Lesson plan for Horstmann's *Computing Concepts with Java 2 Essentials*

This lesson plan matches the chapters of *Computing Concepts with Java 2 Essentials* (second edition), by Cay Horstmann, with activities in *ProgramLive*. Each unit of the lesson plan corresponds to a chapter. Different authors of programming texts introduce material in different orders and emphasize different concepts, so the match between Horstmann and *ProgramLive* is not exact.

Below, we give an overview of each unit together with a checklist for the activities in it. Check each one off as you complete it. But first:

- An activity or lab that is labeled "optional" is in *ProgramLive* but not in *Computing Concepts*.
- Instead of using Horstmann's class `Console` for input and output, *ProgramLive* uses `JLiveRead`. The classes are similar.
- It is possible to skip the lesson in *ProgramLive* on how to use the livetext, but you will save time if you spend half an hour on lesson 0 of *ProgramLive*, learning about the features of a livetext. In addition to the activities, there are a plethora of instructional tools such as the glossary, index, exercises, and labs.

☐ Introduction to livetexts — *PL* Lesson 0-1

☐ Activities — *PL* Lesson 0-2

☐ The lesson book page — *PL* Lesson 0-3

☐ Global features — *PL* Lesson 0-4

☐ Page controls — *PL* Lesson 0-5

☐ Dealing with Java programs — *PL* Lesson 0-6

☐ Learning effectively — *PL* Lesson 0-7

Unit 1. Introduction

This may be your first contact with a programming language, and you will see lots of new terminology and concepts. Don't expect to remember everything from this first look at Java. You will also learn how to run Java programs on the computer, using either an IDE (Interactive Development Environment) or a UNIX or PC command-line environment.

In learning to run Java programs, use material that is appropriate to the system you are using. *ProgramLive* covers the Interactive Development Environments Visual Cafe (lesson 18), CodeWarrior (lesson 19), and the UNIX command-line system (Appendix B of this text).

☐ **Horstmann, Secs. 1.1-1.7, p. 2. Hardware and software**

☐ Hardware and software *PL* Lesson 1-1, page 41
Horstmann does a more thorough job of discussing hardware and software.

☐ **Horstmann, Sec. 1.8-1.10, p. 21. Compiling and errors**

☐ Some simple Java programs *PL* Lesson 1-2, page 43
This lesson page introduces *ProgramLive*'s filing cabinet metaphor to help you understand what a class is. The `import` statement is mentioned, which Horstmann does much later.

☐ Introduction to your IDE. Use whatever materials are available for your method of running Java programs. Visual Cafe is covered in *ProgramLive* lesson 18; CodeWarrior, in lesson 19. There is an introduction to the UNIX command-line system in Appendix B.

☐ Type `char` (advanced) (first activity) *PL* Lesson 6-5, page 121
Horstmann, Sec. 1.8, provides an advanced topic: escape sequences. Look at the footnote for a synopsis of Java escape sequences. One difference: Horstmann does not introduce primitive type **char** until much later, preferring instead to do everything in terms of class `String`.

☐ Components of a Java program (only activities 1–3) *PL* Lesson 1-3, page 47

☐ Good programming practices *PL* Lesson 13-1, page 185

☐ Syntax errors *PL* Lesson 19-5

☐ **Horstmann, Sec. 1.11, p. 31. A first look at objects and classes**

☐ Class and objects *PL* Lesson 3-3, page 79
The class as a way of collecting related pieces of information, and how classes and objects are related.

Unit 2. Fundamental data types

A *type* defines a set of values and operations on them. A *variable* is a named box into which a value can be stored for later use. Chapter 2 explores the use of numerical and `String` types.

☐ **Horstmann, Sec. 2.1, p. 48. Number types (int and double)**

☐ Overview of primitive types *PL* Lesson 6-1, page 117
This lesson page lists all primitive types and shows their ranges and precision (advanced).

