**Advanced Java**

1. What is a transient variable?

A transient variable is a variable that may not be serialized.

2. Which containers use a border Layout as their default layout?

The *Window*, *Frame* and *Dialog* classes use a border layout as their default layout.

3. Why do threads block on I/O?

Threads block on I/O (that is enters the waiting state) so that other threads may execute while the I/O Operation is performed.

4. How are Observer and Observable used?

Objects that subclass the *Observable* class maintain a list of observers. When an *Observable* object is updated it invokes the *update()* method of each of its observers to notify the observers that it has changed state. The Observer interface is implemented by objects that observe *Observable* objects.

5. What is synchronization and why is it important?

With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchronization, it is possible for one thread to modify a shared object while another thread is in the process of using or updating that object's value. This often leads to significant errors.

6. Can a lock be acquired on a class?

Yes, a lock can be acquired on a class. This lock is acquired on the class's *Class* object..

7. What's new with the stop(), suspend() and resume() methods in JDK 1.2?

The *stop()*, *suspend()* and *resume()* methods have been deprecated in JDK 1.2.

8. Is null a keyword?

The *null* is not a keyword.

9. What is the preferred size of a component?

The preferred size of a component is the minimum component size that will allow the component to display normally.

10. What method is used to specify a container's layout?

The *setLayout()* method is used to specify a container's layout.

11. Which containers use a FlowLayout as their default layout?

The *Panel* and *Applet* classes use the *FlowLayout* as their default layout.

12. What state does a thread enter when it terminates its processing?

When a thread terminates its processing, it enters the dead state.

13. What is the Collections API?

The Collections API is a set of classes and interfaces that support operations on collections of objects.

14. Which characters may be used as the second character of an identifier, but not as the first character of an identifier?

The digits 0 through 9 may not be used as the first character of an identifier but they may be used after the first character of an identifier.

15. What is the List interface?

The *List* interface provides support for ordered collections of objects.

16. How does Java handle integer overflows and underflows?

It uses those low order bytes of the result that can fit into the size of the type allowed by the operation.

17. What is the Vector class?

The *Vector* class provides the capability to implement a growable array of objects

18. What modifiers may be used with an inner class that is a member of an outer class?

A (non-local) inner class may be declared as *public*, *protected*, *private*, *static*, *final*, or *abstract*.

19. What is an Iterator interface?

The *Iterator* interface is used to step through the elements of a Collection.

20. What is the difference between the >> and >>> operators?

The *>>* operator carries the sign bit when shifting right. The *>>>* zero-fills bits that have been shifted out.

21. Which method of the Component class is used to set the position and size of a component?

*setBounds()* method is used to set the position and size of a component.

22. What is the difference between yielding and sleeping?

When a task invokes its *yield()* method, it returns to the ready state. When a task invokes its *sleep()* method, it returns to the waiting state.

23. Which java.util classes and interfaces support event handling?

The *EventObject* class and the *EventListener* interface support event processing.

24. Is sizeof a keyword?

The ***sizeof* operator** is **not a keyword**.

25. What are wrapped classes?

Wrapped classes are classes that allow primitive types to be accessed as objects.

26. Does garbage collection guarantee that a program will not run out of memory?

Garbage collection does not guarantee that a program will not run out of memory. It is possible for programs to use up memory resources faster than they are garbage collected. It is also possible for programs to create objects that are not subject to garbage collection.

27. What restrictions are placed on the location of a package statement within a source code file?

A package statement must appear as the first line in a source code file (excluding blank lines and comments).

28. Can an object's finalize() method be invoked while it is reachable?

An object's *finalize()* method cannot be invoked by the garbage collector while the object is still reachable. However, an object's *finalize()* method may be invoked by other objects.

29. What is the immediate superclass of the Applet class?

*Panel*.

30. What is the difference between preemptive scheduling and time slicing?

Under preemptive scheduling, the highest priority task executes until it enters the waiting or dead states or a higher priority task comes into existence. Under time slicing, a task executes for a predefined slice of time and then reenters the pool of ready tasks.

The scheduler then determines which task should execute next, based on priority and other factors.

31. Name three Component subclasses that support painting.

The *Canvas*, *Frame*, *Panel*, and *Applet* classes support painting.

32. What value does readLine() return when it has reached the end of a file?

The *readLine()* method returns *null* when it has reached the end of a file.

33. What is the immediate superclass of the Dialog class?

*Window*.

34. What is clipping?

Clipping is the process of confining paint operations to a limited area or shape.

