**Core Java :: Interview Questions and Answers**

1. **What is the most important feature of Java?**

Java is a platform independent language.

1. **What do you mean by platform independence?**

Platform independence means that we can write and compile the java code in one platform (eg Windows) and can execute the class in any other supported platform eg (Linux,Solaris,etc).

1. **What is a JVM?**

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

1. **Are JVM's platform independent?**

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

1. **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.

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

java.lang.Object

1. **Does Java support multiple inheritance?**

Java doesn't support multiple inheritance.

1. **What is the inheritance?**

Inheritance is a mechanism in which one class acquires the property of another class. For example, a child inherits the traits of his/her parents. With inheritance, we can reuse the fields and methods of the existing class. Hence

1. **Is Java a pure object oriented language?**

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

1. **What is primitive data types?**

A primitive data type specifies the size and type of variable values, and it has no additional methods**.** Primitive data types - includes byte, short, int, long, float, double, boolean and char.

**Numbers :**

Primitive number types are divided into two groups:

Integer types stores whole numbers, positive or negative (such as 123 or -456), without decimals. Valid types are byte, short, int and long. Which type you should use, depends on the numeric value.

Floating point types represents numbers with a fractional part, containing one or more decimals. There are two types: float and double.

1. **Primitive data types and Non-Primitive data type ?**

Non-primitive data types are called reference types because they refer to objects.

The main difference between primitive and non-primitive data types are:

* Primitive types are predefined (already defined) in Java. Non-primitive types are created by the programmer and is not defined by Java (except for String).
* Non-primitive types can be used to call methods to perform certain operations, while primitive types cannot.
* A primitive type has always a value, while non-primitve types can be null.
* A primitive type starts with a lowercase letter, while non-primitive types starts with an uppercase letter.

The size of a primitive type depends on the data type, while non-primitive types have all the same size.

1. **What is data type in java?**

Data type specifies the size and type of values that can be stored in an identifier. The Java language is rich in its data types.

1. What do you mean by variable in Java?

A variable is a container that holds values that are used in a Java program. Every variable must be declared to use a data type

1. What is the difference between variable and data type?

A variable is a container which holds the value while the java program is executed. A variable is assigned with a data type.

Variable is a name of memory location. There are three types of variables in java: **local, instance and static.**

There are two types of data types in java**: primitive and non-primitive.**

1. **What is Local Variable?**

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

A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.

A local variable cannot be defined **with "static" keyword.**

1. **What is Instance Variable?**

*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*

A variable declared inside the class but outside the body of the method, is called instance variable. It is not declared as static.

It is called instance variable because its value is instance specific and is not shared among instances.

1. **What is Static variable?**

A variable which is declared as static is called static variable. It cannot be local. You can create a single copy of static variable and share among all the instances of the class. Memory allocation for static variable happens only once when the class is loaded in the memory.

1. **class** A{
2. **int** data=50;//instance variable
3. **static** **int** m=100;//static variable
4. **void** method(){
5. **int** n=90;//local variable
6. }
7. }//end of class
8. **How to define a constant / final 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 MAX\_LENGTH = 50; is an example for constant.

1. **What is the difference between public, protected, package-private and private in Java?**

│ Class │ Package │ Subclass │ Subclass │ World

│ │ │(same pkg)│(diff pkg)│

────────────┼───────┼─────────┼──────────┼──────────┼────────

public │ + │ + │ + │ + │ +

────────────┼───────┼─────────┼──────────┼──────────┼────────

protected │ + │ + │ + │ + │

────────────┼───────┼─────────┼──────────┼──────────┼────────

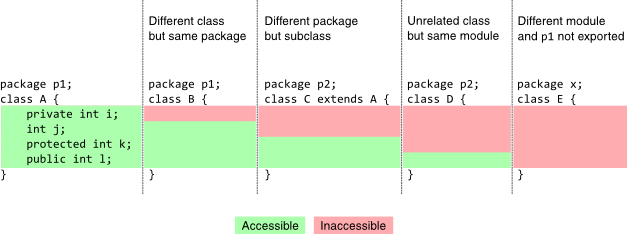
no modifier │ + │ + │ + │ │

────────────┼───────┼─────────┼──────────┼──────────┼────────

private │ + │ │ │ │

+ : accessible blank : not accessible

**Or**

[](https://i.stack.imgur.com/niONO.png)

## Explanations

* A **private** member (i) is only accessible within the same class as it is declared.
* A member with **no access modifier** (j) is only accessible within classes in the same package.
* A **protected** member (k) is accessible within all classes in the same package and within subclasses in other packages.
* A **public** member (l) is accessible to all classes (unless it resides in a [module](http://openjdk.java.net/projects/jigsaw/spec/sotms/) that does not export the package it is declared in).

1. **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.

1. **Java Final Keyword?**

A java variable can be declared using the keyword final. Then the final variable can be assigned only once.

A variable that is declared as final and not initialized is called a blank final variable. A blank final variable forces the constructors to initialise it.

Java classes declared as final cannot be extended. Restricting inheritance!

Methods declared as final cannot be overridden. In methods private is equal to final, but in variables it is not.

final parameters – values of the parameters cannot be changed after initialization. Do a small java exercise to find out the implications of final parameters in method overriding.

Java local classes can only reference local variables and parameters that are declared as final.

A visible advantage of declaring a java variable as static final is, the compiled java class results in faster performance.

1. **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.

1. Can you create an object of an abstract class?

Not possible. Abstract classes can't be instantiated.

1. **Can an Interface extend another Interface?**

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

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

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

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

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

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

No, an object cannot be cast to a primitive value

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

The Object class is extended by all other classes.

1. **What restrictions are placed on method overloading?**

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

1. **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.

1. **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