ABSOLUTE BEGINNER’S GUIDE TO

Minecraft® Mods Programming

No experience necessary!

Second Edition
Rogers Cadenhead
Contents at a Glance

Part I  Java from the Ground Up

1  Dig into Minecraft Programming with Java ........................................... 1
2  Use NetBeans for Minecraft Programming ........................................ 13
3  Create a Minecraft Mod .................................................................. 23
4  Start Writing Java Programs .......................................................... 35
5  Understand How Java Programs Work ............................................. 49
6  Store and Change Information in a Mod ........................................... 61
7  Use Strings to Communicate .......................................................... 77
8  Use Conditional Tests to Make Decisions ......................................... 89
9  Repeat an Action with Loops .......................................................... 105
10 Store Information with Arrays ........................................................ 117

Part II  The World of Java Objects

11 Create Your First Object ............................................................... 129
12 Describe What Your Object Is Like ................................................ 145
13 Make the Most of Existing Objects ................................................ 161
14 Store Objects in Data Structures .................................................. 175
15 Handle Errors in a Mod ................................................................. 189
16 Create a Threaded Mod ................................................................. 207
17 Read and Write Files ..................................................................... 225

Part III  Create Killer Minecraft Mods

18 Spawn a Mob ................................................................................ 243
19 Make One Mob Ride Another ........................................................ 259
20 Take a Census of Mobs and Villages .............................................. 269
21 Transmute Materials in an Inventory .............................................. 283
22 Dig a Giant Hole ......................................................................... 295
23 Chop Down a Forest of Trees ......................................................... 309
24 Respond to Events in the Game ..................................................... 323
25 Display a Mob’s Health During Combat ....................................... 337
26 Make a World Change over Time .................................................. 351
27 Befriend the God of Lightning ....................................................... 361
A  Visit This Book’s Website ............................................................ 373
Index .................................................................................................. 375
This page intentionally left blank
Table of Contents

I Java from the Ground Up

1 Dig into Minecraft Programming with Java ........................................ 1
  Setting Up a Minecraft Server .................................................. 2
  Connecting to the Server ....................................................... 8

2 Use NetBeans for Minecraft Programming ................................. 13
  Installing NetBeans ............................................................... 14
  Creating a New Project ......................................................... 14
  Creating a New Java Class ...................................................... 16
  Running the Application ......................................................... 19
  Fixing Errors ................................................................. 20

3 Create a Minecraft Mod .............................................................. 23
  Creating Your First Mod ....................................................... 24

4 Start Writing Java Programs ...................................................... 35
  What You Need to Write Programs ........................................... 35
  Creating the splash Program ................................................ 36
  Beginning the Program ......................................................... 37
  Storing Information in a Variable ........................................... 41
  Saving the Finished Product .................................................. 42
  Compiling the Program into a Class File .................................. 43
  Fixing Errors ................................................................. 44
  Running a Java Program ....................................................... 45

5 Understand How Java Programs Work ..................................... 49
  Creating an Application ....................................................... 50
  Sending Arguments to Applications ........................................ 52
  The Java Class Library ....................................................... 54

6 Store and Change Information in a Mod .................................... 61
  Statements and Expressions .................................................. 62
  Assigning Variable Types ..................................................... 62
  Naming Your Variables ......................................................... 67
  Storing Information in Variables ............................................ 68
  All About Operators ............................................................ 69
  Using Expressions .............................................................. 73
## ABSOLUTE BEGINNER'S GUIDE TO MINECRAFT MODS PROGRAMMING

### 7 Use Strings to Communicate
- Storing Text in Strings
- Displaying Strings in Programs
- Using Special Characters in Strings
- Pasting Strings Together
- Using Other Variables with Strings
- Advanced String Handling
- Presenting Credits

### 8 Use Conditional Tests to Make Decisions
- if Statements
- if-else Statements
- switch Statements
- The Ternary Operator
- Watching the Clock

### 9 Repeat an Action with Loops
- for Loops
- while Loops
- do-while Loops
- Exiting a Loop
- Naming a Loop
- Testing Your Computer Speed

### 10 Store Information with Arrays
- Creating Arrays
- Using Arrays
- Multidimensional Arrays
- Sorting an Array
- Counting Characters in Strings

### II The World of Java Objects

#### 11 Create Your First Object
- How Object-Oriented Programming Works
- Objects in Action
- What Objects Are
- Understanding Inheritance
- Building an Inheritance Hierarchy
- Converting Objects and Simple Variables
- Creating an Object
# TABLE OF CONTENTS

12 Describe What Your Object Is Like ........................................... 145
   Creating Variables .............................................................. 146
   Creating Class Variables .................................................... 148
   Creating Behavior with Methods ......................................... 149
   Putting One Class Inside Another ........................................ 155
   Using the `this` Keyword .................................................... 156
   Using Class Methods and Variables ..................................... 157

13 Make the Most of Existing Objects ........................................ 161
   The Power of Inheritance .................................................. 162
   Establishing Inheritance ................................................... 164
   Working with Existing Objects ......................................... 166
   Storing Objects of the Same Class in Array Lists .................... 166
   Creating a Subclass ......................................................... 170

14 Store Objects in Data Structures ......................................... 175
   Array Lists ............................................................................. 176
   Hash Maps ............................................................................ 182

15 Handle Errors in a Mod ....................................................... 189
   Exceptions ............................................................................ 190
   Throwing and Catching Exceptions ..................................... 201

16 Create a Threaded Mod .......................................................... 207
   Threads ................................................................................. 208
   Working with Threads ......................................................... 213
   The Constructor ..................................................................... 215
   Catching Errors as You Set Up URLs ................................... 216
   Starting the Thread ............................................................. 216
   Handling Mouse Clicks ....................................................... 218
   Displaying Revolving Links ............................................... 219

17 Read and Write Files ............................................................. 225
   Streams .................................................................................. 226
   Writing Data to a Stream ..................................................... 234
   Reading and Writing Configuration Properties .................... 237
### III Create Killer Minecraft Mods

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td><strong>Spawn a Mob</strong></td>
<td>243</td>
</tr>
<tr>
<td></td>
<td>The Mod Framework</td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>Starting a Mod Project</td>
<td>247</td>
</tr>
<tr>
<td></td>
<td>Writing the Mod’s Code</td>
<td>250</td>
</tr>
<tr>
<td>19</td>
<td><strong>Make One Mob Ride Another</strong></td>
<td>259</td>
</tr>
<tr>
<td></td>
<td>Starting the Project</td>
<td>260</td>
</tr>
<tr>
<td></td>
<td>Writing the Mod</td>
<td>261</td>
</tr>
<tr>
<td></td>
<td>Deploying the Mod</td>
<td>267</td>
</tr>
<tr>
<td>20</td>
<td><strong>Take a Census of Mobs and Villages</strong></td>
<td>269</td>
</tr>
<tr>
<td></td>
<td>Starting the Project</td>
<td>270</td>
</tr>
<tr>
<td></td>
<td>Creating the Project</td>
<td>271</td>
</tr>
<tr>
<td>21</td>
<td><strong>Transmute Materials in an Inventory</strong></td>
<td>283</td>
</tr>
<tr>
<td></td>
<td>Starting the Project</td>
<td>284</td>
</tr>
<tr>
<td></td>
<td>Creating the Project</td>
<td>286</td>
</tr>
<tr>
<td>22</td>
<td><strong>Dig a Giant Hole</strong></td>
<td>295</td>
</tr>
<tr>
<td></td>
<td>Starting the Project</td>
<td>296</td>
</tr>
<tr>
<td></td>
<td>Creating the Project</td>
<td>297</td>
</tr>
<tr>
<td>23</td>
<td><strong>Chop Down a Forest of Trees</strong></td>
<td>309</td>
</tr>
<tr>
<td></td>
<td>Starting the Project</td>
<td>310</td>
</tr>
<tr>
<td></td>
<td>Creating the Project</td>
<td>311</td>
</tr>
<tr>
<td>24</td>
<td><strong>Respond to Events in the Game</strong></td>
<td>323</td>
</tr>
<tr>
<td></td>
<td>Starting the Project</td>
<td>324</td>
</tr>
<tr>
<td></td>
<td>Creating the Project</td>
<td>325</td>
</tr>
<tr>
<td>25</td>
<td><strong>Display a Mob’s Health During Combat</strong></td>
<td>337</td>
</tr>
<tr>
<td></td>
<td>Starting the Project</td>
<td>338</td>
</tr>
<tr>
<td></td>
<td>Creating the Project</td>
<td>340</td>
</tr>
<tr>
<td>26</td>
<td><strong>Make a World Change over Time</strong></td>
<td>351</td>
</tr>
<tr>
<td></td>
<td>Starting the Project</td>
<td>352</td>
</tr>
<tr>
<td></td>
<td>Creating the Project</td>
<td>353</td>
</tr>
</tbody>
</table>
# Table of Contents

27 Befriend the God of Lightning ........................................ 361
  Starting the Project ........................................ 362
  Stepping Through Mod Development. ........................ 362
  Creating the Project........................................ 364

A Visit This Book’s Website ........................................ 373

Index ................................................................. 375
About the Author

Rogers Cadenhead is a writer, computer programmer, and web developer who has written more than 20 books on Internet-related topics, including Sams Teach Yourself Java in 24 Hours. He maintains the Drudge Retort and other websites that receive more than 20 million visits a year. This book’s official website is at www.javaminecraft.com.

