Java Language I.M.P.Questions

1. What is a platform?

   A platform is the hardware or software environment in which a program runs. Most platforms can be described as a combination of the operating system and hardware, like Windows 2000/XP, Linux, Solaris, and MacOS.

2. What is the main difference between Java platform and other platforms?

   The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

   The Java platform has two components:

   1. The Java Virtual Machine (Java VM)
   2. The Java Application Programming Interface (Java API)

3. What is the Java Virtual Machine?

   The Java Virtual Machine is a software that can be ported onto various hardware-based platforms.

4. What is the Java API?

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets.

5. What is the package?

   The package is a Java namespace or part of Java libraries. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages.

6. What is native code?

   The native code is code that after you compile it, the compiled code runs on a specific hardware platform.

7. Is Java code slower than native code?

   Not really. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time bytecode compilers can bring performance close to that of native code without threatening portability.

8. What is the serialization?

   The serialization is a kind of mechanism that makes a class or a bean persistence by having its properties or fields and state information saved and restored to and from storage.

9. How to make a class or a bean serializable?

By implementing either the java.io.Serializable interface, or the java.io.Externalizable interface. As long as one class in a class's inheritance hierarchy implements Serializable or Externalizable, that class is serializable.

---

10. How many methods in the Serializable interface?

There is no method in the Serializable interface. The Serializable interface acts as a marker, telling the object serialization tools that your class is serializable.

---

11. How many methods in the Externalizable interface?

There are two methods in the Externalizable interface. You have to implement these two methods in order to make your class externalizable. These two methods are readExternal() and writeExternal().

---

12. What is the difference between Serializalble and Externalizable interface?

When you use Serializable interface, your class is serialized automatically by default. But you can override writeObject() and readObject() two methods to control more complex object serailization process. When you use Externalizable interface, you have a complete control over your class's serialization process.

---

13. What is a transient variable?

A transient variable is a variable that may not be serialized. If you don't want some field to be serialized, you can mark that field transient or static.

---

14. Which containers use a border layout as their default layout?

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

---

15. 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.

---

16. 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 causes dirty data and leads to significant errors.

---

17. 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.

---

18. 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.

---

19. 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.

---

20. 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.

21. 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.

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

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

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

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

24. What is thread?

A thread is an independent path of execution in a system.

25. What is multithreading?

Multithreading means various threads that run in a system.

26. 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.

---

27. How to create multithread in a program?

You have two ways to do so. First, making your class "extends" Thread class. Second, making your class "implements" Runnable interface. Put jobs in a run() method and call start() method to start the thread.

---

28. Can Java object be locked down for exclusive use by a given thread?

Yes. You can lock an object by putting it in a "synchronized" block. The locked object is inaccessible to any thread other than the one that explicitly claimed it.

---

29. Can each Java object keep track of all the threads that want to exclusively access to it?

Yes.

---

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

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

---

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

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

---

32. 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 communicate each other.

---

33. What are the high-level thread states?

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

---

34. What is the Collections API?

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

---

35. What is the List interface?

The List interface provides support for ordered collections of objects.

---

36. 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.

---

37. What is the Vector class?

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

---

38. 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.

---

39. 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.

---

40. What is an Iterator interface?

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

---

41. How many bits are used to represent Unicode, ASCII, UTF-16, and UTF-8 characters?

Unicode requires 16 bits and ASCII require 7 bits. Although the ASCII character set uses only 7 bits, it is usually represented as 8 bits. UTF-8 represents characters using 8, 16, and 18 bit patterns. UTF-16 uses 16-bit and larger bit patterns.

---

42. 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.

---

43. Is sizeof a keyword?

The sizeof operator is not a keyword.

---

44. What are wrapped classes?

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

---

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

No, it doesn't. 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

---

46. 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.

---

47. Name Component subclasses that support painting.

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

---

48. What is a native method?

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

---

49. How can you write a loop indefinitely?

for(;;)--for loop; while(true)--always true, etc.

---

50. 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.

---

51. 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.

---

52. Which class is the superclass for every class.

Object

---

53. 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.

Operator & has no chance to skip both sides evaluation and && operator does. If asked why, give details as above.

---

54. What is the GregorianCalendar class?

The GregorianCalendar provides support for traditional Western calendars.

---

55. What is the SimpleTimeZone class?

The SimpleTimeZone class provides support for a Gregorian calendar.

---

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

validate()

---

57. What is the Properties class?

The properties class is a subclass of Hashtable that can be read from or written to a stream. It also provides the capability to specify a set of default values to be used.

---

58. What is the purpose of the Runtime class?

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

59. What is the purpose of the System class?

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

60. 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.

61. What is the Locale class?

The Locale class is used to tailor program output to the conventions of a particular geographic, political, or cultural region.

62. 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.

63. What is an abstract method?

An abstract method is a method whose implementation is deferred to a subclass. Or, a method that has no implementation (an interface of a method).

64. What is a static method?

A static method is a method that belongs to the class rather than any object of the class and doesn't apply to an object or even require that any objects of the class have been instantiated.

65. What is a protected method?

A protected method is a method that can be accessed by any method in its package and inherited by any subclass of its class.

66. 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.

67. 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.

68. 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.

---

69. 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.

---

70. What do heavy weight components mean?

Heavy weight components like Abstract Window Toolkit (AWT), depend on the local windowing toolkit. For example, java.awt.Button is a heavy weight component, when it is running on the Java platform for Unix platform, it maps to a real Motif button. In this relationship, the Motif button is called the peer to the java.awt.Button. If you create two Buttons, two peers and hence two Motif Buttons are also created. The Java platform communicates with the Motif Buttons using the Java Native Interface. For each and every component added to the application, there is an additional overhead tied to the local windowing system, which is why these components are called heavy weight.

---

71. Which package has light weight components?

javax.Swing package. All components in Swing, except JApplet, JDialog, JFrame and JWindow are lightweight components.

72. What are peerless components?

The peerless components are called light weight components.

73. 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.

74. 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.

75. 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.

76. 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.

---

77. What is the difference between throw and throws keywords?

The throw keyword denotes a statement that causes an exception to be initiated. It takes the Exception object to be thrown as argument. The exception will be caught by an immediately encompassing try-catch construction or propagated further up the calling hierarchy.

The throws keyword is a modifier of a method that designates that exceptions may come out of the mehtod, either by virtue of the method throwing the exception itself or because it fails to catch such exceptions that a method it calls may throw.