☐ The integral types *PL* Lesson 6-2, page 118
Covers integral constants (literals), and operations on types **int** and **long**. Also covers casting (optional).

☐ A minimalist view of floating point *PL* Lesson 6-3, page 120

☐ Remarks about floating point (advanced) *PL* Lesson 6-4, page 121

☐ **Horstmann, Sec. 2.2, p. 56. Assignment**

☐ Components of a Java program *PL* Lesson 1-3, page 47
Do the last activities, on variables, types, and expressions.

☐ Assignment (only activities 1 and 2) *PL* Lesson 1-4, page 51

☐ Do Lab PGL-1 (Assignment) of this lesson. *PL* Lesson 1

☐ **Horstmann, Sec. 2.3. Type conversion, p. 59**

☐ The integral types *PL* Lesson 6-2, page 118
Do all the activities and read the discussion at the bottom of lesson page 6-3.

☐ Naming conventions *PL* Lesson 13-2, page 185
Read the lesson page and listen to only activity 1, on naming parameters; activity 2, on naming local variables; and activity 4, on naming constants.

☐ **Horstmann, Sec. 2.4-2.5, p. 66. Constants and arithmetic**

☐ Class **Math** *PL* Lesson 3-2, page 77
Arithmetic expressions were covered already in lesson 1-3, activity 5, and lesson page 6-2. This lesson page covers constants and the methods in class `Math`.

☐ **Horstmann, Sec. 2.6, p. 75. Strings**

☐ Strings *PL* Lesson 5-3, page 106

☐ **Horstmann, Sec. 2.7, p. 81 Reading input**

☐ Input/output (activities 1 and 2) *PL* Lesson 1-5, page 56
ProgramLive's class `JLiveRead` is similar to Horstmann's `ConsoleReader`.

☐ **Horstmann, Sec. 2.8, p. 83. Reading input (advanced)**

☐ Reading from the keyboard and files *PL* Lesson 5-7, page 113

Unit 3. An introduction to classes

The study of classes requires you to learn about the interplay between (a) objects and (b) the methods that objects contain. A method is like a recipe; when the recipe (method) is to be carried out, the chef (the computer) follows its instructions. *ProgramLive* and Horstmann introduce objects and methods in different ways. *ProgramLive* first provides a thorough discussion of methods and then proceeds to study objects; Horstmann mingles the study of the two. Therefore, matching this Horstmann chapter to *ProgramLive* is a bit tricky. Further, *ProgramLive* provides a model of memory, along with detailed instructions on executing method calls, which is not covered in Horstmann.

☐ **Horstmann, Sec. 3.1-3.4, p. 104. The basics of classes**

☐ Classes (only the first activity) *PL* Lesson 3-1, page 75
Also, read about the placement of classes in a Java program.

☐ Method *PL* Lesson 2-1, page 61
This lesson page studies executing method calls, but in a simpler, non-object-oriented context.

☐ Method bodies and method calls *PL* Lesson 2-2, page 64
This lesson page provides, in more detail, the material in Horstmann's Sec. 3.2 and 3.4. The model of execution provides understanding on how method calls are carried out.

☐ Functions (only activities 1 and 2) *PL* Lesson 2-4, page 70

☐ Classes and objects *PL* Lesson 3-3, page 79
This lesson page discusses the reason for objects and shows how to write a class. It introduces instance variables (Horstmann, Sec. 3.3).

☐ Creating and initializing objects. *PL* Lesson 3-4, page 82
Objects are created (and stored in the file drawer) during execution.

☐ Nonstatic methods *PL* Lesson 3-6, page 87
Static methods go in the class drawer, nonstatic methods, or *instance methods*, belong in each instance of the class.

☐ **Horstmann, Sec. 3.5, p. 112. Constructors**

☐ Scope boxes and constructors *PL* Lesson 3-5, page 85
Do the first activity, on scope boxes, only if *ProgramLive*'s model of memory is being taught.