Dedication

This book is dedicated to the kids out there who have been inspired by Minecraft to learn computer programming, whether they’re 10, 20, or 50. There’s a lot of great experiences ahead of you, not only in writing mods for a video game but in what you do with your skills beyond the game world.

Acknowledgments

To the folks at Pearson, especially Mark Taber, Andy Beaster, Lori Lyons, Boris Minkin, and San Dee Phillips. No author can produce a book like this on his own. Their excellent work will give me plenty to take credit for later.

To my wife, Mary, and my sons, Max, Eli, and Sam.
We Want to Hear from You!

As the reader of this book, you are our most important critic and commentator. We value your opinion and want to know what we’re doing right, what we could do better, what areas you’d like to see us publish in, and any other words of wisdom you’re willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn’t like about this book—as well as what we can do to make our books better.

*Please note that we cannot help you with technical problems related to the topic of this book.*

When you write, please be sure to include this book’s title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: feedback@quepublishing.com

Mail: Que Publishing
ATTN: Reader Feedback
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at quepublishing.com/register for convenient access to any updates, downloads, or errata that might be available for this book.
This page intentionally left blank
IN THIS CHAPTER
• Create a mod as a new project in NetBeans
• Add the Spigot API to the mod
• Write the Java code to implement the mod
• Configure a mod configuration file
• Organize a mod into the proper files and folders
• Build the mod in NetBeans
• Deploy the mod on a server
• Play the mod-ified Minecraft game

CREATE A MINECRAFT MOD

Now that you have a Spigot server for Minecraft set up and running and have installed the NetBeans integrated development environment (IDE), you’re ready to create and deploy a mod.

Mods are special Java programs that run on a Minecraft server. They can’t be run anywhere else.

Writing a mod requires the use of the Spigot API, a set of Java programs that do all of the background work necessary for the program to function inside a Minecraft game. A Java program also is called a class, so the Spigot API is called a class library.

The Spigot class library handles things like determining the (x,y,z) location of any object in the game, including a player. Everything you interact with in the game is represented in Spigot.
Before this book takes a full trip through Java, from the basics of the language into advanced features, this chapter demonstrates how a mod is created. This will give you a chance to see where all this material is headed. Many programming concepts will be unfamiliar to you, but all will be fully explained in subsequent chapters.

Creating Your First Mod

Mods are packaged as Java archive files, also called JAR files. NetBeans, the free integrated development environment from Oracle used throughout this book, automatically creates JAR files every time you build a project.

When you have finished writing a mod, you will be storing it under the server’s folder in a subfolder named plugins.

The mod you are creating is a simple one that demonstrates the framework you’ll use in every mod you create for Spigot. The mod adds a /petwolf command to the game that creates a wolf mob, adds it to the world, and makes you (the player) its owner.

CAUTION There are two very similar terms you encounter a lot when doing Minecraft programming: mod and mob. A mod is a Java program that runs on a server to add something cool to the game. A mob is any living creature in the game, such as a creeper, chicken, cow, cave spider, or catoblepas.

Actually, there are no catoblepases in Minecraft. I was just making sure you were paying attention.

Each mod will be its own project in NetBeans. To begin this project, follow these steps:

2. In the Categories pane, select Java, and in the Projects pane, select Java Application. Then click Next.
3. In the Project Name field, enter PetWolf (with no spaces and capitalized as shown).
4. Click the Browse button next to the Project Location field. The Select Project Location dialog appears.
5. Find the folder where you installed the Minecraft server. Select it and click Open. The folder appears in the Project Location field.
6. Deselect the Create Main Class check box.
7. Click Finish.

The PetWolf project is created, and two folders appear in the Projects pane, Source Packages and Libraries, as shown in Figure 3.1.

![Viewing a project in the Projects pane.](image)

On each mod project, you must add a Java class library before you begin writing Java code: the Spigot server’s JAR file, which includes the Spigot API. Here’s how to do this:

1. In the Projects pane, right-click the Libraries folder and select the menu command Add Library. The Add Library dialog opens.
2. Click the Create button. The Create New Library dialog appears.
3. In the Library Name field, enter Spigot and click OK. The Customize Library dialog opens.
4. Click the Add JAR/Folder button. The Browse JAR/Folder dialog opens.
5. Use the dialog to find and open the folder where you installed the server. You see a file in that folder named spigotserver.
6. Click that file.
7. Click Add JAR/Folder.
8. Click OK.
9. In the Add Library dialog, the Available Libraries pane now has a Spigot item. Select it and click Add Library.

The Projects pane now contains the JAR file for the Spigot API in the Libraries folder, so you’re ready to begin writing the mod program.
Follow these steps to create the program:

2. In the Categories pane, select Java.
3. In the File Types pane, select Empty Java File; then click Next.
4. In the Class Name field, enter `PetWolf`.
5. In the Package field, enter `com.javaminecraft`.
6. Click Finish.

The file `PetWolf.java` opens in the NetBeans source code editor.

Before you begin entering any code, this section explains the basics of how a mod is structured. Don’t type in anything yet.

Every mod you create for Spigot begins with a framework of standard Java code. Here’s the main part of that code, customized in a few places for this project:

```java
public class PetWolf extends JavaPlugin {

    public static final Logger LOG = Logger.getLogger("Minecraft");

    public boolean onCommand(CommandSender sender, Command command, String label, String[] arguments) {

        if (label.equalsIgnoreCase("petwolf")) {
            if (sender instanceof Player) {
                // do something cool here
                LOG.info("[PetWolf] Howl!");
            }
            return true;
        }
        return false;
    }
}
```

Looking at this framework, the only things that will change when you use it for a different mod are the three things that refer to PetWolf because those are specific to this project:
• The name of the program is PetWolf.

• The argument inside label.equalsIgnoreCase("petwolf") is the command the user will type in the game to run the mod. This program implements the command /petwolf (commands in a mod are preceded by a slash (/) character). The label object, which is sent as an argument to onCommand(), is a string that holds the text of a command entered by the user.

• The statement that calls log.info("[PetWolf] Howl!") sends a log message that is displayed in the Minecraft server window.

Everything a mod does when its command is entered goes at the spot marked by the comment // do something cool here. Comments are messages in a program that explain what it does to humans reading the code. They’re ignored by the computer when the program runs.

The first thing the PetWolf mod needs to do is learn more about the game world, using these three statements:

```java
Player me = (Player) sender;
Location spot = me.getLocation();
World world = me.getWorld();
```

A Player object called me is the character controlled by the person playing the game.

With this Player object, you can call its getLocation() method to learn the exact spot where the player is standing. A method is a section of a Java program that performs a task. Here, the method retrieves the player’s current location as a Location object. Three things you can learn about a location are its (x,y,z) coordinates on the three-dimensional game map.

The Player object has a getWorld() method that responds with the World object that represents the entire game world.

Most of the mods you create need these three Player, Location, and World objects.

This mod creates a new mob that’s a wolf, using the spawn() method of the World object:

```java
Wolf wolfie = world.spawn(spot, Wolf.class);
```

There’s a class for every type of mob in the game. The two arguments to the spawn() method are the location where the wolf should be placed and the class of the mob to create.
This statement creates a Wolf object named wolfie at the same spot as the player:

```java
Wolf wolfie = new Wolf();
```

The color of the wolf’s collar is set in this statement:

```java
cat.setCollarColor(DyeColor.PINK);
```

After the wolf has been created, the player becomes its owner by calling the wolf’s `setOwner()` method with the Player object me as the only argument:

```java
wolf.setOwner(me);
```

Now you can begin typing. Put all this together by entering Listing 3.1 into the source code editor and clicking the Save All button in the NetBeans toolbar (or select File, Save).