---

78. 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 or friendly access. This means that the class can only be accessed by other classes and interfaces that are defined within the same package.

---

79. What is the Map interface?

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

---

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

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

---

81. Name primitive Java types.

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

---

82. 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.

---

83. 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.

---

84. 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.

---

85. 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 accommodate platform-specific differences among windowing systems.

---

86. 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.

---

87. 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.

---

88. 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.

---

89. 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.

---

90. What restrictions are placed on method overloading?

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

---

91. 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.

---

92. 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.

---

93. Name Container classes.

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

---

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

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

---

95. How are this() and super() used with constructors?

this() is used to invoke a constructor of the same class. super() is used to invoke a superclass constructor.

---

96. 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.

---

97. 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.

---

98. 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.

---

99. What is the List interface?

The List interface provides support for ordered collections of objects.

---

100. 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.

---

101. 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.

---

102. 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.

---

103. 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.

---

104. 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.

---

105. 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.

---

106. 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.

---

107. What is Serialization and deserialization?

Serialization is the process of writing the state of an object to a byte stream.
Deserialization is the process of restoring these objects.

---

108. what is tunnelling?

Tunnelling is a route to somewhere. For example, RMI tunnelling is a way to make RMI application get through firewall. In CS world, tunnelling means a way to transfer data.

---

109. Does the code in finally block get executed if there is an exception and a return statement in a catch block?

If an exception occurs and there is a return statement in catch block, the finally block is still executed. The finally block will not be executed when the System.exit(1) statement is executed earlier or the system shut down earlier

or the memory is used up earlier before the thread goes to finally block.

---

110. How you restrict a user to cut and paste from the html page?

Using javaScript to lock keyboard keys. It is one of solutions.

---

111. Is Java a super set of JavaScript?

No. They are completely different. Some syntax may be similar.

---

112. What is a Container in a GUI?

A Container contains and arranges other components (including other containers) through the use of layout managers, which use specific layout policies to determine where components should go as a function of the size of the container.

---

113. How the object oriented approach helps us keep complexity of software development under control?

We can discuss such issue from the following aspects:

- Objects allow procedures to be encapsulated with their data to reduce potential interference.

- Inheritance allows well-tested procedures to be reused and enables changes to make once and have effect in all relevant places.
- The well-defined separations of interface and implementation allows constraints to be imposed on inheriting classes while still allowing the flexibility of overriding and overloading.

---

114. What is polymorphism?

Polymorphism allows methods to be written that needn't be concerned about the specifics of the objects they will be applied to. That is, the method can be specified at a higher level of abstraction and can be counted on to work even on objects of yet unconceived classes.

---

115. What is design by contract?

The design by contract specifies the obligations of a method to any other methods that may use its services and also theirs to it. For example, the preconditions specify what the method required to be true when the method is called. Hence making sure that preconditions are. Similarly, postconditions specify what must be true when the method is finished, thus the called method has the responsibility of satisfying the post conditions.

In Java, the exception handling facilities support the use of design by contract, especially in the case of checked exceptions. The assert keyword can be used to make such contracts.

---

116. What are use cases?

A use case describes a situation that a program might encounter and what behavior the program should exhibit in that circumstance. It is part of the analysis of a program. The collection of use cases should, ideally, anticipate all the standard circumstances and many of the extraordinary circumstances possible so that the program will be robust.

---

117. What is the difference between interface and abstract class?

- interface contains methods that must be abstract; abstract class may contain concrete methods.
- interface contains variables that must be static and final; abstract class may contain non-final and final variables.
- members in an interface are public by default, abstract class may contain non-public members.
- interface is used to "implements"; whereas abstract class is used to "extends".
- interface can be used to achieve multiple inheritance; abstract class can be used as a single inheritance.
- interface can "extends" another interface, abstract class can "extends" another class and "implements" multiple interfaces.
- interface is absolutely abstract; abstract class can be invoked if a main() exists.
- interface is more flexible than abstract class because one class can only "extends" one super class, but "implements" multiple interfaces.
- If given a choice, use interface instead of abstract class.

1. What is the diffrence between an Abstract class and Interface ?
2. What is user defined exception ?

3. What do you know about the garbate collector ?
4. What is the difference between C++ & Java ?
5. Explain RMI Architecture?
6. How do you communicate in between Applets & Servlets ?
7. What is the use of Servlets ?
8. What is JDBC? How do you connect to the Database ?
9. In an HTML form I have a Button which makes us to open another page in 15 seconds. How will do you that ?
10. What is the difference between Process and Threads ?
11. What is the difference between RMI & Corba ?
12. What are the services in RMI ?
13. How will you initialize an Applet ?
14. What is the order of method invocation in an Applet ?
15. When is update method called ?
16. How will you pass values from HTML page to the Servlet ?
17. Have you ever used HashTable and Dictionary ?
18. How will you communicate between two Applets ?
19. What are statements in JAVA ?
20. What is JAR file ?
21. What is JNI ?
22. What is the base class for all swing components ?
23. What is JFC ?
24. What is Difference between AWT and Swing ?
25. Considering notepad/IE or any other thing as process, What will happen if you start notepad or IE 3 times? Where 3 processes are started or 3 threads are started ?
26. How does thread synchronization occurs inside a monitor ?
27. How will you call an Applet using a Java Script function ?
28. Is there any tag in HTML to upload and download files ?
29. Why do you Canvas ?
30. How can you push data from an Applet to Servlet ?

31. What are 4 drivers available in JDBC ?
32. How you can know about drivers and database information ?
33. If you are truncated using JDBC, How can you know ..that how much data is truncated ?
34. And What situation , each of the 4 drivers used ?
35. How will you perform transaction using JDBC ?
36. In RMI, server object first loaded into the memory and then the stub reference is sent to the client ? or whether a stub reference is directly sent to the client ?
37. Suppose server object is not loaded into the memory, and the client request for it , what will happen?
38. What is serialization ?
39. Can you load the server object dynamically? If so, what are the major 3 steps involved in it ?
40. What is difference RMI registry and OSAgent ?
41. To a server method, the client wants to send a value 20, with this value exceeds to 20,. a message should be sent to the client ? What will you do for achieving for this ?
42. What are the benefits of Swing over AWT ?
43. Where the CardLayout is used ?
44. What is the Layout for ToolBar ?
45. What is the difference between Grid and GridbagLayout ?
46. How will you add panel to a Frame ?
47. What is the corresponding Layout for Card in Swing ?
48. What is light weight component ?
49. Can you run the product development on all operating systems ?
50. What is the webserver used for running the Servlets ?
51. What is Servlet API used for conneting database ?
52. What is bean ? Where it can be used ?
53. What is difference in between Java Class and Bean ?
54. Can we send object using Sockets ?
55. What is the RMI and Socket ?

