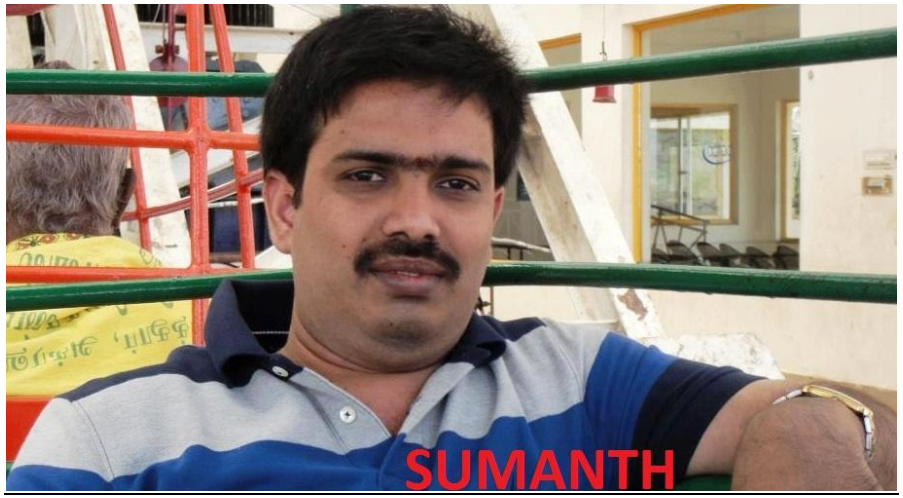


Author : Mr. SUMANTH

INDEX

CORE JAVA BASICS

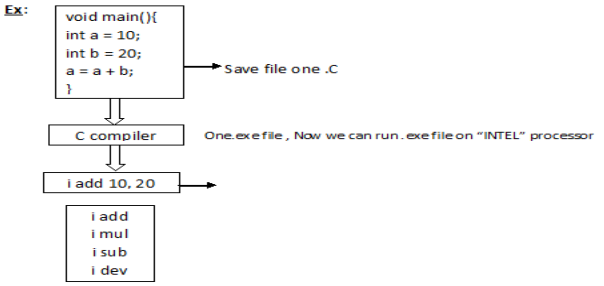
Basics:

There are so many companies who are manufacturing the processors, some of the company names are INTEL, SUN, AMD and etc. the **processors are responsible to execute the programs**. We have developed a 'C' program to add the numbers. The 'C' program developed high level language.

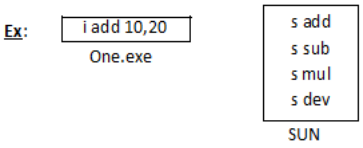
Ex:

```
void main(){
    int a = 10;
    int b = 20;
    a + b = a;
    }
```

If we would like to run this program. We need to compile the program. **It is the responsibility of compiler to convert High level language into machine level language**. The 'C' compiler generates an .exe file. Inside the exe file the 'C' compiler has placed the processor dependent instructions. This 'C' compiler is installed on INTEL processor.



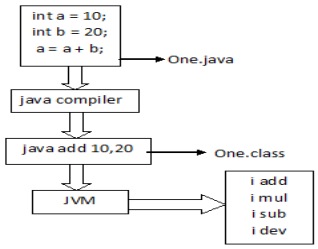
We have taken the same .exe file and try to run it on sun processor, but we are failed to execute the program. This is because sun processor cannot execute the "INTEL PROCESSOR" instructions.



If we got the requirement to run the 'C' program on all the processors we need to recompile the same program on different processor.

When develop a JAVA program to add the numbers we can not run the directly on any computer.

By taking the help of JAVA compiler we have converted .java programming to .class file.
If we would like to run the java application we need to give .class file as Input to JVM.
It is the responsibility of JVM to convert java instructions into processor dependent instructions.



To see what is available in a .class file, we can take the help of java profiler command.
When we use the **javap** command we have observe the following code in Add.class.