**LISTING 3.1 The Full Text of PetWolf.java**

```java
package com.javaminecraft;

import java.util.logging.*;
import org.bukkit.*;
import org.bukkit.command.*;
import org.bukkit.entity.*;
import org.bukkit.plugin.java.*;

public class PetWolf extends JavaPlugin {

    public static final Logger LOG = Logger.getLogger("Minecraft");

    public boolean onCommand(CommandSender sender, Command command, String label, String[] arguments) {

        if (label.equalsIgnoreCase("petwolf")) {
            if (sender instanceof Player) {
                // get the player
                Player me = (Player) sender;
                // get the player’s current location
                Location spot = me.getLocation();
                // get the game world
                World world = me.getWorld();
            }
        }
    }
}
```
24:          // spawn one wolf
25:          Wolf wolf = world.spawn(spot, Wolf.class);
26:          // set the color of its collar
27:          wolf.setCollarColor(DyeColor.PINK);
28:          // make the player its owner
29:          wolf.setOwner(me);
30:          LOG.info("[PetWolf] Howl!");
31:          return true;
32:      }
33:    }
34:  return false;
35: }
36: }
37: }

The import statements in Lines 3–7 of Listing 3.1 make five packages available in the program: one from the Java Class Library and four from Spigot. Packages are groups of Java classes that serve a related purpose. For instance, the org.bukkit.entity package referenced in Line 6 is a group of classes for the mobs in the game (which in Spigot are called entities).

NOTE These classes are used in the PetWolf program: Logger from java.util.logging.Logger, Location and World from org.bukkit, Command from org.bukkit.command, Wolf and Player from org.bukkit.entity, and JavaPlugin from org.bukkit.plugin.java.

You learn more about these packages as you use them to create more mods beginning in Chapter 18, “Spawn a Mob.”

The return statements in Lines 32 and 35 are part of the standard mod framework. A method in Java can return a value when its task is completed. Your mods should return the value true inside the onCommand() method when the mod handles a user command and false when it doesn’t.

You have created your first mod, but it can’t be run yet by the Spigot server. It needs a file called plugin.yml that tells the server about the mod.

This file is a YAML file, which you also can create with NetBeans using these steps:

2. In the Categories pane, scroll down and select Other.
In the File Types pane, select YAML File and click Next.

In the File Name field, enter **plugin**. (Don’t put `.yml` on the end; this is done for you by NetBeans.)

In the Folder field, enter `src`.

Click Finish.

A file named `plugin.yml` opens in the source code editor with two lines in it:

```yaml
## YAML Template.
---
```

Delete these lines. They aren’t needed in this file. Enter the text of Listing 3.2 into the file, and be sure to use the same number of spaces in each line. Don’t use tab characters instead of spaces.

**LISTING 3.2**  The Full Text of This Project’s `plugin.yml`

```yaml
1: name: PetWolf
2: 
3: author: Your Name Here
4: 
5: main: com.javaminecraft.PetWolf
6: 
7: commands:
8:     petwolf:
9:         description: Spawn a wolf as the player’s pet.
10: 
11: version: 1.0
```

Replace **Your Name Here** with your own name. Telling people you wrote a mod is the first step toward becoming a legendary Minecraft coder.

To double-check that you have entered the spaces correctly, there are four spaces in Line 8 before the text `petwolf` and eight spaces in Line 9 before `description`.

The `plugin.yml` file describes the mod’s name, author, Java class file, version, command, and a short description of what the command does.

This file must be in the right place in the project. Look in the Projects pane, where it should be inside the Source Packages folder under a subheading called `<default package>`. This is shown in Figure 3.2.
If the plugin.yml file is in the wrong place, such as under the com.javaveminecraft heading, you can use drag and drop to move it to the proper location. Click and hold the file, drag it to the Source Packages folder icon, and drop it there.

You’re now ready to build your mod. Select the menu command Run, Clean and Build Project. If this is successful, the message Finished Building PetWolf (clean, jar) will appear in the lower-left corner of the NetBeans user interface, way down at the bottom edge.

The mod is packaged as a file called PetWolf.jar in a subfolder of the project. To find it, click the Files tab in the Projects pane, expand the PetWolf folder (if necessary), and then expand the dist subfolder. The Files tab lists all the files that make up the project, as shown in Figure 3.3.

This PetWolf.jar file needs to be copied from the project folder to the Minecraft server. Follow these steps:
1. If the Minecraft server is running, stop it by going to the server window and typing the command stop; then press the spacebar or any other key to close that window.

2. Outside of NetBeans, open the folder where you installed the Minecraft server.

3. Open the PetWolf subfolder.

4. Open the dist subfolder.

5. Select the PetWolf file (a JAR file), and press Ctrl+C to copy it.

6. Go back to the Minecraft server folder.

7. Open the plugins subfolder.

8. Press Ctrl+V to copy PetWolf into it.

You have deployed your new mod on the Minecraft server. Start the server the same way you did before—by clicking the start-server.bat file you created. If you look carefully at the messages that display in the server window as the server loads, you see two new messages in the log file that display as it runs:

   [PetWolf] Loading PetWolf v1.0
   [PetWolf] Enabling PetWolf v1.0

These messages do not appear together. One appears close to the top and another close to the bottom.

If you don’t see these messages, but instead see some long, complicated error messages, double-check everything in PetWolf.java against Listing 3.1 and plugin.yml against Listing 3.2 to ensure they were entered correctly; then rebuild and redeploy the mod.

TIP Still having problems making the PetWolf mod work? The source code, plugin.yml configuration file, and all other files for each book project can be found on www.javaminecraft.com. Visit the website to see files that were compiled successfully and run on a server.
After you run the Minecraft client and connect to your server, enter the command `/petwolf`. You now have a new wolf who will follow you around. Enter the command as many times as you like to keep adding wolves.

Figure 3.4 shows me and 20 wolves. Hostile mobs don’t last long against us. We will rule this world. Hooooooooooooooooowl!

**FIGURE 3.4**
Your own wolf pack, courtesy of your own mod.

**THE ABSOLUTE MINIMUM**

This chapter was a sprint through the subject of how to write a Minecraft mod. The rest of the book will be at a much gentler pace that explains all the concepts in Java programming and the Spigot API that are necessary to create sophisticated mods.

There are a lot of different ways to learn Java. Doing it inside Minecraft is one of the most entertaining. You will put your programming abilities to use in a three-dimensional world where you fight monsters, dig tunnels, build houses, and dodge the legions of the living dead—including some who ride chickens.

Minecraft is an excellent way to learn about object-oriented programming, one of the toughest aspects of the Java language to master. Everything you do in Java is accomplished with objects. You create them in constructors, give them knowledge
in variables, and tell them to do things by calling methods. Objects indicate the tasks they can perform by implementing interfaces.

All these concepts are new to you now, but as you learn them in the next 14 chapters, you will find they directly relate to mod programming. Need a zombie? Create a Zombie object. Want the player to move to a new spot? Call his teleport() method. Did you lose your horse? Call its getLocation() method.

So dig as deep into Minecraft mod programming as you like. You’re going to use these skills far beyond the mines.
Symbols

(\) (backslash character), 80
$ (dollar sign), 67
) (backslashes), filenames, 227
[] (brackets), arrays, 118
{} (brackets), 62
{} curly brackets, arrays, 119
-- (decrement operator), 70
/ (division operator), 70
(*) (double quotation marks), 64
// (double slashes), 41
== (equality operator), 91
= (equal sign), 64, 68
// (forward slashes), 227
> (greater than operator), 90
++ (increment operator), 70
!= (inequality operator), 92
< (less than operator), 90
- (minus sign), 70
* (multiplication operator), 70
% operator, 70
+= operator, 82
+ (plus sign), 70
? (question mark), 97-98
(semicolon), 42, 113
' (single quotation marks), 80
[] (square brackets), 286
_ (underscores)
large numbers, 65
variable names, 67

A

access control
defined, 146
methods, public methods, 150
variables, 147
accessor methods, 151
ActionListener interface, 214
actionPerformed() method, mouse
clicks, 218-219
adding Java virtual machine to
Path, 13
addition operator (+), 70
add() method, 163
array lists, 167, 176
addPotionEffect() method, 266
adjacent blocks, retrieving, 329
alert() method, 151
AnimatedLogo class, 164
annotations
@EventHandler, 329
@Override, 327
Apache Project class libraries, 55
applechicken command, 353
applets, defined, 49
applications
arguments, 52-54
Battlepoint, 179-181
Benchmark, 114-115
Calculator, 191-193
Clock, 98-102
ClockTalk, output, 102
Command, 52-53
Commodity, 96-97
Configurator, 239-240
ConfigWriter, 235-237
Console, 223-224
CreditCardChecker, 198
Credits, 85-86
defined, 40, 49
Dice, 56-57
FindPrimes, 210-213
FontMapper, 185-186
Game, 92-93
HomePage, 201-202
ID3Reader, 229-231
LinkRotator. See LinkRotator
application
NameSorter, 124
NewCalculator, 193-194
NewRoot, 138-139
Nines, 107-108
NumberDivider, 195-196
organizing block statements, 92-94
PageCatalog, 203-204
PlanetWeight, 74-76
PointTester, 172-173
Root, 50-51
running, 19
SpaceRemover, 121-122
Spartacus, 18-20
StringLister, 169-170
Variable, 63-64
Virus, 151-153, 157
VirusLab, 158-159
Wheel of Fortune, 125-128
arguments
onCommand() method, 245
passing methods, 151
passing to applications, 52-53
PotionEffect() constructor, 266
storing, 53
Summon command, 54
array index out of bounds errors, 120
ArrayIndexOutOfBoundsException
class, 190
ArrayList class, 167-168
array lists
adding objects, 167
creating, 167, 176
defined, 176
elements, retrieving, 168
generics, 176
looping through, 168
objects, 167-168, 176-178
object storage, 167-168
outside-the-cube trees, 313
point objects, adding, 179-181
points, creating, 178
size, 177
vectors, compared, 177
arrays, 119
arguments, storing, 53
class
declaring, 121
Wheel of Fortune app, 127
declaring, 118
defined, 117
elements, 118-119
exceptions, 120
initial values, 118
integers
declaring, 118
Wheel of Fortune app, 127
multidimensional, 122-123
pageLink, 215
pageTitle, 214-215
sample application, 121-122
searched locations, 315
sorting, 123-124
strings, 118
upper limits, checking, 120
Wheel of Fortune app, 125-128
Arrays class, 123
attributes, 130
defined, 145
inheritance, hierarchy, 133
autoboxing, 139

