***Object Oriented Programming***

***OOP provide Following four Features.***

***1. Classes***

***2. Objects***

***3. Inheritance***

***4. Polymorphism***

***Note: Any language provide these four features called OOP C++, Java, PHP, .Net, VC++, C#, OOP languages***

***Class: Class is the collection of related methods and data.***

***class Calc{***

***void add(){***

***int a = 10;***

***int b = 5;***

***Println("Addition is: " + (a + b));***

***}***

***void mul(){***

***Scanner ob = new Scanner(System.in);***

***print("Enter 1st value: ");***

***int a = ob.nextInt();***

***print("Enter 2nd value: ");***

***int b = ob.nextInt();***

***println("Multiplication is: " + (a \* b));***

***}***

***void div(){***

***int a = 10;***

***int b = 5;***

***println("division is: " + (a / b));***

***}***

***}***

***class Test{***

***public static void main(String arg[]){***

***Clac ob = new Calc();***

***ob.add();***

***ob.mul();***

***ob.div();***

***}***

***}***

***Passing Arguments:***

***class Clac{***

***void add(int a, int b){***

***println("Addition is: " + (a + b));***

***}***

***void mul(int a, int b){***

***println("Multiplication is: " + (a + b));***

***}***

***void sqr(int num){***

***println("Square is: " + (num \* num));***

***}***

***void cube(int num){***

***println("cube: " + (num \* num \* num));***

***}***

***}***

***class TestArg{***

***public static void main(String arg[]){***

***Calc ob = new Calc();***

***ob.add(10,5);***

***ob.sub(10,7);***

***ob.Sqr(7);***

***ob.cube(10);***

***int a = 10;***

***int b = 5;***

***ob.add(a,b);***

***}***

***}***

***Objects:***

***What is Object?***

***1. In which data is in encapsulated form or is packed form called object.***

***2. Every object have its own memory.***

***3.The memory of object depend upon instance variables of the classes.***

***4.Object is the collection of Data(variables).***

***5.Object is the key of class through which one can access members of the class.***

***class Box{***

***int feet;***

***int inch;***

***}***

***class Box{***

***public static void main(string arg[]){***

***Box ob1 = new Box();***

***Box ob2 = new Box();***

***ob1.feet = 10;***

***ob1.inch = 5;***

***ob2.feet = 100;***

***ob2.inch = 50;***

***println(ob1.feet + ob2.feet);***

***println(ob1.inch + ob2.inch);***

***}***

***}***

***class Rectangle {***

***int x;***

***int y;***

***int width;***

***int height;***

***}***

***void add(){***

***Println("Addition is: " + (feet + inch));***

***}***

***class Box{***

***int feet;***

***int inch;***

***void set(int a, int b){***

***feet = a;***

***inch = b;***

***}***

***void show(){  
println("Feet: " + feet);***

***println("inch: " + inch);***

***}***

***}***

***class TestObj{***

***public static void main(String arg[]){***

***Box ob1 = new Box();***

***Box ob2 = new Box();***

***Box ob3 = new Box();***

***ob1.set(10,5);***

***ob.set(100,50);***

***ob.set(1000,500);***

***ob1.show();***

***ob2.show();***

***ob3.show();***

***}***

***}***

***class LoopSeries{***

***int num;***

***void set(){  
num = n;***

