True/False Review Questions

Chapter 1

1. In object-oriented programming, a class is a data type and an object is an instance of such a type.
2. Object-oriented programming is associated with a design technique known as top-down, functional decomposition.
3. UML is a modeling language associated with object-oriented as opposed to procedural programming.
4. Abstract data types cannot be implemented in procedural languages such as C and Pascal.
5. Object-oriented languages directly support abstract data types through information hiding.
6. Encapsulation means the same thing as information hiding.
7. Methods are typically used to construct objects of a particular class type, whereas constructors are typically used to define operations appropriate to a particular class type.
8. A class encapsulates only public members such as high-level methods and constructors but not private members such as implementation fields.
9. An object obj1 sends a message to an object obj2 by invoking an obj2 method.
10. Under inheritance, a superclass inherits all of the members in all of its subclasses.
11. The term interface is a synonym for hidden implementation.
12. UML models replace programs written in languages such as Java.
Chapter 2

1. A Java source file contains Java code as text and may have either *java* or *class* as its extension.

2. If a Java source file is compiled successfully, the compilation produces one or more files with a *java* extension.

3. Every Java program requires at least one class.

4. A programmer-defined class can have any name except a Java keyword as long as the name is a valid Java identifier.

5. A class's name must begin with an uppercase letter.

6. In a class declaration, the opening left brace { and the closing right brace } must occur alone on a line.

7. A class definition must begin on the first line of a source file and not even a comment is allowed to occur above a class definition.

8. Although methods and fields can be *static*, constructors cannot be *static*.

9. In Java, the identifiers *main*, *Main*, and *mMain* are all distinct.

10. If a programmer uses an *import* statement, the programmer is required to use the fully qualified name for all classes in the imported package.

11. Under the Java naming convention for input and output streams, *readers* and *writers* are character-based streams.

12. If a program generates an *exception*, the program is required to provide an *exception handler* so that the program compiles.
Chapter 3

1. The Java Virtual Machine (JVM) is used to compile as opposed to execute Java programs.

2. A Java bean requires a host program in order to execute.

3. A Java application requires a host program in order to execute.

4. The types application, applet, servlet, and bean partition Java programs in that every Java program belongs to exactly one and only one of these types.

5. Java bytecodes are compiled instructions that execute directly on the host system.

6. Every Java program requires at least one import statement, in particular a statement to import all of the classes in the java.lang package.

7. The import statement

    import java.*;  

does not cause compile-time errors if placed at the very top of a source file.

8. Every source file must be named after the first class declared in the file.

9. Every class file belongs to a package.

10. A package statement can occur either before or after import statements.

11. Java has standard packages but does not allow programmer-defined packages.

12. Every constructor must have void in place of a return type because a constructor cannot return a value.
Chapter 4

1. A valid Java identifier such as the name of a class must begin with an alphabetic character.

2. Local variables, like class fields, have default values such as zero for integers and floating-point numbers.

3. If the programmer fails to declare a local variable’s data type, the type defaults to int.

4. A class C could have a member named c.

5. The expression 0x99 is a hexadecimal constant.

6. Although dividing an integer by zero is an error, dividing a floating-point number by zero is not an error.

7. The code segment
   ```java
   int x = 8;
   System.out.println( x++ );
   ```
   prints 8 to the standard output.

8. An double-quoted expression such as "foo" is a String reference.

9. A boolean value such as true may be assigned with an explicit cast to an int variable because such a cast converts true to 1 and false to 0.

10. The operator = is used for assignment and initialization but not to test for equality.

11. The equality operator == cannot be applied to floating-point types such as doubles.

12. When an integer array is constructed with the new operator, the array’s cells are initialized to zero regardless of whether the array is a field.

13. A try block can occur without an accompanying catch clause or finally clause.

14. A program that throws an uncaught exception generates a compile-time rather than a run-time error.
Chapter 5

1. A programmer-defined class can have only package scope.

2. Class members can have protected scope but the class that encapsulates such members cannot have protected scope.

3. A standard class with package scope is visible only to classes in the same package.

4. Protected scope is broader than package scope.

5. If class C has package scope, then C’s public and package members effectively have the same visibility.

6. Any class member except a constructor can be static.

7. If class C does not define any constructors, then C has a publicly accessible no-argument constructor.

8. A nonstatic method cannot access a static member in the method’s encapsulating class.

9. A static method cannot access a nonstatic member in the method’s encapsulating class.

10. A constructor cannot have private scope.

11. The default scope for a constructor is public, whereas the default scope for any other member is package.

12. A class cannot overload its constructors.
Chapter 6

1. Java supports multiple inheritance for classes but only single inheritance for interfaces.

2. Java supports multiple inheritance for standard classes but only single inheritance for programmer-defined classes.

3. A programmer-defined class has no superclass unless the class is defined explicitly to extend a superclass.

4. In class inheritance, a subclass inherits only the non-private members of the superclass.

5. An interface can extend or implement another interface.

6. The class `java.lang.Object` is the only class, standard or programmer-defined, that has no superclass.

7. If a class is defined as `final`, the class cannot be extended.

8. If a subclass `Sub` overrides method `m` inherited from superclass `Super`, then `m` must have the same signature in `Sub` and `Super`.

