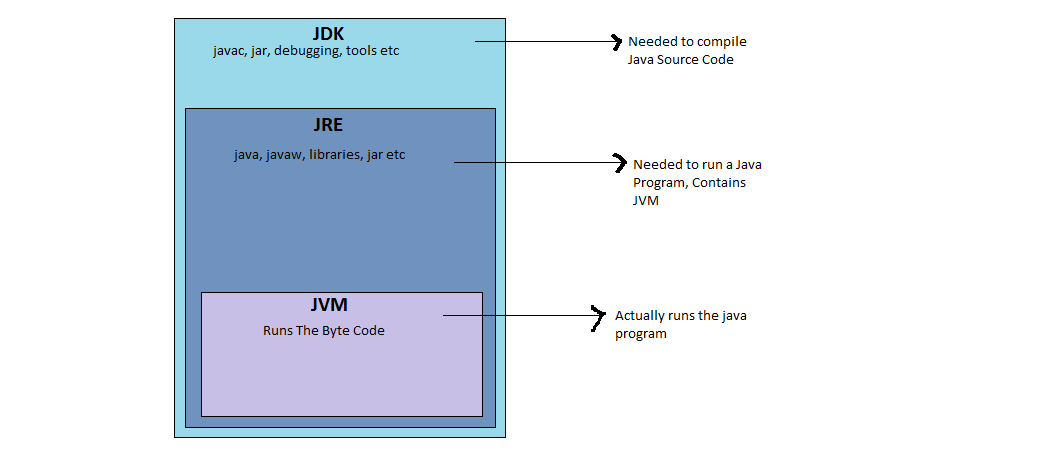
**JAVA Interview Questions**

**Q> what is JDK , JRE and JVM**

Ans:-



**Q> What is wrapper class and why do we need them or where do we use it**

Ans:-Java is an object-oriented language and can view everything as an object. A simple file can be treated as an object , an address of a system can be seen as an object , an image can be treated as an object (with java.awt.Image) and a simple data type can be converted into an object (with wrapper classes). Wrapper classes are used to convert any data type into an object.

The primitive data types are not objects; they do not belong to any class; they are defined in the language itself. Sometimes, it is required to convert data types into objects in Java language. For example, up to JDK1.4, the data structures accept only objects to store. A data type is to be converted into an object and then added to a Stack or Vector etc. For this conversion, the designers introduced wrapper classes.

**What are Wrapper classes?**

As the name says, a wrapper class wraps (encloses) around a data type and gives it an object appearance. Wherever, the data type is required as an object, this object can be used. Wrapper classes include methods to unwrap the object and give back the data type. It can be compared with a chocolate. The manufacturer wraps the chocolate with some foil or paper to prevent from pollution. The user takes the chocolate, removes and throws the wrapper and eats it.

Observe the following conversion.

int k = 100;

Integer it1 = new Integer(k);

The int data type k is converted into an object, it1 using Integer class. The it1 object can be used in Java programming wherever k is required an object.

The following code can be used to unwrap (getting back int from Integer object) the object it1.

int m = it1.intValue();

System.out.println(m\*m); // prints 10000

intValue() is a method of Integer class that returns an int data type.

**Importance of Wrapper classes**

There are mainly two uses with wrapper classes.

**1)** To convert simple data types into objects, that is, to give object form to a data type; here constructors are used.

**2)** To convert strings into data types (known as parsing operations), here methods of type parseXXX() are used.

**Features of the Java wrapper Classes.**

**1)** Wrapper classes convert numeric strings into numeric values.

**2)** The way to store primitive data in an object.

**3)** The valueOf() method is available in all wrapper classes except Character

**4)** All wrapper classes have typeValue() method. This method returns the value of the object as its primitive type.

**Q> javaAutoboxing and unboxing**(HSBC)

Ans:- The automatic conversion of primitive data types into its equivalent Wrapper type is known as boxing and opposite operation is known as unboxing. This is the new feature of Java5. So java programmer doesn't need to write the conversion code.

**Example:-**

**Q> what thing goes to heap and stack**

Stack :- faster

Heap :- slow

All premetive data types and functions calls are stored on stack.

