Programming in Objective-C

International Standard Book Number: 0672325861

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the CD or programs accompanying it.

When reviewing corrections, always check the print number of your book. Corrections are made to printed books with each subsequent printing. To determine the print number of your book, view the copyright page. The print number is the right-most number on the line below the “First Printing” line. For example, the following indicates that this is the 1st printing of this title.

First Printing: November 2003

06 05 04 03   4 3 2 1

First Printing Corrections

<table>
<tr><th>Pg</th><th>Error</th><th>Correction</th></tr>
<tr><td>iv</td><td>First Printing: November 2003

06 05 04 03   4 3 2 1</td><td>First Printing: November 2003

06 05 04   4 3 2</td></tr>
<tr><td>26</td><td>Instances and Methods, p1, line 8:

...but as each car is by is respective owner...</td><td>

...but as each car is **used** by is respective owner...</td></tr>
<tr><td>40</td><td>halfway down the page:

{
   denominator = d;
}

@end</td><td>(delete blank line before @end)

{
   denominator = d;
}

@end</td></tr>
<tr><td>42</td><td>after first two lines of code:

And here are the definitions:</td><td>(should be mono)

And here are the definitions:</td></tr>
<tr><td>45</td><td>Exercises, 1., table
<table><tr><td>Int</td><td>playNextSong</td><td>6_05</td></tr><tr><td>_calloc</td><td>Xx</td><td>alphaBeta Routine</td></tr><tr><td>clearScreen_1312</td><td>z</td><td></td></tr><tr><td>ReInitialize_</td><td>A$</td><td></td></tr></table></td><td>Reformat columns:
<table><tr><td>Int</td><td>playNextSong</td><td>6_05</td></tr><tr><td>_calloc</td><td>Xx</td><td>alphaBetaR outine</td></tr><tr><td>clearScreen</td><td>_1312</td><td>z</td></tr><tr><td>ReInitialize</td><td>_</td><td>A$</td></tr></table></td></tr>
</table>

92	Exercises, 3., line of code 5! = 5 x 4 x 3 x 2 x 1 = 1	Exercises, 3., line of code 5! = 5 x 4 x 3 x 2 x 1 = **120**
120	Program c.10 Output 3 5 7 11 13 17 19 23 31 37 41 43 47	Program c.10 Output **2** 3 5 7 11 13 17 19 23 31 37 41 43 47
141	Program 7.5 Output 1/4 + 1/2 = 3/4 3/4 1/4 + 1/2 = 1/8 = 7/8	(delete third line of code) 1/4 + 1/2 = 3/4 3/4
145	Exercises, 6., final line of code ...(Complex*) complexNum);	Exercises, 6., final line of code ...(Complex * complexNum);
149	Figure 8.3 Add arrows to figure	
152	paragraph 1, line 4 In fact, let's go back to exercise ~~9~~ from Chapter 4...	In fact, let's go back to exercise **7** from Chapter 4...
	middle of page. single line of code: -(void) setWidth: (int) w and Height: (int) h;	-(void) setWidth: (int) w andHeight: (int) h;
	second to last line of code: middle of page. single line of code: -(void) setWidth: (int) w and Height: (int) h	-(void) setWidth: (int) w andHeight: (int) h

