16.10 Random Result from Cursor

```
1: //Cursor photoCursor = managedQuery(Uri.withAppendedPath(PhotoProvider.PHOTOS_URI,
2:                        "99394368305"), null, null, null, null);
3: Cursor photoCursor = managedQuery(PhotoProvider.CONTENT_URI, null,
4:                      "page_id='99394368305'", null, null);
5: int random = (int)(Math.random() * photoCursor.getCount()-1);
6: photoCursor.moveToPosition(random);
7: Photo randomPhoto =  PhotoDBHelper.getPhotoFromCursor(photoCursor);1:
8: mButton2.setEnabled(true);
```

Listing 16.11 Inserting and Updating Using ContentProvider

```
1:  mPagePhotos = Photo.makePhotoList(photoData);
2:  for (Photo currentPhoto : mPagePhotos) {
3:    currentPhoto.pageId = mPageId;
4:    ContentValues newValues = new ContentValues();
5:    if (currentPhoto.id!=null)
6:      newValues.put("photo_id", currentPhoto.id);
7:    if (currentPhoto.pageId!=null)
8:      newValues.put("page_id", currentPhoto.pageId);
9:    if (currentPhoto.name!=null)
10:     newValues.put("name", currentPhoto.name);
11:   if (currentPhoto.source!=null)
12:     newValues.put("source", currentPhoto.source);
13:   if (currentPhoto.picture!=null)
14:     newValues.put("picture", currentPhoto.picture);
15:   if (currentPhoto.createdDate!=0)
16:     newValues.put("createdDate", currentPhoto.createdDate);
17:   if (currentPhoto.modifiedDate!=0)
18:     newValues.put("modifiedDate", currentPhoto.modifiedDate);
19:   Photo exists = photoDbHelper.fetchPhotoById(currentPhoto.id);
20:   if (exists==null){
21:     Uri mNewUri = getContentResolver().insert(
22:                    PhotoProvider.CONTENT_URI,
23:                   newValues);
24:   } else{
25:     getContentResolver().update(
26:            PhotoProvider.CONTENT_URI,
27:            newValues, "photo_id" + "=" + exists.id, null);
28:   }
29: }
```

Listing 16.12 Implementing File Support in PhotoProvider

```
1: @Override
2: public ParcelFileDescriptor openFile(Uri uri, String mode)
3:                                   throws FileNotFoundException {
4:   File root = new File(getContext().getApplicationContext().getCacheDir(),
5:   uri.getEncodedPath());
6:   root.mkdirs();
7:   File imageFile = new File(root,  "image.jpg");
8:   final OutputStream imageOS;
9:   final int imode = ParcelFileDescriptor.MODE_READ_ONLY;
10:   if (imageFile.exists()) {
11:     return ParcelFileDescriptor.open(imageFile, imode);
12:  }
13:  Cursor photoCursor = query(uri, null, null, null, null);
14:  if (photoCursor == null) return null;
15:  if (photoCursor.getCount()==0) return null;
16:  Photo currentPhoto = PhotoDBHelper.getPhotoFromCursor(photoCursor);
17:  final String imageString = currentPhoto.source;
18:  imageOS = new BufferedOutputStream(new FileOutputStream(imageFile));
19:  RetrieveImage ri = new RetrieveImage (uri, imageString, imageOS);
20:  ri.execute();
21:  throw ( new FileNotFoundException());
22: }
```

Listing 16.13 Retrieving an Image in PhotoProvider

```
1:private class RetrieveImage extends AsyncTask<String , String , Long > {
2:  String  mImageString;
3:  OutputStream mImageOS;
4:  Uri mUri;
5:  public RetrieveImage ( Uri uri, String imageString, OutputStream imageOS){
6:    mImageString = imageString;
7:    mImageOS = imageOS;
8:    mUri = uri;
9:  }
10: @Override
11:   protected Long doInBackground(String... params) {
12:   try {
13:     URL imageUrl = new URL(mImageString);
14:     URLConnection connection = imageUrl.openConnection();
15:     connection.connect();
16:     InputStream is = connection.getInputStream();
17:     Bitmap mBitmap = BitmapFactory.decodeStream(is);
18:     mBitmap.compress(Bitmap.CompressFormat.JPEG, 100, mImageOS);
19:     mImageOS.flush();
20:     mImageOS.close();
21:     getContext().getContentResolver().notifyChange(mUri, null);
22:     return (0l);
23:   } catch (MalformedURLException e) {
24:     e.printStackTrace();
25:     return (1l);
26:   }
27:   catch (IOException e) {
28:     e.printStackTrace();
29:     return (1l);
30:   }
31: }
32:}
```

Listing 16.14 Creating a Bitmap Using the ContentProvider

```
1:  @Override
2:  public void onActivityCreated(Bundle savedInstanceState) {
3:    super.onActivityCreated(savedInstanceState);
4:    Bundle b = this.getArguments();
5:    if (b!=null){
6:      mPhotoId=b.getString("PHOTO_ID");
7:    }
8:    Handler handler = new Handler();
```

```
9:    InputStream is;
10:   try {
11:     is = getActivity().getContentResolver().openInputStream
12:                     (Uri.withAppendedPath(PhotoProvider.CONTENT_URI ,mPhotoId));
13:     mBitmap =  BitmapFactory.decodeStream(is);
14:     mImageView.setImageBitmap(mBitmap);
15:   } catch (FileNotFoundException e) {
16:       getActivity().getContentResolver().registerContentObserver(
17:                      Uri.withAppendedPath(PhotoProvider.CONTENT_URI ,mPhotoId),
18:                      false, new PhotoContentObserver(handler));
19:   }
20: }
```

