**1. What is the purpose of garbage collection in Java, and when is it used?**

The purpose of garbage collection is to identify and discard objects that are no longer needed by a program so that their resources can be reclaimed and reused. A Java object is subject to garbage collection when it becomes unreachable to the program in which it is used.

**2. Describe synchronization in respect to multithreading.**

With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchonization, it is possible for one thread to modify a shared variable while another thread is in the process of using or updating same shared variable. This usually leads to significant errors.

**3. Explain different way of using thread?**

The thread could be implemented by using runnable interface or by inheriting from the Thread class. The former is more advantageous, 'cause when you are going for multiple inheritance.the only interface can help.

**4. What are pass by reference and passby value?**

Pass By Reference means the passing the address itself rather than passing the value. Passby Value means passing a copy of the value to be passed.

**5. What is HashMap and Map?**

Map is Interface and Hashmap is class that implements that.

**6. Difference between HashMap and HashTable?**

The HashMap class is roughly equivalent to Hashtable, except that it is unsynchronized and permits nulls. (HashMap allows null values as key and value whereas Hashtable doesnt allow). HashMap does not guarantee that the order of the map will remain constant over time. HashMap is unsynchronized and Hashtable is synchronized.

**7. Difference between Vector and ArrayList?**

Vector is synchronized whereas arraylist is not.

**8. Difference between Swing and Awt?**

AWT are heavy-weight componenets. Swings are light-weight components. Hence swing works faster than AWT.

**9. What is the difference between a constructor and a method?**

A constructor is a member function of a class that is used to create objects of that class. It has the same name as the class itself, has no return type, and is invoked using the new operator. A method is an ordinary member function of a class. It has its own name, a return type (which may be void), and is invoked using the dot operator.

**10. What is an Iterator?**

Some of the collection classes provide traversal of their contents via a java.util.Iterator interface. This interface allows you to walk through a collection of objects, operating on each object in turn. Remember when using Iterators that they contain a snapshot of the collection at the time the Iterator was obtained; generally it is not advisable to modify the collection itself while traversing an Iterator.

**1. What is the difference between an Interface and an Abstract class?**

An abstract class can have instance methods that implement a default behavior. An Interface can only declare constants and instance methods, but cannot implement default behavior and all methods are implicitly abstract. An interface has all public members and no implementation. An abstract class is a class which may have the usual flavors of class members (private, protected, etc.), but has some abstract methods.

**2. What is the purpose of garbage collection in Java, and when is it used?**

The purpose of garbage collection is to identify and discard objects that are no longer needed by a program so that their resources can be reclaimed and reused. A Java object is subject to garbage collection when it becomes unreachable to the program in which it is used.

**3. Describe synchronization in respect to multithreading.**

With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchonization, it is possible for one thread to modify a shared variable while another thread is in the process of using or updating same shared variable. This usually leads to significant errors.

**4. Explain different way of using thread?**

The thread could be implemented by using runnable interface or by inheriting from the Thread class. The former is more advantageous, 'cause when you are going for multiple inheritance..the only interface can help.

**5. What are pass by reference and passby value?**

Pass By Reference means the passing the address itself rather than passing the value. Passby Value means passing a copy of the value to be passed.

**6. What is HashMap and Map?**

Map is Interface and Hashmap is class that implements that.

**7. Difference between HashMap and HashTable?**

The HashMap class is roughly equivalent to Hashtable, except that it is unsynchronized and permits nulls. (HashMap allows null values as key and value whereas Hashtable doesnt allow). HashMap does not guarantee that the order of the map will remain constant over time. HashMap is unsynchronized and Hashtable is synchronized.

**8. Difference between Vector and ArrayList?**

Vector is synchronized whereas arraylist is not.

**9. Difference between Swing and Awt?**

AWT are heavy-weight componenets. Swings are light-weight components. Hence swing works faster than AWT.

**10. What is the difference between a constructor and a method?**

A constructor is a member function of a class that is used to create objects of that class. It has the same name as the class itself, has no return type, and is invoked using the new operator. A method is an ordinary member function of a class. It has its own name, a return type (which may be void), and is invoked using the dot operator.

**1. What is an Iterator?**

Some of the collection classes provide traversal of their contents via a java.util.Iterator interface. This interface allows you to walk through a collection of objects, operating on each object in turn. Remember when using Iterators that they contain a snapshot of the collection at the time the Iterator was obtained; generally it is not advisable to modify the collection itself while traversing an Iterator.