***}***

***void odd(){}***

***void even(){}***

***void factorial(){}***

***void fibonacii(){}***

***Constructor:***

***1.Constructor is special type of Method.***

***2.Constructor invoked once when object is created.***

***3.The main purpose of constructor is to initialize the object.***

***4.Constructor Name and class Name must be same.***

***5.Constructor have no return type.***

***class Box{***

***int feet;***

***int inches;***

***Box(int a, int b){***

***feet = a;***

***inches = b;***

***}***

***void set(int a, int b){***

***feet = a;***

***inch = b;***

***}***

***void show(){***

***println("Feet: " + feet);***

***println("Inches: " + inches);***

***}***

***}***

***class Test{***

***public static void main(String arg[]){***

***Box ob1 = new Box(10,5);***

***Box ob2 = new Box(100,500);***

***ob1.show();***

***ob2.show();***

***ob1.set(100, 500);***

***ob2.set(1000, 2000);***

***ob1.show();***

***ob2.show();***

***}***

***}***

***Constructor Overloading:***

***Same Constructor Name but passing Different type arguments called***

***constructor overloading.***

***class Box(){***

***int feet;***

***int inches;***

***Box(){***

***feet = 0;***

***inches = 0;***

***println("No arg.const-");***

***}***

***Box(int len){***

***Feet = len;***

***inches = len;***

***println("One arg-const");***

***}***

***Box(int a, int b){***

***feet = a;***

***inches = b;***

***println("Two arg.const-");***

***}***

***void show(){***

***println(Feet);***

***println(inch);***

***}***

***}***

***class TestConstOverloading{***

***public static void main(String arg[]){***

***Box ob1 = new Box();***

***Box ob2 = new Box(10);***

***Box ob3 = new Box(100,50);***

***ob1.show();***

***ob2.show();***

***ob3.show();***

***}***

***}***

***How Method Return Value:***

***class Calc{***

***int add(int a, int b){***

***return a+b;***

***}***

***}***

***class Testing{***

***public static void main(String arg[]){***

***Calc ob = new Calc();***

***int result = ob.add(10,5);***

***println("Addition: " + result);***

***}***

***}***

***int cube(int num){***

***int a = num \* num \* num;***

***return a;***

}

***int factorial(int num){***

***.***

***.***

***.***

***return f;***

***}***

***Boolean isPrimeNo(int num){}***

***double per(int obtain, int total){}***

***String grade(double per){}***

***String Seasson(int monthNo){}***

***int max(int a, int b, int c){}***

***int min(int a, int b, int c, int d, int e){}***

***int billCalc(int units){}***

***class Test{***

***public static void main(String arg[]){***

***for(int i=1; i<=50; i++){***

***if(isprimeNo(i))***

***println(i)***

***}***

***}***

***Object Pass As Arguments:***

***class Box{***

***int feet;***

***int inch;***

***Box(int a, int b){***

***feet = a;***

***inch = b;***

***}***

***}***

***class Display{***

***static void add(Box ob){***

***println(ob.feet + ob.inch);***

***}***

***static void add(Box ob1, Box ob2){***

***println(ob1.feet + ob2.feet);***

***println(ob1.inch + ob2.inch);***

***}***

***static void add(Box ob1, Box ob2, Box ob3){***

***println(ob1.feet + ob2.feet + ob3.feet);***

***println(ob1.inch + ob2.inch + ob3.inch);***

***}***

***static void show(Box ob){***

***println("Feet: " + feet);***

***println("Inch: " + inch);***

***}***

***}***

***class