35. What is a native method?

A native method is a method that is implemented in a language other than Java.

36. Can a for statement loop indefinitely?

Yes, a for statement can loop indefinitely. For example, consider the following:

*for(;;) ;*

37. What are order of precedence and associativity, and how are they used?

Order of precedence determines the order in which operators are evaluated in expressions.

Associatity determines whether an expression is evaluated left-to-right or right-to-left.

38. When a thread blocks on I/O, what state does it enter?

A thread enters the waiting state when it blocks on I/O.

39. To what value is a variable of the String type automatically initialized?

The default value of an *String* type is *null*.

40. What is the catch or declare rule for method declarations?

If a checked exception may be thrown within the body of a method, the method must either catch the exception or declare it in its throws clause.

41. What is the difference between a MenuItem and a CheckboxMenuItem?

The *CheckboxMenuItem* class extends the *MenuItem* class to support a menu item that may be checked or unchecked.

42. What is a task's priority and how is it used in scheduling?

A task's priority is an integer value that identifies the relative order in which it should be executed with respect to other tasks. The scheduler attempts to schedule higher priority tasks before lower priority tasks.

43. What class is the top of the AWT event hierarchy?

The *java.awt.AWTEvent* class is the highest-level class in the AWT event-class hierarchy.

44. When a thread is created and started, what is its initial state?

A thread is in the ready state after it has been created and started.

45. Can an anonymous class be declared as implementing an interface and extending a class?

An anonymous class may implement an interface or extend a superclass, but may not be declared to do both.

46. What is the immediate superclass of Menu?

*MenuItem*.

47. What is the purpose of finalization?

The purpose of finalization is to give an unreachable object the opportunity to perform any cleanup processing before the object is garbage collected.

48. Which class is the immediate superclass of the MenuComponent class?

*Object*.

49. What invokes a thread's run() method?

After a thread is started, via its *start()* method or that of the *Thread* class, the JVM invokes the thread's *run()* method when the thread is initially executed.

50. What is the difference between the Boolean & operator and the && operator?

If an expression involving the Boolean *&* operator is evaluated, both operands are evaluated. Then the *&* operator is applied to the operand. When an expression involving the *&&* operator is evaluated, the first operand is evaluated.

If the first operand returns a value of true then the second operand is evaluated. The *&&* operator is then applied to the first and second operands. If the first operand evaluates to false, the evaluation of the second operand is skipped.

51. Name three subclasses of the Component class.

*Box.Filler*, *Button*, *Canvas*, *Checkbox*, *Choice*, *Container*, *Label*, *List*, *Scrollbar*, or *TextComponent*.

52. What is the GregorianCalendar class?

The *GregorianCalendar* class provides support for traditional Western calendars.

53. Which Container method is used to cause a container to be laid out and redisplayed?

*validate()* method is used to cause a container to be laid out and redisplayed.

54. What is the purpose of the Runtime class?

The purpose of the *Runtime* class is to provide access to the Java runtime system.

55. How many times may an object's finalize() method be invoked by the garbage collector?

An object's *finalize()* method may only be invoked once by the garbage collector.

56. What is the purpose of the finally clause of a try-catch-finally statement?

The finally clause is used to provide the capability to execute code no matter whether or not an exception is thrown or caught.

57. What is the argument type of a program's main() method?

A program's *main()* method takes an argument of the *String[]* type.

58. Which Java operator is right associative?

The *=* operator is right associative.

59. Can a double value be cast to a byte?

Yes, a *double* value can be cast to a *byte*.

60. What must a class do to implement an interface?

It must provide all of the methods in the interface and identify the interface in its implements clause.

61. What method is invoked to cause an object to begin executing as a separate thread?

The *start()* method of the *Thread* class is invoked to cause an object to begin executing as a separate thread.

62. Name two subclasses of the TextComponent class.

*TextField* and *TextArea*.

63. Which containers may have a MenuBar?

*Frame*.

64. How are commas used in the intialization and iteration parts of a for statement?

Commas are used to separate multiple statements within the initialization and iteration parts of a *for* statement.

65. What is the purpose of the wait(), notify(), and notifyAll() methods?

The *wait()*, *notify()*, and *notifyAll()* methods are used to provide an efficient way for threads to wait for a shared resource. When a thread executes an object's *wait()* method, it enters the waiting state. It only enters the ready state after another thread invokes the object's *notify()* or *notifyAll()* methods..