154	Paragraph 3, last sentence: The interface and implementation files for your new Square class are shown in Program~~s~~ 8.3~~ and 8.4~~.	The interface and implementation files for your new Square class are shown in Program 8.3.
	Program 8.~~4~~ Square.m Implementation File	Program 8.**3** Square.m Implementation File
155	Paragraph 3, last line: ~~8.5~~ shows the test program and output...	8.**3, "Test Program"** shows the test program and output...
	Program 8.~~5~~ Test Program test2.m	Program 8.**3** Test Program test2.m
	Program 8.~~5~~ Output	Program 8.**3** Output
158	Program 8.~~6~~ Point.m Implementation File	Program 8.**4** Point.m Implementation File
	Program 8.4, first line of code #~~include~~ "Point.h"	#**import** "Point.h"
159	Program 8.~~6~~ Continued	Program 8.**4** Continued
	Program 8.~~7~~ shows the new...	Program 8.**4, "Added Methods,"** shows the new...
	Program 8.~~7~~ Rectangle.m Added Methods	Program 8.**4** Rectangle.m Added Methods
	Program 8.~~7~~ Test Program	Program 8.**4** Test Program
160	Program 8.~~7~~ Continued	Program 8.**4** Continued
	Program 8.~~7~~ Output	Program 8.**4** Output
161	Top of page: Can you explain the output from Program 8.~~8~~?	Can you explain the output from Program 8.**5**?
	Program 8.~~8~~	Program 8.**5**

	Program 8.~~8~~ Output	Program 8.**5** Output
163	Last paragraph: With your modified method, recompiling and rerunning Program 8.~~7~~ produces the following warning messages shown as Program 8.~~9~~.	With your modified method, recompiling and rerunning Program 8.**5** produces the following warning messages shown as Program 8.**5A**.
164	Program 8.~~9~~ Compiler Warning Messages	Program 8.**5A** Compiler Warning Messages
	Program 8.~~9~~ Output	Program 8.**5B** Output
165	First full paragraph: Program 8.~~10~~ shows a simple example to illustrate this concept.	Program 8.**6** shows a simple example to illustrate this concept.
	Program 8.~~10~~	Program 8.**6**
166	Program 8.~~10~~ Continued	Program 8.**6** Continued
	Program 8.~~10~~ Output	Program 8.**6** Output
	Program 8.~~11~~	Program 8.**7**
167	Program 8.~~11~~ Continued	Program 8.**7** Continued
168	Halfway down the page: Now, let's try compiling and running this program again ~~(see Program 8.12)~~.	Now, let's try compiling and running this program again.
	Program 8.~~12~~ Output	Program 8.**7**
169	Paragraph 2, final sentence: The approach used in Program 8.~~9~~ was to have main release that memory with a statement such as follows:	The approach used in Program 8.**6** was to have main release that memory with a statement such as follows:

170	Paragraph 2, final sentence: The two free messages shown in Program 8.~~8~~...	The two free messages shown in Program 8.**5**...
171	Sentence before program listing: Let's put this together in a simple example to illustrate this concept (see Program 8.~~13~~).	Let's put this together in a simple example to illustrate this concept (see Program 8.**8**).
	Program 8.~~13~~	Program 8.**8**
172	Program 8.~~13~~ Continued	Program 8.**8** Continued
	Program 8.~~13~~ Output	Program 8.**8** Output

180	near end of Program 9.2
	[c1 free];
	[f1 free];
	~~[dataValue free];~~
		[c1 free];
		[f1 free];

183	Final paragraph on page, last sentence[remove mono] If frac1 and...	If frac1 and...

198	Code listing, first indented set of lines:
	extern int gCounter;
	++~~c~~ounter;
		extern int gCounter;
		++**gC**ounter;

202	Program 10.3, second to last line of code on this page: [change UC to lc] enum month aMonth;	enum month a**m**onth;
205	Second to last line of code on this page: [change UC to lc] If (userName compare: savedName] ...	**if** (userName compare: savedName] ...