If object is created then only its reference is created on stack.

All non premetive data types such as string, arrays and objects are stored on heap

**Q> what is the difference between hasmap and hastable**(wipro, J & J, Talentica)

**Ans :-** There are several differences between HashMap and Hashtable in Java:

Hashtable is synchronized, whereas HashMap is not. This makes HashMap better for non-threaded applications, as unsynchronized Objects typically perform better than synchronized ones.

Hashtable does not allow null keys or values. HashMap allows one null key and any number of null values.

HashMap is not an ordered collection which means it does not return the keys and values in the same order in which they have been inserted into the HashMap

Code for HashTable

**publicclass**HashTableExample {

**publicstaticvoid**main(String args[]){

Hashtable<Integer,String>ht=**new**Hashtable<Integer,String>();

ht.put(100,"pune");

ht.put(101,"mumbai");

System.***out***.println(ht.get(100));

System.***out***.println(ht.get(101));

}

}

**Q> What is Byte Code?**

**Or**

**What gives java it’s “write once and run anywhere” nature?**

**or**

**What makes Java Platform independent**

Ans:- All Java programs are compiled into class files that contain bytecodes. These byte codes can be run in any platform

and hence java is said to be platform independent.

**Q>Expain the reason for each keyword of public static void main(String args[])?**

Ans:- public- main(..) is the first method called by java environment when a program is executed so it has to accessible from java environment. Hence the access specifier must be public.

static: Java environment should be able to call this method without creating an instance of the class , so this method must be declared as static.

void: main does not return anything so the return type must be void

The argument String indicates the argument type andarg is an array for string given during command line.

**Q> What are the differences between == and .equals()**

**Or**

**what is difference between == and equals**

**Or**

**Difference between == and equals method**

**Or**

**What would you use to compare two String variables, the operator == or the method equals()?**

**Or**

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

Ans:-

The == operator compares two objects to determine if they are the same object in memory i.e. present in the same memory location.

It is possible for two String objects to have the same value, but located in different areas of memory.

== compares references while .equals compares contents. The method public booleanequals(Object obj) is provided by the Object class and can be overridden.

The default implementation returns true only if the object is compared with itself, which is equivalent to the equality operator == being used to compare aliases to the object. String,

BitSet, Date, and File override the equals() method. For two String objects, value equality means that they contain the same character sequence. For the Wrapper classes, value equality means that the primitive values are equal.

Example:-

--------

public class EqualsTest {

public static void main(String[] args) {

String s1 = "abc"; //this may or

String s2 = s1;

String s5 = "abc";

String s3 = new String("abc");

String s4 = new String("abc");

System.out.println("== comparison : " + (s1 == s5)); //== comparison : true

System.out.println("== comparison : " + (s1 == s2)); // == comparison : true

System.out.println("Using equals method : " + s1.equals(s2)); // true

System.out.println("== comparison : " + s3 == s4); //== comparison : false

System.out.println("Using equals method : " + s3.equals(s4)); // true

}

}

Output

-----------

== comparison : true

== comparison : true

Using equals method : true

== comparison : false

Using equals method : true

**Note :-**

[String](http://www.coderanch.com/t/410859/java/java/string-stringbuffer-stringbuilder-performance) a = "abc";   
  
May or may not create a new String object.   
If a String object with the literal "abc" already exists the reference 'a' will only point to it.   
Since String objects are immutable.   
  
Where as,   
String a = new String("abc");   
will always create a new Sting object.

**Q> What if the static modifier is removed from the signature of the main method**

**Or**

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

Ans:-

Program compiles. But at runtime throws an error “NoSuchMethodError”.

**Q> What is the difference between final, finally and finalize? What do you understand by the java final keyword?**

**Or**

**What is final, finalize() and finally?**

**Or**

**What is finalize() method?**

**Or**

**What is the difference between final, finally and finalize?**

**Or**

**What does it mean that a class or member is final?**

Ans:-

final – declare constant

finally – handles exception

finalize – helps in garbage collection

**Q>Describe the principles of OOPS**

Ans:- There are three main principals of oops which are called Polymorphism, Inheritance and Encapsulation.