66. What is an abstract method?

An abstract method is a method whose implementation is deferred to a subclass.

67. How are Java source code files named?

A Java source code file takes the name of a public class or interface that is defined within the file. A source code file may contain at most one public class or interface. If a public class or interface is defined within a source code file, then the source code file must take the name of the public class or interface.

If no public class or interface is defined within a source code file, then the file must take on a name that is different than its classes and interfaces. Source code files use the *.java* extension.

68. What is the relationship between the Canvas class and the Graphics class?

A *Canvas* object provides access to a Graphics object via its *paint()* method.

69. What are the high-level thread states?

The high-level thread states are ready, running, waiting, and dead.

70. What value does read() return when it has reached the end of a file?

The *read()* method returns *-1* when it has reached the end of a file.

71. Can a Byte object be cast to a double value?

No. An object cannot be cast to a primitive value.

72. What is the difference between a static and a non-static inner class?

A non-static inner class may have object instances that are associated with instances of the class's outer class.

A static inner class does not have any object instances.

73. What is the difference between the String and StringBuffer classes?

*String* objects are constants. *StringBuffer* objects are not constants.

74. If a variable is declared as private, where may the variable be accessed?

A private variable may only be accessed within the class in which it is declared.

75. What is an object's lock and which object's have locks?

An object's lock is a mechanism that is used by multiple threads to obtain synchronized access to the object. A thread may execute a synchronized method of an object only after it has acquired the object's lock.

All objects and classes have locks. A class's lock is acquired on the class's *Class* object.

76. What is the Dictionary class?

The *Dictionary* class provides the capability to store key-value pairs.

77. How are the elements of a BorderLayout organized?

The elements of a *BorderLayout* are organized at the borders (North, South, East, and West) and the center of a container.

78. What is the % operator?

It is referred to as the modulo or remainder operator. It returns the remainder of dividing the first operand by the second operand.

79. When can an object reference be cast to an interface reference?

An object reference be cast to an interface reference when the object implements the referenced interface.

80. What is the difference between a Window and a Frame?

The *Frame* class extends *Window* to define a main application window that can have a menu bar.

81. Which class is extended by all other classes?

The *Object* class is extended by all other classes.

82. Can an object be garbage collected while it is still reachable?

A reachable object cannot be garbage collected. Only unreachable objects may be garbage collected..

83. Is the ternary operator written x : y ? z or x ? y : z ?

It is written *x ? y : z*.

84. What is the difference between the Font and FontMetrics classes?

The *FontMetrics* class is used to define implementation-specific properties, such as ascent and descent, of a *Font* object.

85. How is rounding performed under integer division?

The fractional part of the result is truncated. This is known as rounding toward zero.

86. What happens when a thread cannot acquire a lock on an object?

If a thread attempts to execute a synchronized method or synchronized statement and is unable to acquire an object's lock, it enters the waiting state until the lock becomes available.

87. What is the difference between the Reader/Writer class hierarchy and the InputStream/OutputStream class hierarchy?

The *Reader/Writer* class hierarchy is character-oriented, and the *InputStream/OutputStream* class hierarchy is byte-oriented.

88. What classes of exceptions may be caught by a catch clause?

A catch clause can catch any exception that may be assigned to the Throwable type. This includes the Error and Exception types.

89. If a class is declared without any access modifiers, where may the class be accessed?

A class that is declared without any access modifiers is said to have package access. This means that the class can only be accessed by other classes and interfaces that are defined within the same package.

90. What is the SimpleTimeZone class?

The *SimpleTimeZone* class provides support for a Gregorian calendar.

91. What is the Map interface?

The *Map* interface replaces the JDK 1.1 *Dictionary* class and is used associate keys with values.

92. Does a class inherit the constructors of its superclass?

A class does not inherit constructors from any of its superclasses.

93. For which statements does it make sense to use a label?

The only statements for which it makes sense to use a label are those statements that can enclose a *break* or *continue* statement.

94. What is the purpose of the System class?

The purpose of the *System* class is to provide access to system resources.

95. Which TextComponent method is used to set a TextComponent to the read-only state?

*setEditable()*.

96. How are the elements of a CardLayout organized?

The elements of a *CardLayout* are stacked, one on top of the other, like a deck of cards.

97. Is &&= a valid Java operator?

No. It is not a valid java operator.

98. Name the eight primitive Java types.

The eight primitive types are *byte*, *char*, *short*, *int*, *long*, *float*, *double*, and *boolean*.