212	Last block of code before paragraph two: -(Fraction *) add: (Fraction *);	-(Fraction *) add: (Fraction *) **f**;
225	Three line block of code in exercise 3: -~~(void) sin: ~~(double) ~~angle~~; -~~(void) cos: ~~(double) ~~angle~~; -~~(void) tan: ~~(double) ~~angle~~;	-(double) **sin**; -(double) **cos**; -(double) **tan**;

	Block of code in exercise 5:
	{
	  Rectangle *rect;
	}
	-(~~int~~) initWithSide: (int) s;
	-(int) setSide: (int) s;
		{
		  Rectangle *rect;
		}
		-(**square** *) initWithSide: (int) s;
		-(**void**) setSide: (int) s;

229	Last C1 line on the page: return TWO_PI * r ;	return TWO_PI * r**adius** ;
233	Second C1 line on the page: #define MakeFract (x,y) ([Fraction alloc] initWith: x over: y]]	#define MakeFract (x,y) (**[**[Fraction alloc] initWith: x over: y]]
	Paragraph six, sentence two: Without the paren~~e~~theses in the macro...	Without the parentheses in the macro...
237	Program 12.1, line two: [bold text] Enter the number of liters: 55.75	Enter the number of liters: **55.75**
248	Third row after Program 13.1: Fibonacci numbers F_{i-2} and $F_{\cancel{i}-1}$	Fibonacci numbers F_{i-2} and $F_{\mathbf{i}-1}$
278	Fourth to last line of code in Program 13.10: printf ("Today's date is %i/%i/~~.2%~~i: \n",	printf ("Today's date is %i/%i/**%.2**i: \n",

293	Second paragraph, last three sentences: Finally, a union variable can be initialized ~~as follows. If it's a global, static, or automatic union variable, the first member of the union can be initialized toa constant expression. In such a case, the constant expression is listed inside a pair of braces,~~ like so:	Finally, a union variable can be initialized like so:
	Next paragraph: This sets the first member of x, which is c, to the character #.	This sets the first member of x, which is c, to the character #. **A particular member can also be initialized by name, like this:** **union mixed x={.f=123.4;};**
302	Exercise 3, sentence four: Have the program find all prime numbers up to 150.	Have the program find all prime numbers up to **n=**150.
	Exercise 3, step 5: [insert "is not equal to" sign where equals sign is indicated] ...such that ixj<n, set P_{ixj} to 1.	...such that ixj<**=**n, set P_{ixj} to 1.
308	Paragraph 2, second to last sentence: ...(at /Development/Documentation/Cocoa/CocoaTopics.html).	...(at /Development/Documentation/Cocoa/CocoaTopics.html **or /Developer/Documentation/Cocoa/Cocoa.html under Panther**).
339	Program 15.8 Output: 3 5 7 11 13 17 19 23 29 31 37 41 43 47	**2** 3 5 7 11 13 17 19 23 29 31 37 41 43 47

342	Program 15.9 Test Program, 4th line of code: NSAutoreleasePool *pool = [[NSAutorreleasepool alloc] init];	NSAutoreleasePool *pool = [[NSAutorreleasePool alloc] init];
345	Program 15.10 Test Program: [insert as first line] #import<Foundations/NSAutoreleasePool.h>	
	existing third line: NSAutoreleasePool *pool = [[NSAutorreleasepool alloc] init];	NSAutoreleasePool *pool = [[NSAutorreleasePool alloc] init];

347	Program 15.11, before 7th to last line Continued: [add]
	}
	[blank line}
		}
		-(void) dealloc

354	Middle of page, first line of block of code: (BOOL) isEqual (AddressCard *) theCard	-(BOOL) isEqual (AddressCard *) theCard
355	Final line in third block of code: AddressCard *myCard~~ = (AddressBook alloc);~~	AddressCard *myCard

356	Program 15.14 Continued, final two lines of code:
	  [pool release]; return 0;
	}
		[pool release];
		return 0;
		}

370	Exercise currently listed as 7	remove page number
	Exercise currently listed as 8	Renumber as 7.

371	Exercise currently listed as 9	Renumber as 8.
	Exercise currently listed as 10	Renumber as 9.
	Exercise currently listed as 11	Renumber as 10.
377	Program 16.1 Continued, line 9: return 4;	return **3**;
	Program 16.1 Continued, line 15: return ~~5~~;	return **4**;
	Program 16.1 Continued, line 21: return ~~6~~;	return **5**;
	Program 16.1 Continued, line 28: return ~~7~~;	return **6**;
383	Program 16.4, line 4: #import <Foundation/NSAutoreleasePool.h	#import <Foundation/NSAutoreleasePool.h**>**
390	Program 16.6, line 14 NSArray *args = ~~NSProcessInfo~~ arguments];	NSArray *args = **proc** arguments];