B
backing up worlds, 304-305
filenames, 227
backspace special character, 80
bats, summoning with coordinates, 54
battle, mob health state, displaying.
See HealthChecker mod
Battlepoint application, 179-181
digging, 298
behavior (objects), 130. See also methods
inheritance, hierarchy, 133
Benchmark application, 114-116
BestFriendOfZeus mod, 362
Bukkit class library, 362
deploying, 369
event listener, registering, 364
events, receiving, 364
lightening bolts, throwing, 365
mob attack target, 364-365
mob found somebody to attack, retrieving, 364
plug-in configuration file, 363
project, creating, 362
source code, 366-368
bigdig command, 297
BigDig mod
block materials, changing, 299
blocks around players, 298
Bukkit class library, 296
circles around players, digging out, 297-299
deploying, 303

hole completion sound/message, 299
hole size, determining, 297
plug-in configuration file, 296
project, creating, 296
source code, 300-303
binary value variables, 66
blank spaces (Java programs), 46-47
BlockFace enumeration, 330
Block objects
getRelative() method, 329
getType() method, 252
blocks, 40-41
adjacent, retrieving, 329
around players, examining, 298
changing ground under players. See StoneWalker mod
destroying, turning into air, 299
materials. See materials right below players, 330
turning into air, 299
block statements, 62
Game application, 92-93
if statements, 92-94
boolean variables, 66-67
Boole, George, 67
brackets ([ ]), 62
Java programs, 40
brackets [ ], arrays, 118
breaking loops, 111
break statement, 111
browsed method, 219
BufferedInputStream()
constructor, 232
buffered input streams, 232-234
building block materials, 299
Bukkit, website, 59
Bukkit class library
adding BigDig mod, 296
adding to projects
MobCensus, 270
Transmuter mod, 285
ZombieChicken, 260
BestFriendOfZeus mod, 362
event handling methods, 328
event packages, 328
HealthChecker mod, 338
JohnnyApplechicken mod, 352
StoneWalker mod, 324
TreeChopper mod, 310

Bukkit Minecraft server, Java virtual machine can't be found error, 14
running first time, 10
Bukkit Spigot class library
adding to projects, 248
events documentation, 348
Javadoc documentation, 370
Bukkit Spigot class library
documentation website, 291
Bukkit Spigot Plugins Resources directory website, 370
bytecode, 226
bytecode interpreters, 226
bytes, 229, 235
byte streams, 226
byte variables, 65

C
C418, 232
CableModem class, 141
Calculator application, 191-193
cancelling event normal behaviors, 354
cannot find symbol error message, 45
capacity, hash maps, 183
carriage returns character, 80
case, changing strings, 83
case sensitivity, 67-68
case statements, switch statements, 96
casting, 134-136
autoboxing, 139
unboxing, 140
catching exceptions, 190-191
handling something after, 197
HomePage app, 201-202
multiple exceptions, 195
PageCatalog app, 203-204
throwing exceptions, 191
try-catch blocks, 191
Calculator app, 191-193
NewCalculator app, 193-196
census
mobs, 270, 276-280
mobs, project creating, 270-274
villagers, 274-276, 281
changing string case, 83
character arrays
declaring, 121
Wheel of Fortune application, 127
characters
char variables, declaring, 64
counting in strings, 125-128
defined, 63, 78
quotation marks, 64
special, strings, 79-80
streams, 226
char variables, declaring, 64
checkAuthor() method, 156
CheckDatabase class, 198
chickens
speed, setting, 266
turning into tree laying mutants. See JohnnyApplechicken mod
ezombies riding. See ZombieChicken mod
ChickenStorm mod
Bukkit Spigot class library, adding, 248
chickens, spawning, 250-253
deploying, 256-257
JAR file, finding, 255
overpopulating, 251
project, creating, 247
source code, 253-255
YAML file, 248-250
chopAdjacentTrees() method, 314-315
chopping down trees. See TreeChopper mod
circles, calculating radius, 297
classes, 130
AnimatedLogo, 164
Applet, methods, 163
ArrayList, 167-168
ArrayIndexOutOfBoundsException, 190
ArrayList, 167-168
Arrays, 123
Bukkit, website, 59
CableModem, 141
CheckDatabase, 198
Console, 232-234
declaring, 39
threads, 214
defined, 23, 55
Double, 297
DslModem, 142
encapsulation, 151
event listeners, 323
Exception, 191
file, 227
FileInputStream, 234
FileOutputStream, 234
FileReader, 234
FileWriter, 234
Font, 185
helper, 155
hierarchy, 162-164
HighSpeedModem, 133
importing, 262-263
inheritance, 133, 162-163
inner, defined, 313
interfaces. See interfaces
ItemSpawnEvent, 354
ItemStack, 283
JApplet
inheritance, 162-163
methods, 163
collections, adding to projects, 248, 260
LinkRotator, 214
Loc, 313
Logger, 246
Material, 283, 286
Java documentation, 291
Math
documentation, 57-58
floor() method, 56
Random() method, 56, 251
sqrt() method, 51
methods, declaring, 153
Modem, 132, 141
ModemTester, 142-143
nesting, 155
NetBeans, 16-19
NetBeans automatic importing, 262-263
packages, defined, 29
PetWolf program, 29
PlayerInventory, 283
PlayerMoveEvent, 328
Point, 171
Point3D, 170-172
PotionEffect, 266
Properties, 237
RuntimeException, 201
subclasses, 134
creating, 164-165, 170-172
superclasses, 134
synchronization, 186
System, 232
testing, 172
Thread, 208-209, 213
threaded, 209-213
variables, 146-147
Vector, 183
Villager, 275
Virus, identifier variable, 146
class libraries, defined, 23
class statement, 39, 132
event listeners, 326
class variables, 148
Clock application, 98-103
ClockTalk application, 102
close() method, 235
streams, 227
closing streams, 227, 235
collections, defined, 170
colors, text output (selecting), 342-343
Command application, 52-53
commandOn variable, 353
commands
applechicken, 353
bigdig, 297
stonewalk, 324
stopstonewalk, 324
Summon, arguments, 54
CommandSender object
verification, 245
comments, 27, 41
Commodity application, 96-97
com object, creating, 132-133
comparing strings 82
equal/not equal, 91
less/greater than, 90-91
compiling
classes, NetBeans, 17
Java programs, 43-44
complex for loops, 113
computer speed, testing. See Benchmark application
concatenating strings, 80-81
concatenation operator (+), 80-81
conditionals, 90
Clock application, 98-102
ClockTalk application, 102
if, 90-94
if-else, 94-95
switch, 95
ternary operator (?), 97-98
configuration settings. See properties
Configurator app, 239-240
ConfigWriter app, 235-237
connections, Minecraft servers
creating, 14-15
fixing, 15-17
Console application, 232-234
Console class, 232-234
console input, 232
constants, 69
constructor methods, 152
CONSTRUCTORS