☐ Do Lab PGL-1 (Writing constructors) of this lesson. *PL* Lesson 3

☐ Do Lab PGL-2 (Drawing objects) of this lesson. *PL* Lesson 3

☐ Consequences of using objects *PL* Lesson 3-7, page 90
Listen to the first activity only if *ProgramLive*'s model of memory is being taught. Listen to the rest of this page only if method `toString` is being emphasized.

☐ Do Lab PGL-3 (Drawing frames) of this lesson. *PL* Lesson 3

☐ Describing variables *PL* Lesson 13-5, page 191

☐ Style considerations concerning classes (optional) *PL* Lesson 13-2, page 185
Listen only to the third activity (p. 187) on naming instance variables and class variables, the last activity (p. 188) on naming classes, and the footnote on lesson page 13-3 (p. 189) on indenting components of a class.

☐ **Horstmann, Sec. 3.7, p. 120. Discovering classes**

☐ Object-oriented design *PL* Lesson 3-8, page 93

☐ **Horstmann, Sec. 3.9, p. 127. The null reference**

☐ Classes and objects (last activity) *PL* Lesson 3-3, page 79

Unit 4. Applets and graphics

An *application* is a Java program whose execution starts when the system calls method `main` of some class. An *applet* is a Java program whose execution starts when a browser (e.g. Netscape or Internet Explorer) loads an html page that contains a command to start the applet.

Both applications and applets can be used as graphics programs —i.e. programs that draw graphics (line, rectangles, circles, etc.) in graphics windows.

☐ **Horstmann, Secs. 4.1-4.3, p. 141. Applets**

☐ Applets *PL* Lesson 16-1, page 205

☐ Examples of applets *PL* Lesson 16-3, page 208

☐ HTML and applet commands *PL* Lesson 16-2, page 206

☐ **Horstmann, Secs. 4.4-4.7, p. 150. Graphics**

☐ Input/Output *PL* Lesson 1-5, page 56
The material in these sections of Horstmann is not covered in *ProgramLive*. The closest the livetext comes is activity 5 and activity 6 on page 1-5.

☐ **Horstmann, Sec. 4.8, p. 161. Reading text input**

☐ Examples of applets *PL* Lesson 16-3, page 208
Listen to activity 2 for a discussion of class `JOptionPane`. Look in the index of this *Companion* to get to a specification of this class.

Unit 5. Decisions

This section discusses conditional statements and the boolean expressions that are used as conditions of such statements.

☐ **Horstmann, Sec. 5.1, p. 184. The if statement**

☐ Three statements . . . (activities 3, 4, 6, and 7) *PL* Lesson 1-4, page 51

☐ Conventions for indentation *PL* Lesson 13-3, page 189
Do activity 1 and read the footnotes on indenting the if- and if-else-statements.

☐ Do Lab PGL-2 (if-statement) of this lesson. *PL* Lesson 1

☐ Do Lab PGL-3 (if-else-statement) of this lesson. *PL* Lesson 1

☐ **Horstmann, Sec. 5.2, p. 189. Comparing values**

☐ Three statements . . . (only activity 5) *PL* Lesson 1-4, page 51
For comparisons of floating-point numbers, read lesson page 6-4.

☐ Strings (comparing) *PL* Lesson 5-3, page 106
Listen to activity 6 and peruse the third footnote.

☐ Objects (comparing) *PL* Lesson 3-7, page 90
Read the section on equality and aliasing, including the footnotes.

☐ **Horstmann, Sec. 5.3, p. 195. Multiple alternatives**

☐ Three statements . . . (only activity 5) *PL* Lesson 1-4, page 51

☐ Occasionally useful statements *PL* Lesson 1-7, page 60
Read the section on the **switch** statement, including the footnotes.

☐ **Horstmann, Sec. 5.4, p. 207. Boolean expressions**

☐ Type **boolean** *PL* Lesson 6-6, page 123

☐ Assertions in programs *PL* Lesson 1-6, page 58
This material is covered more briefly in Horstmann.