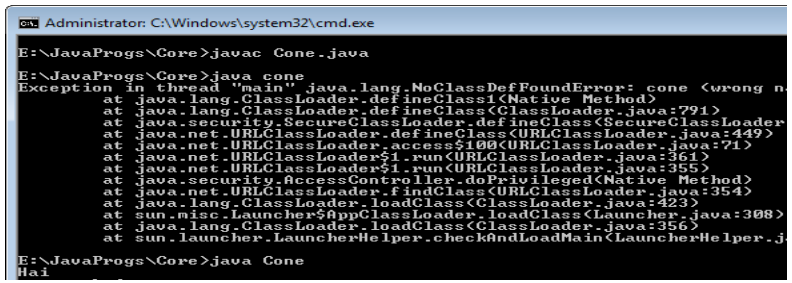
Ex: Public class Add extends java.lang.object{
public Add();
public static void main(java.lang.String[]) //save Add.class

If we want to see all the instructions also we have to use an option "-c"

Ex: Javap –c Add ⏎

- **We can invoke (call) the JVM by using the command java.** When we invoke the JVM it searches for appropriate .class file. If the class file is available **Class Loader** loads the class into JVM's memory. If the class is not available JVM throws an Exception java.lang.NoclassDefFoundError: sub.

After the class is loaded JVM checks for main method. If it is available JVM starts the execution from main method. If the main method is not available JVM throws an Exception NoSuchMethodFoundError: main



- **When the complier will add the constructor?**

When the programmer provides constructor in the java programmer the compiler will not add default constructor other wise compiler will Add default constructor.

Ex: public class Add{
public Add(){
}
} // save Add.java

- When the compiler will add the code to inherit properties of java.lang.object?

When the class is not Inheritance the properties of any other class then the compiler will try to inherit the properties of java.lang.object class
In the following code the compiler try's to inherit the properties of java.lang.object

Ex: public class Cone { } // save Cone.java

In the following example java compiler will not inherit the properties of java.lang.object

Ex: Public class Ctwo extends Cone { } //save Ctwo.java

We can compile and run the java programs using the following commands :
Compile : javac –g programname.java
Run : java –Djava.compiler=NONE programname (or)
java –Djava.compiler=none programname (or)
java –Djava.compiler programname

When ever we call the JVM , operating system loads the JVM program into RAM. Now the operating system allocates default memory (256mb) to JVM in a RAM.

EX :

After the program execute is finished, memory will be cleared from the RAM.
As start of the JVM we have a program class loader. **It is the responsibility of a class loader to load the class into JVM's memory.**
When ever JVM's starts the execution of a method it allocates two memory blocks. They are 1) Stack and 2) Method area
Stack is mainly used to perform any operations. Stack is a temporary memory location. If we want to the data permanently, We store the data in method area.
JVM's removes stack, method area after finishing the execution of method.
This is the Java Instructions for "Add.class" , **Command is javap –c add**

public class Add extends java.lang.object{
public Add();
code:

```
0: aload-0
1: invoke special #1;//Method java/lang/object."<init>"()u
4: return
public  static void main(java.lang.String[]);
code:
0: bipush 10
2: istore-1
3: bipush 20
5: istore-2
6: iload-1
7: iload-2
8: iadd
9: istore-1
10: return
 }
```

They are 3 types of variables in java.

1. **Local variables**
2. **Instance variables**
3. **Static variables**

- **The variables which are declared in side a method are called as local variables. The method parameters are also called as local variables.**

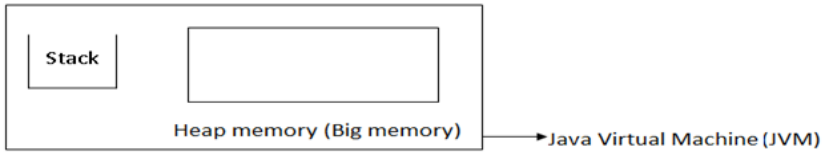
Ex:

```
public class One{
int a;
        //Instance Variables
int b;
                    Static Variables
static int aa;
public void methodOne(int d, int e){
int x;
        //Local Variables
int y;
      }
}
```

The memory for the local variables will be allocated in stack.

The variables which are declared in a class are called as Instance variables. In the above examples the variables 'a' and 'b' are called as a Instance variables ,

the memory will be allocated for the Instance variables in heap memory.



We have developed a class Cone with two Instance Variables as shown below.

Ex:

```
public class Cone {
int a;
int b;
}       //Cone.java
```

We have developed a reusable components Cone. Any body can use it by creating object to Cone class.