constructors
BufferedInputStream(), 232
FileInputStream(), 228
FileOutputStream(), 234
Location, 251
PotionEffect(), 266
threads, 215
containsKey() method, hash maps, 184
contains() method, arrays, 168
containsValue() method, 184
continue statement, 111
controlling access. See access control
converting. See also casting
properties to numerical values, 238
variables to objects, 137-138
coordinates (players), finding, 251
counters, 106
counting characters in strings, 125-128
createNewFile() method, 227
Creative Commons license, 231
CreditCardChecker app, 198
credit card purchase verification application, 198
Credits application, 85-86
curly brackets {}, arrays, 119
current location, retrieving, 27
currentThread() method, 217
customizing mobs, adult/child forms, 265
damage() method, 349
data types, 42
binary values, 66
boolean, 66-67
byte, 65
class variables, 148
class variables, 148
characters, 66
boolean, 66-67
char, 64
decrementing variables, 70-72
decrement operator (--), 70
default statements, 96
default variables, 147
decleting() method, files, 228
decleting files, 228
displaying
credits. See Credits application
Java program errors, 44
mob health checker, 340
mob health state during battle. See HealthChecker mod
mods source code on GitHub, 371
properties, 238
reversing links, 219-222
strings
println() method, 78-79
special characters, 79-80
deleting blocks, turning into air, 299
determining string lengths, 83
decreasing values, 70-72
decrement operator (--), 70
default statements, 96
default variables, 147
deleting() method, files, 228
decleting files, 228
displaying
ghosts. See GameUtility class
carpet, 306
monsters, 268
rare creatures, 73
fire, 353
notices, 118
name tag, 328
dialog, 329
emotions, 58
entities. See packages
EntityDamageEvent object, 340
EntityTargetEvent object, 341
EntityTargetEvent.TargetReason
enumeration, 365
ticks
E

effects, potions (adding to mobs), 266-267
elements, 118-119
array lists, retrieving, 168
initial values, 118
else statements, if-else, 94-95
capsulation, 151
endless loops, 110
end-user license agreements (EULAs), 9
ties. See packages
EntityDamageEvent object, 340
EntityTargetEvent object, 341
equality, string comparisons, 91
equalsIgnoreCase() method, 84, 245
equals() method, 82, 163
equality, string comparisons, 91
equivalence, string comparisons, 91

error handling, 195

errors
arrays, 120
defined, 189
exceptions. See exceptions handling in threads, 216
IndexOutOfBoundsException, 201
Java programs, 44-45
MalformedURLException, Java virtual machine can't be found error, 12-14
Spigot server JAR file can't be found error, 11
NetBeans, fixing, 20, 261
NullPointerException, 201
NumberFormatException, 194, 198

escape sequences, 79
EULAs (end-user license agreements), 9
EventHandler annotation (@before), 329
events, 323
documentation, 348-349
explosions, 348
handling, 323, 326
adjacent blocks, 329
block materials, 330
blocks right below players, 330
@EventHandler annotation, 329
graphical user interfaces, 326
methods, 328
mod on status, checking, 329
player locations, 329
registerEvents() method, 327
lightening bolts, throwing, 365
listeners, 323
mods as, registering, 326-327, 364
listening. See listening, events
mob attack target reasons, 365
mob found somebody to attack, retrieving, 364
mob targets, identifying, 364
mods receiving, 364
mouse clicks, 218-219
normal behavior, cancelling, 354
packages, 328
player caused, 328
Exception class, 191
exceptions, 120, 190
array index out of bounds, 120
ArrayIndexOutOfBoundsException, 190
catching, 190-191
multiple, 194-196
multiple exceptions, 195
try-catch blocks, 191-194
catching versus throwing, 191
creating, 200
defined, 189
Exception class, 191
handling something after, 197
HomePage app, 201-202
ignoring, 200
PageCatalog app, 203-204
throwing, 197-199
unchecked, 201
exclamation points (mob health), 342
executeCommand() method, 246, 309
exists() method, 227
exiting loops, 111
explosions, 348
expressions, 73. See also operators advantages, 74
defined, 62
increment/decrement operators, 71
operator precedence, 72-73
PlanetWeight app, 74-76
extends statement, 141, 164

---

File class, 227
FileNotFoundException class, 234
FileInputStream class, 234
FileInputStream() constructor, 228
FileOutputStream class, 234
FileOutputStream() constructor, 234
FileReader class, 234
data,
creating, 227
deleting, 228
existence, checking, 227
File class, 227
FILE
ChickenStorm, 255
ZombieChicken, 267
names, backslashes, 227
properties, 237-240
reading, 228-231
renaming, 228
size, finding, 227
writing to, 234-235
bytes, 235
ConfigWriter application, 235-237
output stream, creating, 234
write() method, 235
YAML, 248-250
BestFriendOfZeus mod, 363
BigDig mod, 296
HealthChecker mod, 339
JohnnyApplechicken mod, 352
MobCensus mod, 270-271
plugin.yml, 249
property names, 249
StoneWalker mod, 324
storing, 250
Transmuter mod, 285-286
TreeChopper mod, 310
ZombieChicken mod, 260-261
FileWriter class, 234
finding
blocks right below players, 330
mods, 370
outside-the-cube trees, 313-315
player coordinates, 251
strings, 84
FindPrimes application, 210-213
fixing errors, NetBeans, 20
floating-point numbers
declaring, 63
rounding down, 56
float statement, 63
float variable, 63
floor() method, 56
folders, 228
Font class, 185
FontMapper app, 185-186
Football application, 92-93
for loops, 106-107
array lists, 168
collection, 113
counter variables, 106
empty sections, 113
exit, 111
Nines application, 107-108
spawning chickens, 353
syntax, 106-107
formfeeds character, 80
for loop, 258
frame, 244-247
Java programs, 44-45
StringWriter application, 235-237
JAR
spigot server JAR file can't be found error, 11
---