Unit 6. Iteration

The loop is the most difficult statement to understand, and the best way to understand a loop is in terms of a "loop invariant". *ProgramLive* introduces the loop invariant almost immediately, and just about every loop studied is presented in terms of a loop invariant. Horstmann, on the other hand, treats the loop invariant as an advanced topic (on p. 256).

☐ **Horstmann, Sec. 6.1, p. 224. while loops**

☐ Iteration *PL* Lesson 7-1, page 127

☐ Do Lab PGL-1 (Executing a `while` loop) of this lesson. *PL* Lesson 7

☐ Several examples of loops *PL* Lesson 7-2, page 132
This lesson page discusses the development of loop invariants and then gives three examples of the development of loops.

☐ Do Lab PGL-2 (Developing loops from invariants) of this lesson. *PL* Lesson 7

☐ (Optional) Do Lab PGL-3 (Developing loops . . . II) of this lesson. *PL* Lesson 7

☐ Conventions for indentation *PL* Lesson 13-3, page 189
Read the footnote on conventions for indenting loops.

☐ Loop schemata *PL* Lesson 7-3, page 134
Rather than write each loop from scratch, learn to use loop schemata.

☐ Do Lab PGL-4 (Using loop schemata) of this lesson. *PL* Lesson 7

☐ **Horstmann, Sec. 6.2. For-loops**

☐ The for-loop *PL* Lesson 7-4, page 136
The for-loop is an abbreviation of a while-loop that uses a "loop counter". It is extremely useful when the number of iterations to perform is known before execution of the loop.

☐ Do Lab PGL-4 (Translating whiles into fors) of this lesson. *PL* Lesson 7

☐ **Horstmann, Sec. 6.3, 227, p. 231. Do loops**

☐ Miscellaneous points (the do-while loop) *PL* Lesson 7-6, page 140
Read the section on the do-while loop, including the footnote.

☐ Making progress and stopping (activity 1) *PL* Lesson 7-5, page 138

☐ **Horstmann, Common errors 6.2-6.4, p. 232**

☐ Making progress and stopping *PL* Lesson 7-5, page 138
For "off-by-one errors", do activity 4.

☐ Miscellaneous points *PL* Lesson 7-6, page 140
For"a semicolon too many", read the warning at the top of the lesson page.

☐ **Horstmann, Quality tip 6.2, p. 234** *PL* Lesson 7-5, page 138
Activity 2 deals with the use of `!=` in the loop condition and comes to the opposite conclusion of Quality tip 6.2. This sort of disagreement is very common in the programming world. Think about both sides of the argument and use whichever convention your instructor prefers.

☐ **Horstmann, Sec. 6.4, p. 237. Nested loops**

☐ Miscellaneous points (nested loops) *PL* Lesson 7-6, page 140
Activities 1 and 2 make the point that you should not think in terms of nested loops, even if they are in the program. The third activity shows how not to develop a loop. There is no need to read the rest of the page.

☐ **Horstmann, Sec. 6.5. Processing input, p. 239**

☐ Loop schemata (activities 1 and 2) *PL* Lesson 7-3, page 134

☐ **Horstmann, Advanced topic. 6.3, p. 239**

☐ Miscellaneous points *PL* Lesson 7-6, page 140
The **break** and **continue** statements are discussed at the bottom of this lesson page.

Unit 7. Methods

In this chapter, Horstmann provides a more in-depth overview of methods in reference to classes. *ProgramLive* studies methods thoroughly in isolation before introducing classes. This is why the match of Horstmann and *ProgramLive* is not clean.

☐ **Horstmann, Sec. 7.1, p. 270. Method parameters**

☐ Methods (activities 2, 4, and 5) *PL* Lesson 2-1, page 61
A parameter is initialized to the value of the corresponding argument when the method in which it is declared is called. A parameter can also be assigned other values within the method body.