Ex:

```
Public class MyApp {
Public static void main(String[] args)  {
Cone c1 = new Cone();
      }
}            // MyApp.java
```

When we execute the above java program it will divided creation of object line into two parts, they are

Cone c1;	part1 (c1 = Reference Variable)
c1 = new Cone();	part2 (c1 create object to Cone class)

When the part1 line is executed JVM allocates memory for c1 Reference Variable inside the Stack (c1 Reference (local variable)).

When part2 is executed JVM try to create object to Cone class.

While try to create Cone class performs the following steps.

1. **JVM checks for Cone.class** and try to load Cone.class files into JVM's memory.
2. Now JVM **opens Cone.class** and **check for no of Instance variables** in a Cone.class
3. Now the **JVM calculates the required memory for the Instance variables**.
4. Now the **JVM's checks where the suficiant free space** is available to allocated the memory **for instance variables in JVM's Heap memory**.
5. After the memory is allocate , **JVM place the starting address of object into c1 Reference variable.**

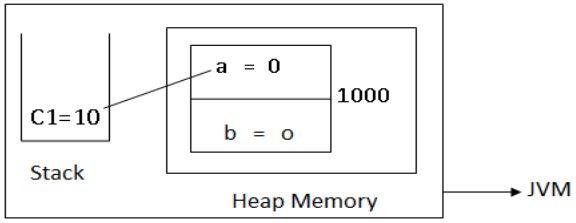
Note:

As the c1 variable is hold the address of object of Cone,then we call c1 as Reference variable.

When ever we create object or Instance for ever copy it contains one copy of variables. In java all the local variable must be initializing with some values, without initialize the local variables when we try to compilation a program we get a compilation error.

Ex: Public class MyApp{
Public static void main(String[] args){
int a;
System.out.println(a);
}
} // MyApp.java

All the instance variables are initializing with default values when the object is created.



We can access methods of a class based on reference variables or objects.

Ex: public class Cone {
int x;
int y;
public void methodOne() {
System.out.println("we are Cone methodOne");
System.out.println("x value is: " +x);
System.out.println("y value is:" +y);
}
} // Cone.java

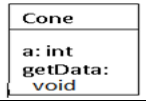
public class MyApp {
public static void main(String[] args) {
Cone c1 = new Cone();
Cone c2 = new Cone();
c1.X = 1;
c1.y = 2;
c2.x = 11;
c2.y = 22;
c1.methodOne();
c2.methodOne();

}
} // MyApp.java

Generally the team leaders of responsible to develop the UML diagrams.
Once the UML diagrams release we are responsible to convert UML diagrams to corresponding java program.

Note: UML diagrams are used for any programming language.
The following is an example class diagram.

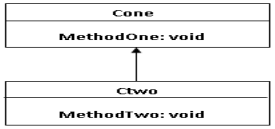
Ex:



The compiler compiles all the dependent class of a java program.

Ex: Public class MyApp{
Public static void main(String[] args){
Cone c1 = new Cone();
Ctwo c2 = new Ctwo();
C.methodOne();
}
}

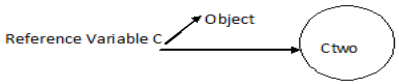
When we compile the java program the java compiler compile three different programs they are Cone.java, Ctwo.java and MyApp.java
The team leader has given the UML diagrams.



public class Cone {
void methodOne() {
System.out.println("we are in Cone method()");
}
} //Cone.java

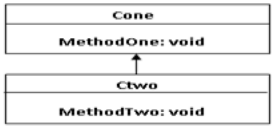
public class Ctwo{
void methodTwo(){
System.out.println("we are in Ctwo methodTwo()");
}
} //Ctwo.java

It is always recommended to create object to sub class. By using sub class object we can access the methods of that class as well as super class of that class.



From the above diagram we have understand that 'C' is reference variable is hold in Ctwo class.

*The following is a UML diagram given by the leader.



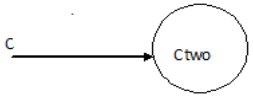
Now we have developed two classes Cone and Ctwo.

```
public class Cone{
void methodOne(){
System.out.println("we are in Cone methodOne()");
      }
}

public class Ctwo extends Cone{
void methodTwo(){
System.out.println("we are in Ctwo methodTwo()");
      }
}

public class MyApp{
public static void main(String[] args){
Ctwo c = new Ctwo();        ----> Step - 1
c.methodOne();              ----> Step -2
     }
}                           // MyApp.java
```

When JVM execute it creates reference variable C and Ctwo class object.