**2.State the significance of public, private, protected, default modifiers both singly and in combination and state the effect of package relationships on declared items qualified by these modifiers.**

public : Public class is visible in other packages, field is visible everywhere (class must be public too) private : Private variables or methods may be used only by an instance of the same class that declares the variable or method, A private feature may only be accessed by the class that owns the feature. protected : Is available to all classes in the same package and also available to all subclasses of the class that owns the protected feature.This access is provided even to subclasses that reside in a different package from the class that owns the protected feature. default :What you get by default ie, without any access modifier (ie, public private or protected).It means that it is visible to all within a particular package.

**3.What is an abstract class?**

Abstract class must be extended/subclassed (to be useful). It serves as a template. A class that is abstract may not be instantiated (ie, you may not call its constructor), abstract class may contain static data. Any class with an abstract method is automatically abstract itself, and must be declared as such. A class may be declared abstract even if it has no abstract methods. This prevents it from being instantiated.

**4.What is static in java?**

Static means one per class, not one for each object no matter how many instance of a class might exist. This means that you can use them without creating an instance of a class. Static methods are implicitly final, because overriding is done based on the type of the object, and static methods are attached to a class, not an object. A static method in a superclass can be shadowed by another static method in a subclass, as long as the original method was not declared final. However, you can't override a static method with a nonstatic method. In other words, you can't change a static method into an instance method in a subclass.

**5.What is final?**

A final class can't be extended ie., final class may not be subclassed. A final method can't be overridden when its class is inherited. You can't change value of a final variable (is a constant).

**6.What if the main method is declared as private?**

The program compiles properly but at runtime it will give "Main method not public." message.

**7.What if the static modifier is removed from the signature of the main method?**

Program compiles. But at runtime throws an error "NoSuchMethodError".

**8.What if I write static public void instead of public static void?**

Program compiles and runs properly.

**9.What if I do not provide the String array as the argument to the method?**

Program compiles but throws a runtime error "NoSuchMethodError".

**1.What is the first argument of the String array in main method?**

The String array is empty. It does not have any element. This is unlike C/C++ where the first element by default is the program name.

**2.If I do not provide any arguments on the command line, then the String array of Main method will be empty or null?**

It is empty. But not null.

**3.How can one prove that the array is not null but empty using one line of code?**

Print args.length. It will print 0. That means it is empty. But if it would have been null then it would have thrown a NullPointerException on attempting to print args.length.

**4. What environment variables do I need to set on my machine in order to be able to run Java programs?**

CLASSPATH and PATH are the two variables.

**5.Can an application have multiple classes having main method?**

Yes it is possible. While starting the application we mention the class name to be run. The JVM will look for the Main method only in the class whose name you have mentioned. Hence there is not conflict amongst the multiple classes having main method.

**6.Can I have multiple main methods in the same class?**

No the program fails to compile. The compiler says that the main method is already defined in the class.

**7.Do I need to import java.lang package any time? Why ?**

No. It is by default loaded internally by the JVM.

**8.Can I import same package/class twice? Will the JVM load the package twice at runtime?**

One can import the same package or same class multiple times. Neither compiler nor JVM complains abt it. And the JVM will internally load the class only once no matter how many times you import the same class.

**9.What are Checked and UnChecked Exception?**

A checked exception is some subclass of Exception (or Exception itself), excluding class RuntimeException and its subclasses. Making an exception checked forces client programmers to deal with the possibility that the exception will be thrown. eg, IOException thrown by java.io.FileInputStream's read() method� Unchecked exceptions are RuntimeException and any of its subclasses. Class Error and its subclasses also are unchecked. With an unchecked exception, however, the compiler doesn't force client programmers either to catch the exception or declare it in a throws clause. In fact, client programmers may not even know that the exception could be thrown. eg, StringIndexOutOfBoundsException thrown by String's charAt() method� Checked exceptions must be caught at compile time. Runtime exceptions do not need to be. Errors often cannot be.

**10.What is Overriding?**

When a class defines a method using the same name, return type, and arguments as a method in its superclass, the method in the class overrides the method in the superclass. When the method is invoked for an object of the class, it is the new definition of the method that is called, and not the method definition from superclass. Methods may be overridden to be more public, not more private.