Listing 16.15 ContentObserver Defined in ImageViewFragment

```
1:  class PhotoContentObserver extends ContentObserver {
2:    public PhotoContentObserver(Handler handler) {
3:      super(handler);
4:    }
5:    @Override
6:    public void onChange(boolean selfChange) {
7:      InputStream is;
8:      try {
9:        if (ImageViewFragment.this.isAdded()){
10:         is = getActivity().getContentResolver().openInputStream
11:              Uri.withAppendedPath(PhotoProvider.CONTENT_URI ,mPhotoId));
12:          mBitmap =  BitmapFactory.decodeStream(is);
13:          mImageView.setImageBitmap(mBitmap);
14:       }
15:     } catch (FileNotFoundException e) {
16:    }
17:   }
18: }
```

Hour 17

Listing 17.1 ImageViewFragment Loader Setup

```
1:  public class ImageViewFragment extends Fragment implements LoaderCallbacks<Cursor>{
2:    ImageView mImageView;
3:    String mPageId;
4:    Bitmap mBitmap;
5:    String mPhotoId;
6:    @Override
7:    public void onActivityCreated(Bundle savedInstanceState) {
8:      super.onActivityCreated(savedInstanceState);
9:      Bundle b = this.getArguments();;
10:     mPageId=b.getString("PAGE_ID");
11:     getLoaderManager().initLoader(0, null, this);
12:   }
13:   @Override
14:   public View onCreateView(LayoutInflater inflater,
15:               ViewGroup container, Bundle savedInstanceState) {
16:     mImageView  = (ImageView) inflater.inflate(R.layout.image_fragment,
17:                     container, false);
18:     return mImageView;
19:   }
20:   @Override
21:   public Loader<Cursor> onCreateLoader(int id, Bundle arg1) {
22:     CursorLoader cursorLoader = new CursorLoader(getActivity(),
23:     Uri.withAppendedPath(PhotoProvider.PHOTOS_URI ,mPageId),
24:                                    null, null, null, null);
25:     return cursorLoader;
26:   }
27:   @Override
28:   public void onLoaderReset(Loader<Cursor> arg0) {
29:   }
```

Listing 17.2 ImageViewFragment Using the Data

```
1:  @Override
2:  public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {
3:    int random = (int)(Math.random() * cursor.getCount()-1);
4:    cursor.moveToPosition(random);
5:    Photo randomPhoto =  PhotoDBHelper.getPhotoFromCursor(cursor);
6:    mPhotoId = randomPhoto.id;
7:    Handler handler = new Handler();
8:    InputStream is;
9:    try {
10:     if (ImageViewFragment.this.isAdded()){
11:       is = getActivity().getContentResolver(). openInputStream
12:           (Uri.withAppendedPath(PhotoProvider.CONTENT_URI ,mPhotoId));
13:       mBitmap =  BitmapFactory.decodeStream(is);
14:       mImageView.setImageBitmap(mBitmap);
15:     }
16:   } catch (FileNotFoundException e) {
17:     getActivity().getContentResolver(). registerContentObserver(
18:     Uri.withAppendedPath(PhotoProvider.CONTENT_URI ,mPhotoId), false,
19:     new PhotoContentObserver(handler));
20:   }
21: }
22: class PhotoContentObserver extends ContentObserver {
23:   public PhotoContentObserver(Handler handler) {
24:     super(handler);
25:
26:     @Override
27:     public void onChange(boolean selfChange) {
28:       InputStream is;
29:       try {
30:         if (ImageViewFragment.this.isAdded()){
31:           is = getActivity().getContentResolver().
32:           openInputStream(Uri.withAppendedPath(PhotoProvider.CONTENT_URI,mPhotoId));
33:           mBitmap =  BitmapFactory.decodeStream(is);
34:           mImageView.setImageBitmap(mBitmap);
35:        }
36:      } catch (FileNotFoundException e) {
37:    }
38:  }
39: }
```

Listing 17.3 Showing Photo Data in a List with PhotoListFragment.java

```
1:  public class PhotoListFragment extends ListFragment
2:              implements LoaderCallbacks<Cursor>  {
3:    SimpleCursorAdapter mAdapter;
4:    String mPageId;
5:    @Override
6:    public void onActivityCreated(Bundle savedInstanceState) {
7:      super.onActivityCreated(savedInstanceState);
8:      Bundle b = this.getArguments();;
9:      mPageId=b.getString("PAGE_ID");
10:     getLoaderManager().initLoader(0, null, this);
11:     mAdapter = new SimpleCursorAdapter(getActivity(),
12:       android.R.layout.simple_list_item_2,
13:       null, //Cursor
14:       new String[] {"photo_id", "source"},
15:       new int[] { android.R.id.text1, android.R.id.text2 }, 0);
16:       setListAdapter(mAdapter);
17:   }
18:   @Override
19:   public Loader<Cursor> onCreateLoader(int id, Bundle args) {
20:     CursorLoader cursorLoader = new CursorLoader(getActivity(),
21:     Uri.withAppendedPath(PhotoProvider.PHOTOS_URI ,mPageId),
22:                          null, null, null, null);
23:     return cursorLoader;
24:   }
25:   @Override
26:   public void onLoadFinished(Loader<Cursor> Loader, Cursor cursor) {
27:     mAdapter.swapCursor(cursor);
```

```
28:   }
29:   @Override
30:   public void onLoaderReset(Loader<Cursor> arg0) {
31:     mAdapter.swapCursor(null);
32:   }
33: }
```