	last three lines:
	[NSFm file ExistsAtPath: dest isDirectory: &is Dir];
	if (isDir == YES)
	   dest = [dest stringByAppendingPathComponent:
		fileExists = [NSFm file ExistsAtPath: dest isDirectory: &is Dir];
		if (**fileExists == YES &&** isDir == YES)
		   dest = [dest stringByAppendingPathComponent:

397	Running head: Basic File Operations: NSFileHandle~~p~~	Basic File Operations: NSFileHandle

402	Third block of code at the top of the page: ~~[~~myInt release]; ~~[~~printf ("after release = %x	[myInt release]; [printf ("after release = %x
410	Program 17.5 Continued,line 11 of code: printf ("~~Foo~~ dealloc\n");	printf ("**ClassA** dealloc\n");
	Program 17.5 Output, last line ~~Foo~~ dealloc	**ClassA** dealloc
411	Paragraph 3, final sentence: We did this just to verify that the ~~Foo~~ object is deallocated properly when the autorelease pool is released.	We did this just to verify that the **ClassA** object is deallocated properly when the autorelease pool is released.
445	Program 19.10, line 17: // Insert code from Program 19.4 to create and Address Book	// Insert code from Program 19.**6** to create and Address Book
448	Program 19.12, add as line 5:	**#import <Foundation/NSArrqy.h>**
484	Last paragraph: ~~Because the sizeof operator is evaluated at compile time, it~~ can be used in constant expressions (refer to the section "Constant Expressions").	**If a is a variable length array, then the expression is evaluated at runtime; otherwise, it is evaluated at compile time and** can be used in constant expressions (refer to the section "Constant Expressions").
498	First block of code. Add italic: @interface className (categoryName) <protocol,...> methodDeclaration	@interface *className (categoryName) <protocol,...>* *methodDeclaration*

	methodDeclaration ... @end	*methodDeclaration* ... @end
	next paragraph: [add italic] This defines the category categoryName for the class specified by className with the associated listed methods.	This defines the category *categoryName* for the class specified by *className* with the associated listed methods.
	next paragraph: [add italic] The compiler must know about className through a previous interface section declaration for the class.	The compiler must know about *className* through a previous interface section declaration for the class.
	Paragraph 7: [add italic] Categories are uniquely defined by className/cateoryName pairs.	Categories are uniquely defined by *className/cateoryName* pairs.
499	Protocol Definition, block of code: [add italic] @protocol protocolName <protocol, ...> methodDeclaration methodDeclaration ... @end	@protocol *protocolName <protocol, ...>* *methodDeclaration* *methodDeclaration* ... @end
	Next paragraph: [add italic] The protocal called protocolName is defined with associated methods. If other protocols are listed, protocolName also adopts the listed protocols.	The protocal called *protocolName* is defined with associated methods. If other protocols are listed, *protocolName* also adopts the listed protocols.

	Last paragraph: [add italic] A class conforms to the protocolName protocol...	A class conforms to the *protocolName* protocol...

503	The do Statement: [add italic]
	do
	programStatement
	while { expression };
		do
		programStatement
		while { *expression* };

507	Last paragraph, last sentence: As an example, the following defines a macro called myPrintf to take a ~~leading format string followed by a~~ variable number of arguments.	As an example, the following defines a macro called myPrintf to take a variable number of arguments.
535	First two lines of text on the page: ...there, as well as an HTML version (open thefile FoundationTOC.html in that folder).	...there, as well as an HTML version (open thefile FoundationTOC.html **or index.html under Panther,** in that folder).