☐ Naming conventions *PL* Lesson 13-2, page 185
Read the general guidelines and listen to the activity on naming parameters.

☐ Method bodies and method calls *PL* Lesson 2-2, page 64
At this point, this material should be a review.

☐ Guidelines for writing methods *PL* Lesson 13-4, page 190
The first two activities try to convince you of the importance of method specifications. The last three activities provide insight on how to structure method bodies when they get long.

☐ **Horstmann, Sec. 7.2, p. 273. Accessor methods, mutator methods, and side effects**

☐ Nonstatic methods *PL* Lesson 3-6, page 87
This lesson page reviews instance methods. The fourth activity talks about "getter" and "setter" methods, *ProgramLive*'s terminology for Horstmann's accessor and mutator methods.

☐ **Horstmann, Sec. 7.3, p. 276. Static methods**

☐ Classes *PL* Lesson 3-1, page 75

☐ **Horstmann, Sec. 7.4, p. 281. The return statement**

☐ Two components of method bodies (last activity) *PL* Lesson 2-3, page 67

☐ Form of a function call (activity 1) *PL* Lesson 2-4, page 70

☐ Do Lab PGL-4 (writing simple functions) of this lesson. *PL* Lesson 2

☐ **Horstmann, Sec. 7.5, p. 283. Static variables**

☐ Class `Math` (activity 2) *PL* Lesson 3-2, page 77

☐ Naming conventions (activity 2) *PL* Lesson 13-2, page 185

☐ **Horstmann, Sec. 7.6, p. 287. Variable lifetime, initialization, and scope**
These terms are introduced and discussed at various places within *ProgramLive*. For a summary of "lifetime" and "scope" of variables, look the words up in *ProgramLive*'s glossary.

For the default initial values of static and instance variables, look up "default" values in the *ProgramLive* glossary. A parameter is initialized to the value of the corresponding argument. A local variable is not initialized.

☐ **Horstmann, Sec. 7.7, p. 291. Comments**

☐ Guidelines for writing methods *PL* Lesson 13-4, page 190
This lesson page describes (in detail) specifications of methods, as well as the use of statement-comments.

☐ Describing variables *PL* Lesson 13-5, page 191

☐ Javadoc *PL* Lesson 13-1, page 185
Javadoc is described in a footnote at the bottom of this lesson page and in more detail on p. 245 of this *Companion*.

☐ **Horstmann, Sec. 7.8, p. 296. Preconditions**

☐ Assertions in programs (last activity) *PL* Lesson 1-6, page 58

☐ **Horstmann, Sec. 7.9, p. 298. Recursion**

☐ Recursion *PL* Lesson 15-1, page 197
Lesson 15-1 of *ProgramLive* provides a more thorough study of recursion than Horstmann.

Unit 8. Testing and debugging

☐ **Horstmann, Sec. 8.1-8.5, p. 314. Introduction to testing**

☐ Introduction to testing and debugging *PL* Lesson 14-1, page 193

☐ Testing strategies (second activity) *PL* Lesson 14-2, page 194
Class `JLiveRead` is similar to Horstmann's class `ConsoleReader`.

☐ Selecting test cases and checking them ... *PL* Lesson 14-3, page 195

☐ **Horstmann, Sec. 8.6, p. 323. The debugger**

☐ Debugging *PL* Lesson 14-4, page 196

☐ The CodeWarrior debugger *PL* Lesson 19-1

☐ Breakpoints and expression in the IDE *PL* Lesson 19-2

☐ **Horstmann, Sec. 8.7, p. 332. Debugging strategies**

☐ Debugging *PL* Lesson 14-4, page 196

Unit 9. Inheritance and interfaces

☐ **Horstmann, Sec. 9.1, p. 342. Introduction to inheritance**

☐ Subclasses *PL* Lesson 4-1, page 97

☐ Do Lab PGL-1 (Drawing objects II) of this lesson. *PL* Lesson 4

☐ **Horstmann, Sec. 9.1, p. 345. Converting between class types**

☐ Casting and a new model of execution *PL* Lesson 4-3, page 101
This lesson page introduces *ProgramLive*'s final method of execution. Don't listen to the last activity yet.