Hour 18

Listing 18.1 Creating the GridView

```
1:  @Override
2:  public View onCreateView(LayoutInflater inflater, ViewGroup container,
3:              Bundle savedInstanceState) {
4:    grid = (GridView) inflater.inflate(R.layout.grid_fragment, container, false);
5:    return grid;
6:  }
7:  @Override
8:  public Loader<Cursor> onCreateLoader(int id, Bundle args) {
9:    CursorLoader cursorLoader = new CursorLoader(getActivity(),
10:   Uri.withAppendedPath(PhotoProvider.PHOTOS_URI ,mPageId),
11:             null, null, null, null);
12:   return cursorLoader;
13: }
14: @Override
15: public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {
16:   gridAdapter.swapCursor(cursor);
17: }
18: @Override
19: public void onLoaderReset(Loader<Cursor> loader) {
20:   gridAdapter.changeCursor(null);
21: }
```

Listing 18.2 GridView Layout

```
1:  <?xml version="1.0" encoding="utf-8"?>
2:  <GridView
3:      xmlns:android="http://schemas.android.com/apk/res/android"
4:      android:layout_width="fill_parent"
5:      android:layout_height="fill_parent"
6:      android:numColumns="4">
7:  </GridView>
```

Listing 18.3 Adding an Adapter in onActivityCreated

```
1:  public void onActivityCreated(Bundle savedInstanceState) {
2:    super.onActivityCreated(savedInstanceState);
3:    Bundle b = this.getArguments();;
4:    mPageId=b.getString("PAGE_ID");
5:    currentActivity = this.getActivity();
6:    getLoaderManager().initLoader(2, null, this);
7:    gridAdapter = new PhotoGridCursorAdapter(currentActivity, null, grid);
8:    grid.setAdapter(gridAdapter);
9:    grid.setOnItemClickListener(new OnItemClickListener() {
10:     public void onItemClick(AdapterView<?> parent, View view,int position,long id) {
11:       Bundle args = new Bundle();
12:       args.putString("PAGE_ID", mPageId);
13:       args.putInt("PHOTO_POSITION", position);
14:       PhotoAdapterViewFragment pgFragment = new PhotoAdapterViewFragment();
15:       pgFragment.setArguments(args);
16:       FragmentTransaction ft = getFragmentManager().beginTransaction();
17:       ft.replace(R.id.linearLayout1, pgFragment, "Image Viewer");
18:       ft.commit();
19:     }
20:   });
21: }
```

Listing 18.4 Creating the PhotoCursorGridAdapter

```
1:  public class PhotoGridCursorAdapter extends android.widget.CursorAdapter  {
2:    Context context;
3:    Activity mActivity;
4:    Cursor c;
5:    Bitmap mBitmap;
6:    GridView mGrid;
7:    private LayoutInflater inflater;
8:    public PhotoGridCursorAdapter(Context context, Cursor c, GridView g) {
9:      super(context, c);
10:     this.context = context;
11:     this.inflater = LayoutInflater.from(context);
12:     this.c = c;
13:     this.mGrid=g;
14:     this.mActivity = (Activity) context;
15:   }
16:   @Override
17:   public View newView(Context context, Cursor cursor, ViewGroup parent) {
18:     View v = inflater.inflate(R.layout.photo_grid_item, parent, false);
19:     return (v);
20:   }
```

Listing 18.5 Grid Item Layout

```
1:  <?xml version="1.0" encoding="utf-8"?>
2:  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:      android:id="@+id/relativeLayout1"
4:      android:layout_width="wrap_content" android:layout_height="wrap_content"
5:      android:cacheColorHint="#00000000"  android:padding="12dp" >
6:      <ImageView
7:          android:id="@+id/photoImage"
8:          android:layout_width="200dp"
9:          android:layout_height="200dp"
10:         android:layout_centerHorizontal="true"
11:         android:adjustViewBounds="true"
12:         android:paddingBottom="2dp"
13:         android:paddingLeft="4dp"
14:         android:paddingRight="4dp"
15:         android:paddingTop="8dp"
16:         android:scaleType="centerCrop"
17:    </ImageView>
18:  </RelativeLayout>
```

Listing 18.6 PhotoCursorGridAdapter BindView

```
1:   @Override
2:   public void bindView(View v, Context context, Cursor c) {
3:     PhotoViewHolder vh = (PhotoViewHolder) v.getTag();
4:     if(vh==null){
5:       vh = new PhotoViewHolder();
6:       vh.imageView = (ImageView) v.findViewById(R.id.photoImage);
7:       v.setTag(vh);
8:     }
9:     int idCol = c.getColumnIndex("photo_id");
10:    String id = c.getString(idCol);
11:    vh.imageView.setTag(id);
12:    Handler handler = new Handler();
13:    InputStream is;
14:    try {
15:      is =  mActivity.getContentResolver().openInputStream
16:           (Uri.withAppendedPath(PhotoProvider.CONTENT_URI ,id));
17:      mBitmap =  BitmapFactory.decodeStream(is);
18:      vh.imageView.setImageBitmap(mBitmap);
19:    } catch (FileNotFoundException e) {
20:       mActivity.getContentResolver().registerContentObserver
21:       (Uri.withAppendedPath(PhotoProvider.CONTENT_URI ,id), false,
22:       new PhotoGridContentObserver(handler, id));
23:    }
24:  }
```

```
25:  public static class PhotoViewHolder{
26:    public long imageId;
27:    public ImageView imageView;
28:  }
```