---
GAME APPLICATION

G

Game application, 92-93

gameOver variable, 67

Gauntlet, 46

generics, 176

getAmount() method, materials, 288

getBlock() method, 252, 329

getEntity() method

BestFriendOfZeus mod, 364

mob damage, 340

getFrom() method, 329

getHealth() method, 341

getIdentiﬁ er() method, 151

getInventory() method, 287

getItem() method, player inventories, 288

getKey() method, 274

getLivingEntities() method, 272

getLocation() method, 27, 246

getMaterial() method, 287

getMaxHealth() method, 341

getMaxStackSize() method

itm stacks, 288

get() method

array lists, 168

hash maps, 183-184

FontMapper application, 185-186

mobs, 272-274

health bars, 342

HealthChecker mod, 338

Bukkit class library, adding, 338

deploying, 347

display time, 348

health bars, 342-343

mobs taking damage, 340

mobs targeting players for attack, 341

plug-in conﬁ guration ﬁ le, 339

project, creating, 338

source code, 344-347

turning on/off, 340

healthcheck on/healthcheck off commands, 340

health state of mobs during battle. See HealthChecker mod

helper classes, 155

hexadecimal variables, 66

hierarchies, classes, 162-164

HighSpeedModem class, 133

holes, digging. See digging

HomePage application, source code, 201-202

I

ID3Reader application, 229-231

identiﬁ er variable, 146

IDEs (integrated development environments), 13

NetBeans. See NetBeans

if-else statements, 94-95

if statements, 90-94

ignoring exceptions, 200

implements keyword class statement, 326

Runnable interface, 209

importing classes, 262-263

incrementing variables, 70-72

increment operator (++), 70

indexOff() method, array lists, 178

IndexOff() method, 84-85

IndexOutOfBoundsException error, 201

inequality operator (!=), 92

inﬁ nite loops, 110

inheritance, 133, 161-163

constructors, 152

hierarchy, 133

JApplet class, 162-163

initializing counter variables, 106

inner classes, deﬁ ned, 313

input/output. See I/O

installing NetBeans, 14

instance variables, 132

integer arrays, Wheel of Fortune application, 127

integers

arrays, declaring, 118

binary values, 66

byte, 65

declaring, 63

floating-point, 56

hexadecimal values, 66

identiﬁ er variable, 146

long, 65

random, creating, 56

short, 65

underscores, 65

integrated development environments. See IDEs

interfaces, 134

ActionListener, 214

Listener, 326

Runnable, 208-209

int statement, 63

inventories

loading, 287

looping through, 288

materials. See materials

slots, determining, 287

I/O (input/output), 225

console input, 232

ﬁ les. See ﬁ les

streams. See streams

isAdult() method, 275

isStoneWalking variable, 325, 329

ItemSpawnEvent class, 354

ItemSpawnEvent object, 354

ItemStack class, 283

item stacks, 287-288

iteration. See loops

iterators, 107

J

JApplet class, 162-163

JARs (Java archive ﬁ les), 8

ChickenStorm, 255

PetWolf, ﬁ nding, 31

ZombieChicken, 267
Java, Material class documentation, 291
Java Class Library, 54-55
java.io package, 56, 225
javaminecraft.com, 373
Java programs
adding to open projects, 37
blank spaces/whitespace, 46-47
blocks, 40-41
[.] (brackets), 40
category, choosing, 38
class statement, 39
comments, 41
compiling, 43-44
defined, 36
errors, 44-45
main statement, 40
running, 45
saving, 42-43
source code, entering, 38-39
Splash, 36
variables, 41-42
java.time package, 56
java.util package, 56
Properties class, 237
Java virtual machine, adding to Path, 13
JDK, 13
JohnnyApplechicken mod
Bukkit class library, 352
cancelling chickens normal egg laying behavior, 354
deploying, 359
event handler, registering, 354
player location, holding, 353
plug-in configuration file, 352
project, creating, 352
source code, 356-359
spawning chickens, 353
tree laying capabilities, turning on/off, 353
tree sapling supported blocks, 355
JRE, 13
JRE, 13
keys, hash maps, 184
keywords
implements
class statement, 326
Runnable interface, 209
this, 327
locations
block materials, 252
mob/player distances, 275
mobs, 251-252
retrieving, 275
players
holding, 353
listening, 329
searched, storing, 315
Log class, 313
Logger class, 246
logging server messages, 246
LOG variable, 246
long variables, 65
loops, 105
array lists, 168
Benchmark app, 114-115
blocks around players, 298
do-while, 110-111
exiting, 111
for, 106-107
array lists, 168
complex, 113
counter variables, 106
door, 112-113
names, 112-113
nested, Wheel of Fortune app, 128
player inventories, 288
TreeChopper mod, 314
while, 109
lowercase, 83
L
length, strings, 83
length() method, 83
length variable, 123
less-than conditional, 90-91
less than operator (<), 90
letter frequency application. See Wheel of Fortune app
libraries
Bukkit. See Bukkit class library
BukkitSpigot, adding to projects, 248
class, defined, 23
Java Class, 54-55
lightening bolts, throwing, 365
linking strings with variables, 81-82
LinkRotator application
class declaration, 214
constructors, 215
error handling, 216
mouse clicks, handling, 218-219
revolving link, displaying, 219-222
threads
running, 217-218
starting, 216
stopping, 222
variables, setting up, 214-215
LinkRotator class, 214
links, revolving, 219-222
Listener interface, 326
listeners, events, 323
mods as, registering, 326-327, 364
listening, events
mob current/maximum health, determining, 341
mobs taking damage, 340
mobs targeting players for attack, 341
player locations, 329
StoneWalker mod, 326
listFiles() method, 228
list() method, properties, 238
lists, arrays (outside-the-cube trees), 313
literals, defined, 68
load factors, hash maps, 183
loading properties, 238
Location constructor, 251
Location object, 27
getBlock() method, 252
spot variable, 353
Location objects, 130
M
main statement, 40
makeBarGraph() method, 342
MalformedURLException errors, 200
maps, hash. See hash maps
Material class, 283, 286
Java documentation, 291
materials
blocks, 312
changing, 299
tree saplings, planting, 355
blocks into air, changing, 299
names, 286
transmuting. See Transmuter mod
turning to stone eligibility, 330
materials (blocks), 252
Math class
documentation, 57-58
floor() method, 56
random() method, 56
Random() method, 251
sqrt() method, 51
memory
controlling, 159
server lags, 312
messages
colors, selecting, 342-343
digging holes completion, 300
TreeChopper mod, 316
methods
accessor, 151
actionPerformed(), mouse clicks, 218-219
add(), 163
array lists, 167, 176
addPotionEffect(), 266
alert(), 151
arguments, 151
browse(), 219
checkAuthor(), 156
chopAdjacentTrees(), 314-315
classes, declaring, 153
close(), 235
streams, 227
constructors, 152
contains(), array lists, 168
containsKey(), hash maps, 184
containsValue(), 184
createNewFile(), 227
currentThread(), 217
damage(), 349
declaring, 149
defined, 27, 83, 145, 149
delete(), files, 228
displaySpeed(), 132-133
damage(), 274
hash maps, 184
equals(), 82, 163
equalsIgnoreCase(), 245
event handling, 328
events, learning, 348-349
executeCommand(), 246
events(), 227
floor(), 56
get()
array lists, 168
hash maps, 184
getAmount(), materials, 288
getBlock(), 252, 329
getBytes(), 235
getEntity()
 BestFriendOfZeus mod, 364
 mob damage, 340
getFrom(), 329
getHealth(), 341
getInventory(), 287
getItem(), 288
ggetKey(), 274
getLivingEntities(), 272
getLocation(), 27, 246
getMaterial(), 287
getMaxHealth(), 341
getMaxStackSize(), 288
getName(), 227
getOrDefault(), 184
getPlayer(), 329
getPluginManager(), 327
getProperty(), 238
getReason(), 365
getRelative(), 329
getSeconds(), 151
ggetSize(), 287
getTarget(), 364
getTo(), 329
ggetType(), 252, 273
getURL(), 216
ggetValue(), 274
ggetVirusCount(), 158
ggetWorld(), 27, 246
indexOf(), array lists, 178
IndexOf(), 84-85
isAdult(), 275
length(), 83
list(), properties, 238
listFiles(), 228
makeBarGraph(), 342
mod on status, checking, 329
okToTransform(), 330
onCommand(), 271
onEnable(), 364
onEntityDamage(), 340
onEntityTarget(), 341
onItemSpawn(), 354
onPlayerMove(), 328
overriding, 163-164, 327
paint(), overriding, 164
parseDouble(), 297
parseltnt(), 138, 153
playSound(), 299
println(), 75, 149
public, 150
put(), hash maps, 183
random(), 56, 251
read(), input streams, 229
readLine(), 234
recursion, 315
registerEvents(), 327
renameTo(), 228
return values, 150
run(), 209-210
sendMessage(), 273
setAmount(), materials, 288
setBaby(), 265
setBackgroundColor(), 163
setCustomName(), 342
setIdentifier(), 151
setLayout(), 163
setPassenger, 265
setProperty(), 238
setType(), materials, 288
setYield(), 348
shoot(), 181
showHealth(), 341
showVirusCount(), 153
signatures, 151
size() array lists, 177
hash maps, 184
skip(), input streams, 229
sleep(), 208-209
sort(), Arrays class, 123
spawn(), 27, 253
sqrt(), 51
start(), runner threads, 216
stop(), 213
store(), 239
strikeLightning(), 365
substring(), 231
toCharArray(), 121
toLowerCase(), 83
toUpperCase(), 83
toUpperCase(), 83
toUpperCase(), 83
variable scope, 153
write(), 235
Minecraft EULA website, 10
Minecraft Mods Programming
 Absolute Beginner’s Guide
website, 373
Minecraft project, 37
Minecraft servers, 8
minecraf connections, 14-17
EULA check, 9
Java virtual machine can’t be found
error, 12-14
messages, logging, 246
player profiles, changing, 17
running first time, 10
Spigot API and server downloads, 8
Spigot server JAR file can’t be found error, 11
starting, 9
Minecraft website, 15
minus sign (-), 70
MobCensus mod, 270
Bukkit class library, 270
commands, executing, 271
current mobs, listing, 272
deploying, 279-280
player messages, sending, 273
plug-in configuration file, 270-271
project, creating, 270
source code, 276-279
type counts, storing, 272-274
villagers, 281
creating, 274-276
mobs, 269
census, 270
Bukkit class library, 270
commands, executing, 271
current mobs, listing, 272
deploying, 279-280
player messages, 273
plug-in configuration file, 270-271
project, creating, 270
source code, 276-279
type counts, storing, 272-274
current health, calculating, 342
current/maximum health, 341
customing adults/child forms, 265
defined, 24
found somebody to attack, retrieving, 364
health state during battle, displaying. See HealthChecker mod
interacting with other mobs, riding other mobs. See ZombieChicken mod
lightening bolts, throwing at, 365
locations, 251-252
locations, retrieving, 275
numbers, 250
overpopulating, 251
passengers, adding, 265
player distance from, calculating, 275
potions, adding, 266-267
taking damage, 340
targeting players for attack, 341
targets, 364-365
types, 272
wolf, creating, 27
classes, 29
deploying, 32-33
full text, 28-30
JAR file, finding, 31
troubleshooting, 32
mobs hash map, 272-274
Modem class, 132, 141
Modem objects, 131
modems, 141-143
ModemTester class, 142-143
mods as event listeners, registering, 326-327, 364
BestFriendOfZeus. See BestFriendOfZeus mod
BigDig. See BigDig mod
ChickenStorm. See ChickenStorm mod
defined, 24
finding, 370
framework, 244-247
HealthChecker. See HealthChecker mod
JohnnyApplechicken. See JohnnyApplechicken mod
MobCensus. See MobCensus mod
on status, checking, 329
player information, 27
program, creating, 26-27
projects, 24-25
source code, displaying on GitHub, 371
StoneWalker. See StoneWalker mod
storing, 24
Transmutor. See Transmutor mod
TreeAssist, 370
TreeChopper. See TreeChopper mod
turning on/off, 325
wolf. See wolf
ZombieChicken. See ZombieChicken mod
modulo operator (%), 70
Monitor objects, 131
mouse clicks, handling, 218-219
MP3 files, reading, 229-231
multidimensional arrays, 122
multiplication operator (*), 70
mutitasking, 207
mutithreading, 208
names
event handling methods, 328
files, backslashes, 227
loops, 112-113
materials, 286
packages, 56
properties, 237
property, YAML files, 249
variables, 67-68
worlds, 305
NameSorter application, 124
nested loops, Wheel of Fortune, 128
nesting classes, 155
NetBeans automatic class importing, 262-263
classes, 16-19
errors, fixing, 20, 261
installing, 14
overview, 13
projects, creating, 15-16
source editor, displaying errors, 44
user interface, 14
website, 21
NewCalculator application, 193-194
New Empty Java File dialog, 38
New File Wizard, 37
newline characters, 80
New Project dialog, 37
New Project Wizard, NetBeans, 15
NewRoot application, 138-139
new statement, 118, 152
Nines application, 107-108
NullPointerException error, 201
NumberFormatException errors, 194, 198
numbers floating-point, declaring, 63
integers, declaring, 63
prime sequence, displaying, 210-213