**Q>Explain the Polymorphism principle. Explain the different forms of Polymorphism.**

Ans:- Polymorphism in simple terms means one name many forms. Polymorphism enables one entity to be used as a general category for different types of actions. The specific action is determined by the exact nature of the situation.

Polymorphism exists in three distinct forms in Java:

• Method overloading

• Method overriding through inheritance

• Method overriding through the Java interface

**Q> In System.out.println(), what is System, out and println.**

Ans:- System is a predefined final class, out is a PrintStream object and println is a built-in overloaded method in the out object.

**Q> What are Java Access Specifiers?**

**Or**

**What is the difference between public, private, protected and default Access Specifiers?**

**Or**

**What are different types of access modifiers?**

Ans:- Access specifiers are keywords that determine the type of access to the member of a class. These keywords are for allowing

privileges to parts of a program such as functions and variables. These are:

• Public : accessible to all classes

• Protected : accessible to the classes within the same package and any subclasses.

• Private : accessible only to the class to which they belong

• Default : accessible to the class to which they belong and to subclasses within the same package

**Q> Which class is the superclass of every class?**

Ans:- Object.

**Q> What is the difference between static and non-static variables?**

**Or**

**What are class variables?**

**Or**

**What is static in java?**

**Or**

**What is a static method?**

Ans:-

A static variable is associated with the class rather than with specific instances of a class. Each object will share a common copy of the static variables i.e. there is only one copy per class, no matter how many objects are created from it. Class variables or static variables are declared with the static keyword in a class. These are declared outside a class and stored in static memory. Class variables are mostly used for constants. Static variables are always called by the class name. This variable is created when the program starts and gets destroyed when the programs stops. The scope of the class variable is same an instance variable. Its initial value is same as instance variable and gets a default value when its not initialized corresponding to the data type. Similarly, 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.

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, if the original method was not declared final. However, you can’t override a static method with a non-static method. In other words, you can’t change a static method into an instance method in a subclass.

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

**Q> Can I have multiple main methods in the same class?**

Ans:- We can have multiple overloaded main methods but there can be only one main method with the following signature :

public static void main(String[] args) {}

**Q> Does Java support multiple inheritance?**

Ans :- No java does not support it. but there is a workaround. we can achieve it using interfaces.

**Q> What is constructor?**

Ans:- Constructor is just like a method that is used to initialize the state of an object. It is invoked at the time of object creation.

**Q> Difference between method overloading and method overriding**

Ans:-

Method Overloading

------------------

1) Method overloading increases the readability of the program.

2) method overlaoding is occurs within the class.

3) In this case, parameter must be different.

Method Overriding

------------------

1) Method overriding provides the specific implementation of the method that is already provided by its super class.

2) Method overriding occurs in two classes that have IS-A relationship.

3) In this case, parameter must be same.

**Q>Can we override static method?**

Ans:- No, you can't override the static method because they are the part of class not object

**Q> What is Runtime Polymorphism.**

Ans:- Runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time.

In this process, an overridden method is called through the reference variable of a super class. The determination of the method to be called is based on the object being referred to by the reference variable.

**Q> What is difference between abstract class and interface?**

Ans:-

Abstract class

--------------

1)An abstract class can have method body (non-abstract methods).

2)An abstract class can have instance variables.

3)An abstract class can have constructor \*\*\*\*\*\*\*\*\*\*\*\*\*\*

4)An abstract class can have static methods.

5)You can extends one abstract class.

Interface

---------

1)Interface have only abstract methods.

2)An interface cannot have instance variables.

3)Interface cannot have constructor.

4)Interface cannot have static methods.

5)You can implement multiple interfaces.

**Q>Does** [**Interfaces allows constructors?**](http://www.instanceofjava.com/2016/02/java-interface-constructor-example.html)

**Ans:-**

* No. Interfaces does not allow constructors.
* Why interface does not have constructor? The variables inside interfaces are static final variables means constants and we can not create object for interface so there is no need of constructor in interface that is the reason interface doesn't allow us to create constructor.