**1.What are different types of inner classes?**

Nested top-level classes, Member classes, Local classes, Anonymous classes Nested top-level classes- If you declare a class within a class and specify the static modifier, the compiler treats the class just like any other top-level class. Any class outside the declaring class accesses the nested class with the declaring class name acting similarly to a package. eg, outer.inner. Top-level inner classes implicitly have access only to static variables.There can also be inner interfaces. All of these are of the nested top-level variety. Member classes - Member inner classes are just like other member methods and member variables and access to the member class is restricted, just like methods and variables. This means a public member class acts similarly to a nested top-level class. The primary difference between member classes and nested top-level classes is that member classes have access to the specific instance of the enclosing class. Local classes - Local classes are like local variables, specific to a block of code. Their visibility is only within the block of their declaration. In order for the class to be useful beyond the declaration block, it would need to implement a more publicly available interface.Because local classes are not members, the modifiers public, protected, private, and static are not usable. Anonymous classes - Anonymous inner classes extend local inner classes one level further. As anonymous classes have no name, you cannot provide a constructor.

**2.Are the imports checked for validity at compile time? e.g. will the code containing an import such as java.lang.ABCD compile?**

Yes the imports are checked for the semantic validity at compile time. The code containing above line of import will not compile. It will throw an error saying,can not resolve symbol symbol : class ABCD location: package io import java.io.ABCD;

**3.Does importing a package imports the subpackages as well? e.g. Does importing com.MyTest.\* also import com.MyTest.UnitTests.\*?**

No you will have to import the subpackages explicitly. Importing com.MyTest.\* will import classes in the package MyTest only. It will not import any class in any of it's subpackage.

**4.What is the difference between declaring a variable and defining a variable?**

In declaration we just mention the type of the variable and it's name. We do not initialize it. But defining means declaration + initialization.

e.g String s; is just a declaration while String s = new String ("abcd"); Or String s = "abcd"; are both definitions.

**5.What is the default value of an object reference declared as an instance variable?**

null unless we define it explicitly.

**6.Can a top level class be private or protected? A: No. A top level clas.s can not be private or protected. It can have either "public" or no modifier. If it**

does not have a modifier it is supposed to have a default access.If a top level class is declared as private the compiler will complain that the "modifier private is not allowed here". This means that a top level class can not be private. Same is the case with protected.

**7.What type of parameter passing does Java support?**

In Java the arguments are always passed by value .

**8.Primitive data types are passed by reference or pass by value?**

Primitive data types are passed by value.

**1.Objects are passed by value or by reference?**

Java only supports pass by value. With objects, the object reference itself is passed by value and so both the original reference and parameter copy both refer to the same object .

**2.What is serialization?**

Serialization is a mechanism by which you can save the state of an object by converting it to a byte stream.

**3.How do I serialize an object to a file?**

The class whose instances are to be serialized should implement an interface Serializable. Then you pass the instance to the ObjectOutputStream which is connected to a fileoutputstream. This will save the object to a file.

**4.Which methods of Serializable interface should I implement?**

The serializable interface is an empty interface, it does not contain any methods. So we do not implement any methods.

**5.How can I customize the seralization process? i.e. how can one have a control over the serialization process?**

Yes it is possible to have control over serialization process. The class should implement Externalizable interface. This interface contains two methods namely readExternal and writeExternal. You should implement these methods and write the logic for customizing the serialization process.

**6.What is the common usage of serialization?**

Whenever an object is to be sent over the network, objects need to be serialized. Moreover if the state of an object is to be saved, objects need to be serilazed.

**7.What is Externalizable interface?**

Externalizable is an interface which contains two methods readExternal and writeExternal. These methods give you a control over the serialization mechanism. Thus if your class implements this interface, you can customize the serialization process by implementing these methods.

**8.When you serialize an object, what happens to the object references included in the object?**

The serialization mechanism generates an object graph for serialization. Thus it determines whether the included object references are serializable or not. This is a recursive process. Thus when an object is serialized, all the included objects are also serialized alongwith the original obect.

**9.What one should take care of while serializing the object?**

One should make sure that all the included objects are also serializable. If any of the objects is not serializable then it throws a NotSerializableException.

**10.What happens to the static fields of a class during serialization?**

There are three exceptions in which serialization doesnot necessarily read and write to the stream. These are 1. Serialization ignores static fields, because they are not part of ay particular state state. 2. Base class fields are only hendled if the base class itself is serializable. 3. Transient fields.