O
object-oriented programming. See OOP
objects. See also classes
advantages, 131
array lists, 167-168, 176-178
attributes, 130
defined, 145
behavior, 130
Block
getRelative() method, 329
type, 252
casting, 136
classes, 130
creating to variables, 137-138
debugging, 131
Desktop, 218
EntityDamageEvent, 340
EntityTargetEvent, 341
hash maps, 183-184
inheritance, 133, 161-163
instance variables, 132
ItemSpawnEvent, 354
Location, 27, 130
getBlock(), 252
spot variable, 353
Modem, 131
Monitor, 131
Player, 27
creating, 246
PlayerInventory, 287
Plugin Manager, 327
point, creating, 179-181
refererencing this statement, 156
Server, creating, 327
sharing, 166
storing array lists, 167-168
thisThread, 217
unboxing, 140
variables, 146-147
Wolf, 28
World, 27
accessing, 246
okToTransform() method, 330
onCommand() method, 271, 309
arguments, 245
StoneWalker mod, 326
onEnable() method, 364
onEntityDamage() method, 340
onEntityTarget() method, 341
onItemSpawn() method, 354
onPlayerMove() method, 328
OOP (object-oriented programming), 129-130
advantages, 131
debugging, 131
encapsulation, 151
inheritance, 133, 161-163
objects. See objects
opening Minecraft project, 37
operators
+=, 82
addition (+), 70
concatenation (+), 80-81
decrement (−), 70
division (/), 70
equality (==), 91
greater than (>), 90
increment (++) and (+=), 70
inequality (!=), 92
less than (<), 90
modulo (%), 70
multiplication (*), 70
precedence, 72-73
prefixing/postfixing, 71-72
subtraction (-), 70
ternary (?), 97-98
order of precedence, operator, 72-73
organizing applications, block statements, 92-94
output, colors (selecting), 342-343
output. See I/O (input/output)
outside-the-cube trees, 313
finding, 313-315
overpopulating with mobs, 251
overriding methods, 163-164, 327

P
packages, 147
defined, 29, 55
events, 328
java.io, 56, 225
java.time, 56
java.util, 56
names, 56
PageCatalog application, 203-204
PageLink array, 215
pageTitle array, 214-215
paint() method, overriding, 164
parseDouble() method, 297
parseInt() method, 138, 153
passengers (mobs), adding, 265
passing arguments, methods, 151
arguments to applications, 52-53
pasting strings, 80-81
Path, Java virtual machine

(adding to), 13
percent sign (%), 70
performance, server lag, 312
PetWolf mod
classes, 29
deploying, 32-33
full text, 28-29
JAR file, finding, 31
player information, 27
plugin, creating, 29-30
program, creating, 26-27
project, 24-25
troubleshooting, 32
Wolf object, 27
phonebook hash map, 183-184
PlanetWeight application, 74-76
planting tree saplings 355
player information, 27
PlayerListInventory class, 283
PlayerListInventory object, 287
PlayerMoveEvent class, 328
Player object, 27
creating, 246
player profiles, changing, 17
players
40 block cube around, 311
blocks right below, 330
coordinates, finding, 251
events, causing, 328
ground under, changing. See StoneWalker mod
inventories
block materials, 299, 312
loading, 287
looping through, 288
material names, 286
material transmutations. See Transmuter mod
slots, determining, 287
locations
holding, 353
listening, 329
retrieving, 246
mob distance from, calculating, 275
targeting for attack, 341
world, accessing, 246
playing sounds, digging holes
completion, 299
playSound() method, 299
Plugin Manager object, 327
plugins, PetWolf mod, 29-30
plugin.yml
BestFriendOfZeus mod, 363
BigDig mod, 296
HealthChecker, 339
JohnnyApplechicken mod, 352
MobCensus mod, 270-271
SLEEP() METHOD

StoneWalker mod, 324
Transmuter mod, 285-286
TreeChopper mod, 310
plugin.yml file
storing, 250
ZombieChicken mod, 260-261
plugin.yml file full text listing,
249-250
plus sign (+), 70
Point3D class, 170-172
Point class, 171
point objects, creating, 179-181
points, creating, 178
PointTester application, 172-173
postfixing, 71-72
PotionEffect class, 266
PotionEffect() constructor, 266
potions, 266-267
precedence, operators, 72-73
prefixing, 71-72
prime numbers sequence, displaying,
210-213
println(), 17
printing strings, 78-80
println() method, 75, 149
strings, displaying, 78-79
private variables, 147
programs
comments, defined, 27
creating, 26-27
Java. See Java
projects
BestFriendOfZeus mod, 362
BigDig, 296
Bukkit class library, 260
Bukkit Spigot class library,
adding, 248
ChickenStorm, creating, 247
creating, 24
HealthChecker mod, 338
JohnnyApplechicken, 352
Minecraft, 37
MobCensus, 270
NetBeans, creating, 15-16
StoneWalker, 324
Transmuter mod, 285
TreeChopper, 310
viewing, 25
ZombieChicken, 260
properties, 237-238
Configurator app, 239-240
Properties class, 237
protected variables, 147
public methods, 150

public statements, 132
public variables, 147
put() method, hash maps, 183
Pythagorean theorem, 275

question mark (?), 97-98
quotations marks
double ("), 64
escape sequences, 79
escape codes, 80
single ('), 64

radius, calculating, 297
random() method, 56, 251
random numbers, creating, 56
reading
buffered input streams, 232
files, 228-231
readLine() method, 234
read() method, 229
recursion, 315
redwoods, 312
referencing objects, this statement, 156
registerEvents() method, 327
registering mods as event listeners,
326-327, 364
removing
objects from array lists, 168
objects from arrays, 177
renameTo() method, 228
renaming files, 228
restricting access, 147
retrieving
array list elements, 168
objects, hash maps, 184
properties, 238
return values (methods), 150
revolving links, displaying, 219-222
Rock Proper, 231
Root application, 50-51
Rosenfeld, Daniel, 231
run() method, 209-210
threads, 217-218
Runnable interface, 208-209
running
applications, 19
Java programs, 45
threads, 217-218
RuntimeException class, 201

saving
Java programs, 42-43
properties, 238
worlds, 304-305
scale, health bars, 342
scope (variables), 153
seed numbers, 305
semi-colon, 42, 113
sendMessage() method, 273
server lag, 312
server messages, TreeChopper
mod, 316
Server objects, creating, 327
setAmount() method, 288
setBaby() method, 265
setBackground() method, 163
setCustomName() method, 342
setIdentifier() method, 151
setLayout() method, 163
setPassenger() method, 265
setProperty() method, 238
setting up Minecraft servers
EULA check, 9
Java virtual machine can't be
found error, 12-14
running first time, 10
Spigot API and server,
downloads, 8
Spigot server JAR file can't be
found error, 11
starting, 9
setType() method, 288
setYield() method, 348
sharing objects, 166
shoot(), 182
shooting targets, creating, 179-181
shoot() method, 181
short variables, 65
showHealth() method, 341
showVirusCount() method, 153
signatures, methods, 151
single quotation marks ('), 64
escape code, 80
size
array lists, 177
hash maps, 184
holes, determining, 297
size() method
array lists, 177
hash maps, 184
skip() method, 229
sleep() method, 208-209
SLOWING DOWN APPLICATIONS