Listing 18.7 PhotoCursorGridAdapter BindView

```
1: class PhotoGridContentObserver extends ContentObserver {
2:   String mId;
3:   public PhotoGridContentObserver(Handler handler, String id) {
4:     super(handler);
5:     mId = id;
6:   }
7:   @Override
8:   public void onChange(boolean selfChange) {
9:     InputStream is;
10:    try {
11:        is = mActivity.getContentResolver().openInputStream
12:            (Uri.withAppendedPath(PhotoProvider.CONTENT_URI ,mId));
13:        mBitmap =  BitmapFactory.decodeStream(is);
14:        ImageView imageViewByTag = (ImageView) mGrid.findViewWithTag(mId);
15:        if (imageViewByTag != null) {
16:           imageViewByTag.setImageBitmap(mBitmap);
17:        }
18:    } catch (FileNotFoundException e) {
19:    }
20:  }
21:}
```

Listing 18.8 PhotoAdapterViewFragment onLoadFinished

```
1: @Override
2: public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {
3:   adapter.swapCursor(cursor);
4:   mAdapterView.setSelection(photoPosition);
5:   mAdapterView.setAutoStart(true);
6:   mAdapterView.startFlipping();
7: }
```

Listing 18.9 Layout view_fragment.xml

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <AdapterViewFlipper
3: xmlns:android="http://schemas.android.com/apk/res/android"
4: android:layout_width="fill_parent"
5: android:layout_height="fill_parent" >
6: </AdapterViewFlipper>
```

Listing 18.10 Layout view_fragment.xml

```
1: @Override
2: protected void onPostExecute(Long result) {
3: getContentResolver().notifyChange
4:                  (Uri.withAppendedPath(PhotoProvider.PHOTOS_URI ,mPageId), null);
5: }
```

Listing 18.11 ShowPagePhoto Method

```
1:  public void showPagePhoto(String id){
2:    LoadPhotos lp = new LoadPhotos(id );
3:    lp.execute();
4:    GridFragment plFragment = new GridFragment();
5:    Bundle args = new Bundle();
6:    args.putString("PAGE_ID", id);
7:    plFragment.setArguments(args);
8:    FragmentTransaction ft = getFragmentManager().beginTransaction();
9:    ft.addToBackStack(id);
10:   ft.replace(R.id.linearLayout1, plFragment, "page "+ id);
```

```
11:   ft.commit();
12: }
```

Listing 18.12 Defining the ActionBar

```
1:   @Override
2:   public void onCreate(Bundle savedInstanceState) {
3:     super.onCreate(savedInstanceState);
4:     setContentView(R.layout.main);
5:     mActionBar = getActionBar();
6:     mTab1= mActionBar.newTab().setText("Coca Cola").
7:            setTabListener(new PagePhotoTabListener());
8:     mTab2= mActionBar.newTab().setText("Pepsi").
9:            setTabListener(new PagePhotoTabListener());
10:    mTab3= mActionBar.newTab().setText("More Choices").
11:           setTabListener(new PagePhotoTabListener());
12:    mActionBar.setDisplayShowTitleEnabled(false);
13:    mActionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
14:    mActionBar.addTab(mTab1);
15:    mActionBar.addTab(mTab2);
16:    mActionBar.addTab(mTab3);
17:  }
18:  private class PagePhotoTabListener implements ActionBar.TabListener {
19:  @Override
20:    public void onTabSelected(Tab tab, FragmentTransaction ft) {
21:      if (tab ==mTab1){
22:        showPagePhoto("99394368305");
23:      }
24:      if (tab ==mTab2){
25:        showPagePhoto("123490744049");
26:      }
27:      if (tab==mTab3){
28:        ListDialogFragment listDialog = new ListDialogFragment();
29:        listDialog.show(getFragmentManager(), "list_dialog");
30:      }
31:    }
```

Listing 18.13 ListDialogFragment onCreateView

```
1:   @Override
2:   public View onCreateView(LayoutInflater inflater, ViewGroup
3:                container, Bundle savedInstanceState) {
4:     View v = inflater.inflate(R.layout.list_dialog_fragment, container, false);
5:     mListView = (ListView)v.findViewById(R.id.listView1);
6:     Button button = (Button)v.findViewById(R.id.button1);
7:     button.setOnClickListener(new OnClickListener() {
8:       public void onClick(View v) {
9:         ListDialogFragment.this.dismiss();
10:      }
11:    });
12:    return v;
13: }
```

Listing 18.14 ListDialogFragment onActivityCreated

```
1:   public class ListDialogFragment extends DialogFragment   {
2:     ListView mListView;
3:     HashMap mValues = new HashMap();
4:     @Override
5:     public void onActivityCreated(Bundle savedInstanceState) {
6:       super.onActivityCreated(savedInstanceState);
7:       mValues.put("BFF Media","304703469573709");
8:       mValues.put("Nurse Jackie","183648859772");
9:       mValues.put("True Blood","81916858562");
10:      getDialog().setTitle("Choose Page");
11:      final Object[] values = mValues.keySet().toArray();
12:      ArrayAdapter adapter = new ArrayAdapter(
13:        getActivity(),
14:        android.R.layout.simple_list_item_1,
```

```
15:        values);
16:      mListView.setAdapter(adapter);
17:      mListView.setOnItemClickListener(new OnItemClickListener() {
18:       public void onItemClick(AdapterView<?> parent, View v, int position, long id){
19:          Hour18Activity activity = (Hour18Activity)getActivity();
20:          activity.showPagePhoto((String)mValues.get(values[position]));
21:          ListDialogFragment.this.dismiss();
22:       }
23:      });
24:  }
```

Hour 19

Listing 19.1 AndroidManifest.xml

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3:     package="com.bffmedia.content"
4:     android:versionCode="1"
5:     android:versionName="1.0" >
6:     <application
7:         android:icon="@drawable/icon"
8:         android:label="@string/app_name" >
9:         <activity
10:            android:name=".Hour18Activity"
11:            android:label="@string/app_name" >
12:            <intent-filter>
13:                <action android:name="android.intent.action.MAIN" />
14:                <category android:name="android.intent.category.LAUNCHER" />
15:            </intent-filter>
16:        </activity>
17:        <provider
18:            android:name="com.bffmedia.content.PhotoProvider"
19:            android:authorities="com.bffmedia.content.PhotoProvider"
20:            android:multiprocess="true" >
21:        </provider>
22:    </application>
23:    <uses-sdk android:minSdkVersion="12" />
24:    <uses-permission android:name="android.permission.INTERNET" >
25:    </uses-permission>
26: </manifest>
```