56. How to communicate 2 threads each other ?
57. What are the files generated after using IDL to Java Compilet ?
58. What is the protocol used by server and client ?
59. Can I modify an object in CORBA ?
60. What is the functionality stubs and skeletons ?
61. What is the mapping mechanism used by Java to identify IDL language ?
62. Diff between Application and Applet ?
63. What is serializable Interface ?
64. What is the difference between CGI and Servlet ?
65. What is the use of Interface ?
66. Why Java is not fully objective oriented ?
67. Why does not support multiple Inheritance ?
68. What it the root class for all Java classes ?
69. What is polymorphism ?
70. Suppose If we have variable ' I ' in run method, If I can create one or more thread each thread will occupy a separate copy or same variable will be shared ?
71. In servlets, we are having a web page that is invoking servlets username and password ? which is cheks in the database ? Suppose the second page also If we want to verify the same information whethe it will connect to the database or it will be used previous information?
72. What are virtual functions ?
73. Write down how will you create a binary Tree ?
74. What are the traverses in Binary Tree ?
75. Write a program for recursive Traverse ?
76. What are session variable in Servlets ?
77. What is client server computing ?
78. What is Constructor and Virtual function? Can we call Virtual funciton in a constructor ?
79. Why we use OOPS concepts? What is its advantage ?
80. What is the middleware ? What is the functionality of Webserver ?

81. Why Java is not 100 % pure OOPS ? (EcomServer)
82. When we will use an Interface and Abstract class ?
83. What is an RMI?
84. How will you pass parameters in RMI ? Why u serialize?
85. What is the exact difference in between Unicast and Multicast object ? Where we will use ?
86. What is the main functionality of the Remote Reference Layer ?
87. How do you download stubs from a Remote place ?
88. What is the difference in between C++ and Java ? can u explain in detail ?
89. I want to store more than 10 objects in a remote server ? Which methodology will follow ?
90. What is the main functionality of the Prepared Statement ?
91. What is meant by static query and dynamic query ?
92. What are the Normalization Rules ? Define the Normalization ?
93. What is meant by Servelet? What are the parameters of the service method ?
94. What is meant by Session ? Tell me something about HTTPSession Class ?
95. How do you invoke a Servelt? What is the difference in between doPost and doGet methods ?
96. What is the difference in between the HTTPServlet and Generic Servlet ? Expalin their methods ? Tell me their parameter names also ?
97. Have you used threads in Servelet ?
98. Write a program on RMI and JDBC using StoredProcedure ?
99. How do you sing an Applet ?
100. In a Container there are 5 components. I want to display the all the components names, how will you do that one ?

101. Why there are some null interface in java ? What does it mean ? Give me some null interfaces in JAVA ?
102. Tell me the latest versions in JAVA related areas ?
103. What is meant by class loader ? How many types are there? When will we use them ?
104. How do you load an Image in a Servlet ?
105. What is meant by flickering ?
106. What is meant by distributed Application ? Why we are using that in our applications ?
107. What is the functionality of the stub ?
108. Have you used any version control ?
109. What is the latest version of JDBC ? What are the new features are added in that ?
110. Explain 2 tier and 3 -tier Architecture ?
111. What is the role of the webserver ?
112. How have you done validation of the fileds in your project ?
113. What is the main difficulties that you are faced in your project ?
114. What is meant by cookies ? Explain ?
115. Problem faced in your earlier project
116. How OOPS concept is achieved in Java
117. Features for using Java
118. How does Java 2.0 differ from Java 1.0
119. Public static void main – Explain
120. What are command line arguments
121. Explain about the three-tier model
122. Difference between String & StringBuffer
123. Wrapper class. Is String a Wrapper Class
124. What are the restriction for static method
125. Purpose of the file class
126. Default modifier in Interface
127. Difference between Interface & Abstract class
128. Can abstract be declared as Final

129. Can we declare variables inside a method as Final Variables
130. What is the package concept and use of package
131. How can a dead thread be started
132. Difference between Applet & Application
133. Life cycle of the Applet
134. Can Applet have constructors
135. Differeence between canvas class & graphics class
136. Explain about Superclass & subclass
137. Difference between TCP & UDP
138. What is AppletStub
139. Explain Stream Tokenizer
140. What is the difference between two types of threads
141. Checked & Unchecked exception
142. Use of throws exception
143. What is finally in exception handling
144. Vector class
145. What will happen to the Exception object after exception handling
146. Two types of multi-tasking
147. Two ways to create the thread
148. Synchronization
149. I/O Filter
150. How can you retrieve warnings in JDBC
151. Can applet in different page communicate with each other
152. Four driver Manager
153. Features of JDBC 20
154. Explain about stored procedures
155. Servlet Life cycle
156. Why do you go for servlet rather than CGI
157. How to generate skeleton & Stub classes
158. Explain lazy activation
159. Firewalls in RMI

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 value 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()

22. How many bits are used to represent Unicode, ASCII, UTF-16, and UTF-8 characters?

Unicode requires 16 bits and ASCII require 7 bits. Although the ASCII character set uses only 7 bits, it is usually represented as 8 bits. UTF-8 represents characters using 8, 16, and 18 bit patterns. UTF-16 uses 16-bit and larger bit patterns.

23What 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.

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

The EventObject class and the EventListener interface support event processing.

25. Is sizeof a keyword?

The sizeof operator is not a keyword.

26. What are wrapped classes?

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

27. 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

28. 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).

29. 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.

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

Panel

31. 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.

32. Name three Component subclasses that support painting.

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

33. 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.

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

Window

35. What is clipping?

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

36. What is a native method?

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

37. Can a for statement loop indefinitely?

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

for(;;) ;

38. 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

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

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

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

The default value of an String type is null.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

46. 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.

47. What is the range of the short type?

The range of the short type is -(2^15) to 2^15 - 1.

48. What is the range of the char type?

The range of the char type is 0 to 2^16 - 1.

49. In which package are most of the AWT events that support the event-delegation

model defined?

