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

Ans. Simple Inner Class, Local Inner Class, Anonymous Inner Class, Static Nested Inner Class.

1. **Difference between Public, Private, Default and Protected?**

Ans. Private - Not accessible outside object scope.   
Public - Accessible from anywhere.   
Default - Accessible from anywhere within same package.  
Protected - Accessible from object and the sub class objects.

1. **What are the Wrapper classes available for primitive types?**

Ans. **Boolean** - java. lang. Boolean  
**byte** - java.lang.Byte  
**char** - java. lang. Character  
**double** - java.lang.Double  
**float** - java. lang. Float  
**int** - java. lang. Integer  
**long** - java. lang. Long  
**short** - java. lang. Short  
**void** - java.lang.Void

1. **What are concepts introduced with Java 5?**

Ans. Generics, Enums, Autoboxing, Annotations and Static Import.

**Does Constructor create the object?**

Ans. New operator in Java creates objects. Constructor is the later step in object creation. Constructor's job is to initialize the members after the object has reserved memory for itself.

1. **Can we access instance variables within static methods?**

Ans. Yes.  
  
we cannot access them directly, but we can access them using object reference.  
Static methods belong to a class and not objects whereas non-static members are tied to an instance. Accessing instance variables without the instance handler would mean an ambiguity regarding which instance the method is referring to and hence its prohibited.

1. **Does Java support Multiple Inheritance?**

Ans. Interfaces doesn't facilitate inheritance and hence implementation of multiple interfaces doesn't make multiple inheritance. Java doesn't support multiple inheritance.

**Q13. Difference between == and. equals ()?**

Ans. "equals" is the member of object class which returns true if the content of objects is same whereas "==" evaluate to see if the object handlers on the left and right are pointing to the same object in memory.

1. **What is a Final Variable?**

Ans. Final variable is a variable constant that cannot be changed after initialization.

1. **What is a final method?**

Ans. It’s a method which cannot be overridden. Compiler throws an error if we try to override a method which has been declared final in the parent class.

1. **Does java allow overriding static methods?**

Ans. No. Static methods belong to the class and not the objects. They belong to the class and hence doesn't fit properly for the polymorphic behavior.

1. **When are static variables loaded in memory?**

Ans. They are loaded at runtime when the respective Class is loaded.

1. **What are the common uses of "this" keyword in java?**

Ans. "this" keyword is a reference to the current object and can be used for following -  
  
1. Passing itself to another method.  
2. Referring to the instance variable when local variable has the same name.  
3. Calling another constructor in constructor chaining.

1. **How can we run a java program without making any object?**

Ans. By putting code within either static method or static block.

1. **Why java doesn't support multiple Inheritance?**

Ans. class A {  
void test() {  
System.out.println("test() method");  
}  
}  
  
class B {  
void test() {  
System.out.println("test() method");  
}  
}  
Suppose if Java allows multiple inheritance like this,  
class C extends A, B {  
}  
  
A and B test() methods are inheriting to C class.  
  
So which test() method C class will take? As A & B class test () methods are different, so here we would Facing Ambiguity.

1. **What are transient variables in java?**

Ans. Transient variables are variable that cannot be serialized.

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

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

1. **What is a String Pool?**

Ans. String pool (String intern pool) is a special storage area in Java heap. When a string is created and if the string already exists in the pool, the reference of the existing string will be returned, instead of creating a new object and returning its reference.

1. **Why is String immutable in Java?**

Ans. 1. String Pool  
  
When a string is created and if the string already exists in the pool, the reference of the existing string will be returned, instead of creating a new object. If string is not immutable, changing the string with one reference will lead to the wrong value for the other references.  
  
2. To Cache its Hash code  
  
If string is not immutable, one can change its hash code and hence not fit to be cached.  
  
3. Security  
  
String is widely used as parameter for many java classes, e.g. network connection, opening files, etc. Making it mutable might possess threats due to interception by the other code segment.

1. **Which are the different segments of memory?**

Ans. 1. Stack Segment - contains local variables and Reference variables (variables that hold the address of an object in the heap)  
2. Heap Segment - contains all created objects in runtime, objects only plus their object attributes (instance variables)  
3. Code Segment - The segment where the actual compiled Java bytecodes resides when loaded

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

Ans. 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

1. **Describe what happens when an object is created in Java?**

Ans. 1. Memory is allocated from heap to hold all instance variables and implementation-specific data of the object and its super classes. Implementation-specific data includes pointers to class and method data.  
2. The instance variables of the objects are initialized to their default values.  
3. The constructor for the most derived class is invoked. The first thing a constructor does is call the constructor for its super classes. This process continues until the constructor for java. lang. Object is called,  
as java. lang. Object is the base class for all objects in java.  
4. Before the body of the constructor is executed, all instance variable initializers and initialization blocks are executed. Then the body of the constructor is executed. Thus, the constructor for the base class completes first and constructor for the most derived class completes last.

1. **Describe, in general, how java's garbage collector works?**

Ans. The Java runtime environment deletes objects when it determines that they are no longer being used. This process is known as garbage collection. The Java runtime environment supports a garbage collector that periodically frees the memory used by  
objects that are no longer needed. The Java garbage collector is a mark-sweep garbage collector that scans Java's dynamic memory areas for objects, marking those that are referenced. After all possible paths to objects are investigated, those objects that are not marked (i.e. are not referenced) are known to be garbage and are collected.

1. **What is RMI?**

Ans. RMI stands for Remote Method Invocation. Traditional approaches to executing code on other machines across a network have been confusing as well as tedious and error-prone to implement. The nicest way to think about this problem is that some object happens to live on another machine, and that you can send a message to the remote object and get a result as if the object lived on your local machine. This simplification is exactly what Java Remote Method Invocation (RMI) allows you to do.