Hour 20

Listing 20.1 Accessing Data in the Channel Provider

```
1:  package com.bffmedia.channel;
2:  import android.app.Activity;
3:  import android.database.Cursor;
4:  import android.net.Uri;
5:  import android.os.Bundle;
6:  import android.widget.Toast;
7:  public class Hour20ChannelActivity extends Activity {
8:  @Override
9:  public void onCreate(Bundle savedInstanceState) {
10:   super.onCreate(savedInstanceState);
11:   setContentView(R.layout.main);
12:   Uri channelUri = Uri.parse("content://"
13:                              + "com.google.android.tv.provider"
14:                              + "/" + "channel_listing");
15:   String mWhere = "callsign = 'CNN'";
16:   Cursor mCursor = getContentResolver().query(channelUri,
17:                    null,  mWhere, null, null
18:   );
19:   if (null == mCursor) {
20:      Toast.makeText(this,"Empty cursor",Toast.LENGTH_LONG).show();
21:      return;
```

```
22:   }
23:   if (mCursor.getCount() == 0 ) {
24:      Toast.makeText(this,"Callsign Not Found",Toast.LENGTH_LONG).show();
25:      return;
26:   }
27:   mCursor.moveToFirst();
28:   Toast.makeText(this,
29:       mCursor.getString(mCursor.getColumnIndex("callsign"))
30:       + "\n Channel " + mCursor.getString(mCursor.getColumnIndex("channel_number"))
31:       + "\n" + mCursor.getString(mCursor.getColumnIndex("channel_name")),
32:       Toast.LENGTH_LONG).show();
33: }    // end OnCreate
34:}     // end Activity
```

Listing 20.2 Channels in a ListFragment

```
1:   public class ChannelListFragment extends ListFragment
2:                implements LoaderCallbacks<Cursor>  {
3:     SimpleCursorAdapter mAdapter;
4:     @Override
5:     public void onActivityCreated(Bundle savedInstanceState) {
6:       super.onActivityCreated(savedInstanceState);
7:       getLoaderManager().initLoader(0, null, this);
8:       mAdapter = new SimpleCursorAdapter(getActivity(),
9:           android.R.layout.simple_list_item_2,
10:          null,
11:          new String[] {"callsign", "channel_name"},
12:          new int[] { android.R.id.text1, android.R.id.text2 }, 0);
13:      setListAdapter(mAdapter);
14:    }
15:    @Override
16:    public Loader<Cursor> onCreateLoader(int id, Bundle args) {
17:      Uri channelProviderUri = Uri.parse("content://"
18:                               + "com.google.android.tv.provider"
19:                               + "/" + "channel_listing");
20:      CursorLoader cursorLoader = new CursorLoader(getActivity(),
21:      channelProviderUri ,
22:          null, null, null, "callsign ASC");
23:      return cursorLoader;
24:    }
25:    @Override
26:    public void onLoadFinished(Loader<Cursor> Loader, Cursor cursor) {
27:      mAdapter.swapCursor(cursor);
28:    }
29:    @Override
30:    public void onLoaderReset(Loader<Cursor> arg0) {
31:      mAdapter.swapCursor(null);
32:    }
33:  }
```

Listing 20.3 GridView Fragment for Changing Channels

```
1:public class GridFragment extends Fragment implements LoaderCallbacks<Cursor> {
2:   GridView grid;
3:   Activity currentActivity;
4:   ChannelGridCursorAdapter gridAdapter;
5:   @Override
6:   public void onActivityCreated(Bundle savedInstanceState) {
7:     super.onActivityCreated(savedInstanceState);
8:     currentActivity = this.getActivity();
9:     getLoaderManager().initLoader(0, null, this);
10:    gridAdapter = new ChannelGridCursorAdapter(currentActivity, null, grid);
11:    grid.setAdapter(gridAdapter);
12:    grid.setOnItemClickListener(new OnItemClickListener() {
13:     public void onItemClick(AdapterView<?> parent,View view,int position,long id) {
14:       ChannelViewHolder channelInfo = (ChannelViewHolder) view.getTag();
15:       System.out.println(channelInfo.channelUri);
16:       Uri channelUri = Uri.parse(channelInfo.channelUri);
17:       startActivity(new Intent(Intent.ACTION_VIEW, channelUri));
18:     }
```

```
19:    });
20:  }
21:  @Override
22:  public View onCreateView(LayoutInflater inflater, ViewGroup container,
23:                Bundle savedInstanceState) {
24:      grid = (GridView) inflater.inflate(R.layout.grid_fragment, container, false);
25:      return grid;
26:  }
27:  @Override
28:  public Loader<Cursor> onCreateLoader(int id, Bundle args) {
29:     Uri channelProviderUri = Uri.parse("content://"
30:         + "com.google.android.tv.provider" + "/" + "channel_listing");
31:     CursorLoader cursorLoader = new CursorLoader(getActivity(),
32:     channelProviderUri ,
32:     null, null, null, "callsign ASC");
34:     return cursorLoader;
35:  }
```