Most of the AWT-related events of the event-delegation model are defined in the java.awt.event package. The AWTEvent class is defined in the java.awt package.

50. What is the immediate superclass of Menu?

MenuItem

51. 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.

52. Which class is the immediate superclass of the MenuComponent class.

Object

53. 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.

54. 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.

55. Name three subclasses of the Component class.

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

56. What is the GregorianCalendar class?

The GregorianCalendar provides support for traditional Western calendars.

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

validate()

58. What is the purpose of the Runtime class?

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

59. 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.

60. 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.

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

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

62. Which Java operator is right associative?

The = operator is right associative.

63. What is the Locale class?

The Locale class is used to tailor program output to the conventions of a particular geographic, political, or cultural region.

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

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

65. What is the difference between a break statement and a continue statement?

A break statement results in the termination of the statement to which it applies (switch, for, do, or while). A continue statement is used to end the current loop iteration and return control to the loop statement.

66. 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.

67. 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.

68. Name two subclasses of the TextComponent class.

TextField and TextArea

69. What is the advantage of the event-delegation model over the earlier event-inheritance model?

The event-delegation model has two advantages over the event-inheritance model. First, it enables event handling to be handled by objects other than the ones that generate the events (or their containers). This allows a clean separation

between a component's design and its use. The other advantage of the event-delegation model is that it performs much better in applications where many events are generated. This performance improvement is due to the fact that the event-delegation model does not have to repeatedly process unhandled events, as is the case of the event-inheritance

model.

70. Which containers may have a MenuBar?

Frame

71. How are commas used in the intialization and iterationparts of a for statement?

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

72. 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.

73. What is an abstract method?

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

74. 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.

75. 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.

76. What are the high-level thread states?

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

77. 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.

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

No, an object cannot be cast to a primitive value.

79. 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.

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

String objects are constants. StringBuffer objects are not.

81. 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.

82. 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.

83. What is the Dictionary class?

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

84. 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.

85. 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.

86. 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.

87. 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.

88. Which class is extended by all other classes?

The Object class is extended by all other classes.

89. 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..

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

It is written x ? y : z.

91. 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.

92. How is rounding performed under integer division?

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

93. 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.

94. 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.

95. 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.

96. 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.

97. What is the SimpleTimeZone class?

The SimpleTimeZone class provides support for a Gregorian calendar.

98. What is the Map interface?

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

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

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

100. 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.

101. What is the purpose of the System class?

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

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

setEditable()

103. 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.

104. Is &&= a valid Java operator?

No, it is not.

105. Name the eight primitive Java types.

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

106. 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.

107. 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.

108. Is "abc" a primitive value?

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

109. 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.

110. 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.

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

An interface may be declared as public or abstract.

112. Is a class a subclass of itself?

A class is a subclass of itself.

113. 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.

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

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

115. 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.

116. What is the difference between a while statement and a dostatement?

A while statement checks at the beginning of a loop to see whether the next loop iteration should occur. A do statement checks at the end of a loop to see whether the next iteration of a loop should occur. The do statement will

always execute the body of a loop at least once.

117. 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.

118. 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.

119. 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.

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

A local inner class may be final or abstract.

121. 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.

122. 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.

123. 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.

124. Can an exception be rethrown?

Yes, an exception can be rethrown.

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

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

126. 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.

127. 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.

128. 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.

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

Component

130. 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.

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

By associating Checkbox objects with a CheckboxGroup.

132. 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

133. What restrictions are placed on method overloading?

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

134. 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.

135. 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.

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

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

137. Name four Container classes.

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

138. 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.

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

The Java runtime system generates RuntimeException and Error exceptions.

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

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

141. 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.

142. 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.

143. How are this() and super() used with constructors?

this() is used to invoke a constructor of the same class. super() is used to invoke a superclass constructor.

144. 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.

145. 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.

146. 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.

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

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

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

getState()

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

An executing thread is in the running state.

150. 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.

151. How are the elements of a GridLayout organized?

The elements of a GridBad layout are of equal size and are laid out using the squares of a grid.

152. 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.

153. 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.

154. 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.

155. 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.

156. What are E and PI?

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

157. Are true and false keywords?

The values true and false are not keywords.

158. What is a void return type?

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

159. 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.

160. 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.

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

The result is a String object.

162. 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..

163. Which package is always imported by default?

The java.lang package is always imported by default.

164. 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.

165. 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.

166. What is the purpose of garbage collection?

The purpose of garbage collection is to identify and discard objects that are no longer needed by a program so that their resources may be reclaimed and

reused.

167. What is a compilation unit?

A compilation unit is a Java source code file.

168. What interface is extended by AWT event listeners?

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

169. 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.

170. How can a dead thread be restarted?

A dead thread cannot be restarted.

171. 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.

172. What is a layout manager?

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

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

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

174. 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.

175. Can an abstract class be final?

An abstract class may not be declared as final.

176. 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.

177. 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.

178. 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.

179. 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.

180. 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.

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

The default value of the boolean type is false.

182. Can try statements be nested?

Try statements may be tested.

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

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

184. What is the purpose of a statement block?

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

185. 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.

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

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

187. 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.

188. 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.

189. 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.

190. 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.

191. What method must be implemented by all threads?

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

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

getLabel() and setLabel()

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

Canvas

194. 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.

195. 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.

196. 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 sizingand positioning that will work within the constraints imposed by each windowing system.

197. What is the difference between an if statement and a switch statement?

The if statement is used to select among two alternatives. It uses a boolean expression to decide which alternative should be executed. The switch statement is used to select among multiple alternatives. It uses an int expression to determine which alternative should be executed.

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

The result is a String object.

199. What is the List interface?

The List interface provides support for ordered collections of objects.