When JVM executes Step-2 JVM executes C reference variable is holding which class object JVM finds that C reference variable is holding Ctwo class object.

Now the JVM open Ctwo class object and check method one() is available as part of Ctwo class or not. If available in Ctwo class JVM executes it. If it is not available JVM checks whether Ctwo class is inheriting the properties of any other class.

In our example Ctwo class is inheriting the properties of Cone class. Now the JVM open's Cone class and check whether methodOne() is available in Cone class. JVM executes methodOne() Cone class.

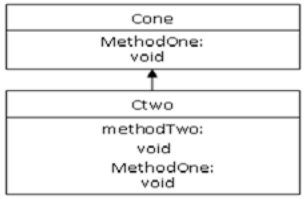
In **java we can achieve polymorphism by using inheritance**.

What is polymorphism?

An object which exhibits multiple behaviors is called as polymorphism.

In java a super class reference variable can hold sub class object.

According to the above UML diagram the following code is valid.

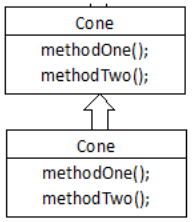



```
Ex: Cone c = new Cone();
  Object o = new Ctwo();
Ex: public class MYApp{
     Public static void main(String[] args){
     Cone c = new Ctwo();
      c.methodOne();
          }
     }
```

When we got an error message like javaC is not recognized as internal or external commands we can resolve the problem by setting path and environment variable in the command prompt.

Ex: Set PATH=c:\program files\java\jdk1.6_29\bin

If a super class reference variable can holds sub class object, by reference variable we can access the methods of that class or super class.



This UML diagram we want to achieve polymorphism for that we have created a reference variable to Cone class. It is holding Ctwo class object.

Ex: Cone c = new Ctwo();
By using c we can access the methods of methodOne() and methodTwo(). But we can't access methodThree() by using 'C' reference variable.

A reference variable can hold an object. By using a method getClass() we can find which class object is holed by the reference variable.

Ex: Cone c1 = new Cone();
System.out.println(c1.getClass());

A reference variable can hold the address of another object.

Ex:
```
public class MyApp{
public static void main(String[] args){
Cone c1 = new Cone();
Cone c2 = new Cone();
c1.x = 10;
c1.y = 20;
c2.x = 60;
c2.y = 70;
System.out.println(c1.x);
System.out.println(c1.y);
System.out.println(c2.x);
System.out.println(c2.y);
c1 = c2;
System.out.println(c1.x);
System.out.println(c1.y);
System.out.println(c2.x);
System.out.println(c2.y);
    }
}
```

A super class reference variable can hold sub class object. **By using the super class reference variable we can access the methods of that class only.**

But we can't access the specific methods of sub class , to access the specific methods of sub class **we must type cast super class reference variables into sub class**.

Ex:
```
public class MYApp {
public static void main(String args[]){
Object o = new Ctwo();
Cone c1 = (cone)o;
Ctwo c2 = (ctwo)o;
c2.methodTwo();
    }
}
```

We have developed Cone.java with two methods?
```
public class Cone{
public void methodOne(){
System.out.println("we are in Cone methodOne()");
this.methodTwo();
    }
```



```
public void mehodTwo{
System.out.println("we are in Cone methodTwo()");
    }
 }
```

We have develop Ctwo class by overriding methodTwo() in it?
```
public class Ctwo extends Cone{
public void methodTwo(){
System.out.println("we are in methodTwo()");
    }
}
```

In the main() we have create object to Ctwo class and call methodOne()
```
public class MyApp{
public static void main(String[] args){
Ctwo c = new Ctwo();
c.methodOne();
    }                    // Step 1
}
```

When we execute the above program JVM checks which class object is holed by the reference variable 'C'.

Now the JVM opens that reference variable and check for methodOne(). In our example 'C' reference variable is holding Ctwo class object.

Now the JVM checks mehtodOne() is available in Ctwo class or not. If it is not available JVM has checks the super class Cone. Now the JVM executes methodOne() of Cone class.

The internal code of methodOne() calling methodTwo(). Now the JVM checks whether methdTwo() is available in the current object or not , in our example in current object is Ctwo so the JVM open Ctwo class object and check for methodTwo(). If it is available it will execute or it go to super class of Cone and check for methodTwo().