99. Which class should you use to obtain design information about an object?

The *Class* class is used to obtain information about an object's design.

100. What is the relationship between clipping and repainting?

When a window is repainted by the AWT painting thread, it sets the clipping regions to the area of the window that requires repainting.

101. Is "abc" a primitive value?

The *String* literal "abc" is not a primitive value. It is a *String* object.

102. What is the relationship between an event-listener interface and an event-adapter class?

An event-listener interface defines the methods that must be implemented by an event handler for a particular kind of event.

An event adapter provides a default implementation of an event-listener interface.

103. What restrictions are placed on the values of each case of a switch statement?

During compilation, the values of each case of a *switch* statement must evaluate to a value that can be promoted to an *int* value.

104. What modifiers may be used with an interface declaration?

An interface may be declared as *public* or *abstract*.

105. Is a class a subclass of itself?

A class is a subclass of itself.

106. What is the highest-level event class of the event-delegation model?

The *java.util.EventObject* class is the highest-level class in the event-delegation class hierarchy.

107. What event results from the clicking of a button?

The *ActionEvent* event is generated as the result of the clicking of a button.

108. How can a GUI component handle its own events?

A component can handle its own events by implementing the required event-listener interface and adding itself as its own event listener.

109. How are the elements of a GridBagLayout organized?

The elements of a *GridBagLayout* are organized according to a grid. However, the elements are of different sizes and may occupy more than one row or column of the grid. In addition, the rows and columns may have different sizes.

110. What advantage do Java's layout managers provide over traditional windowing systems?

Java uses layout managers to lay out components in a consistent manner across all windowing platforms. Since Java's layout managers aren't tied to absolute sizing and positioning, they are able to accomodate platform-specific differences among windowing systems.

111. What is the Collection interface?

The *Collection* interface provides support for the implementation of a mathematical bag - an unordered collection of objects that may contain duplicates.

112. What modifiers can be used with a local inner class?

A local inner class may be *final* or *abstract*.

113. What is the difference between static and non-static variables?

A static variable is associated with the class as a whole rather than with specific instances of a class.

Non-static variables take on unique values with each object instance.

114. What is the difference between the paint() and repaint() methods?

*The paint()* method supports painting via a Graphics object. The *repaint()* method is used to cause *paint()* to be invoked by the AWT painting thread.

115. What is the purpose of the File class?

The *File* class is used to create objects that provide access to the files and directories of a local file system.

116. Can an exception be rethrown?

Yes, an exception can be rethrown.

117. Which Math method is used to calculate the absolute value of a number?

The *abs()* method is used to calculate absolute values.

118. How does multithreading take place on a computer with a single CPU?

The operating system's task scheduler allocates execution time to multiple tasks. By quickly switching between executing tasks, it creates the impression that tasks execute sequentially.

119. When does the compiler supply a default constructor for a class?

The compiler supplies a default constructor for a class if no other constructors are provided.

120. When is the finally clause of a try-catch-finally statement executed?

The finally clause of the try-catch-finally statement is always executed unless the thread of execution terminates or an exception occurs within the execution of the finally clause.

121. Which class is the immediate superclass of the Container class?

*Component*.

122. If a method is declared as protected, where may the method be accessed?

A *protected* method may only be accessed by classes or interfaces of the same package or by subclasses of the class in which it is declared.

123. How can the Checkbox class be used to create a radio button?

By associating Checkbox objects with a *CheckboxGroup*.

124. Which non-Unicode letter characters may be used as the first character of an identifier?

The non-Unicode letter characters *$* and *\_* may appear as the first character of an identifier

125. What restrictions are placed on method overloading?

Two methods may not have the same name and argument list but different return types.

126. What happens when you invoke a thread's interrupt method while it is sleeping or waiting?

When a task's *interrupt()* method is executed, the task enters the ready state. The next time the task enters the running state, an *InterruptedException* is thrown.

127. What is casting?

There are two types of casting, casting between primitive numeric types and casting between object references.

Casting between numeric types is used to convert larger values, such as *double* values, to smaller values, such as *byte* values.

Casting between object references is used to refer to an object by a compatible *class, interface*, or *array* type reference.

128. What is the return type of a program's main() method?

A program's *main()* method has a *void* return type.

129. Name four Container classes.

*Window*, *Frame*, *Dialog*, *FileDialog*, *Panel*, *Applet*, or *ScrollPane*.

130. What is the difference between a Choice and a List?

A Choice is displayed in a compact form that requires you to pull it down to see the list of available choices. Only one item may be selected from a Choice.