slowing down applications, 208-209
sorting arrays, 123-124
sort() method, Arrays class, 123
sounds, playing (digging holes completion), 299
source code editors, 36
source editor (NetBeans), 44
sources (casting), 135
SpaceRemover app, 121-122
spacing (Java programs), 46-47
Spartacus application, 18-20
spawning chickens, 353
spawning mobs, 250-252
spawn() method, 27, 253
special characters, strings, 79-80
speed, chickens, 266
Spigot class library, events documentation, 349
Spigot Minecraft server connections, 14-17
downloading, 8
EULA check, 9
Java virtual machine can't be found error, 12
player profiles, changing, 17
Spigot server JAR file can't be found error, 11
starting, 9
Spigot Project website, 9
splash messages website, 46
Splash program adding to open projects, 37
blank spaces/whitespace, 46-47
blocks, 40-41
{} (brackets), 40
category, choosing, 38
class statement, 39
comments, 41
compiling, 43-44
creating, 36-37
errors, 45
line-by-line breakdown, 43
main statement, 40
running, 45
saving, 42-43
source code, 38-39
variables, 41-42
spot variable, 353
sqrt() method, 51
square brackets ([[]), 286
starting
Minecraft servers, 9
threads, 209-213, 216
starting values, variables, 68-69
start() method, 216
statements
blocks, 40-41, 62
break, 111
switch statements, 96
case, switch statements, 96
class, 39, 132
event listeners, 326
conditionals. See conditionals continue, 111
default, switch statements, 96
defined, 62
expressions, 73
advantages, 74
defined, 62
PlanetWeight application, 74-76
extends, 141, 164
float, 63
if, 90-94
if-else, 94-95
int, 63
literals, 68
loops, Benchmark application, 114-115
defined, 105
do-while, 110-111
for, 111
See for loops
infinite, 110
names, 112-113
while, 109
main, 40
new, 118, 152
operators. See operators
public, 132
quotation marks, 64
static, 148, 153
super
class declarations, 165
Point3D class, 172
switch, 95
switch-case, 96
this, 156
class declarations, 165
Point3D class, 172
throw, 198
try-catch, 216
try-catch blocks
exceptions, catching, 191-194
multiple exceptions, catching, 194-196
try-catch-finally blocks, 197
variables, creating, 62-63
void, 150
static statement, 148, 153
static variables. See class variables
stonewalk command, 324
StoneWalker mod
block material eligibility for stone, checking, 330
block right below player, retrieving, 330
Bukkit class library, 324
deploying, 334
events, 326-327
on status, checking, 329
player location methods, 329
plug-in configuration file, 324
project, creating, 324
source code, 330-333
turning blocks to stone, 330
stop() method, 213
stopping threads, 222
stopstonewalk command, 324
storage, variables. See variables
store() method, 239
storing
arguments, 53
mob type counts, 272-274
mods, 24
objects, array lists, 167-168
properties, 238
searched locations, 315
values in variables, 68-69
YAML files, 250
streams, 225
buffered input, 232-234
bytes, 226
caracter, 226
closing, 227, 235
defined, 226
reading data from, 228-231
types, 226
writing to, 234-237
strikeLightning() method, 365
String data type, 42
StringLister application, 169-170
strings
adding to, 81
arrays, 118
case, changing, 83
class, 78
counting, 125-128
comparing, 82
equal/not equal, 91
less/greater than, 90-91
concatenating, 80-81
Credits application, 85-86
defined, 63, 77
displaying
println() method, 78-79
special characters, 79-80
finding, 84
length, determining, 83
linking with variables, 81-82
variables, declaring, 64, 78
subclasses, 134
creating, 170
extends statement, 164
Point3D class, 171-172
super statement, 165
this statement, 165
substring() method, 231
subtraction operator (-), 70
Summon command, 54
summoning
bats, 54
zombie pigmans, 54
superclasses, 134
super statement
class declarations, 165
Point3D class, 172
switch-case statements, 96
switch statements, 95
synchronization, 186
System class, 232
threads, 208
class declarations, 214
constructors, 215
creating, 209-213
error handling, 216
mouse clicks, 218-219
reversing links, 219-222
Runnable interface, 208
running, 217-218
slowing down applications, 208-209
starting, 216
stopping, 222
Thread class, 208
variables, setting up, 214-215
throwing exceptions, 197-199
catching exceptions, 191
HomePage application, 201-202
PageCatalog application, 203-204
throwing lightening bolts, 365
throw statements, 216
this keyword, 327
timekeeping, 98-102
ticks, 267
time, 267
toCharArray() method, 121
toLowerCase() method, 83
tops variable, 63
toUpperCase() method, 83
transforming block materials,
StoneWalker mod, 330
Transmuter mod, 284
Bukkit class library, adding, 285
deploying, 291
input/output materials, defining,
286-287
player inventory, looping
trough, 288
player inventory, loading, 287
player inventory slots, 287
plug-in configuration file, 285-286
project, creating, 285
source code, 288-291
TreeAssist mod, 370
TreeChopper mod, 310
40 block cube around players, 311
block material, 312
Bukkit class library, 310
deploying, 320-321
outside-the-cube trees, 313-315
plug-in configuration file, 310
project, creating, 310
server messsage, 316
source code, 317-320
trees
chopping down. See
TreeChopper mod
creating with mutated
chickens. See
JohnnyApplechicken mod
outside-the-cube, 313
finding, 313-315
redwoods, 312
saplings, planting, 355
website, 313
troubleshooting
Minecraft servers
connection problems, 15-17
Java virtual machine can't be
found error, 12-14
Spigot server JAR file can't be
found error, 11
PetWolf mod, 32
try-catch blocks
exceptions, catching, 191
Calculator app, 191-193
multiple classes, 194-196
NewCalculator app, 193-194
try-catch-finally blocks, 197
try-catch statements, 216
turning blocks into air, 299
turning on/off, mods, 325
types, mobs, 272
unboxing, 140
unchecked exceptions, 201
underscores (_)
large numbers, 65
variable names, 67
uppercase, 83
upper limits (arrays), 120
URIStreamException errors, 216
URIs, MalformedURLException
errors, 200
user interfaces, NetBeans, 14
values
class variables, changing, 148
hash maps, presence, 184
return, methods, 150
starting, 68-69
variables
incrementing/decrementing,
70-72
storing, 68-69
Variable application, 63-64
variables, 41
access control, 146-147
arrays. See arrays
casting, 135-136
class, 148
commandOn, 353
constants, 69
contents, displaying, 42
converting to objects, 137-138
counter, 106
creating, 62-63
data types, 42
declaring, 146
default, 147
defined, 61
display, 340
= (equal signs), 64
identifier, 146
instance, 132
isStoneWalking, 325, 329
length, 123
arrays, 120
LinkRotator application, 214-215
LOG, 246
names, 67-68
private, 147
protected, 147
quotation marks, 64
referring to this statement, 156
scope, 153
spot, 353
strings. See strings
types
binary values, 66
boolean, 66-67
byte, 65
char, 64, 78
floating-point, 63
hexadecimal values, 66
integers, 63
long, 65
short, 65
string, 64
underscores, 65
values, 68-72
warning, 154
Vector class, 183
vectors, 177
Villager class, 275
villagers, census
creating, 274-276
deploying, 281
Virus application
class constructor, 152
methods, 151
showVirusCount() method, 153
source code, 157
Virus class, identifier variable, 146
VirusLab application, 158-159
void statement, 150
Volume Alpha (Rosenfeld), 232
Volume Beta (Rosenfeld), 232

W-X
warning variable, 154
Web-Adventures website, 234
websites
Apache Project, 55
Bukkit, 59
Bukkit Spigot class library, 291
Bukkit Spigot class library Javadoc documentation, 370
Bukkit Spigot Plugins Resources directory, 370
Creative Commons license, 231
events documentation, 348
Java Class Library documentation, 55
Minecraft, 15
Minecraft EULA, 10
Minecraft Mods Programming Absolute Beginner’s Guide, 373
Minecraft splash messages, 46
NetBeans, 21
Rock Proper, 231
Spigot API and server downloads, 8
Spigot Project, 9
TreeAssist mod directory page, 370
trees, 313
Web-Adventures, 234
Wheel of Fortune application, 125-128
while loops, 109
writing
files, 234-235
Confi gWriter application, 235-237
streams, 234-235
Confi gWriter application, 235-237

Y
YAML (Yet Another Markup Language) fi les, 248-250
BestFriendOfZeus mod, 363
BigDig mod, 296
HealthChecker mod, 339
JohnnyApplechicken mod, 352
MobCensus mod, 270-271
plugin.yml, 249
property names, 249
StoneWalker mod, 324
storing, 250
Transmutor mod, 285-286
TreeChopper mod, 310
ZombieChicken mod, 260-261

Z
ZombieChicken mod, 260
Bukkit class library, adding, 260
chicken speed, setting, 266
deploying, 267
full source code, 263-266
NetBeans automatic class importing, 262-263
plug-in configuration file, creating, 260-261
project, creating, 260
zombie flesh, transmuting into diamonds, 284
zombie pigmans, summoning with coordinates, 54
zombies, adult/child forms, 265
zombies mounted on chickens. See ZombieChicken mod