**1.How does an exception permeate through the code?**

An unhandled exception moves up the method stack in search of a matching When an exception is thrown from a code which is wrapped in a try block followed by one or more catch blocks, a search is made for matching catch block. If a matching type is found then that block will be invoked. If a matching type is not found then the exception moves up the method stack and reaches the caller method. Same procedure is repeated if the caller method is included in a try catch block. This process continues until a catch block handling the appropriate type of exception is found. If it does not find such a block then finally the program terminates.

**2.What are the different ways to handle exceptions?**

There are two ways to handle exceptions, 1. By wrapping the desired code in a try block followed by a catch block to catch the exceptions. and 2. List the desired exceptions in the throws clause of the method and let the caller of the method hadle those exceptions.

**3.What is the basic difference between the 2 approaches to exception handling.**

1> try catch block and 2> specifying the candidate exceptions in the throws clause?

**4.When should you use which approach?**

In the first approach as a programmer of the method, you urself are dealing with the exception. This is fine if you are in a best position to decide should be done in case of an exception. Whereas if it is not the responsibility of the method to deal with it's own exceptions, then do not use this approach. In this case use the second approach. In the second approach we are forcing the caller of the method to catch the exceptions, that the method is likely to throw. This is often the approach library creators use. They list the exception in the throws clause and we must catch them. You will find the same approach throughout the java libraries we use.

**5.Is it necessary that each try block must be followed by a catch block?**

It is not necessary that each try block must be followed by a catch block. It should be followed by either a catch block OR a finally block. And whatever exceptions are likely to be thrown should be declared in the throws clause of the method.

**6.If I write return at the end of the try block, will the finally block still execute?**

Yes even if you write return as the last statement in the try block and no exception occurs, the finally block will execute. The finally block will execute and then the control return.

**7.If I write System.exit (0); at the end of the try block, will the finally block still execute?**

No in this case the finally block will not execute because when you say System.exit (0); the control immediately goes out of the program, and thus finally never executes.

**8.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.

**9.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.

**10.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.

**1. What is a transient variable?**

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

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

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

**3. Why do threads block on I/O?**

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

**4. How are Observer and Observable used?**

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

**5. What is synchronization and why is it important?**

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

**6. Can a lock be acquired on a class?**

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

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

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

**8. Is null a keyword?**

The null is not a keyword.

**9. What is the preferred size of a component?**

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

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

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

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

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

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

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

**13. What is the Collections API?**

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

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

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

**15. What is the List interface?**

The List interface provides support for ordered collections of objects.

**16. How does Java handle integer overflows and underflows?**

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

**17. What is the Vector class?**

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

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

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

**19. What is an Iterator interface?**

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

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

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

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

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

**22. What is the difference between yielding and sleeping?**

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

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

The EventObject class and the EventListener interface support event processing.

**24. Is sizeof a keyword?**

The sizeof operator is not a keyword.

**25. What are wrapped classes?**

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

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

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

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

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

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

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

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

Panel.

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

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

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

**31. Name three Component subclasses that support painting.**

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

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

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

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

Window.

**34. What is clipping?**

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

**35. What is a native method?**

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

**36. Can a for statement loop indefinitely?**

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

for(;;) ;

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

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

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

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

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

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

The default value of an String type is null.

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

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

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

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

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

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

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

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

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

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

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

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

**46. What is the immediate superclass of Menu?**

MenuItem.

**47. What is the purpose of finalization?**

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

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

Object.

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

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

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

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

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

**51. Name three subclasses of the Component class.**

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

**52. What is the GregorianCalendar class?**

The GregorianCalendar class provides support for traditional Western calendars.

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

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

**54. What is the purpose of the Runtime class?**

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

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

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

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

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

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

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

**58. Which Java operator is right associative?**

The = operator is right associative.

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

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

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

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

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

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

**62. Name two subclasses of the TextComponent class.**

TextField and TextArea.

**63. Which containers may have a MenuBar?**

Frame.

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

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

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

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

**66. What is an abstract method?**

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

**67. How are Java source code files named?**

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

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

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

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

**69. What are the high-level thread states?**

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

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

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

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

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

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

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

A static inner class does not have any object instances.

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

String objects are constants. StringBuffer objects are not constants.

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

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

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

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

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

**76. What is the Dictionary class?**

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

**77. How are the elements of a BorderLayout organized?**

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

**78. What is the % operator?**

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

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

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

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

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

**81. Which class is extended by all other classes?**

The Object class is extended by all other classes.

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

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

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

It is written x ? y : z.

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

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

**85. How is rounding performed under integer division?**

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

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

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

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

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

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

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

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

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

**90. What is the SimpleTimeZone class?**

The SimpleTimeZone class provides support for a Gregorian calendar.