Coding Interview on Database - JDBC (java.sql)

Connecting to a Database
This example uses the JDBC-ODBC bridge to connect to a database called "mydatabase".

```
try {

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

String url = "jdbc:odbc:mydatabase";

Connection con = DriverManager.getConnection(

url, "login", "password");

} catch (ClassNotFoundException e) {

} catch (SQLException e) {

}
```

Creating a Table

This example creates a table called "mytable" with three columns: COL_A which holds strings, COL_B which holds integers, and COL_C which holds floating point numbers.

```
try {

Statement stmt = con.createStatement();

stmt.executeUpdate("CREATE TABLE mytable (
```

```
COL_A VARCHAR(100), COL_B INTEGER, COL_C
FLOAT)");

} catch (SQLException e) {

}
```

Entering a New Row into a Table

This example enters a row containing a string, an integer, and a floating point number into the table called "mytable".

```
try {

Statement stmt = connection.createStatement();

stmt.executeUpdate("INSERT INTO mytable

VALUES ('Patrick Chan', 123, 1.23)");

connection.close();

} catch (SQLException e) {

}
```

Getting All Rows from a Table

This example retrieves all the rows from a table called "mytable". A row in "mytable" consists of a string, integer, and floating point number.

```
try {

Statement stmt = connection.createStatement();

// Get data using colunm names.

ResultSet rs = stmt.executeQuery(
```

```
"SELECT * FROM mytable");
while (rs.next()) {
String s = rs.getString("COL_A");
int i = rs.getInt("COL_B");
float f = rs.getFloat("COL_C");
process(s, i, f);
}
// Get data using colunm numbers.
rs = stmt.executeQuery(
"SELECT * FROM mytable");
while (rs.next()) {
String s = rs.getString(1);
int i = rs.getInt(2);
float f = rs.getFloat(3);
process(s, i, f);
}
} catch (SQLException e) {
}
```

Getting Particular Rows from a Table

This example retrieves all rows from a table called "mytable" whose column COL_A equals ``Patrick Chan". A row in "mytable" consists of a string, integer, and floating point number.

```
try {

Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery(

"SELECT * FROM mytable WHERE COL_A = 'Patrick Chan'");

rs.next();

String s = rs.getString("COL_A");

int i = rs.getInt("COL_B");

float f = rs.getFloat("COL_C");

process(s, i, f);

} catch (SQLException e) {

}
```

Updating a Row of Data in a Table

This example updates a row in a table called ``mytable". In particular, for all rows whose column COL_B equals 123, column COL_A is set to "John Doe".

```
try {

Statement stmt = connection.createStatement();

int numUpdated = stmt.executeUpdate(

"UPDATE mytable SET COL_A = 'John Doe'
```

```
WHERE COL_B = 123");

connection.close();

} catch (SQLException e) {

}
```

Using a Prepared Statement

A prepared statement should be used in cases where a particular SQL statement is used frequently. The prepared statement is more expensive to set up but executes faster than a statement. This example demonstrates a prepared statement for getting all rows from a table called "mytable" whose column COL_A equals "Patrick Chan". This example also demonstrates a prepared statement for updating data in the table. In particular, for all rows whose column COL_B equals 123, column COL_A is set to "John Doe".

```
try {

// Retrieving rows from the database.

PreparedStatement stmt = connection.prepareStatement(

"SELECT * FROM mytable WHERE COL_A = ?");

int columm = 1;

stmt.setString(columm, "Patrick Chan");

ResultSet rs = stmt.executeQuery();

// Updating the database.

stmt = connection.prepareStatement(
```

"UPDATE mytable SET COL_A = ? WHERE COL_B = ?");

columm = 1;

stmt.setString(columm, "John Doe");

columm = 2;

stmt.setInt(columm, 123);

int numUpdated = stmt.executeUpdate();

} catch (SQLException e) {

}

Examplets ™ provided by permission of the publisher, Addision-Wesley, and Author Patrick Chan.

Handling Events with an Anonymous Class

If an event handler is specific to a component (that is, not shared by other components), there is no need to declare a class to handle the event. The event handler can be implemented using an anonymous inner class. This example demonstrates an anonymous inner class to handle key events for a component.

component.addKeyListener(new KeyAdapter() {

public void keyPressed(KeyEvent evt) {

}

});

Handling Action Events

Action events are fired by subclasses of AbstractButton and includes buttons, checkboxes, and menus.

AbstractButton button = new JButton(quot;OK");

button.addActionListener(new MyActionListener());

public class MyActionListener

implements ActionListener {

public void actionPerformed(ActionEvent evt) {

// Determine which abstract

// button fired the event.

AbstractButton button =

(AbstractButton)evt.getSource();

}

}

Handling Key Presses

You can get the key that was pressed either as a key character (which is a Unicode character) or as a key code (a special value representing a particular key on the keyboard).

component.addKeyListener(new MyKeyListener());

public class MyKeyListener extends KeyAdapter {

public void keyPressed(KeyEvent evt) {

// Check for key characters.

```
if (evt.getKeyChar() == 'a') {
process(evt.getKeyChar());
}
// Check for key codes.
if (evt.getKeyCode() == KeyEvent.VK_HOME) {
process(evt.getKeyCode());
}
}
}
```

Handling Mouse Clicks

```
component.addMouseListener(
new MyMouseListener());
public class MyMouseListener
extends MouseAdapter {
public void mouseClicked(MouseEvent evt) {
if ((evt.getModifiers() &
InputEvent.BUTTON1_MASK) != 0) {
processLeft(evt.getPoint());
}
if ((evt.getModifiers() &
```

```
InputEvent.BUTTON2_MASK) != 0) {
processMiddle(evt.getPoint());
}
if ((evt.getModifiers() &
InputEvent.BUTTON3_MASK) != 0) {
processRight(evt.getPoint());
}
}
}
```

Handling Mouse Motion

```
component.addMouseMotionListener(
new MyMouseMotionListener());
public class MyMouseMotionListener
extends MouseMotionAdapter {
public void mouseMoved(MouseEvent evt) {
// Process current position of cursor
// while all mouse buttons are up.
process(evt.getPoint());
}
public void mouseDragged(MouseEvent evt) {
```

```
// Process current position of cursor
// while mouse button is pressed.
process(evt.getPoint());
}
}
```

Detecting Double and Triple Clicks

```
component.addMouseListener(
new MyMouseListener());
public class MyMouseListener extends MouseAdapter {
public void mouseClicked(MouseEvent evt) {
if (evt.getClickCount() == 3) {
// triple-click
} else if (evt.getClickCount() == 2) {
// double-click
}
}
}
```

Handling Focus Changes

```
component.addFocusListener(
new MyFocusListener());
```

```
public class MyFocusListener
extends FocusAdapter {
public void focusGained(FocusEvent evt) {
// The component gained the focus.
}
public void focusLost(FocusEvent evt) {
// The component lost the focus.
}
}
```

Files, Streams, I/O (java.io)

Constructing a Path

On Windows, this example creates the path \blash a\blash b. On Unix, the path would be /a/b.

```
String path = File.separator +
"a" + File.separator + "b";
```