Listing 20.4 ChannelGridCursorAdapter Snippet

```
1: @Override
2:  public void bindView(View v, Context context, Cursor c) {
3:    ChannelViewHolder vh = (ChannelViewHolder) v.getTag();
4:    if(vh==null){
5:      vh = new ChannelViewHolder();
6:      vh.callsignView = (TextView) v.findViewById(R.id.callsign);
7:      vh.channelNumberView = (TextView) v.findViewById(R.id.channel);
8:      vh.channelNameView = (TextView) v.findViewById(R.id.name);
9:      v.setTag(vh);
10:   }
11:   int callsignCol = c.getColumnIndex("callsign");
12:   String callsign = c.getString(callsignCol);
13:   vh.callsignView.setText(callsign);
14:   int channelNumberCol = c.getColumnIndex("channel_number");
15:   String channelNumber = c.getString(channelNumberCol);
16:   vh.channelNumberView.setText(channelNumber);
17:   int channelNameCol = c.getColumnIndex("channel_name");
18:   String channelName = c.getString(channelNameCol);
19:   vh.channelNameView.setText(channelName);
20:   int channelUriCol = c.getColumnIndex("channel_uri");
21:   vh.channelUri = c.getString(channelUriCol);
22:   v.setTag(vh);
23: }
24: public static class ChannelViewHolder{
25:   public TextView callsignView;
26:   public TextView channelNumberView;
27:   public TextView channelNameView;
28:   public String channelUri;
29: }
```

Listing 20.5 Requesting AudioFocus

```
1: AudioManager audioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
2: int result = audioManager.requestAudioFocus(this, AudioManager.STREAM_MUSIC,
3:     AudioManager.AUDIOFOCUS_GAIN);
4: if (result != AudioManager.AUDIOFOCUS_REQUEST_GRANTED) {
5: }
```

Listing 20.6 Handling a Change in AudioFocus

```
1: public void onAudioFocusChange(int focusChange) {
2:     switch (focusChange) {
3:         case AudioManager.AUDIOFOCUS_GAIN:
4:             // gained focus
5:             break;
6:         case AudioManager.AUDIOFOCUS_LOSS:
7:             // Lost focus
8:             break;
9:         case AudioManager.AUDIOFOCUS_LOSS_TRANSIENT:
```

```
10:            // Lost focus temporarily.  Stop playback
11:            break;
12:        case AudioManager.AUDIOFOCUS_LOSS_TRANSIENT_CAN_DUCK:
13:            // Lost focus temporarily. Lower volume.
14:            break;
15:    }
16:}
```

Hour 22

Listing 22.1 The Blackjack "Hit" Button Code

```
1: hit.setOnClickListener(new OnClickListener() {
2:   @Override
3:   public void onClick(View v) {
4:   // Sends Keycode H for 'Hit'.
5:      sendKeyEvent(KeyEvent.KEYCODE_H);
6:   }
7: });
```

Listing 22.2 Sending a Key to TV App

```
1:  private void sendKeyEvent(final int keyEvent) {
2:    if (anymoteSender == null) {
3:        Toast.makeText(BlackJackRemoteActivity.this, "Waiting for connection",
4:                       Toast.LENGTH_LONG).show();
5:        return;
6:     }
7:     anymoteSender.sendKeyPress(keyEvent);
8:  }
```

Listing 22.3 Handling a Key Sent from the Remote App

```
1:  @Override
2:    public boolean onKeyDown(int keyCode, KeyEvent event) {
3:      if (keyCode == KeyEvent.KEYCODE_H) {
4:           mBoundService.hitPlayer();
5:       }
```

Listing 22.4 Android Manifest for Remote App

```
1:   <service android:name=
2:      "com.example.google.tv.anymotelibrary.client.AnymoteClientService">
3:   </service>
4:   <activity
5:      android:name=
6:      "com.example.google.tv.anymotelibrary.connection.PairingActivity"
7:      android:configChanges="orientation"
8:      android:label="Pairing with TV"
9:      android:screenOrientation="portrait" />
```

Listing 22.5 Android Manifest for Remote App

```
1:  @Override
2:    protected void doInitializationPhase()
3:        throws PoloException, IOException {
4:      logDebug("Sending PairingRequest... " + mServiceName);
5:      PairingRequestMessage msg = new PairingRequestMessage(mServiceName);
6:      sendMessage(msg);
7:      logDebug("Waiting for PairingRequestAck ...");
8:      PairingRequestAckMessage ack = (PairingRequestAckMessage) getNextMessage(
9:          PoloMessageType.PAIRING_REQUEST_ACK);
10:      if (ack.hasServerName()) {
11:        mPeerName = ack.getServerName();
12:        logDebug("Got PairingRequestAck with server name = " + mPeerName);
13:   } else {
```

```
14:     mPeerName = null;
15:   }
16:   logDebug("Sending Options ...");
17:   sendMessage(mLocalOptions);
18:   logDebug("Waiting for Options...");
19:   OptionsMessage serverOptions = (OptionsMessage) getNextMessage(
20:       PoloMessageType.OPTIONS);
21:  // Compare compatibility with server options, and save config.
22:   System.out.println("Local config = " + mLocalOptions);
23:   System.out.println("Server options = " + serverOptions);
24:   setConfiguration(mLocalOptions.getBestConfiguration(serverOptions));
25:   }
```