**Q>**[**Can we create static constructor in java**](http://www.instanceofjava.com/2016/02/java-static-constructor-method.html)

**Ans:-**

* No. We can not create constructor with static.
* If we try to create a static constructor compile time error will come: Illegal modifier for the constructor

**Q> What is difference between Checked Exception and Unchecked Exception?**

Ans:-

1)Checked Exception

The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions e.g.IOException,SQLException etc. Checked exceptions are checked at compile-time.

2)Unchecked Exception

The classes that extend RuntimeException are known as unchecked exceptions e.g. ArithmeticException,NullPointerException etc. Unchecked exceptions are not checked at compile-time.

**Q>Is it necessary that each try block must be followed by a catch block?**

Ans:- 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.

**Q>What is finally block?**

Ans:-finally block is a block that is always executed

**Q> What is difference between throw and throws?**

Ans:-

throw keyword

-------------

1)throw is used to explicitly throw an exception. throws is used to declare an exception.

2)checked exceptions can not be propagated with throw only. Un checked exception can be propagated with throws.

3)throw is followed by an instance. throws is followed by class.

4)throw is used within the method. throws is used with the method signature.

5)You cannot throw multiple exception You can declare multiple exception e.g. public void method()throws IOException,SQLException.

throws keyword

--------------

1)throws is used to declare an exception.

2)checked exception can be propagated with throws.

3)throws is followed by class.

4)throws is used with the method signature.

5)You can declare multiple exception e.g. public void method()throws IOException,SQLException.

**Q> What is the difference between String, StringBuffer and StringBuilder ?**

Ans:-

String:-

------

String is **immutable**  ( once created can not be changed )object . The object created as a String is stored in the Constant String Pool .

Every immutable object in Java is thread safe ,that implies String is also thread safe . String can not be used by two threads simultaneously.

String once assigned can not be changed.

Example:-

String demo = " hello " ;

// The above object is stored in constant string pool and its value can not be modified.

demo="Bye" ; //new "Bye" string is created in constant pool and referenced by the demo variable

// "hello" string still exists in string constant pool and its value is not overrided but we lost reference to the "hello"string

**StringBuffer( Thread safe)**

---------------

StringBuffer is mutable means one can change the value of the object . The object created through StringBuffer is stored in the heap .StringBuffer has the same methods as the StringBuilder , but each method in StringBuffer is synchronized that is StringBuffer is thread safe .

Due to this it does not allow two threads to simultaneously access the same method . Each method can be accessed by one thread at a time .

But being thread safe has disadvantages too as the performance of the StringBuffer hits due to thread safe property .Thus StringBuilder is faster than the StringBuffer when calling the same methods of each class.

StringBuffer value can be changed , it means it can be assigned to the new value . Nowadays its a most common interview question ,the differences between the above classes .

String Buffer can be converted to the string by using

toString() method.

Example:-

StringBuffer demo1 = new StringBuffer("Hello") ;

// The above object stored in heap and its value can be changed .

demo1=new StringBuffer("Bye");

// Above statement is right as it modifies the value which is allowed in the StringBuffer

**StringBuilder (Not a thread safe)**

--------------

StringBuilder is same as the StringBuffer , that is it stores the object in heap and it can also be modified . The main difference between the StringBuffer and StringBuilder is that StringBuilderis not thread safe.

StringBuilder is fast as it is not thread safe .

Example:-

StringBuilder demo2= new StringBuilder("Hello");

// The above object too is stored in the heap and its value can be modified

demo2=new StringBuilder("Bye");

// Above statement is right as it modifies the value which is allowed in the StringBuilder

Q> What is the difference between abstraction & encapsulation?

Ans :-

|  |  |
| --- | --- |
|  | [**Abstraction**](https://en.wikipedia.org/wiki/Abstraction_%28computer_science%29#Abstraction_in_object_oriented_programming) occurs during class level design, with the objective of **hiding** the implementation complexity of **how** the features offered by an API / design / system were implemented, in a sense simplifying the 'interface' to access the underlying implementation.  For example, a Java developer can make use of the high level features of [FileInputStream](https://docs.oracle.com/javase/7/docs/api/java/io/FileInputStream.html) without concern for how it works (i.e. file handles, file system security checks, memory allocation and buffering will be managed internally, and are hidden from consumers). This allows the implementation of FileInputStream to be changed, and if the API (interface) to FileInputStream remains consistent, code built against previous versions will still work.  Similarly, when designing your classes, you will want to hide internal implementation from others as far as possible.  [**Encapsulation**](https://en.wikipedia.org/wiki/Information_hiding), often referred to as "Information Hiding", revolves more specifically around the hiding of the internal data (e.g. state) owned by a class instance, and enforcing access to the internal data in a controlled manner.  For example, class fields can be made private by default, and only if external access to these was required, would a get () and/or set () (or Property) be exposed from the class. (In modern day OO languages, fields can be marked as read only / final / immutable which further restricts change, even within the class). |