1. **What is JDBC? Describe the steps needed to execute a SQL query using JDBC.**

Ans. The JDBC is a pure Java API used to execute SQL statements. It provides a set of classes and interfaces that can be used by developers to write database applications.  
  
The steps needed to execute a SQL query using JDBC:  
  
1. Open a connection to the database.  
2. Execute a SQL statement.  
3. Process th results.  
4. Close the connection to the database.

1. **What are the methods of Object Class?**

Ans. clone() - Creates and returns a copy of this object.  
equals() - Indicates whether some other object is "equal to" this one.  
finalize() - Called by the garbage collector on an object when garbage collection determines that there are no more references to the object  
getClass() - Returns the runtime class of an object.  
hashCode() - Returns a hash code value for the object.  
toString() - Returns a string representation of the object.  
notify(), notifyAll(), and wait() - Play a part in synchronizing the activities of independently running threads in a program.

1. **Explain EJB (Enterprise Java Beans)?**

Ans. EJB Provides a mechanism that make easy for Java developers to use advanced features in their components, such as remote method invocation (RMI), object/ relational mapping (that is, saving Java objects to a relational database), and distributed transactions across multiple data sources.

1. **What are the default or implicitly assigned values for data types in java?**

Ans. Boolean ---> false  
byte ----> 0  
short ----> 0  
int -----> 0  
long ------> 0l  
char -----> /u0000  
float ------> 0.0f  
double ----> 0.0d  
any object reference ----> null

1. **What is difference between Encapsulation and Abstraction?**

Ans. 1. Abstraction solves the problem at design level while encapsulation solves the problem at implementation level  
  
2.Abstraction is used for hiding the unwanted data and giving relevant data. while Encapsulation means hiding the code and data into a single unit to protect the data from outside world.  
  
3. Abstraction lets you focus on what the object does instead of how it does it while Encapsulation means hiding the internal details or mechanics of how an object does something.  
  
4.For example: Outer Look of a Television, like it has a display screen and channel buttons to change channel it explains Abstraction but Inner Implementation detail of a Television how CRT and Display Screen are connect with each other using different circuits , it explains Encapsulation.

1. **Explain static blocks in Java?**

Ans. A static initialization block is a normal block of code enclosed in braces, {}, and preceded by the static keyword. Here is an example:  
  
static {  
// whatever code is needed for initialization goes here  
}  
  
A class can have any number of static initialization blocks, and they can appear anywhere in the class body. The runtime system guarantees that static initialization blocks are called in the order that they appear in the source code.

1. **Which access specifier can be used with Class?**

Ans. For top level class we can only use "public" and "default". We can use private with inner class.

1. **What are few of the Annotations pre-defined by Java?**

Ans. @Deprecated annotation indicates that the marked element is deprecated and should no longer be used. The compiler generates a warning whenever a program uses a method, class, or field with the @Deprecated annotation.   
  
@Override annotation informs the compiler that the element is meant to override an element declared in a superclass.   
  
@SuppressWarnings annotation tells the compiler to suppress specific warnings that it would otherwise generate.   
  
@SafeVarargs annotation, when applied to a method or constructor, asserts that the code does not perform potentially unsafe operations on its varargsparameter. When this annotation type is used, unchecked warnings relating to varargs usage are suppressed.  
  
@FunctionalInterface annotation, introduced in Java SE 8, indicates that the type declaration is intended to be a functional interface, as defined by the Java Language Specification.

1. **Difference between Abstract and Concrete Class?**

Ans. Abstract classes are only meant to be sub classed and not meant to be instantiated whereas concrete classes are meant to be instantiated.

1. **What is Serialization?**

Ans. Storing the state of an object in a file or other medium is called serialization.

1. **What is the use of Transient Keyword?**

Ans. It in Java is used to indicate that a field should not be serialized.

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

Ans. Final variable is a constant variable. Variable value can't be changed after instantiation.

1. **What is a Final Method?**

Ans. A Method that cannot be overridden in the sub class.

1. **What is a Final Class?**

Ans. A Class that cannot be sub classed.

1. **What is an Immutable Object?**

Ans. Object that can't be changed after instantiation.

1. **What is an immutable class?**

Ans. Class using which only immutable (objects that cannot be changed after initialization) objects can be created.

1. **How to implement an immutable class?**

Ans. We can make a class immutable by  
  
1. Making all methods and variables as private.  
2. Setting variables within constructor.  
  
Public Class ImmutableClass{  
private int member;  
ImmutableClass(int var){  
member=var;  
}   
}   
  
and then we can initialize the object of the class as  
ImmutableClass immutableObject = new ImmutableClass(5);  
Now all members being private , you can't change the state of the object.

1. **Does Declaring an object "final" makes it immutable?**

Ans. Only declaring primitive types as final makes them immutable. Making objects final means that the object handler cannot be used to target some other object, but the object is still mutable.

1. **Difference between object instantiation and construction?** Show

Ans. Though It's often confused with each other, Object Creation (Instantiation) and Initialization (Construction) are different things in Java. Construction follows object creation.  
  
Object Creation is the process to create the object in memory and returning its handler. Java provides New keyword for object creation.   
  
Initialization is the process of setting the initial / default values to the members. Constructor is used for this purpose. If we don't provide any constructor, Java provides one default implementation to set the default values according to the member data types.

1. **Can we override static methods? Why?**

Ans. No.   
  
Static methods belong to the class and not the objects. They belong to the class and hence doesn't fit properly for the polymorphic behavior.   
  
A static method is not associated with any instance of a class so the concept of overriding for runtime polymorphism using static methods is not applicable.