**Java Abstraction and Encapsulation**  
   
**Java Object Oriented Programming**  
  
i) Inheritance  
  
ii) Polymorphism  
----------------  
iii) Abstraction  
  
iv) Encapsulation  
--------------------------------------  
**Abstraction**  
  
> It is a process of hiding implementation details and showing only functionality to the user.  
  
Two types of Methods in Java  
  
1) Concrete Methods (The methods which are having body)  
  
Example:  
  
public void add(){  
Statements  
---------  
--------  
-----------  
}  
  
2) Abstract Methods (The methods which are not having body)  
  
Ex:  
  
public void add();  
  
--------------------------------------  
> If we know the method name, but don't know the method functionality, then we go for Abstract methods.  
  
> Java Class contains 100% concrete methods.  
  
> Abstract class contains one or more abstract methods.  
--------------------------------------  
  
Example:  
  
Class 1 (having 10 methods)  
  
10 methods are concrete methods  
  
It is a Java Class  
  
Class 2 (having 10 methods)  
  
(5 concrete methods and 5 abstract methods)  
  
Abstract class  
  
Class 3 (having 10 methods)  
  
(All 10 methods are abstract methods)  
  
Abstract class

----------------------------  
Example:  
  
Abstract Class  
  
public abstract class Bike {  
  
public void handle(){  
System.out.println("Bikes have Handle");  
}

public void seat(){  
System.out.println("Bikes have Seats");  
}  
  
public abstract void engine();  
  
public abstract void wheels();  
  
public static void main (String [] args){  
Bikes obj = new Bikes();  
obj.engine();  
obj.wheels();  
}  
}  
----------------------------------  
Sub Class   
  
public class HeroHonda extends Bike{  
@Override  
public void engine() {  
System.out.println("Bikes have Engine");  
}  
  
@Override  
public void wheels() {  
System.out.println("Bikes have Wheels");  
}  
  
public static void main (String [] args){  
//Create Object  
HeroHonda objHH = new HeroHonda();      
objHH.engine();  
objHH.seat();  
objHH.wheels();  
objHH.handle();  
}  
}  
--------------------------

**Interfaces**  
  
Selenium IDE has User Interface (Integrated Development Environment)  
  
Selenium WebDriver is a Programming Interface.  
---------------------------  
UFT/QTP has both IDE as well as Programming Interface.  
------------------------------  
Java Interfaces  
  
> Interface is a Java type definition block which is 100% abstract  
  
> All the Interface methods by default public and abstract.  
  
> static and final modifiers are not allowed for interface methods.  
  
> In Interfaces variables have to initialize at the time of declaration.  
  
int a;//Incorrect  
  
int a =10; //Correct  
  
> In Interfaces variables are public, static, and final by default.  
  
> Interface is going to be used using "implements" keyword.  
---------------------------------------------------  
**Example for Java Interface:**  
  
1) Create an Interface  
  
public interface Interface1 {  
public void engine();  
public void wheels();  
public void seat();  
public void handle();  
  
public static void main (String [] args){  
ClassNew objx = new ClassNew();      
objx.engine();  
objx.wheels();  
}  
}  
  
2) Reuse Methods from Interface to Class  
  
public class ClassNew implements Interface1{  
  
      
public void engine() {  
System.out.println("Bikes have Engine");  
}  
      
public void wheels() {  
System.out.println("Bikes have Wheels");  
}  
  
      
public void seat() {  
System.out.println("Bikes have Seats");  
}  
  
public void handle() {  
System.out.println("Bikes have Handle");  
}  
  
public static void main (String [] args){  
ClassNew obj = new ClassNew();  
obj.seat();  
obj.wheels();  
obj.engine();  
obj.handle();  
}  
}  
---------------------------------------------  
Note:  
  
From Class (Concrete class or Abstract Class) to Class we use "extends" keyword  
  
From Interface to Class we use "implements" keyword  
-----------------------------------

**Encapsulation:**  
  
It is a process of wrapping code and data into a single unit.  
  
Encapsulation is the technique making the fields in a class private and providing access via public methods.  
  
> It provides control over the data.  
  
> By providing setter and getter methods we can make a class read only or write only.  
  
Example:  
Class1:  
  
public class Class1 {  
private String name = "Test Automation";  
  
public String getName(){  
return name;  
}  
public void setName(String newName){  
name=newName;      
}  
  
public static void main (String [] args){  
Class1 obj = new Class1();  
System.out.println(obj.getName());  
}  
}  
--------------------------------  
Class2:  
public class Class2 extends Class1 {  
  
public static void main(String[] args) {  
Class2 obj2 = new Class2();  
obj2.setName("Selenium");  
System.out.println(obj2.getName());  
}  
}  
--------------------------------------  
Java for Selenium  
  
Java Environment setup  
Java Program Structure  
**A) Java Fundamentals**  
    1) Comments in Java  
    2) Data Types  
    3) Modifiers  
    4) Variables  
    5) Operators  
    6) Flow Control  
        Conditional Statements  
        Loop Statements  
    8) Arrays  
    9) String Handling in Java  
    10) I/O operations and File handling  
    11) Java Methods  
        Built in methods  
        User defined methods  
    13) Exception handling  
-------------------------------  
**B) Java OOPS**  
  
    1) Inheritance  
    2) Polymorphism  
    3) Abstraction  
    4) Encapsulation  
-------------