A List may be displayed in such a way that several List items are visible. A List supports the selection of one or more List items.

131. What class of exceptions are generated by the Java run-time system?

The Java runtime system generates *RuntimeException* and Error exceptions.

132. What class allows you to read objects directly from a stream?

The *ObjectInputStream* class supports the reading of objects from input streams.

133. What is the difference between a field variable and a local variable?

A field variable is a variable that is declared as a member of a class.

A local variable is a variable that is declared local to a method.

134. Under what conditions is an object's finalize() method invoked by the garbage collector?

The garbage collector invokes an object's *finalize()* method when it detects that the object has become unreachable.

135. What is the relationship between a method's throws clause and the exceptions that can be thrown during the method's execution?

A method's throws clause must declare any checked exceptions that are not caught within the body of the method.

136. What is the difference between the JDK 1.02 event model and the event-delegation model introduced with JDK 1.1?

The JDK 1.02 event model uses an event inheritance or bubbling approach. In this model, components are required to handle their own events. If they do not handle a particular event, the event is inherited by (or bubbled up to) the component's container. The container then either handles the event or it is bubbled up to its container and so on, until the highest-level container has been tried.

In the event-delegation model, specific objects are designated as event handlers for GUI components. These objects implement event-listener interfaces. The event-delegation model is more efficient than the event-inheritance model because it eliminates the processing required to support the bubbling of unhandled events.

137. How is it possible for two String objects with identical values not to be equal under the == operator?

The *==* operator compares two objects to determine if they are the same object in memory. It is possible for two String objects to have the same value, but located indifferent areas of memory.

138. Why are the methods of the Math class static?

So they can be invoked as if they are a mathematical code library.

139. What Checkbox method allows you to tell if a Checkbox is checked?

*getState()*.

140. What state is a thread in when it is executing?

An executing thread is in the running state.

141. What are the legal operands of the instanceof operator?

The left operand is an object reference or null value and the right operand is a *class*, *interface*, or *array* type.

142. How are the elements of a GridBagLayout organized?

The elements of a *GridBagLayout* are of equal size and are laid out using the squares of a grid.

143. What an I/O filter?

An I/O filter is an object that reads from one stream and writes to another, usually altering the data in some way as it is passed from one stream to another.

144. If an object is garbage collected, can it become reachable again?

Once an object is garbage collected, it ceases to exist. It can no longer become reachable again.

145. What is the Set interface?

The *Set* interface provides methods for accessing the elements of a finite mathematical set. Sets do not allow duplicate elements.

146. What classes of exceptions may be thrown by a throw statement?

A *throw* statement may throw any expression that may be assigned to the Throwable type.

147. What are E and PI?

*E* is the base of the natural logarithm and *PI* is mathematical value pi.

148. Are true and false keywords?

The values *true* and *false* are not keywords.

149. What is a void return type?

A *void* return type indicates that a method does not return a value.

150. What is the purpose of the enableEvents() method?

The *enableEvents()* method is used to enable an event for a particular object. Normally, an event is enabled when a listener is added to an object for a particular event.

The *enableEvents()* method is used by objects that handle events by overriding their event-dispatch methods.

151. What is the difference between the File and RandomAccessFile classes?

The *File* class encapsulates the files and directories of the local file system.

The *RandomAccessFile* class provides the methods needed to directly access data contained in any part of a file.

152. What happens when you add a double value to a String?

The result is a *String* object.

153. What is your platform's default character encoding?

If you are running Java on English Windows platforms, it is probably Cp1252. If you are running Java on English Solaris platforms, it is most likely 8859\_1.

154. Which package is always imported by default?

The *java.lang package* is always imported by default.

155. What interface must an object implement before it can be written to a stream as an object?

An object must implement the *Serializable* or *Externalizable* interface before it can be written to a stream as an object.

156. How are this and super used?

*this* is used to refer to the current object instance.

*super* is used to refer to the variables and methods of the superclass of the current object instance.

157. What is a compilation unit?

A compilation unit is a Java source code file.

158. What interface is extended by AWT event listeners?

All AWT event listeners extend the *java.util.EventListener* interface.

159. What restrictions are placed on method overriding?

Overridden methods must have the same name, argument list, and return type.

The overriding method may not limit the access of the method it overrides.

The overriding method may not throw any exceptions that may not be thrown by the overridden method.

160. How can a dead thread be restarted?

A dead thread cannot be restarted.