Reading Text from Standard Input

```
try {
BufferedReader in = new BufferedReader(
```

```
new InputStreamReader(System.in));
String str = "";
while (str != null) {
System.out.print("> prompt ");
str = in.readLine();
process(str);
}
} catch (IOException e) {
}
```

Reading Text from a File

```
try {
BufferedReader in = new BufferedReader(
new FileReader("infilename"));
String str;
while ((str = in.readLine()) != null) {
process(str);
}
in.close();
} catch (IOException e) {
}
```

Writing to a File

If the file does not already exist, it is automatically created.

```
try {
BufferedWriter out = new BufferedWriter(
new FileWriter("outfilename"));
out.write("aString");
out.close();
} catch (IOException e) {
}
```

Creating a Directory

```
(new File("directoryName")).mkdir();
```

Appending to a File

```
try {
BufferedWriter out = new BufferedWriter(
new FileWriter("filename", true));
out.write("aString");
out.close();
} catch (IOException e) {
}
```

Deleting a File

```
(new File("filename")).delete();
```

Deleting a Directory

```
(new File("directoryName")).delete();
```

Creating a Temporary File

```
try {
// Create temp file.
File temp = File.createTempFile(
"pattern", ".suffix");
// Delete temp file when program exits.
temp.deleteOnExit();
// Write to temp file
BufferedWriter out = new BufferedWriter(
new FileWriter(temp));
out.write("aString");
out.close();
} catch (IOException e) {
}
```

Using a Random Access File

```
try {
File f = new File("filename");
```

```
RandomAccessFile raf =
new RandomAccessFile(f, "rw");
// Read a character.
char ch = raf.readChar();
// Seek to end of file.
raf.seek(f.length());
// Append to the end.
raf.writeChars("aString");
raf.close();
} catch (IOException e) {
}
```

Serializing an Object

The object to be serialized must implement java.io.Serializable.

```
try {
ObjectOutput out = new ObjectOutputStream(
new FileOutputStream("filename.ser"));
out.writeObject(object);
out.close();
} catch (IOException e) {
}
```

Deserializing an Object

This example deserializes a java.awt.Button object.

```
try {
ObjectInputStream in = new ObjectInputStream(
new FileInputStream("filename.ser"));
AnObject object = (AnObject) in.readObject();
in.close();
} catch (ClassNotFoundException e) {
} catch (IOException e) {
}
```

Traversing a Directory

```
public static void traverse(File f) {
process(f);
if (f.isDirectory()) {
String[] children = f.list();
for (int i=0; i<children.length; i++) {
traverse(new File(f, children[i]));
}
}
}
```

Reading UTF-8 Encoded Data

```
try {
BufferedReader in = new BufferedReader(
new InputStreamReader(new FileInputStream(
"infilename"), "UTF8"));
String str = in.readLine();
} catch (UnsupportedEncodingException e) {
} catch (IOException e) {
}
```

Writing UTF-8 Encoded Data

```
try {
Writer out = new BufferedWriter(
new OutputStreamWriter(new FileOutputStream(
"outfilename"), "UTF8"));
out.write(aString);
out.close();
} catch (UnsupportedEncodingException e) {
} catch (IOException e) {
}
```

Reading ISO Latin-1 Encoded Data

```
try {
BufferedReader in = new BufferedReader(
new InputStreamReader(new FileInputStream(
"infilename"), "8859_1"));
String str = in.readLine();
} catch (UnsupportedEncodingException e) {
} catch (IOException e) {
}
```

Writing ISO Latin-1 Encoded Data

```
try {
Writer out = new BufferedWriter(
new OutputStreamWriter(new FileOutputStream(
"outfilename"), "8859_1"));
out.write(aString);
out.close();
} catch (UnsupportedEncodingException e) {
} catch (IOException e) {
}
```

Networking (java.net)

Creating a URL

```
try {
// With components.
URL url = new URL("http","hostname", 80, "index.html");
// With a single string.
url = new URL(
"http://hostname:80/index.html");
} catch (MalformedURLException e) {
}
```

Parsing a URL

```
try {
URL url = new URL("http://hostname:80/index.html#_top_");
String protocol = url.getProtocol();// http
String host = url.getHost();// hostname
int port = url.getPort();// 80
String file = url.getFile(); // index.html
String ref = url.getRef();// _top_
} catch (MalformedURLException e) {
}
```

Reading Text from a URL

```
try {
URL url = new URL("http://hostname:80/index.html");
BufferedReader in = new BufferedReader(
new InputStreamReader(url.openStream()));
String str;
while ((str = in.readLine()) != null) {
process(str);
}
in.close();
} catch (MalformedURLException e) {
} catch (IOException e) {
}
```

Resolving a Hostname

Creating a Client Socket

```
try {
InetAddress addr = InetAddress.getByName("java.sun.com");
int port = 80;
Socket sock = new Socket(addr, port);
} catch (IOException e) {
}
```

Creating a Server Socket

```
try {
int port = 2000;
ServerSocket srv = new ServerSocket(port);
// Wait for connection from client.
Socket socket = srv.accept();
} catch (IOException e) {
}
```

Reading Text from a Socket

```
try {
BufferedReader rd = new BufferedReader(
new InputStreamReader(socket.getInputStream()));
String str;
while ((str = rd.readLine()) != null) {
process(str);
}
rd.close();
} catch (IOException e) {
}
```

Writing Text to a Socket

```
try {
BufferedWriter wr = new BufferedWriter(
new OutputStreamWriter(socket.getOutputStream()));
wr.write("aString");
wr.flush();
} catch (IOException e) {
}
```

Sending a Datagram

```
public static void send(InetAddress dst,
int port, byte[] outbuf, int len) {
try {
DatagramPacket request = new DatagramPacket(
outbuf, len, dst, port);
DatagramSocket socket = new DatagramSocket();
socket.send(request);
} catch (SocketException e) {
} catch (IOException e) {
}
}
```

Receiving a Datagram

```
try {
byte[] inbuf = new byte[256]; // default size
DatagramSocket socket = new DatagramSocket();
// Wait for packet
DatagramPacket packet = new DatagramPacket(
inbuf, inbuf.length);
socket.receive(packet);
// Data is now in inbuf
int numBytesReceived = packet.getLength();
} catch (SocketException e) {
} catch (IOException e) {
}
```

Joining a Multicast Group

```
public void join(String groupName, int port) {
try {
MulticastSocket msocket = new MulticastSocket(port);
group = InetAddress.getByName(groupName);
msocket.joinGroup(group);
} catch (IOException e) {
}
```

```
}
```