Interfaces: An interface can contain set of abstract methods. In an interface we cannot provide method with body. By default all the methods in an interface are "Abstract".

As part of an interface we can declare the variable, **when we declare variables in the interface, by default they are "static and final" variables**.

Ex:
```
public  interface MyInterface{
int a = 10;
public void methodOne();
}
```

We cannot create an object to an interface?

MyInterface m = new MyInterface(); // Invalid

We can create reference variable to an interface?

MyInterface m;

Once the interface is released, any body can provide the implementation to an interface. Providing implementation means writing the method body.

The final variable values will not able to modify. We can assign the value to final variable only once. (at a time)

Once if we declared final, no body will be able to override in the sub class.

If we declared final classes no body will be able to inherit properties of final classes.

String class is the final class this is because no body could like to inherit properties of string class.

for example some body as created the following MyInterface with two abstract methods.

```
public interface MyInterface{
public void methodOne();
public void methodTwo();
   }                                    // MyInterface.java
```

If we would like to provide the implementation to the above interface we must provide the implementation to all the abstract methods, other wise java complier will not allow us to compile the program.

Ex:

```
public class MyImp implements MyInteface{
public void methodTwo();{
   }
}                                       //MyImp.java
```

When we compile the above java program we get a compilation error MYImp is not abstract and does not override all the abstract methods().

To an interface we can create the reference variable the reference variable holds an object of a class which provides the implementation of a interface.

```
public  class MyApp{
public static void main(String[] args){
MyInterface m;
m = new MyImp();
m = methodOne();
m = methodTwo();
```



```
   }
}                                       // MyApp.java
```

In a class we can provide abstract methods. If we provide abstract methods we must be declared that class as an abstract.

Ex:

```
abstract class Cone{
   public void methodOne(){
   }
   public void methodTwo(){
   }                                    // Cone.java
```

We cannot create object to any abstract class. We can create only reference variable.

We can declare of abstract if we don't allow any body to create object to a class.

Ex:

```
public abstract class Cone{
   public void methodOne(){
   }
   Public abstract void methodTwo();
   }
```

To the above Cone class we cannot create the object.

```
Cone c = new Cone();          // invalid
```

For an abstract class we can create only reference variable. i.e.

```
Cone c;
```

Once if we have an abstract class any body can inherit the properties of abstract class.

When they inherit properties of abstract class, they must provide the implementation to all the abstract methods.

Ex:

```
public class Ctwo extends Cone{
public void methodTwo(){
   }
}
```

We are training to a project to calculate electricity bill as well as water bill. The following programs calculate electricity bill as well as water bill **with out polymorphism**.

```
Public class ElecBill {
public void CalcElecBill(int units){
double bamount = units*2;
System.out.println("ElecBill amount is:" + bamount);
   }
}                                       // ElecBill.java

public class WaterBill{
```

```
public void CalcBill(int units){
double bamount = units*0.50;
System.out.println("water bill amount is:" +bamount);
    }
}                                    // WaterBill.java

public class APBill{
public static void main(string[] args){
ElecBill e = new ElecBill();
e.calElecBill(100);
WateBill w = new WaterBill();
w.calWaterBill(200);
    }
}                                    // APBill.java
```

In the above ApBill.java, we are calculating ElecBill by using an object 'e' we are using 'w' to calculate WaterBill.

As we are using two different objects to calculate ElecBill as well as WaterBill, we cannot call this program as polymorphism.

The following example demonstrates how you achieve polymorphism for the same above project. **To achieve polymorphism we take the help of interface** will force every developer to provide the implementation to that interface.

```
public Interface Bill{
public void calBill(){
  }
}
public class ElecBill implements Bill {
public void calBill(int units) {
double bamount = units * 2;
System.out.println("ElecBill amount is:" + bamount);
    }
}                                    //ElecBill.java

public class WaterBill implements Bill{
public void calBill(int units){
double bamount = units *0.50;
System.out.println("WaterBill amount is:" + bamount);
    }
}                                    //WaterBill.java

public class APBill{
```



```
public static void main(String[] args){
Bill b;
b = new ElecBill();
b.calBill(100);
b = new WaterBill();
b.calBill(100);
   }
}                                    //APBill.java
```

In the above APBill.java the object 'b' is used to calculate both ElecBill as well as WaterBill. Because of this reasons we calAPBill.java as polymorphism code.