TestPassObject{***

***public static void main(String arg[]){***

***Box ob1 = new Box(10,5);***

***Box ob2 = new Box(7,3);***

***Box ob3 = new Box(100,5);***

***Display.add(ob1);***

***Display.add(ob2);***

***Display.add(ob3);***

***Display.show(ob1);***

***Display.show(ob2);***

***Display.show(ob3);***

***Display.add(ob1, ob2);***

***Display.add(ob2, ob2);***

***Display.add(ob3, ob3);***

***Display.add(ob1, ob2, ob3);***

***Display.add(ob1, ob1, ob1);***

***}***

***}***

***class Display{***

***void show(School std){***

***println("Name: " + std.name);***

***println("Subject: " + std.subject);***

***println("RollNo: " + std.rollNo);***

***}***

***}***

***School std1 = new School();***

***School std1 = new School();***

***School std1 = new School();***

***std1.set("Ali", "Java", "2k18/ITE/09");***

***std2.set("Imran", "Oracle", "2k18/ITE/38");***

***std3.set(....);***

***Display ob = new Display();***

***ob.show(std1);***

***ob.show(std2);***

***ob.show(std3);***

***How the method return Object:***

***class Box{***

***int feet;***

***int inch;***

***Box(){}***

***Box(int a, int b){***

***feet = a;***

***inch = b;***

***}***

***void show(){***

***-------***

***-------***

***}***

***}***

***class Display{***

***static Box add(Box ob1, Box ob2){***

***Box ob = new Box();***

***ob.feet = ob1.feet + ob2.feet;***

***ob.inch = ob1.inch + ob2.inch;***

***return ob;***

***}***

***}***

***class ReturnObjectDemo{***

***public static void main(String arg[]){***

***Box ob1 = new Box(10,5);***

***Box ob2 = new Box(100,50);***

***Box ob3 = Display.add(ob1, ob2);***

***ob3.show();***

***Box ob4 = Display.add(ob2, ob2);***

***ob4.show();***

***}***

***}***

***class Box{***

***int feet;***

***int inch;***

***Box(){***

***feet = 0;***

***inch = 0;***

***Box(int a, int b){***

***feet = a;***

***inch = b;***

***}***

***Box add(Box ob){***

***Box obj = new Box();***

***ob.feet = feet + ob.feet;***

***ob.inch = inch + ob.inch;***

***return obj;***

***}***

***Box add(Box ob1, Box ob2){***

***Box obj = new Box();***

***obj.feet = feet + ob1.feet + ob2.feet;***

***obj.inch = inch + ob1.inch + ob2.feet;***

***return obj;***

***}***

***void show(){***

***println("Feet: " + feet);***

***println("Inch: " + inch);***

***}***

***}***

***class ReturnDemo{***

***public static void main(String arg[]){***

***Box ob1 = new Box(10,5);***

***Box ob2 = new Box(100,50);***

***Box ob3 = ob1.add(ob2);***

***Box ob4 = ob1.add(ob1, ob2);***

***Box ob5 = ob1.add(ob1, ob1);***

***ob3.show();***

***ob4.show();***

***ob5.show();***

***}***

***}***

***class Rectangle{***

***int x;***

***int y;***

***int width;***

***int height***

***....***

***....***

***....***

***Static Keyword:***

***There are four uses of static keyword***

***1. static keyword.***

***2. static method.***

***3. static block.***

***4. static inner class.***

***Static Variable:***

***\* static variable is also known as class Variable.***

***\* static variable cover only one time memory.***

***\* when class first time load inside memory.***

***\* static variable working on sharing basis- single memory of static variable will be shared with all objects of the class.***