9. An abstract class can be implemented but not extended.

10. An interface must be declared explicitly as `abstract`.

11. If class `C` implements interface `IFace` but fails to define all of the methods declared in `IFace`, then `C` is abstract.

12. Neither an abstract class nor an interface can be instantiated as an object.
Chapter 7

1. Classes for AWT and the Swing set graphics reside in different packages.
2. Every Swing set component is lightweight.
3. In the Swing set’s implementation of the model/view/controller architecture, the model and the controller are integrated as the UI delegate.
4. In the Swing set hierarchy, classes descended from `JComponent` are used to construct lightweight components.
5. Java’s basic event model requires that an event listener be the container in which an event source such as a button is embedded.
6. If class C implements the `WindowListener` interface, then C must define all of the method declared in this interface in order to be concrete.
7. Framed windows such as `Frames` and `JFrames` are constructed by default as visible.
8. A menu can contain nested submenus to an arbitrary level.
9. A `graphics context` is the same as a layout manager.
10. An override of the `paint` method can invoke the superclass version in order to clear the drawing area.
12. Even relatively simple Swing set components such as `JButtons` give the programmer access to the component’s underlying model.
Chapter 8

1. Every interface must declare at least one method.

2. The Serializable interface declares two methods, writeObject and readObject.

3. Object cloning is an alternative term for object construction.

4. To clone an object is to copy a reference to the object.

5. Primitive types such as ints and doubles cannot be serialized but can be written to the same binary stream as a serializable object.

6. Deserialization restores transient fields to their default values.

7. If an object’s class does not implement Serializable but the object’s superclass does implement this interface, then the object is serializable.

8. A program executes a constructed Thread by invoking the Thread’s method run.

9. The Runnable interface is empty.

10. If thread T<sub>1</sub> runs at priority 1 and thread T<sub>2</sub> runs at priority 2, then T<sub>2</sub> is guaranteed to execute exactly twice as many times as T<sub>1</sub>.

11. In the code segment

   ```java
   Thread t = new Thread( obj );
   ```

   obj must refer to a Runnable target, that is, an object whose class implements the Runnable interface.

12. A program continues to run as long as at least one daemon thread is alive.

13. Java’s synchronized construct ensures progress and mutual exclusion but not fairness.

14. The synchronized construct makes it impossible for a multithreaded program to deadlock.
Chapter 9

1. The DatagramSocket class is associated with the UDP transport protocol, whereas the ServerSocket and Socket classes are associated with the TCP transport protocol.

2. A ServerSocket has an associated input and output stream for communications with clients.

3. Invoking the accept method on a ServerSocket causes a block until a client connects.

4. Serialization over a socket can be enabled by constructing an ObjectOutputStream and an ObjectInputStream from the output and input streams associated with a client socket.

5. An applet operating under sandbox security cannot read from a local file.

6. An applet operating under sandbox security cannot open a network connection to an arbitrary host.

7. Applets operating in the same context can communicate with one another.

8. An RMI client receives a reference to an RMI server rather than a copy of an RMI server.

9. An RMI client may invoke any public method defined in an RMI server.

10. Under RMI, a stub is a proxy for the server and a skeleton is a proxy for a client.

11. RMI activation is a type of Exception.

12. CORBA supports location transparency, whereas RMI does not.
Chapter 10

1. Because the standard Component class implements Serializable, every instance of a class in the Component hierarchy is technically a bean.

2. Under the event-delegation model, one object can be a listener for property change events in another object.

3. The bytecode verifier is a compile-time rather than a run-time utility.

4. The access controller implements security measures through a system of permissions and privileges.

5. Because a message digest is a secure one-way function, the original message from which the digest is generated cannot be reconstructed from the digest itself even if the underlying algorithm is known.

6. A digital signature is an encrypted message digest.

7. In an authentication system based on digital signatures and using private key/public key technology, the sender signs a message using the sender's public key and a receiver verifies the signature using the sender's private key.

8. Every object Obj encapsulates the getClass method, which returns a reference to an object that represents the class that Obj instantiates.

9. In the java.lang.reflect package, instances of the class Method represent both methods and constructors.

10. Java reflection technology can be used to obtain run-time information about method definitions and not simply about method declarations.

11. A servlet, like an applet, typically executes on a client machine.

12. JDBC technology must be used in conjunction with servlets.