161. What happens if an exception is not caught?

An uncaught exception results in the *uncaughtException()* method of the thread's *ThreadGroup* being invoked, which eventually results in the termination of the program in which it is thrown.

162. What is a layout manager?

A layout manager is an object that is used to organize components in a container.

163. Which arithmetic operations can result in the throwing of an ArithmeticException?

Integer */* and *%* can result in the throwing of an *ArithmeticException*.

164. What are three ways in which a thread can enter the waiting state?

A thread can enter the waiting state by invoking its *sleep()* method, by blocking on I/O, by unsuccessfully attempting to acquire an object's lock, or by invoking an object's *wait()* method. It can also enter the waiting state by invoking its (deprecated) *suspend()* method.

165. Can an abstract class be final?

An *abstract* class may not be declared as *final*.

166. What is the ResourceBundle class?

The *ResourceBundle* class is used to store locale-specific resources that can be loaded by a program to tailor the program's appearance to the particular locale in which it is being run.

167. What happens if a try-catch-finally statement does not have a catch clause to handle an exception that is thrown within the body of the try statement?

The exception propagates up to the next higher level try-catch statement (if any) or results in the program's termination.

168. What is numeric promotion?

Numeric promotion is the conversion of a smaller numeric type to a larger numeric type, so that integer and floating-point operations may take place. In numerical promotion, *byte*, *char*, and *short* values are converted to *int* values. The int values are also converted to *long* values, if necessary. The *long* and *float* values are converted to *double* values, as required.

169. What is the difference between a Scrollbar and a ScrollPane?

A *Scrollbar* is a Component, but not a Container. A *ScrollPane* is a Container.

A *ScrollPane* handles its own events and performs its own scrolling.

170. What is the difference between a public and a non-public class?

A public class may be accessed outside of its package.

A non-public class may not be accessed outside of its package.

171. To what value is a variable of the boolean type automatically initialized?

The default value of the *boolean* type is *false*.

172. Can try statements be nested?

*Try* statements may be tested.

173. What is the difference between the prefix and postfix forms of the ++ operator?

The prefix form performs the increment operation and returns the value of the increment operation.

The postfix form returns the current value all of the expression and then performs the increment operation on that value.

174. What is the purpose of a statement block?

A statement block is used to organize a sequence of statements as a single statement group.

175. What is a Java package and how is it used?

A Java package is a naming context for classes and interfaces. A package is used to create a separate name space for groups of classes and interfaces.

Packages are also used to organize related classes and interfaces into a single API unit and to control accessibility to these classes and interfaces.

176. What modifiers may be used with a top-level class?

A top-level class may be *public*, *abstract*, or *final*.

177. What are the Object and Class classes used for?

The *Object* class is the highest-level class in the Java class hierarchy. The *Class* class is used to represent the classes and interfaces that are loaded by a Java program..

178. How does a try statement determine which catch clause should be used to handle an exception?

When an exception is thrown within the body of a *try* statement, the catch clauses of the *try* statement are examined in the order in which they appear. The first *catch* clause that is capable of handling the exception is executed. The remaining *catch* clauses are ignored.

179. Can an unreachable object become reachable again?

An unreachable object may become reachable again. This can happen when the object's *finalize()* method is invoked and the object performs an operation which causes it to become accessible to reachable objects.

180. When is an object subject to garbage collection?

An object is subject to garbage collection when it becomes unreachable to the program in which it is used.

181. What method must be implemented by all threads?

All tasks must implement the *run()* method, whether they are a subclass of *Thread* or implement the *Runnable* interface.

182. What methods are used to get and set the text label displayed by a Button object?

*getLabel()* and *setLabel()*.

183. Which Component subclass is used for drawing and painting?

*Canvas*.

184. What are synchronized methods and synchronized statements?

Synchronized methods are methods that are used to control access to an object. A thread only executes a synchronized method after it has acquired the lock for the method's object or class.

Synchronized statements are similar to synchronized methods. A synchronized statement can only be executed after a thread has acquired the lock for the object or class referenced in the synchronized statement.

185. What are the two basic ways in which classes that can be run as threads may be defined?

A thread class may be declared as a subclass of *Thread*, or it may implement the *Runnable* interface.

186. What are the problems faced by Java programmers who don't use layout managers?

Without layout managers, Java programmers are faced with determining how their GUI will be displayed across multiple windowing systems and finding a common sizing and positioning that will work within the constraints imposed by each windowing system.