***\* static variable will be accessed by outside of the class through its class name.***

***\* No need to create object for accessing static variable.***

***\* Without object static variable can be accessed.***

***\* Object do not cover the memory of static variable.***

***class Box{***

***int feet; //Instance variable of feet***

***int inch; //Instance variable of inch***

***static int score; // class variable of score***

***}***

***class Test{***

***public static void main(String arg[]){***

***Box.score = 500;***

***println(Box.score);***

***Box ob1 = new Box();***

***Box ob2 = new Box();***

***ob1.feet = 10;***

***ob1.inch = 5;***

***ob2.feet = 100;***

***ob2.inch = 50;***

***println(ob1.feet); // 10***

***println(ob1.inch); // 5***

***println(ob1.score); // 500***

***println(ob2.feet); // 100***

***println(ob2.inch); // 50***

***println(ob2.score); // 500***

***ob1.score = 1000;***

***println(Box.score); // 1000***

***println(ob1.score); // 1000***

***println(ob2.score); // 1000***

***ob2.score = 5000;***

***println(Box.score); // 5000***

***println(ob1.score); // 5000***

***println(ob2.score); // 5000***

***}***

***}***

***Static Method:***

***\* Static method it will be invoked through its class name.***

***\* Static method invoked outside of the class without object.***

***\* Static method can not use Instance variables of the current***

***class.***

***\* Static method do not call other instance methods of the current***

***class.***

***\* Static Method do not use "this" keyword.***

***class Calc{***

***static void add(){***

***-------***

***-------***

***}***

***static void mul(){***

***-------***

***-------***

***}***

***static void sub(){***

***-------***

***-------}}***

***System class and Math class provide List of static Methods***

***double a = Math.sqr(10);***

***b = Math.pow(10,5);***

***c = Math.sin(10);***

***d = Math.cos(10);***

***class Test{***

***public static void main(String arg[]){***

***hello();***

***test();***

***}***

***static void hello(){  
println("Hello");***

***}***

***static void test(){***

***println("test");***

***}***

***}***

***this Keyword:***

***this keyword tells that "I am the object of current class"-***

***this keyword refer the object of current class by which method***

***invoked.***

***what is "this" keyword?***

***Answer is so easy - it is object***

***Local Variable:***

***Local Variable declare inside method- therefore it can be used inside that method only if means it can only be accessed within the method where it declear- when method terminate their execution local variable destory their memory.***

***class Box{***

***int feet; // instance variable***

***int inch; // instance variable***

***Box(int feet, int inch){ //local variables***

***this.feet = feet;***

***this.inch = inch;***

***}***

***void show(){***

***Display.add(this);***

***Display.add(this);***

***}***

***}***

***class TestThisKey{***

***public static void main(String arg[]){***

***Box ob1 = new Box(10,5);***

***Box ob2 = new Box(100,500);***

***ob1.show();***

***ob2.show();***

***}***

***}***

***class Display{***

***static void add(Box ob){***

***println(ob.feet + ob.inch);***

***}***

***static void sub(Box ob){***

***println(ob.feet - ob.inch);***

***}***

***static void mul(Box ob){***

***println(ob.feet + ob.inch);***

***}***

***}***

***"this" keyword don't use in static method as static method invoked through its class name and this keyword represent object of current class by which method has been invoked.***