**Q> What is static block**

Ans :- Static block is used for initializing the static variables.This block gets executed when the class is loaded in the memory. A class can have multiple Static blocks, which will execute in the same sequence in which they have been written into the program.

classJavaExample{

staticintnum;

staticStringmystr;

static

{

num = 97;

mystr = "Static keyword in Java";

}

publicstaticvoidmain(Stringargs[])

{

System.out.println("Value of num: "+num);

System.out.println("Value of mystr: "+mystr);

}

}

**What is the feature of Java?**

Java is a platform independent language.

**What is mean by platform independence?**

Platform independence means that we can write and compile the

java code in one platform (e.g. Windows) and can execute the

class in any other supported platform e.g. (Linux, Solaris, etc).

**Are JVM’s platform independent?**

JVM’s are not platform independent. JVM’s are platform specific run time implementation provided by the vendor.

**What is a JVM?**

JVM is Java Virtual Machine which is a run time environment for the compiled java class files.

**What is the difference between a JDK and a JVM?**

JDK is Java Development Kit which is for development purpose and it includes execution environment also. But JVM is purely a run time environment and hence you will not be able to compile your source files using a JVM.

**What is the base class of all classes?**

java.lang.Object

**Does Java support multiple inheritance?**

Java doesn’t support multiple inheritance using class.

**Is Java a pure object-oriented language?**

Java uses primitive data types and hence is not a pure object-oriented language.

**Are arrays primitive data types?**

In Java, Arrays are objects.

**What is difference between Path and Classpath?**

Path and Classpath are operating system level environment variables. Path is used define where the system can find the executables(.exe) files and classpath is used to specify the location .class files.

**What are local variables?**

Local variables are those which are declared within a block of code like methods. Local variables should be initialised before accessing them.

**What are instance variables?**

Instance variables are those which are defined at the class level. Instance variables need not be initialized before using them as they are automatically initialized to their default values.

**How to define a constant variable in Java?**

The variable should be declared as static and final. So only one copy of the variable exists for all instances of the class and the value can’t be changed also.

static final int PI = 2.14; is an example for constant.

**Should a main method be compulsorily declared in all java classes?**

No not required. main method should be defined only if the source class is a java application.

**What is the return type of the main method?**

Main method doesn’t return anything hence declared void.

**Why is the main method declared static?**

main method is called by the JVM even before the instantiation of the class hence it is declared as static.

**What is the argument of main method?**

main method accepts an array of String object as argument.

**Can a main method be overloaded?**

Yes. You can have any number of main methods with different method signature and implementation in the class.

**Can a main method be declared final?**

Yes. Any inheriting class will not be able to have its own default main method.

**Does the order of public and static declaration matter in main method?**

No, it doesn’t matter but void should always come before main().

**Can a source file contain more than one Class declaration?**

Yes, a single source file can contain any number of Class declarations but only one of the class can be declared as public.

**What is a package?**

Package is a collection of related classes and interfaces. package declaration should be first statement in a java class.

**Which package is imported by default?**

java.lang package is imported by default even without a package declaration.

**Can a class declared as private be accessed outside its package?**

Not possible.

Can a class be declared as protected?

A class can’t be declared as protected. only methods can be declared as protected.

**What is the access scope of a protected method?**

A protected method can be accessed by the classes within the same package or by the subclasses of the class in any package.

What is the purpose of declaring a variable as final?

A final variable’s value can’t be changed. final variables should be initialized before using them.

**What is the impact of declaring a method as final?**

A method declared as final can’t be overridden. A sub-class can’t have the same method signature with a different implementation.