Receiving from a Multicast Group

```
public void read(MulticastSocket msocket,
byte[] inbuf) {
try {
DatagramPacket packet = new DatagramPacket(
inbuf, inbuf.length);
// Wait for packet
msocket.receive(packet);
// Data is now in inbuf
int numBytesReceived = packet.getLength();
} catch (IOException e) {
}
}
```

Sending to a Multicast Group

```
byte[] outbuf = new byte[1024];
int port = 1234;
try {
DatagramSocket socket = new DatagramSocket();
InetAddress groupAddr = InetAddress.getByName(
```

```
"228.1.2.3");
DatagramPacket packet = new DatagramPacket(
outbuf, outbuf.length, groupAddr, port);
socket.send(packet);
} catch (SocketException e) {
} catch (IOException e) {
}
```

Defining and Exporting a Remote Object

1. Define the remote interface.

```
import java.rmi.*;
public interface RObject extends Remote {
void aMethod() throws RemoteException;
}
```

Looking Up a Remote Object and Invoking a Method

```
try {
// Look up remote object
RObject robj = (RObject) Naming.lookup(
"//localhost/RObjectServer");
// Invoke method on remote object
robj.aMethod();
```

```
} catch (MalformedURLException e) {

} catch (UnknownHostException e) {

} catch (NotBoundException e) {

} catch (RemoteException e) {

}
```

Passing Parameters to a Remote Method

Arguments to remote methods must be primitive, serializable, or Remote. This example demonstrates the declaration and use of all three parameter types.

1. Define the remote interface.

```
import java.rmi.*;

public interface RObject extends Remote {

// This parameter is primitive.

void primitiveArg(int num) throws RemoteException;

// This parameter implements Serializable.

void byValueArg(Integer num) throws RemoteException;

// This parameter implements Remote.

void byRefArg(ArgObject arg) throws RemoteException;

}

public interface ArgObject extends Remote {

int aMethod() throws RemoteException;
```

```
}
```

2. Define the remote object implementation.

```
import java.rmi.*;

import java.rmi.server.UnicastRemoteObject;

public class RObjectImpl extends UnicastRemoteObject
implements RObject {

public RObjectImpl() throws RemoteException {

super();

}

public void primitiveArg(int num) throws RemoteException {

}

public void byValueArg(Integer num) throws RemoteException {

}

public void byRefArg(ArgObject arg) throws RemoteException {

}

}
```

3. Compile the remote object implementation.

```
> javac RObject.java RObjectImpl.java
```

4. Generate the skeletons and stubs.

```
> rmic RObjectImpl
```

5. Create an instance of RObjectImpl and bind it to the RMI Registry.

```
try {

RObject robj = new RObjectImpl();

Naming.rebind("//localhost/RObjectServer", robj);

} catch (MalformedURLException e) {

} catch (UnknownHostException e) {

} catch (RemoteException e) {

}
```

6. Look Up the Remote object and pass the parameters.

```
try {

// Look up the remote object

RObject robj = (RObject)
Naming.lookup("//localhost/RObjectServer");

// Pass a primitive value as argument

robj.primitiveArg(1998);

// Pass a serializable object as argument

robj.byValueArg(new Integer(9));

// Pass a Remote object as argument

robj.byRefArg(new ArgObjectImpl());

} catch (MalformedURLException e) {
```

```
} catch (UnknownHostException e) {

} catch (NotBoundException e) {

} catch (RemoteException e) {

}
```

Returning Values from a Remote Method

Return values from remote methods must be primitive, serializable, or Remote. This example demonstrates the declaration and use of all three return types. 1. Define the remote interface.

```
import java.rmi.*;

public interface RObject extends Remote {

// This return value is primitive.

int primitiveRet() throws RemoteException;

// This return value implements Serializable.

Integer byValueRet() throws RemoteException;

// This return value implements Remote.

ArgObject byRefRet() throws RemoteException;

}

public interface ArgObject extends Remote {

int aMethod() throws RemoteException;

}
```

2. Define the remote object implementation.

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
public class RObjectImpl extends UnicastRemoteObject
implementsRObject {
public RObjectImpl() throws RemoteException {
super();
}
public int primitiveRet() throws RemoteException {
return 3000;
}
public Integer byValueRet() throws RemoteException {
return new Integer(2000);
}
public ArgObject byRefRet() throws RemoteException {
return new ArgObjectImpl();
}
}
```

3. Compile the remote object implementation.

> javac RObject.java RObjectImpl.java

4. Generate the skeletons and stubs.

> rmic RObjectImpl

5. Create an instance of RObjectImpl and bind it to the RMI Registry.

```
try {

RObject robj = new RObjectImpl();

Naming.rebind("//localhost/RObjectServer", robj);

} catch (MalformedURLException e) {

} catch (UnknownHostException e) {

} catch (RemoteException e) {

}
```

6. Look Up the Remote object, invoke the methods, and receive the return values.

```
try {

// Look up the remote object

RObject robj = (RObject) Naming.lookup(

"//localhost/RObjectServer");

// Receive the primitive value as return value

int r1 = robj.primitiveRet();

// Receive the serializable object as return value

Integer r2 = robj.byValueRet();

// Receive the Remote Object as return value
```

```
ArgObject aobj =robj.byRefRet();
} catch (MalformedURLException e) {
} catch (UnknownHostException e) {
} catch (NotBoundException e) {
} catch (RemoteException e) {
}
```

Throwing an Exception from a Remote Method

1. Define the remote interface.