General Collections

* 1. **What is a load factor ?**
  2. **Which of the collection class would you user if you have to implement a queque behavior ?**
  3. **What is size and capacity of a Vector ?**
  4. **Which situation would lead a Vector to throw an ArrayIndexOutOfBoundsException ?**
  5. **Can you iterate a map ?**No, Map does not implement Iterable interface, only Collection ( thus , list and set ) do , so the Collection implements iterable <E> , make the set and the list iteratble. The Map can't be iterable only the key part can be iterable for example the following code is how you traverse the map

Map <String,String> map= new HashMap<String,String>();

Ierator i =map.keySet().iterator();

while(i.hasNext()){

String value=map.get(i.next());

}

1. **How can you make your custom  class iterate through advanced for loop ?**You need to make your class implement the Iterable interface, override the iterator() method , and create Iterator class , which can iterate your items in the class.

**HashCode,Equality , toString,identity**

1. **Suppose that I am designing a class it is really mandatory to override the hashCode() method , even if I am sure none of these object will ever be used in Hash based collections like HashMap, Hashtable or HashSet ?**It is not necessary to override the hashCode method even if you don't use it in a hashbased collections. But If you have your equals method overriden , which means that you would want your object to be tested for equality. And the contract with the equals method is that , if two objects are equal you must override the hashcode method. Lets take the following example

[?](http://www.bullraider.com/java/core-java/advanced-interview-questions)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | class Animal {      private int i=12;  }  public class Test {      public static void main(String [] args) throws Exception {          Animal a1=new Animal();          Animal a2=new Animal();          System.out.println(a1==a2);          System.out.println(a1.equals(a2));      }  } |

Looking at the code you certainly know that the a1 and a2 objects are equal meaningfully, But the output would say otherwise, ( false false), This is not expected. What the above equals does is , it invokes the equals method on Object , and the object checks if they are pointing to the same object, which is not the right way to check. So essentially you would have to override the equals method, and as per contract with hashCode, the whenever the equals is overriden it is necessary to override the hashCode(), who knows you might need this objects added to the hash based collections, so better do it now.

1. **When I invoke system.out.println() on an object which class doesn't override the toString() method , what is the output and what it means ?**  
   The output comes in a format of <fully qualified class name>@hashcode**,**the hashcode (in hexadecimal) is invoked from the Object if hashCode method is not overriden
2. **What is the problem if you return a random number in your hashCode(), I think it is most efficient , since it would create unique bukkets ?**
3. **How good idea is that to return 1 ( a contstant value ) in your hashCode()?**  
   In such case all your object will fall into one single bucket , and the searching algorithm will end up invoking the equals method to find the object you are looking for , which is a highly inefficient way , but legal.

**Seriallization , Deseriallization, Externalization**

1. **Lets say you need to introduce a marker interface in your library, how would be add capability to the class which implements it?**
2. **Static fields don't get seriallized, but lets say I would want to save this the state of the field , how would I do it ?**

**Generics**

1. **If  methods overloaded on basic of type parameters, what is the best alternative to overload ?**

**Java Memory Management**

1. **How do you set the stack size of java runtime ?**

-Xss can be used to set the stack size

1. **What is the difference between stackoverflowerror vs outofmemoryerror ?**

 stackoverflow happens in overflow of stack ,and outofmemory ocuures if you dreained your heap.

1. **How to generate StackOverlFlow error and OutOfMemoryError problematically ?**
2. **What is package-private ?**  
    Package-private is a different name of default access specifier.
3. **What is Class Loader in java ?**

Java Runtime Environment that dynamically loads Java classes into the Java Virtual Machine on demand .

There are three types of class loaders. There are

**Bootstrap class loader**

             The bootstrap class loader loads the core Java libraries[5] located in the {JAVA\_HOME}/lib directory  
 **Extensions class loader**

The extensions class loader loads the code in the extensions directories {JAVA\_HOME}/lib/ext directory  
   **System Class  Path Loader**

             The system class loader loads code found on java.class.path, which maps to the system CLASSPATH variable

