Chapter Objectives

Chapter 1: Object-Oriented Programming

1. To introduce the basic concepts and advantages of object-oriented technology.

2. To contrast object-oriented technology with alternatives such as top-down functional decomposition/procedural programming.

3. To clarify the relationship between object-oriented design and object-oriented programming.

4. To introduce UML, the de facto standard object-oriented modeling language.

5. To explain how object-oriented languages directly and powerfully support general programming goals such as modularity, abstract data types, and code robustness.

6. To introduce the basic vocabulary of object-oriented programming: class, object, interface, message passing and method invocation, encapsulation, "instance" versus "class" member, information hiding, inheritance, and polymorphism.

7. To explain the concepts of *server* and *client* in the context of object-oriented technology.

8. To clarify the advantages of object-oriented technology in component-based programming.

9. To illustrate briefly the Java syntax for general object-oriented constructs such as *class* and *interface*.

Chapter 2: Introductory Programs

1. To explain the basic steps in coding, documenting, compiling, and executing a Java program.

2. To provide and clarify in detail full programs that the user can study, execute, and adapt.

3. To explain the role of the JVM.

4. To introduce basic programming Java constructs such as classes, class members, `if` statements, and loops.

5. To illustrate the syntax and basic use of constructors, methods, and fields.

6. To introduce the `String`, `StringBuffer`, `StringTokenizer`, `Vector`, and `Hashtable` classes.

7. To explain and illustrate basic input/output operations, including writing to the standard output and files and reading from the standard input and files.

8. To show the convenience of utility classes such as `Random` and `Date`.

9. To introduce exceptions and exception-handling.

10. To provide full programs that illustrate the "look and feel" of coding in Java.

11. To introduce programmer-define classes and test clients for testing their functionality.

Chapter 3: Programs and Packages

1. To survey the basic Java program types: application, applet, servlet, and bean.

2. To review and expand the discussion of *java* and *class* files

3. To introduce packages: standard, default, and programmer-defined.

4. To review the `import` statement and fully qualified names.

5. To explain the role of the `CLASSPATH` environment variable.

6. To illustrate in a sample application the use of programmer-defined packages.

Chapter 4: Language Fundamentals

1. To explain basic language constructs such as identifiers, variables, primitive and class types, and operators.

2. To examine in detail the difference between constructors and methods.

3. To clarify further arrays, bounds checking, and "arrays of arrays."

4. To examine basic control structures such as `if/else`, `switch`, `break`, and `continue` constructs, and loops.

5. To highlight the usefulness of utility classes such as `Vector`s and `Random` number generators.

6. To illustrate the object-oriented principle of "programming to the interface."

7. To examine in detail the syntax and use of exceptions.

8. To present small but realistic programs that show how fundamental language features are used.

Chapter 5: Classes

1. To explain, in technical detail, scope for classes and their members.

2. To introduce advanced features of constructors, methods, and fields.

3. To clarify constructor and method overloading.

4. To review and extend the discussion of “class” and “instance” members.

5. To review the distinction between `static` and non`static` members and the access rule for `static` ones.

6. To provide a `BasicInput` and `BasicOutput` class as examples of high-level wrappers to facilitate input/output operations.

7. To provide a utility for file copying that includes a GUI.

8. To introduce the integration of graphics components into applications.

9. To illustrate and clarify the process of writing test clients for library classes.

Chapter 6: Inheritance, Interfaces, and Abstract Classes

1. To explain single inheritance for classes and multiple inheritance for interfaces.
2. To highlight the key role of `Object` as the root of the inheritance hierarchy.
3. To explain the role of the constructors in general and the no-argument constructor in particular within inheritance hierarchies.
4. To explain the connection between method overriding and polymorphism.
5. To illustrate the power of polymorphism through examples and a sample application on polymorphic input and output.
6. To review the importance of overriding `Object` methods.
7. To review the syntax and role of interfaces.
8. To illustrate "programming to the interface" with standard and programmer-defined interfaces.
9. To introduce the syntax and role of abstract classes.
10. To compare and contrast concrete classes, interfaces, and abstract classes.