***Static block:***

***static block invoked automatically when class load inside memory.***

***the main purpose of static block is to initialize the static variables.***

***static block once invoked in the life of program.***

***class Test{***

***static {***

***println("I am static block");***

***}***

***public static void main(String arg[]){***

***println("I am main Method");}***

***class Test{***

***static {***

***int a = sqr(10);***

***println("Sqr is: " + a);***

***}***

***public static void main(String arg[]){***

***println("I am main");***

***}***

***private static int sqr(int num){***

***println("I am sqr");***

***return (num \* num);***

***}***

***}***

***}***

***class Test{***

***static {***

***println("I am static");***

***}***

***class Testing{***

***public static void main(String arg[]){  
println("I am main");***

***Test ob = new Test();***

***}***

***}***

***class Test{***

***static int num = getValue();***

***static {***

***println("Num: " + num);***

***}***

***public static void main(String arg[]){  
println("I am main");***

***}***

***private static int getValue(){***

***println("I am getValue()");***

***return 10 \* 10;***

***}***

***}***

***Nested Classes:***

***Class within another class called nested class.***

***class A{***

***Methods***

***and variables.***

***class B{***

***-----***

***------***

***Note: Inner class can use methods and variables of Outer class***

***}***

***}***

***class Box{***

***int feet;***

***int inch;***

***Rectangle r1 = new Rectangle(100,200);***

***Rectangle r2 = new Rectangle(300,400);***

***Box(int a, int b){***

***feet = a;***

***inch = b;***

***}***

***static class Rectangle{***

***int width;***

***int height;***

***Rectangle(int w, int h){***

***width = w;***

***height = h;***

***}***

***void display(){***

***println("Feet: " + feet);***

***println("Inch: " + inch);***

***println("Width: " + width);***

***println("Height: " + height);***

***}***

***}//end inner class***

***}// end outer class***

***class TestNested {***

***public static void main(String arg[]){***

***Box ob1 = new Box(10,20);***

***Box ob2 = new Box(30,40);***

***ob1.r1.display();***

***ob1.r2.display();***

***ob2.r1.display();***

***ob2.r2.display();***

***}***

***}***

***Box.Rectangle r1 = new Box.Rectangle(100,500);***

***r1.display();***

***Static Inner class do not used Instance variable and***

***Instance Methods of its outer class.***

***Inheritance:***

***C ob = new C();***

***ob.a();***

***ob.b();***

***ob.c();***

***class A{***

***void a(){***

***println("I am a()");***

***}***

***}***

***class B extends A{***

***void b(){***

***println("I am b()");***

***}***

***}***

***class C extends B{***

***void c(){***

***println("I am c()");***

***}***

***}***

***In Object Oriented Programming class inherit the properties of Higher class called Inheritance.***

***\* Inheritance provide a way to modify the existing project/ or create new versions.***

***\* Inheritance provide similar concept of Father and Child.***

***\* Father class known Super class and child class known "sub" class.***

***In c++ Father class known as "Base" class and child class known Drived class.***

***class PhysicsAndTech{***

***void computerTehnology{***

***println("Three Year Computer Tech");***

***}***

***void electronics(){  
println("Three Years Electronics");***

***}***

***void teleCom(){***

***println(Three Year Telecom);***

***}***

***void lab(){***

***println("Labortory....");***

***}***

***}***

***class IICT extends PhysicsAndTech{***

***void IT(){***

***println("Four year IT");***

***}***

***void softEng(){***

***println("Four year Soft Eng");***

***}***

***void electronics(){ //overriding***

***super.electronics();***

***println("Four years Electronics");***

***}***

***void teleCom(){***

***super.teleCom(){***

***println("Four year telecom");***

***}***

***}***

***class Test{***

***public static void main(String arg[]){***

***IICT ob = new IICT();***

***ob.IT();***

***ob.softEng();***

***ob.lab();***

***ob.computerTechnology();***

***ob.teleCom();***

***ob.electronics();***

***}***

***}***

***How the constructor are invoked in Inheritance:***

***class A{***

***A(){***

***println("A");***

***}***

***}***

***class B extends A{***

***B(){***

***println("B");***

***}***

***}***

***class C extends B{***

***C(){***

***println("C");***

***}}***

***C ob = new C();***

***C ob2 = new C();***

***class A{***

***A(){***

***Println("No arg.const of A");***

***}***

***A(int a){***

***println("One arg const of A");***

***}***

***}***

***class B extends A{***

***}***

***class C extends B{***

***}***

***C ob1 = new C();***

***C ob2 = new C(10);***

***C ob3 = new C(100,200);***

***class