**Java At A Glance**

* **String** is a **final** class from **java.lang package**.
* **“System**” is a final class from java.lang package. “**out**” is static member of System class of type PrintStream. “**println**” is a method of PrintStream class.
* **Setting path variable and classpath variable**

set path=I:\Java\jdk1.7.0\_25\bin

echo %path%

>javac  -d  (Specify the path where to save generated .class files)  FileName.java

>java  -classpath  (path of generated .class files)  ClassName

> set classpath=I:\Classes

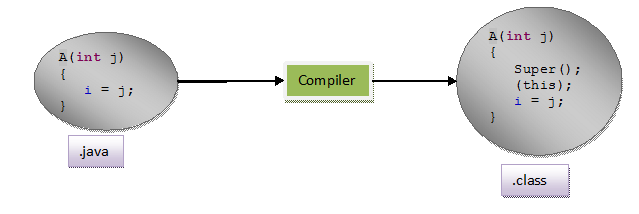
> echo %classpath%

**SIB**

* Static Initialization Block(**SIB**) is used to initialize only **static** variables.

**IIB**

* IIB stands for **Instance Initialization Block**. As the name suggest this block is used to initialize **state of an object**. State of an object is indicated by instance variables or non-static variables. So, IIB is used to initialize instance variables or non-static variables.
* We all know that [first statement of constructor](http://javaconceptoftheday.com/constructors-in-java/) is super() or this(). After executing first statement, IIB blocks are called. After executing IIB blocks, remaining statements are executed.
* So, when the constructor is called while creating an object (Line 19), compiler will treat constructor code like this,



where (this) is a calling statement to IIB block.

You can keep any number of IIB blocks in a class. All blocks are called after super() in the constructor in the order they appear.

* IIB blocks will not be called from the constructor in which **this()** statement is written as a first statement.

**Constructor**

* First statement in a constructor must be either **super**() or **this**(). If you put any other statements you will get compile time error.If you don’t include these statements, by default compiler will keep **super**() calling statement. **super**() – It is a calling statement to default constructor of super class. **this**()- it is a calling statement to constructor of the same class.

**Super**

* super keyword is used to access super class members inside the sub class. Using super keyword, we can access super class methods, super class fields and super class constructors in the sub classes.
* super class constructor is called by **super()** calling statement.You can’t use super() calling statement outside the constructor. By default, super() calling statement is the first statement in any constructor.
* If you want same implementation as that of super class method in the sub class, but want to add some more extra statements to it, in such cases, super keyword will be very useful. First call the super class method using super keyword and after it add extra statements according to requirements in the sub class method.

**This**

* this keyword is used to access other members of the same class. Using this keyword, you can access methods, fields and constructors of the same class within the class. this refers to current instance of the class.
* this() is the calling statement to same class constructor. It must be used within constructor only. If it is used, it must be the first statement in the constructor.
* You can’t use super and this keywords in a static method and in a static initialization block even though you are referring static members.

**Final**

* **final keyword in java** can be used with a class, with a variable and with a method. final keyword restricts the further modification. When you use final keyword with an entity (class or variable or method), it gets the meaning that entity is complete and can not be modified further.
* We can’t create a subclass to the class or we can’t extend a class or we can’t modify a class which is declared as **final**.
* We can’t override a method or we can’t modify a method in the sub class which is declared as **final** in the super class.
* The value of a final variable can not be changed in the whole execution once it got initialized.
* Any class or any method can be either **abstract or final** but not both. abstract and final are totally opposite. Because, abstract class or abstract method must be implemented or modified in the sub classes but final does not allow this. This creates an ambiguity.
* final method can be overloaded and that overloaded method can be overridden in the sub class.
* final variable can not be re-initialized but final variable can be used to initialize other variables.
* When an array reference variable is declared as final, only variable itself is final but not the array elements.
* When a reference variable is declared as final, you can’t re-assign a new object to it once it is referring to an object. But, you can change the state of an object to which final reference variable is referring.
* Static variables, non-static variables and local variables all can be final. once the final variables are initialized, even you can’t re-assign the same value.
* If the global variables are not initialized explicitly, they get default value at the time of object creation. But final global variables don’t get default value and they must be explicitly initialized at the time of object creation. Uninitialized final field is called **Blank Final Field**.
* final non-static global variable must be initialized at the time of declaration or in all constructors or in any one of IIBs – Instance Initialization Blocks.
* final static global variable must be initialized at the time of declaration or in any one of SIBs – Static Initialization Blocks. (final static global variable can’t be initialized in constructors)

**Type Casting**

* **byte < short < int < long < float < double.**
* **Auto Widening -** When you are converting data from small sized data type to big sized data type, i.e when you are converting data from left-placed data type to right-placed data type in the above order, auto widening will be used. For example, when you are converting byte to short or short to int, auto widening will be used.
* **Explicit Narrowing -** When you are converting data from big sized data type to small sized data type, i.e when you are converting data from right-placed data type to left-placed data type in the above order, explicit narrowing will be used. For example, when you are converting double to float or float to int, explicit narrowing will be used.
* **Auto-Up Casting -** Auto-Up Casting is used to change the type of object from sub class type to super class type. i.e an object of sub class type is automatically converted to an object of super class type.
* **Explicit Down Casting -** Explicit down Casting is used to change the type of object from super class type to sub class type. i.e you have to explicitly convert an object of super class type to an object of sub class type.

**Abstraction**

**Interface**

* Interface methods must be implemented as public. Because, interface methods are public by default and you should not reduce the visibility of any methods while overriding.
* interface fields are static and final by default and you can’t change their value once they are initialized
* Interfaces can’t have constructors.
* Interfaces can’t have initializers.
* Interfaces can’t be local members of a method.