☐ **Horstmann, Sec. 9.3, p. 348. Inheritance hierarchies**

☐ Constructors and inherited methods (last activity) *PL* Lesson 4-2, page 99

☐ **Horstmann, Sec. 9.4- 9.5, p. 351. More about subclasses**

☐ Constructors and inherited methods *PL* Lesson 4-2, page 99

☐ A last look at classes `Employee` ... (last activity) *PL* Lesson 4-3, page 101

☐ Do Lab PGL-2 (Writing constructors II) of this lesson. *PL* Lesson 4

☐ Do Lab PGL-3 (Drawing frames II) of this lesson. *PL* Lesson 4

☐ Object-oriented design *PL* Lesson 4-4, page 103

☐ **Horstmann, Sec. 9.6, p. 359. Polymorphism**

☐ Classes (overloading method names, last activity) *PL* Lesson 3-1, page 75
This activity treats polymorphism, under the title "overloading method names".

☐ **Horstmann, Sec. 9.7, p. 362. Interfaces**

☐ Interfaces *PL* Lesson 12-1, page 179

☐ The interface as a type *PL* Lesson 12-2, page 180

☐ Interface `Comparable` *PL* Lesson 12-3, page 182

☐ **Horstmann, Advanced topic 9.2, p. 366. Abstract classes**

☐ Abstract classes *PL* Lesson 4-5, page 104

☐ (Optional) Do Lab PGL-4 (Practice with shapes) of this lesson. *PL* Lesson 4

☐ **Horstmann, Sec. 9.8, p. 368. Access control**

☐ Constructors and inherited methods (last activity) *PL* Lesson 4-2, page 99
Look at item *access modifier* in *ProgramLive*'s glossary for a short summary.

☐ **Horstmann, Sec. 9.9, p. 371. The cosmic superclass**

☐ Constructors and inherited methods (last activity) *PL* Lesson 4-2, page 99

☐ Consequences of using objects *PL* Lesson 3-7, page 90
Look at the activities on method `toString` and read about equality testing.

☐ **Horstmann, Sec. 9.10, p. 380. Packages**

☐ Packages *PL* Lesson 11-1, page 177

Unit 10. Event handling

This chapter is about GUIs and the events that occur within them.

☐ GUIs and event-driven programming *PL* Lesson 17-1, page 209

☐ Components and container *PL* Lesson 17-2, page 210

☐ Layout managers *PL* Lesson 17-3, page 211

☐ Listening to a GUI *PL* Lesson 17-4, page 212

☐ Interfaces —ActionListener (last activity) *PL* Lesson 12-1, page 179

Unit 11. Arrays and vectors

An "array" is a collection of elements of the same (primitive or class) type —`int`, `String`, `JLiveWindow`, etc. With arrays we can discuss algorithms such as searching arrays and sorting arrays into ascending order.

☐ **Horstmann, Sec. 11.1, p. 432. Using arrays**

☐ Introduction to arrays. *PL* Lesson 8-1, page 143
This lesson page introduces all the technical details concerning arrays.

☐ Talking about array segments *PL* Lesson 8-2, page 146
The notation makes it easier to discuss algorithms that manipulate arrays.