A{***

***int feet;***

***int inch;***

***A(){***

***feet = 0;***

***inch = 0;}***

***A(int len){***

***feet = inch = len;***

***println("one arg.const of A");***

***}***

***A(int a, int b){***

***feet = a;***

***inch = b;***

***println("Two arg.const of A");***

***}***

***A(A ob){***

***feet = ob.feet;***

***inch = ob.inch;***

***println("Onj pass from A");***

***}***

***void show(){***

***println("Feet: " + feet);***

***println("Inch: " + inch);***

***}***

***}***

***class B extends A{***

***int width;***

***B(){***

***feet = inch = width = 0;***

***println("No arg.const of B");***

***}***

***B(int len){***

***feet = inch = width = len;***

***println("one arg.const of B");***

***}***

***B(int a, int b, int c){***

***feet = a;***

***inch = b;***

***width = c;***

***println("Three arg.const of B");***

***}***

***B(B ob){***

***feet = ob.feet;***

***inch = ob.inch;***

***width = width;***

***println("obj pass from B");***

***}***

***void show(){***

***println("Feet: " + feet);***

***println("inch: " + inch);***

***println("width: " + width);***

***}***

***}***

***class C extends B{***

***int height;***

***C(){  
feet = inch = width = height = 0;***

***println(no arg.const of C);***

***}***

***C(int len){***

***feet = inch = width = height = len;***

***println("one arg.const of C");***

***}***

***C(int a, int b, int c, int d){***

***feet = a;***

***inch = b;***

***width = c;***

***height = d;***

***println("Four arg.const of C");***

***}***

***C(C ob){***

***feet = ob.feet;***

***inch = ob.inch;***

***width = ob.width;***

***height = ob.height;***

***println("Obj pass from C");***

***}***

***void show(){ //overriding  
println("Feet: " + feet);***

***println("Inch: " + inch);***

***println("width: " + width);***

***println("height: " + height);***

***}***

***}***

***C ob1 = new C();***

***C ob2 = new C(10);***

***C ob3 = new C(100, 200, 300, 400);***

***C ob5 = new C(ob3);***

***ob1.show();***

***ob2.show();***

***ob3.show();***

***ob4.show();***

***Dynamic Method Dispatching(Runtime Polymorphism)***

***Dynamic = Runtime;***

***Dispatching = calling;***

***\*It is also called Runtime Polymorphism.***

***\*It is a mechanism through which one can invoked overridden***

***method at runtime through super object.***

***class Animal {***

***void show(){***

***println("Sound...");***

***void eyes(){***

***println("two eyes");***

***}***

***void legs(){***

***println("Four legs");***

***}***

***}***

***class Cat extends Animal{***

***void sound(){***

***println("Mayaoon miyaoon mimum..");***

***}***

***}***

***class Dog extends Animal{***

***void sound(){***

***println("Wawo wawoo wwa")***

***}***

***}***

***class Human extends Animal{***

***void sound(){***

***println("Natural lang..");***

***}***

***void legs(){***

***println("Two legs");***

***}***

***}***

***class DynamicMethodDis{***

***public static void main(String arg[]){***

***Cat c = new Cat();***

***Dog d = new Dog();***

***Human h = new Human();***

***Extra Thing:***

***Animal a = d; // reference variable***

***a.sound();***

***a.eyes();***

***a.legs();***

***a=d; a = h;***

***a.sound(); a.sound();***

***a.eyes(); a.eyes();***

***a.legs(); a.legs();***

***------------------------------------------***

***display.show(c);***

***display.show(d);***

***display.show(h);***

***class Display{***

***static void show(Animal a){***

***a.sound();***

***a.eyes();***

***a.legs();***

***}***

***}***

***Abstraction:***

***abstract class usindh{***

***abstract void BBA();***

***abstract void IT();***

***}***

***class Imsa extends Usindh{***

***void BBA(){***

***println("Imsa BBA");***

***}***

***void IT(){***

***println("Imsa IT");***

***}***

***}***

***class AbsDemo{***

***public static void main(String arg[]){  
Imsa im = new Imsa();***

***Cms cm = new Cms();***

***Mit m = new Mit();***

***Display.show(in);***

***Display.show(cm);***

***Display.show(m);***

***class Cms extends Usindh{***

***void BBA(){***

***println("Cms BBA");***

***}***

***void IT(){  
println("Cms IT");***

***}***

***}***

***class Mit extends Usindh{***

***void BBA(){  
println("mit BBA");***