The following example demonstrates use of abstract class.

```
public interface Car{
public void ty();
public void eng();
public void st();
public void br();
   }                                 //Car.java
```

MRF guis are interest to provide implementation only some methods of the car interface.

```
public abstract class MRF implements car{
public void ty(){
System.out.println("MRF Types");
    }
}                                    //MRF.java
```

MRF guys are any other fellows how want to a manufacture a car they engine the properties of MRF as shown below.

```
public class Mar extends MRF{
public void eng(){
System.out.println("Mar using suzi");
    }
public void st(){
System.out.println("Mar using power Streeing");
   }
public void br(){
System.out.println("mar using drum braks");
```

```
        }
}                                   //Mar.java
```

1. How do we organize the files in the project?
2. How do we deliver the project to the customer?
3. How do we resolve the problems like package not do not excite or class not found Exception?

Every project organizers the files in a better manner by create a folder.
Most of the people follow a procedure do organizer the files in the project?
Step1: Create a folder and the folder name must be project name (BMS).
Step2: Create the following folders in the above created project.

src = browser files
bin\classes-class files
lib-library files (jar files)
doc-documentation
tmp-temporary

We would like to develop 3 java programs in src folders and compile and placed .class files in the bin folder. If we would like to place the .class files into a specific folders we use an option '-d' for javac command.

Ex: **Javac –d <destination-folders> sourcefileName**
Javac –d d:\work\bms\bin welcome.java

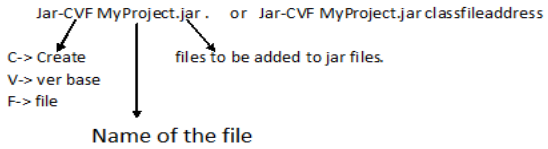
If always recommended to create the java program with packages. It is not at all recommended to create the java programs with out packages.
To create the packages every company follows their own procedures. Some of the company users the following pattern.
Package domain.companyname.projectname.module (Or)
Package companyname.projectname.modulename
To deliver the project to the customer we can use any of the following.

1. JAR(javar a relive)
2. WAR(web a relive)
3. EAR (enterprise a relive)

We deliver the software in the form of jar if it's corejava related projects.
All the web based applications (servlets, jsp, structs) will release in the form of WAR files.
The enterprise applications like EJB's will release in the from of EAR files.
We have only one file or command JAR.EXE to create jar file or war file or ear files.

D:\> javac –d

*__Procedure to create JAR file__:
Step1: Goto a folder where all .class files are available use the following command to create JAR file.
Jar –cvf jarfilename files-to-be-added
Ex:



To extract contents of jar file we use the following command.

Jar –XFV MyProject.jar

When we extract the contents of jar file we have found on extra folder META-INF.
Inside this folder we found an extra file MANIFEST.MF
As part of manifest.MF we specify version information of software.
The following are examples of creating a WAR file and EAR files.

Jar -CVF MyProject.war
Jar -CVF Myproject.ear

The main advantage of java is reusability. In java we can reuse the classes developed by somebody by creating the objects. If we would like to call the projects we must aware of all the methods of that class.
The **javadoc** command will help us increasing the doc as per the documentation we can find all the field summary, constructor summary & method summary.
The following is an example of using java doc command.

Javadoc Cone.java

The java documentation will generate so many .HTML files. We can navigate the documentation from index.HTML file.
Javac –d D:\work\lic\bin or D:\work\lic\src*.java
bin> java info.inetsolv.jee.mb.welcome
delivering project.jar file
cd work\lic\bin>jar –cvf lib.jar
\bin>cd src
>src> java doc –d
D:\work\lic\doc welcome.java

*__Environment Variables__:

These are the variables which will be created for us to create Environment variable.
We use keyword set.
Eg: D:\>work>set ONE = 1
To see all the environment variables of our computer we use a keyword "set".
To check the value of environment variable we use a command echo
Ex: echo% one %

When we run above command. If environment variable one is available we get value of it, if it is not available we get the value as % one %
Set one = 1;2;3 one is Variable name
Echo % one %
1;2;3
C:\> set one = 1
E:\> echo % one %
Set one = % one % 4,5;6;7
Echo % one % 1; 2; 3; 4; 5; 6; 7
Set one = % one % ;3;4;5
Echo % one % 1; 2; 3; 4; 5
Set one = 0; % one %
Echo % one % 0; 1; 2; 3; 4; 5

To append new value to the existing environment variable he will use following command.
Scope:
Where we can access variable is scope by default the scope of environment variable is until we close session (close the window) procedure to set the environment variable is until permanently.