I don’t want my class to be inherited by any other class. What should I do?

You should declare your class as final. But you can’t define your class as final, if it is an abstract class. A class declared as final can’t be extended by any other class.

**Can you give few examples of final classes defined in Java API?**

java.lang.String,java.lang.Math are final classes.

**How is final different from finally and finalize?**

final is a modifier which can be applied to a class or a method or a variable. final class can’t be inherited, final method can’t be overridden, and final variable can’t be changed.

finally, is an exception handling code section which gets executed whether an exception is raised or not by the try block code segment.

finalize() is a method of Object class which will be executed by the JVM just before garbage collecting object to give a final chance for resource releasing activity.

**Can a class be declared as static?**

No, a class cannot be defined as static. Only a method, a variable or a block of code can be declared as static.

**When will you define a method as static?**

When a method needs to be accessed even before the creation of the object of the class then we should declare the method as static.

**I want to print “Hello” even before main is executed. How will you achieve that?**

Print the statement inside a static block of code. Static blocks get executed when the class gets loaded into the memory and even before the creation of an object. Hence it will be executed before the main method. And it will be executed only once.

**What is the importance of static variable?**

static variables are class level variables where all objects of the class refer to the same variable. If one object changes the value, then the change gets reflected in all the objects.

**Can we declare a static variable inside a method?**

Static variables are class level variables and they can’t be declared inside a method. If declared, the class will not compile.

**What is an Abstract Class and what is its purpose?**

A Class which doesn’t provide complete implementation is defined as an abstract class. Abstract classes enforce abstraction.

**Can an abstract class be declared final?**

Not possible. An abstract class without being inherited is of no use and hence will result in compile time error.

**What is use of an abstract variable?**

Variables can’t be declared as abstract. only classes and methods can be declared as abstract.

**Can you create an object of an abstract class?**

Not possible. Abstract classes can’t be instantiated.

**Can an abstract class be defined without any abstract methods?**

Yes, it’s possible. This is basically to avoid instance creation of the class.

**Class C implements Interface I contain method m1 and m2 declarations. Class C has provided implementation for method m2. Can I create an object of Class C?**

No not possible. Class C should provide implementation for all the methods in the Interface I. Since Class C didn’t provide implementation for m1 method, it must be declared as abstract. Abstract classes can’t be instantiated.

Can a method inside an Interface be declared as final?

No not possible. Doing so will result in compilation error. public and abstract are the only applicable modifiers for method declaration in an interface.

**Can an Interface implement another Interface?**

Interfaces doesn’t provide implementation hence an interface cannot implement another interface.

**Can an Interface extend another Interface?**

Yes, an Interface can inherit another Interface, for that matter an Interface can extend more than one Interface.

**Can a Class extend more than one Class?**

Not possible. A Class can extend only one class but can implement any number of Interfaces.

Why is an Interface being able to extend more than one Interface but?

**Class can’t extend more than one Class.**

Basically, Java doesn’t allow multiple inheritance, so a Class is restricted to extend only one Class. But an Interface is a pure abstraction model and doesn’t have inheritance hierarchy like classes(do remember that the base class of all classes is Object). So, an Interface is allowed to extend more than one Interface.

**Can an Interface be final?**

Not possible. Doing so will result in compilation error.

**Can a class be defined inside an Interface?**

Yes, it’s possible.

**Can an Interface be defined inside a class?**

Yes, it’s possible.

**What is a Marker Interface?**

An Interface which doesn’t have any declaration.

**Which OO Concept is achieved by using overloading and overriding?**

Polymorphism.

**If I only change the return type, does the method become overloaded?**

No, it doesn’t. There should be a change in method arguments for a method to be overloaded.

**Why does Java not support operator overloading?**

Operator overloading makes the code very difficult to read and maintain. To maintain code simplicity, Java doesn’t support operator overloading.

**Can we define private and protected modifiers for variables in interfaces?**

No

**What modifiers are allowed for methods in an Interface?**

Only public and abstract modifiers are allowed for methods in interfaces.