```
import java.rmi.*;
public interface RObject extends Remote {
void aMethod() throws RemoteException;
}
```

2. Define the remote object implementation.

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
public class RObjectImpl extends
UnicastRemoteObject implements RObject {
public RObjectImpl() throws RemoteException {
super();
}
```

```
public void aMethod() throws RemoteException {

// The actual exception must be wrapped in

// a RemoteException

throw new RemoteException(

"message", new FileNotFoundException("message"));

}

}
```

3. Compile the remote object implementation.

```
> javac RObject.java RObjectImpl.java
```

4. Generate the skeletons and stubs.

```
> rmic RObjectImpl
```

5. Create an instance of RObjectImpl and bind it to the RMI Registry.

```
try {

RObject robj = new RObjectImpl();

Naming.rebind("//localhost/RObjectServer", robj);

} catch (MalformedURLException e) {

} catch (UnknownHostException e) {

} catch (RemoteException e) {

}
```

6. Look up the Remote object, invoke the method, and catch the exception.

```
try {

// Look up the remote object.

RObject robj = (RObject) Naming.lookup(

"//localhost/RObjectServer");

// Invoke the method.

robj.aMethod();

} catch (MalformedURLException e) {

} catch (UnknownHostException e) {

} catch (NotBoundException e) {

} catch (RemoteException e) {

// Get the actual exception that was thrown.

Throwable realException = e.detail;

}
```

Strings (java.lang)

Constructing a String

If you are constructing a string with several appends, it may be more efficient to construct it using a StringBuffer and then convert it to an immutable String object.

```
StringBuffer buf = new StringBuffer("Initial Text");

// Modify

int index = 1;

buf.insert(index, "abc");

buf.append("def");

// Convert to string

String s = buf.toString();
```

Getting a Substring from a String

```
int start = 1;

int end = 4;

String substr = "aString".substring(start, end);// Str
```

Searching a String

```
String string = "aString";

// First occurrence.

int index = string.indexOf('S');// 1

// Last occurrence.

index = string.lastIndexOf('i');// 4

// Not found.

index = string.lastIndexOf('z');// -1
```

Replacing Characters in a String

```
// Replace all occurrences of 'a' with 'o'
String newString = string.replace('a', 'o');
```

Replacing Substrings in a String

```
static String replace(String str,
String pattern, String replace) {
int s = 0;
int e = 0;
StringBuffer result = new StringBuffer();
while ((e = str.indexOf(pattern, s)) >= 0) {
result.append(str.substring(s, e));
result.append(replace);
s = e+pattern.length();
}
result.append(str.substring(s));
return result.toString();
}
```

Converting a String to Upper or Lower Case

```
// Convert to upper case
String upper = string.toUpperCase();
// Convert to lower case
```

```
String lower = string.toLowerCase();
```

Converting a String to a Number

```
int i = Integer.parseInt("123");
long l = Long.parseLong("123");
float f = Float.parseFloat("123.4");
double d = Double.parseDouble("123.4e10");
```

Converting Unicode to UTF-8

```
try {
String string = "\u5639\u563b";
byte[] utf8 = string.getBytes("UTF8");
} catch (UnsupportedEncodingException e) {
}
```

Converting UTF-8 to Unicode

```
public static String toUnicode(byte[] utf8buf) {
try {
return new String(utf8buf, "UTF8");
} catch (UnsupportedEncodingException e) {
}
return null;
}
```

Determining a Character's Unicode Block

```
char ch = '\u5639';
Character.UnicodeBlock block =
Character.UnicodeBlock.of(ch);
```

Breaking a String into Words

```
String aString = "word1 word2 word3";
StringTokenizer parser =
new StringTokenizer(aString);
while (parser.hasMoreTokens()) {
processWord(parser.nextToken());
}
```

Java and EJB - Interview Questions

1. What is Entity Bean and Session Bean ?
2. What are the methods of Entity Bean?
3. How does Stateful Session bean store its state ?
4. Why does Stateless Session bean not store its state even though it has ejbActivate and ejbPassivate ?
5. What are the services provided by the container ?
6. Types of transaction ?
7. What is bean managed transaction ?
8. Why does EJB needs two interface(Home and Remote Interface) ?
9. What are transaction attributes ?
10. What is the difference between Container managed persistent bean and Bean managed persistent entity bean ?

11. What is J2EE ?
12. What is JTS ?
13. How many entity beans used and how many tables can u use in EJB project ?
14. What is scalable,portability in J2EE?
15. What is Connection pooling?Is it advantageous?
16. Method and class used for Connection pooling ?
17. How to deploy in J2EE(i.e Jar,War file) ?
18. How is entity bean created using Container managed entity bean ?
19. Sotware architechture of EJB ?
20. In Entity bean will the create method in EJB home and ejbCreate in Entity bean have the same parameters ?
21. What methods do u use in Servlet – Applet communication ?
22. What are the types of Servlet ?
23. Difference between HttpServlet and Generic Servlets ?
24. Difference between doGet and doPost ?
25. What are the methods in HttpServlet?
26. What are the types of SessionTracking?
27. What is Cookie ? Why is Cookie used ?
28. If my browser does not support Cookie,and my server sends a cookie instance What will happen ?
29. Why do u use Session Tracking in HttpServlet ?
30. Can u use javaScript in Servlets ?
31. What is the capacity the doGet can send to the server ?
32. What are the type of protocols used in HttpServlet ?
33. Difference between TCP/IP and IP protocol ?
34. Why do you use UniCastRemoteObject in RMI ?
35. How many interfaces are used in RMI?
36. Can Rmi registry be written in the code, without having to write it in the command prompt and if yes where?
37. Why is Socket used ?
38. What are the types of JDBC drivers ?
39. Explain the third driver(Native protocol driver) ?

40. Which among the four driver is pure Java driver ?
41. What are the Isolation level in JDBC transaction ?
42. How do you connect with the database ?
43. How do you connect without the Class.forName (" ") ?
44. What does Class.forName return ?
45. What are the types of statement ?
46. Why is preparedStatement,CallableStatement used for?
47. In real time project which driver did u use ?
48. Difference between AWT and Swing compenents ?
49. Is there any heavy weight component in Swings ?
50. Can the Swing application if you upload in net, be compatible with your browser?
51. What should you do get your browser compatible with swing components?
52. What are the methods in Applet ?
53. When is init(),start() called ?
54. When you navigate from one applet to another what are the methods called ?
55. What is the difference between Trusted and Untrusted Applet ?
56. What is Exception ?
57. What are the ways you can handle exception ?
58. When is try,catch block used ?
59. What is finally method in Exceptions ?
60. What are the types of access modifiers ?
61. What is protected and friendly ?
62. What are the other modifiers ?
63. Is synchronised modifier ?
64. What is meant by polymorphism ?
65. What is inheritance ?
66. What is method Overloading ? What is this in OOPS ?
67. What is method Overriding ? What is it in OOPS ?
68. Does java support multi dimensional arrays ?
69. Is multiple inheritance used in Java ?

70. How do you send a message to the browser in JavaScript ?
71. Does javascript support multidimensional arrays ?
72. Why is XML used mainly?
73. Do use coding for database connectivity in XML?
74. What is DTD ?
75. Is there any tool in java that can create reports ?