1. **What is the order of execution of Class Loader in java?**  
   Java applications  involve three classloaders – Bootstrap, Extensions and System-Classpath classloaders  
   Bootstrap classloader is the parent of all classloaders ,extensions Classloader is the immediate child of Bootstrap classloader and System-Classpath classloader is the immediate child of Extensions classloader. Generally  Order of execution happens from sub-sequent children to parent.   
   So order of execution is starts from System-classpath-loader then extension then Bootstrap Loader.
2. **What is differenence between ClassNotFoundException and NoClassDefFoundError ?**  
   **ClassNotFoundException** is generated by a call to Class.forName() with a String that contains a class not available on the bundle's class path. For example when we are dynamically want to load the class using  Class.forName("oracle.jdbc.driver.OracleDriver"); if that class is not aviailable in runtion, it'll throw ClassNotFoundException  
   **NoClassDefFoundError** is thrown if the Java Virtual Machine or a ClassLoader instance tries to load in the definition of a class and that class or its dependancies are not available  in run time. For example if Class x extends Class y and Class y is extends  class Z .When class loader tries to load class x if any of the depenacies is missing it'll give
3. **If we are using importing a class e.g java.util.ArrayList , and we are not using that class in side our program, does class loader will load the class?**  
   No, it won't
4. **Local variable, instance variable, method and object, amonng of these which one will store in heap and which one will store in stack?**  
   local variable and method will store in stack  
   instance variable and object will store in heap
5. **Emp e=new Employee(), where the "e" and "new Employee" will store?**  
   "e" will store in stack and Emploee() ( or the object )  will store in heap.
6. **Where we can can declare local variable ,where instance variable and where static variable?**  
   Local : inside mehod, constructor, and block  
   instance: outside method, constructor and block  
   static: outside method, constructor and block
7. **Can I write like this?**  
   void methodA()  
   {  
   int a;  
   System.out.println("I am in methodA");  
   }  
   yes, ( a stupid question for smart people )
8. **Can I write like this?**  
   void methodA()  
   {  
   int a;  
   System.out.println("I am in methodA " + (a+5));  
   }  
   no, because as long as we are not using local variable it won't give any error, but we should initialise it before using it(using of a).
9. **Can we declare a local variable with in static block, like below?**  
   public class TestVariable {  
           static{  
            int i=9;  
            System.out.println("print  "+i);  
       }  
   yes
10. **Can we use an instance variable with in static block ?**  
    public class TestVariable {  
        int i=9;  
            static{  
                 System.out.println("print  "+i);  
        }  
      
    No, because instance variable alwasys accessable wth a reference type.
11. **Can we write like this?**  
    public class TestVariable {  
                    static{  
             static int i=9;  
             System.out.println("print  "+i);  
        }  
    no, static variable always declares out side of block?
12. **Can I write like this?**  
      
    public class TestVariable {  
        int y=9;  
        static{  
             System.out.println("print  "+new TestVariable().y);  
        }  
    }  
    yes
13. **If I am writing static and non static block in my program like below, what is output?**  
    public class TestBlock {  
      
        static{  
      
            System.out.println("I am in static block");  
        }  
      
        {  
            System.out.println("I am in non static block");  
        }  
        public static void main(String a[]){  
            TestBlock block=new TestBlock();  
      
        }  
    }  
    Ans: I am in static block  
    I am in non static block
14. **What will be output for this program?**  
    public class TestBlock {  
      
        static{  
      
            System.out.println("I am in static1 block");  
        }  
      
        {  
            System.out.println("I am in non static block");  
        }  
      
        static{  
      
            System.out.println("I am in static2 block");  
        }  
        public static void main(String a[]){  
            TestBlock block=new TestBlock();  
      }  
    }  
    Ans  
    I am in static1 block  
    I am in static2 block  
    I am in non static block
15. **What will be the out put for folowing code?**  
      
    public class TestBlock {  
      
        static{  
      
            System.out.println("I am in static1 block");  
        }  
      
        {  
            System.out.println("I am in non static block");  
        }  
      
      
        public TestBlock() {  
            System.out.println("I am in non argumented constructor");  
        }  
          
        public TestBlock(String s) {  
            System.out.println("I am in argumented constructor  "+s);  
        }  
      
        public static void main(String a[]){  
            TestBlock block=new TestBlock("hello");  
      
        }  
    }  
    I am in static1 block  
    I am in non static block  
    I am in argumented constructor  hello
16. **Can I use instance variable in static variable like below?**  
    public class TestBlock {  
        int z=98;  
        static{  
            System.out.println("I am in static block"+new TestBlock().z);  
        }  
        public static void main(String a[]){  
            TestBlock block=new TestBlock();  
        }  
    }  
    yes, you can execute this program, out put will  
    I am in static block98