1. Computer Right click -> properties -> Advanced system settings.
2. Choose an option advanced -> environment variables.
3. In system variables select new button and enter variable name and value.

If we want operating system to search for some files we use an environment variable path.
Ex: set PATH=C:\programfiles\java\jdk1.6.0_29\bin
Path environment variable environment variable is used to search for the files. This variable is used by operating system.

We use the class path environment variable to search for the .class files this environment variable to search for the .class files this environment variable is used by JVM.
\bin>jar –CVF MyProject.jar
C:\> set CLASSPATH=C:\MyProject.jar;.;
C:\> java info.inetsolv.jee.mb.welcome
package info.inetsolv.jee.mb
javac –d d:\work\bin
D:\work\lic\src *.java
java info.inetsolv.jee.mb.welcome
set CLASSPATH=D:\work\lib\bin;.;
java info.inetsolv.jee.mb.welcome

When we deliver jar file to customer. He set CLASSPATH to jar file & execute it.
Ex:

```
C:\> set CLASSPATH=C:\MyProject.jar;.;   } .cmd file
C:\> java info.inetsolv.jee.mb.welcome   } b (or)
C:\>pause                                } sh for unix
```

To release project to customers we create jar file if we give only jar file customer is responsible to set the CLASSPATH and type the commands to run the project. To resolve this problem we develop a CMD file or data file. In this we supply all the DOS commands.
set CLASSPATH=c:\MyProject.jar;.;
java info.inetsolv.jee.mb.welcome
pause //run.cmd
creating an object's INSTANCE
***static variables**:
In a class we can declare static variables , **for the static variables memory will be allocated only once. This allocation will happen at the time of loading the class into JVM's memory**.
When ever we create object the class is loaded into JVM's memory.

```
Public class Cone{
static int a;
public void methodOne(){
a = 10;
    }
Static{
System.out.println(a);
```

}

Memory Allocation

Local ⟶ Stack

Instance ⟶ Heap (one copy is created) variables

Static ⟶ Memory is not allocated to every object only one copy is allocated.

When class is loaded in JVM's memory, memory is allocated.

```
public class Cone{
   static int a;
       public static void methodOne(){
          a = 10;
                    }
       Static {
              System.out.println(a);
             }
}
```

When we run above Cone.java by using java Cone, JVM loads Cone.class into JVM's memory.

When the class is loaded **JVM checks for static variables JVM allocates the memory for all the static variables and these variables are initialized with default values**.

Now the JVM checks are there any **static methods are available. If they are available it will allocate memory to all the static methods**. How the JVM checks are there any **static blocks are available. If they are available JVM allocates memory to static blocks**.

Order of Execution is Static Variable → Static Methods →Static Blocks

Now, the JVM checks are there any static blocks are available. If they are available JVM Executed static blocks. When ever we run a program JVM checks for public static void main() method. If it is available it starts the Execution from main()

```
public class Cone{
int a;
static int a;
static {
System.out.println("we are in Cone");
        }
}                                          // save Cone.java

public class MyApp{
public static void main(String[] args){
Cone c1 = new Cone();
```



```
Cone c2 = new Cone();
      }
}                                          // save MyApp.java
```

Memory for static variables will be allocated only once when class is loaded into JVM's memory.

```
public class Cone{
int a;
static int b;
static{
System.out.println("we are in Cone");
       }
}                                          // save Cone.java

public class MyApp{
public static void main(String[] args){
Cone c1 = new Cone();
c1.a = 10;
Cone.b = 99;
System.out.println(c1.a);
System.out.println(Cone.b);
       }
 }                                         // save MyApp.java

public class MyApp{
public static void main(String args[]){
Cone c1 = new Cone();
Cone c2 = new Cone();
c1.a = 10;
c1.b = 99;
c2.a = 20;
c2.b = 70;
System.out.println(c1.a);
System.out.println(c1.b);
System.out.println(c2.a);
      }
}                                          // save MyApp.java

public class MyApp{
```

```
public static void main(String[] args){
Cone c1 = new Cone();
Cone c2 = new Cone();
c1.a = 10;
c1.b = 99;
c2.a = 20;
System.out.println(c1.a);
System.out.println(c1.b);
System.out.println(c2.a);
System.out.println(c2.b);
    }
}                                   // save MyApp.java
```

***Hard coding**:

Fixing the value to a variable is called as hard coding.