Chapter 7: Graphics and Event Handling

1. To present an overview of the similarities and differences between AWT and Swing set graphics.

2. To highlight architectural similarities between AWT and Swing set graphics.

3. To clarify basic graphics terminology such as *component*, *container*, *top-level window*, *dialog window*, and the like.

4. To explain the event-delegation model at the core of Java event handling.

5. To clarify the relationship between event-driven programming and graphical user interfaces.

6. To illustrate through examples basic GUI constructs such as buttons, check boxes, docked and popup menus, menu bars and tool bars, scrollbars, and the like.

7. To introduce sophisticated Swing set components such as `JTree`s and `JTable`s.

8. To introduce basic drawing and explain the role of the `Graphics` context.

9. To clarify the model/view/controller architecture at the foundation of Swing set graphics.

10. To explain and illustrate “pluggable look and feel.”

Chapter 8: Three Interface: `Cloneable`, `Serializable`, and `Runnable`

1. To explain the use of empty or *marker* interfaces.

2. To introduce object cloning.

3. To highlight and illustrate through examples the challenges of cloning objects that contain object or array references.

4. To explain how cloning can be disabled.

5. To introduce and illustrate the basic syntax and the rich uses of serialization.

6. To explain the concept of *object state*.

7. To clarify the distinction between serializing objects and writing primitive types to binary streams.

8. To illustrate through examples serialization and deserialization.

9. To show how serialization can be customized.

10. To explain and illustrate the connection between serialization and object persistence.

11. To explain and illustrate what happens to `transient` and `static` fields during serialization.

12. To explain potential problems with serialization.

13. To introduce and illustrate the syntax and uses of multithreading.

14. To explain the priority-based, preemptive scheduling of the JVM.

15. To contrast program termination under single-threaded and multithread conditions.

16. To introduce the distinction between user and daemon threads.

17. To highlight and illustrate through examples the need for thread synchronization.

18. To explain the `synchronized` construct and its relationship to `wait`, `notify`, and `notifyAll`.

17. To introduce through examples core thread methods such as `start`, `run`, `join`, `sleep`, and the like.

18. To introduce the terminology of critical section problems and to highlight how Java thread constructs address such problems.

19. To explain and illustrate deadlock.

20. To summarize multithreading issues with a sample application on the dining philosophers problem.

Chapter 9: Network Programming

1. To present an overview of networking concepts in general and TCP/IP in particular.

2. To explain client and server socket implementations in Java through a series of examples.

3. To explain datagram packets and sockets, including multicast sockets.

4. To consolidate socket material through a sample application with a multithreaded server.

5. To explain the role of applets in distributed client/server applications.

6. To clarify "sandbox security" for applets.

7. To illustrate through a sample application the integration of applet and socket technology.

8. To present the steps for creating an RMI client and an RMI server.

9. To explain the socket-based infrastructure for RMI.

10. To clarify how advanced Java technologies such as Jini leverage RMI.

11. To introduce object request brokers in general and CORBA in particular.

12. To present the steps for creating a CORBA client and a CORBA server.

13. To compare and contrast RMI and CORBA technologies.

Chapter 10: Selected Topcis

1. To explain in detail bean technology as a type of component-based programming.

2. To clarify the role of property sheets in beans.

3. To introduce the Bean Box utility.

4. To underscore the importance of property change events with respect to beans.

5. To explain why beans need to be serializable.

6. To present an overview of security constructs in Java, including the compiler, bytecode verifier, security manager, and access controller.

7. To present Java-based authentication through digital signatures in a series of related examples.

8. To introduce Java's reflection technology.

9. To illustrate through examples practical uses of reflection.

10. To introduce servlets.

11. To introduce JDBC.

12. To integrate servlets and JDBC in a sample application.