Hour 23

Listing 23.1 ServiceConnection Declaration

```
1: private AnymoteSender anymoteSender;
2: private ServiceConnection mConnection = new ServiceConnection() {
3:   public void onServiceConnected(ComponentName name, IBinder service) {
4:      mAnymoteClientService=((AnymoteClientService.AnymoteClientServiceBinder)
5:      service).getService();
6:      mAnymoteClientService.attachClientListener(ExampleOneActivity.this);
7:   }
8:   public void onServiceDisconnected(ComponentName name) {
9:      mAnymoteClientService.detachClientListener(ExampleOneActivity.this);
10:      mAnymoteClientService = null;
11:   }
12:};
```

Listing 23.2 Remote Activity onCreate Method

```
1:  @Override
2:  public void onCreate(Bundle savedInstanceState) {
3:    super.onCreate(savedInstanceState);
4:    mContext = this;
5:    setContentView(R.layout.main);
6:    progressBar = (ProgressBar) findViewById(R.id.a_progressbar);
7:    progressBar.setVisibility(View.VISIBLE);
8:    Button go = (Button) findViewById(R.id.go);
9:    final EditText destination = (EditText) findViewById(R.id.destination);
10:   go.setOnClickListener(new OnClickListener() {
11:     @Override
12:     public void onClick(View v) {
13:       String url = destination.getText().toString();
14:       final Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
15:       anymoteSender.sendUrl (intent.toUri(Intent.URI_INTENT_SCHEME));
16:     }
17:   });
18:   handler = new Handler();
19:   Intent intent = new Intent(ExampleOneActivity.this, AnymoteClientService.class);
20:   bindService(intent, mConnection, Context.BIND_AUTO_CREATE);
21: }
```

Listing 23.3 Required Methods for ClientListener

```
1:   @Override
2:   public void onConnected(final AnymoteSender anymoteSender) {
3:     this.anymoteSender = anymoteSender;
4:     handler.post(new Runnable() {
5:       public void run() {
6:         progressBar.setVisibility(View.INVISIBLE);
7:       }
8:     });
9:   }
10:  @Override
11:  public void onDisconnected() {
12:    this.anymoteSender = null;
```

```
13:  }
14:  @Override
15:  public void onConnectionFailed() {
16:    System.out.println("connection failed");
17:  }
```

Listing 23.4 Cleaning Up in the onDestroy Method

```
1:  @Override
2:  protected void onDestroy() {
3:    if (mAnymoteClientService != null) {
4:      mAnymoteClientService.detachClientListener(this);
5:    }
6:    unbindService(mConnection);
7:    super.onDestroy();
8:  }
```

Listing 23.5 Facebook Random Image Remote App

```
1:  @Override
2:  public void onCreate(Bundle savedInstanceState) {
3:    super.onCreate(savedInstanceState);
4:    setContentView(R.layout.main);
5:    LoadPhotos lp = new LoadPhotos("99394368305" );
6:    lp.execute();
7:    progressBar = (ProgressBar) findViewById(R.id.progressBar1);
8:    progressBar.setVisibility(View.VISIBLE);
9:    mButton2= (Button)findViewById(R.id.button2);
10:   mButton2.setEnabled(false);
11:   mButton2.setOnClickListener(new OnClickListener() {
12:     public void onClick(View v) {
13:       ImageViewFragment fbImg = new ImageViewFragment();
14:       Bundle args = new Bundle();
15:       int random = (int)(Math.random() * mPagePhotos.size());
16:       String url = mPagePhotos.get(random).source;
17:       args.putString("URL", url);
18:       final Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
19:       anymoteSender.sendUrl (intent.toUri(Intent.URI_INTENT_SCHEME));
20:       fbImg.setArguments(args);
21:       FragmentTransaction ft = getFragmentManager().beginTransaction();
22:       ft.replace(R.id.linearLayout1, fbImg, "Image from Facebook");
23:       ft.commit();
24:     }
25:   });
26:   handler = new Handler();
27:   Intent intent = new Intent(ExampleTwoActivity.this, AnymoteClientService.class);
28:   bindService(intent, mConnection, Context.BIND_AUTO_CREATE);
29: }
```

Listing 23.6 onKeyDown and onKeyUp Methods

```
1:  @Override
2:    public boolean onKeyDown(int keyCode, KeyEvent event) {
3:      anymoteSender.sendKey(Code.valueOf(keyCode),Key.Action.DOWN);
4:      return true;
5:    }
6:    @Override
7:      public boolean onKeyUp(int keyCode, KeyEvent event) {
8:        anymoteSender.sendKey(Code.valueOf(keyCode),Key.Action.UP);
9:        return true;
10:   }
```

Listing 23.7 Companion Activity on TV for Echoing Keystrokes

```
1:  public class Hour23Echo extends Activity {
2:    EditText echoText;
3:    @Override
4:      public void onCreate(Bundle savedInstanceState) {
```

```
5:        super.onCreate(savedInstanceState);
6:        setContentView(R.layout.main);
7:        echoText= (EditText)findViewById(R.id.echo);
8:    }
9:  }
```

Listing 23.8 Starting the Companion App from the Remote App

```
1:  @Override
2:  public void onConnected(final AnymoteSender anymoteSender) {
3:    this.anymoteSender = anymoteSender;
4:    //Add this section to open the Companion App
5:    final Intent echoIntent = new Intent("android.intent.action.MAIN");
6:    echoIntent.setComponent(new ComponentName(
7:        "com.bffmedia.hour23echo",
8:        "com.bffmedia.hour23echo.Hour23Echo"));
9:    anymoteSender.sendIntent(echoIntent);
10:   handler.post(new Runnable() {
11:     public void run() {
12:       progressBar.setVisibility(View.INVISIBLE);
13:     }
14:   });
15: }
```