☐ About array schemas (activities 1 and 2) *PL* Lesson 8-3, page 148

☐ **Horstmann, Sec. 11.2, p 443. Array parameters and return values**

☐ Some programs that use arrays (last two activities) *PL* Lesson 8-3, page 148

☐ **Horstmann, Sec. 11.3, p. 444. Simple array algorithms**

☐ Some programs that use arrays (activities 4 and 5) *PL* Lesson 8-3, page 148

☐ Some basic array algorithms *PL* Lesson 8-5, page 156

☐ Do Lab PGL-1 (Using arrays) of this lesson. *PL* Lesson 11

☐ **Horstmann, Sec. 11.5, p. 455. Arrays as object data**

☐ Arrays and classes (first three activities) *PL* Lesson 8-4, page 153

☐ **Horstmann, Sec. 11.6, p. 461. Vectors**

☐ Arrays and classes *PL* Lesson 8-4, page 153
Do the last three activities; they introduce the concept of a dynamic array using a class that is slightly different from class `Vector`.

☐ Class `Vector` *PL* Lesson 5-5, page 111

☐ **Horstmann, Sec. 11.7, 465. Two-dimensional arrays**

☐ Multidimensional arrays *PL* Lesson 9-1, page 163

☐ Programs that use two-dimensional arrays *PL* Lesson 9-2, page 165

☐ Do Lab PGL-1 (Rectangular arrays) of this lesson. *PL* Lesson 9

☐ The Java concept of a multidim. array (optional) *PL* Lesson 9-3, page 168
You will see that a two-dimenssional array is really an array whose elements are arrays, whose elements can be different lengths!

☐ Programs that use ragged arrays (advanced) *PL* Lesson 9-4, page 170

Unit 12. Graphical user interfaces

This topic is covered in a cursory manner in *ProgramLive*.

☐ GUIs and event-driven programming *PL* Lesson 17-1, page 209

☐ Components and container *PL* Lesson 17-2, page 210

☐ Layout managers *PL* Lesson 17-3, page 211

☐ Listening to a GUI *PL* Lesson 17-4, page 212

Unit 13. Streams and exceptions

- ☐ **Horstmann, Sec. 13.1-13.2, p. 516. Reading and writing streams**
- ☐ Reading from the keyboard and files — *PL* Lesson 5-7, page 113
- ☐ Writing to the Java console and files — *PL* Lesson 5-8, page 116
- ☐ **Horstmann, Sec. 13.3-13.4, p. 522. Exception handling**
- ☐ Output of thrown exceptions and errors — *PL* Lesson 10-1, page 173
- ☐ The throwable object — *PL* Lesson 10-2, page 173
- ☐ Catching a thrown exception — *PL* Lesson 10-3, page 174
- ☐ The throw-statement — *PL* Lesson 10-4, page 175
- ☐ Checked exceptions and the throws clause — *PL* Lesson 10-5, page 176
- ☐ Hints on using exceptions — *PL* Lesson 10-6, page 176

Unit 14. Object-oriented design

The material on designing and developing programs, including object-oriented design, is covered in several different lessons in *ProgramLive*. For this chapter of Horstmann, we list those lesson pages.

- ☐ Top-down programming — *PL* Lesson 2-5, page 72
  This may be a review.
- ☐ Object-oriented design — *PL* Lesson 3-8, page 93
  How to develop a program in an object-oriented framework.
- ☐ Object-oriented design with subclasses — *PL* Lesson 4-4, page 103

Unit 15. Algorithms

- ☐ **Horstmann, Secs. 15.1-15.3, p. 614. Selection sort**
- ☐ Selection sort and insertion sort — *PL* Lesson 8-6, page 161
- ☐ **Horstmann, Sec. 15.4-15.5, p. 622. Merge sort**
- ☐ Some interesting recursive methods (last activity) — *PL* Lesson 15-3, page 202
- ☐ Quicksort — *PL* Lesson 15-4, page 202
  This sorting algorithm, which is perhaps the most famous and most widely used sorting algorithm, is not covered in Horstmann.
- ☐ **Horstmann, Sec. 15.6, p. 630. Linear search**

☐ Some basic array algorithms (first two activities) *PL* Lesson 8-5, page 156

☐ **Horstmann, Sec. 15.7, p. 632. Binary search**

☐ Some basic array algorithms (last activity) *PL* Lesson 8-5, page 156