Ex:

```
public class MyApp{
public static void main(String[] args){
int a = 10;
System.out.println(a);
    }
}                                   // save MyApp.java
```

In the above program 'a' variable value is hard coded with a value '10'.

When we remove the hard coding project will be flexible. In a project it is not at all recommended to hard code the values we can use any of the following two techniques to remove hard coding. They are two types

1. Command line arguments
2. System properties

***Command line arguments**:

```
public class MyApp{
public static void main(String args[]){
 String color = ar[0];
int font = Integer.pharseInt(ar[1]);
System.out.println(color);
System.out.println(font);
    }
}                                   // save MyApp.java
```

To run the above java program we are using the following command.


```
>java MyApp blue 13
```

The disadvantage of above approach is if we inter change the values the project will be failed to resolve this problem. We use System properties.

***System properties**:

```
public class MyApp{
public static void main(String[] args){
String color = System.getProperty("c1");
int font = Integer.parseInt(System.getProperty("f"));
System.out.println(color);
System.out.println(font);
    }
}
```

To run the above program we use the following command.

Ex: >java –D cl = green –DF =12 MyApp

Define System variable name — Value

A method which returns an object's factory method.

It's always recommended to remove hard coding.

If we remove hard coding project will be very flexble.

***class.forName()**:

Static method to class

It is used to load the class into memory without creating the object.

Ex:

```
public class MyApp{
public static void main(String[] args){
try{
   class.forName("Cone");
  }
catch(class not found Exception c){
System.out.println("class not available");
        }
    }
}
```

All ways recommended handling the checked Exception by using any of the following two techniques.

1. try catch block
2. by using throws

When ever we use class.forName() to load the class we must specify fully qualified name or absolute class name (class name with package).

Ex:
```
public class MyApp{
public static void main(String[] args)throws classNotFoundException{
class.forName("java.lang.String");
    }
  }
```

class.forName return class object this object contains two information they are

1. Name of the loaded class and its package.
2. We can get this information by using a methods like getName() get package();

Ex:
```
class MyApp{
public static void main(String[] args)throws classNotFoundException{
class c = class.forName("java.lang.Object");
System.out.println(c.getName());
System.out.println(c.getPackage());
      }
}
```

Get package method returns null value if the class does not contain package.

Develop a java program to create object to a class without using new operator.

***Using newInstance Operator**:

The following java program creates object to Cone class without using new operator and able to call methods.

```
public class MyApp{
public static void main(String[] args)throws Exception{
class c = class.forName("Cone");
Object O = c.new Instance();
Cone c1 = (Cone)O;
c1.methodOne();
    }
}
```

Ex:

```
public class Cone{
}
```

✓

```
public class Cone{
public Cone(){
    }
}
```

✓


```
public class Cone{
public Cone(int dummy){
    }
}
```

✗

```
public class Cone{
public Cone{
    }
public Cone(int dummy){
}
```

✓

The following java program crates the object to any class dynamically.

```
public class MyApp{
public static void main(String[] agrs)throws Exception{
class c = class.forName(ar[0]);
object o = c.newInstance();
System.out.println(o.getClass());
      }
}
```

New instantiation method create object to a class if the class contains default constructor().

If class contains only parameter constructures() class.forName() fails in create a the object.

JDBC

***API (Application Programming Interface)**:

API is a document. (not a software)

They are two types of API's are available. They are

1. Public API
2. Proprietary API

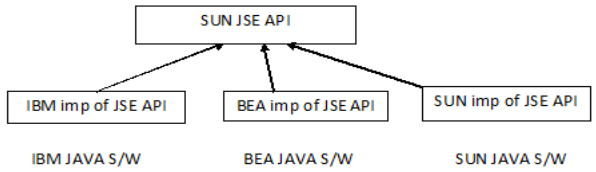
Any body can use the public API. Proprietary API can be used by only that company.

In java, API document contains set of class and Interfaces.

In case of 'C' language, API document contains set of predefined functions.

Once if the API is released any body can use the API.

The meaning of using API is providing the implementation.



Sun micro system as released JSE API. This is a public API released by sun micro system.