***}***

***void IT(){***

***println("mit It");***

***}***

***}***

***class Display{***

***static void show(Usindh u){***

***u.BBA();***

***u.IT();***

***}***

***}***

***\*Abstraction class is a incomplete class because abstract class contain***

***abstract methods.***

***\*Abstract class do not create its object.***

***\*The subclass of abstract class must override abstract methods of its super class(Abstraction class).***

***\*Abstract Methods have No body.***

***\*Abstract Methods also known prototype Method.***

***\*In Abstract class Some methods are completely defined and some methods are abstract.***

***abstract class Animal{***

***abstract void sound(){***

***void eyes(){***

***println("Two Eyes");}***

***void legs(){***

***println("Four legs");***

***}***

***}***

***class cat extends Animal{***

***.....***

***,,,,,***

***Final Keyword:***

***Final keyword is used to create constants.***

***There are three uses of Final keyword.***

***1.final variable.***

***2.final method.***

***3.final class.***

***Final Vs Abstraction:***

***\*Final keyword break the Inheritance .***

***\*Final method do not override in sub class while abstract method must override in sub class .***

***\*Final class do not create sub class.***

***\*while abstract class must create its sub class.***

***Final Variable:***

***Final Variable only one time initialized.***

***class Test{***

***public static void main(String arg[]){***

***final int a = 10;***

***println(a \* a);***

***a+=10;//error***

***a = 50;//error***

***println(a);***

***}***

***}***

***Final variable do not change its value.***

***final int a;***

***int b = 10;***

***println(b \* b);***

***a = 10; //ok;***

***a = 50; // error***

***class Box{***

***int feet;***

***int inch;***

***Box(int a, int b){***

***feet = a;***

***inch = b;***

***}***

***}***

***Final Box ob1 = new Box(10,5);***

***ob1.feet = 100; //ok***

***ob1.inch = 50; //ok***

***//ob1 = new Box(10,5) //error***

***ob1 = ob2; // error***

***Interface:***

***\*Interface is pure abstract.***

***\*Interface is similar to class.***

***\*Bydefault all variables of interface are abstract, public.***

***\*Bydefault all variables of the interface are public, static and final***

***\*Class who have implemented any interface is responsible to override all methods of the interface.***

***interface Calc{***

***void add(int a, int b);***

***void sub(int a, int b);***

***void mul(int a, int b);***

***}***

***class Abc implements Clac{***

***public void add(int a, int b){***

***println("Addition: " + (a + b));***

***}***

***public void sub(int a, int b){***

***println(a-b);***

***}***

***public void mul(int a, int b){***

***println(a \* b);***

***}***

***}***

***//optional***

***abstract interface Calc{***

***//optional***

***public abstract void add(int , int );***

***}***

***\*class can implements two or more interfaces at a time. But class***

***do not extends two or more classes at a time.***

***interface A{***

***void a();***

***}***

***interface B{***

***void b();***

***}***

***class AB implements A,B{***

***public void a(){***

***----------***

***----------***

***----------***

***}***

***public void b(){***

***--------***

***-------***

***}***

***}***

***interface A{***

***void a();***

***}***

***interface B extends A{***

***void b();***

***}***

***interface C extends B{***

***void C();***

***}***

***class ABC impements C{***

***public void a(){***

***----***

***---***

***--***

***}***

***public void b(){***

***---***

***}***

***public void c(){***

***----***

***}***

***}//end class***

***Note: interface can extends other interface.***

***interface A{***

***void a();***

***}***

***interface B{***

***void b();***

***}***

***interface C extends A,B{***

***void c();***

***}***

***class Abc implements C{***

***public void a(){-----}***

***public void b(){-----}***

***public void c(){-----}***

***}***

***final class Calc{***

***int sqr(int num){***

***return num\*num;***

***}}***

***class Abc extends Calc{***

***//error***

***//final class do not***

***create its subclass.***

***}***

***class A{***

***----***

***----***

***}***

***class B extends A{***

***----***

***----***

***}***

***final class C extends B{***

***------------***

***-----------***

***}***

***class Test{***

***final int sqr(int num){***

***return num \* num;***

***}***

***final int cube(int num){***

***return num\*num\*num;***

***}***

***}***

***class XYZ extends Test{***

***/\* here final method of its super class***

***will not be override in sub.***

***}***

***class Test{***

***public static void main(String arg[]){***

***final int data[] = {90,60,70,80};***

***data[0]=100; //ok***

***data[1]=200; //ok***

***data[2]=300; //ok***

***data[3]=400; //ok***

***data[0] = 1000; //ok***

***data = new int[10]; //error***

***Final Array don't change its memory space or size.***

***class Display{***

***static void show(A a){***

***a.a();***

***}***

***static void show(B b){***

***b.a();***

***b.b();***

***}***

***static void show(C c){***

***c.a();***

***c.b();***

***c.c();***

***}***

***}***

***Abc ob = new Abc();***

***// Display.show(ob);***

***Display.show((A) ob);***

***Display.show((B) ob);***

***or***

***Display.show1(ob); // change methods name***

***Display.show2(ob);***