Listing 23.9 Accelerometer App onCreate

```
1:  Button click = (Button) findViewById(R.id.button1);
2:  click.setOnClickListener(new OnClickListener() {
3:    @Override
4:    public void onClick(View v) {
5:      anymoteSender.sendKey (Code.BTN_MOUSE, Action.DOWN);
6:      anymoteSender.sendKey (Code.BTN_MOUSE, Action.UP);
7:    }
8:  });
```

Listing 23.10 Defining SensorManager in onConnected Method

```
1:  @Override
2:  public void onConnected(final AnymoteSender anymoteSender) {
3:    this.anymoteSender = anymoteSender;
4:    sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
5:    sensorManager.registerListener(this,
6:    sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
7:        SensorManager.SENSOR_DELAY_FASTEST);
8:    handler.post(new Runnable() {
9:      public void run() {
10:       progressBar.setVisibility(View.INVISIBLE);
11:     }
12:   });
13: }
```

Listing 23.11 Detecting Sensor Changes

```
1:  @Override
2:  public void onAccuracyChanged(Sensor Sensor, int accuracy) {
3:  }
4:  @Override
5:  public void onSensorChanged(SensorEvent event) {
6:    if(event.sensor.getType()==Sensor.TYPE_ACCELEROMETER){
7:      float x=event.values[0];
8:      float y=event.values[1];
9:      float z=event.values[2];
10:     anymoteSender.sendMoveRelative((int)x*-1,(int) ((y-1)*-1));
11:   }
12: }
```

Hour 24

Listing 24.1 Keycode Proto File Snippet

```
1:   enum Code {
2:       KEYCODE_UNKNOWN            = 0;
3:       KEYCODE_SOFT_LEFT          = 1;
4:       KEYCODE_SOFT_RIGHT         = 2;
5:       KEYCODE_HOME               = 3;
6:       KEYCODE_BACK               = 4;
7:   …
8:       KEYCODE_DPAD_UP            = 19;
9:       KEYCODE_DPAD_DOWN          = 20;
10:       KEYCODE_DPAD_LEFT          = 21;
11:       KEYCODE_DPAD_RIGHT         = 22;
12:       KEYCODE_DPAD_CENTER        = 23;
13:   …
14:       BTN_BACK                   = 278;
15:       BTN_TASK                   = 279;
16:   }
17:   enum Action {
18:       // Key released
19:       UP     = 0;
20:       // Key pressed
21:      DOWN   = 1;
22:   }
```

Listing 24.2 Request Messages in the Remote Proto File

```
1:    / Sends a key event to the server
2:   message KeyEvent {
3:     // Key code
4:     required Code keycode = 1;
5:     // Action (Up/Down)
6:    required Action action = 2;
7:   }
8:   // Sends a mouse event to the server
9:   message MouseEvent {
10:     // Relative movement of the cursor on the xAxis
11:     required int32 x_delta = 1;
12:     // Relative movement of the cursor on the yAxis
13:     required int32 y_delta = 2;
14:   }
15:   // Sends a mouse wheel to the server
16:   message MouseWheel {
17:     // Scrolling along the x-axis
18:     required int32 x_scroll = 1;
19:     // Scrolling along the y-axis
20:     required int32 y_scroll = 2;
21:   }
22:   message Connect {
23:     // Remote device name
24:     required string device_name = 1;
25:     // Version number for a given device software
26:     optional int32 version = 2;
27:   }
28:   message Fling {
29:     // Flinged URI
30:     required string uri = 1;
31:   }
```

Listing 24.3 Interpreting a Request Message

```
1:    /*
2:     * Interprets a request message.
3:     * @param message the request message
4:     */
5:    private void interpretRequest(RequestMessage message, Integer sequenceNumber) {
6:      ResponseMessage.Builder builder = ResponseMessage.newBuilder();
```

```
7:      boolean reply = sequenceNumber != null;
8:    if (message.hasKeyEventMessage()) {
9:       reply = false;
10:      onKeyEvent(message.getKeyEventMessage());
11:    }
12:    if (message.hasMouseEventMessage()) {
13:      reply = false;
14:      onMouseEvent(message.getMouseEventMessage());
15:    }
16:    if (message.hasMouseWheelMessage()) {
17:      reply = false;
18:      onMouseWheel(message.getMouseWheelMessage());
19:    }
20:    if (message.hasDataMessage()) {
21:      reply = false;
22:     onData(message.getDataMessage());
23:    }
24:    if (message.hasConnectMessage()) {
25:      reply = false;
26:      onConnect(message.getConnectMessage());
27:    }
28:    if (message.hasFlingMessage()) {
29:      reply = true;
30:      builder.setFlingResultMessage(
31:          onFling(message.getFlingMessage(), sequenceNumber));
32:    }
33:    if (reply) {
34:     sendResponse(builder, sequenceNumber);
35:    }
36:   }
```

Listing 24.4 PairingRequest and PairingRequestAck

```
1: message PairingRequest {
2:   // String name of the service to pair with.  The name used should be an
3:   // established convention of the application protocol.
4:   required string service_name = 1;
5:   // Descriptive name of the client.
6:   optional string client_name = 2;
7: }
8: message PairingRequestAck {
9:   // Descriptive name of the server.
10:  optional string server_name = 1;
11:}
```

Listing 24.5 Challenge Option Message

```
1:  message Encoding {
2:      enum EncodingType {
3:        ENCODING_TYPE_UNKNOWN = 0;
4:        ENCODING_TYPE_ALPHANUMERIC = 1;
5:        ENCODING_TYPE_NUMERIC = 2;
6:        ENCODING_TYPE_HEXADECIMAL = 3;
7:        ENCODING_TYPE_QRCODE = 4;
8:      }
9:      required EncodingType type = 1;
10:     required uint32 symbol_length = 2;
11:   }
```