**What is a local, member and a class variable?**

Variables declared within a method are “local” variables. Variables declared within the class i.e. not within any methods are “member” variables (global variables). Variables declared within the class i.e. not within any methods and are defined as “static” are class variables

**What is an abstract method?**

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

**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.

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

The Object class is extended by all other classes.

**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.

**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.

What do you understand by private, protected and public?

These are accessibility modifiers. Private is the most restrictive, while public is the least restrictive. There is no real difference between protected and the default type (also known as package protected) within the context of the same package, however the protected keyword allows visibility to a derived class in a different package.

**What is Down casting ?**

Down casting is the casting from a general to a more specific type, i.e. casting down the hierarchy

**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.

**What does it mean that a method or field is “static”?**

Static variables and methods are instantiated only once per class. In other words, they are class variables, not instance variables. If you change the value of a static variable in an object, the value of that variable changes for all instances of that class. Static methods can be referenced with the name of the class rather than the name of an object of the class (though that works too). That’s how library methods like System.out.println()

work. out is a static field in the java.lang.System class.

**Is null a keyword?**

The null value is not a keyword. 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.

**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.

**Does a class inherit the constructors of its superclass?**

A class does not inherit constructors from any of its super classes.

Name the eight primitive Java types.

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

**What is the difference between a while statement and a do statement?**

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.

What modifiers can be used with a local inner class?

A local inner class may be final or abstract.

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.

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.

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.

Are true and false keywords?

The values true and false are not keywords.

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

The result is a String object.

101.What is the difference between inner class and nested class?

When a class is defined within a scope of another class, then it becomes inner class. If the access modifier of the inner class is static, then it becomes nested class.

102.Can an abstract class be final?

An abstract class may not be declared as final

103.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

104.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.

105.To what value is a variable of the Boolean type automatically initialized?

The default value of the Boolean type is false

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

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

107.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.

108.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.

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

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

110.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.

111.What are the practical benefits, if any, of importing a specific class rather than an entire package (e.g. import java.net.\* versus import java.net.Socket)?

It makes no difference in the generated class files since only the classes that are used are referenced by the generated class file. There is another practical benefit to importing single classes, and this arises when two (or more) packages have classes with the same name. Take java.util.Timer and javax.swing.Timer, for example. If I import java.util.\* and javax.swing.\* and then try to use “Timer”, I get an error while compiling (the class name is ambiguous between both packages). Let’s say what you really wanted was the javax.swing.Timer class, and the only classes you plan on using in java.util are Collection and HashMap. In this case, some people will prefer to import java.util.Collection and import java.util.HashMap instead of importing java.util.\*. This will now allow them to use Timer, Collection, HashMap, and other javax.swing classes without using fully qualified class names in.

112.Can a method be overloaded based on different return type but same argument type ? No, because the methods can be called without using their return type in which case there is ambiquity for the compiler

113.What happens to a static var that is defined within a method of a class ?

Can’t do it. You’ll get a compilation error

114.How many static in it can you have ?

As many as you want, but the static initializers and class variable initializers are executed in textual order and may not refer to class variables declared in the class whose declarations appear textually after the use, even though these class variables are in scope.

115.What is the difference between method overriding and overloading?

Overriding is a method with the same name and arguments as in a parent, whereas overloading is the same method name but different arguments

116.What is constructor chaining and how is it achieved in Java ?

A child object constructor always first needs to construct its parent (which in turn calls its parent constructor.). In Java it is done via an implicit call to the no-args constructor as the first statement.

117.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.

118.Which Java operator is right associative?

The = operator is right associative.

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

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

120.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.

121.Can a for statement loop indefinitely?

Yes, a fort statement can loop indefinitely. For example, consider the following: for(;;).

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

The default value of a String type is null.

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.

123.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.

124.What does it mean that a class or member is final?

A final class cannot be inherited. A final method cannot be overridden in a subclass. A final field cannot be changed after it’s initialized, and it must include an initializer statement where it’s declared.

125.What does it mean that a method or class is abstract? An abstract class cannot be instantiated. Abstract methods may only be included in abstract classes. However, an abstract class is not required to have any abstract methods, though most of them do. Each subclass of an abstract class must override the abstract methods of its super classes or it also should be declared abstract.