***Dt : 29/9/2022***

***\*imp***

***Sorting process on User defined class Objects:***

***step-1 : The user defined class must be implemented from 'Comparable'***

***interface***

***step-2 : User defined class must construct body for compareTo()***

***method and which is abstract method of 'Comparable<T>'***

***interface***

***step-3 : This compareTo() method is declared with***

***sorting-specification-logic***

***step-4 : Execute Arrays.sort() method to perform sorting process.***

***Ex:***

***Product.java***

***package test;***

***@SuppressWarnings("rawtypes")***

***public class Product implements Comparable***

***{***

***public String code,name;***

***public float price;***

***public int qty;***

***public Product(String code,String name,float price,int qty)***

***{***

***this.code=code;***

***this.name=name;***

***this.price=price;***

***this.qty=qty;***

***}***

***public String toString()***

***{***

***return code+"\t"+name+"\t"+price+"\t"+qty;***

***}***

***public int compareTo(Object o)***

***{***

***Product p = (Product)o;***

***if(price==p.price) return 0;***

***else if(price>p.price) return 1;***

***else return -1;***

***}***

***}***

***DemoInterface10.java(MainClass)***

***package maccess;***

***import java.util.\*;***

***import test.Product;***

***public class DemoInterface10 {***

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

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

***System.out.println***

***("Enter the size of Array to hold Product Objects");***

***int size = Integer.parseInt(s.nextLine());***

***Product a[] = new Product[size];***

***System.out.println("Enter "+size+" ProductDetails :");***

***for(int i=0;i<a.length;i++)***

***{***

***System.out.println("===Product-"+(i+1)+"====");***

***System.out.println("Enter the ProdCode:");***

***String code = s.nextLine();***

***System.out.println("Enter the ProdName:");***

***String name = s.nextLine();***

***System.out.println("Enter the ProdPrice:");***

***float price = Float.parseFloat(s.nextLine());***

***System.out.println("Enter the ProdQty:");***

***int qty = Integer.parseInt(s.nextLine());***

***a[i] = new Product(code,name,price,qty);***

***}//end of loop***

***System.out.println("===Display before Sorting====");***

***for(Product k : a)***

***{***

***System.out.println(k.toString()+" ");***

***}//end of loop***

***System.out.println("===Display after Sorting====");***

***Arrays.sort(a);//Sorting process***

***for(Product k : a)***

***{***

***System.out.println(k.toString()+" ");***

***}//end of loop***

***s.close();***

***}***

***}***

***o/p:***

***Enter the size of Array to hold Product Objects***

***3***

***Enter 3 ProductDetails :***

***===Product-1====***

***Enter the ProdCode:***

***A121***

***Enter the ProdName:***

***Mou***

***Enter the ProdPrice:***

***123.67***

***Enter the ProdQty:***

***12***

***===Product-2====***

***Enter the ProdCode:***

***A001***

***Enter the ProdName:***

***CDR***

***Enter the ProdPrice:***

***98.67***

***Enter the ProdQty:***

***10***

***===Product-3====***

***Enter the ProdCode:***

***A111***

***Enter the ProdName:***

***FDD***

***Enter the ProdPrice:***

***110.10***

***Enter the ProdQty:***

***10***

***===Display before Sorting====***

***A121 Mou 123.67 12***

***A001 CDR 98.67 10***

***A111 FDD 110.1 10***

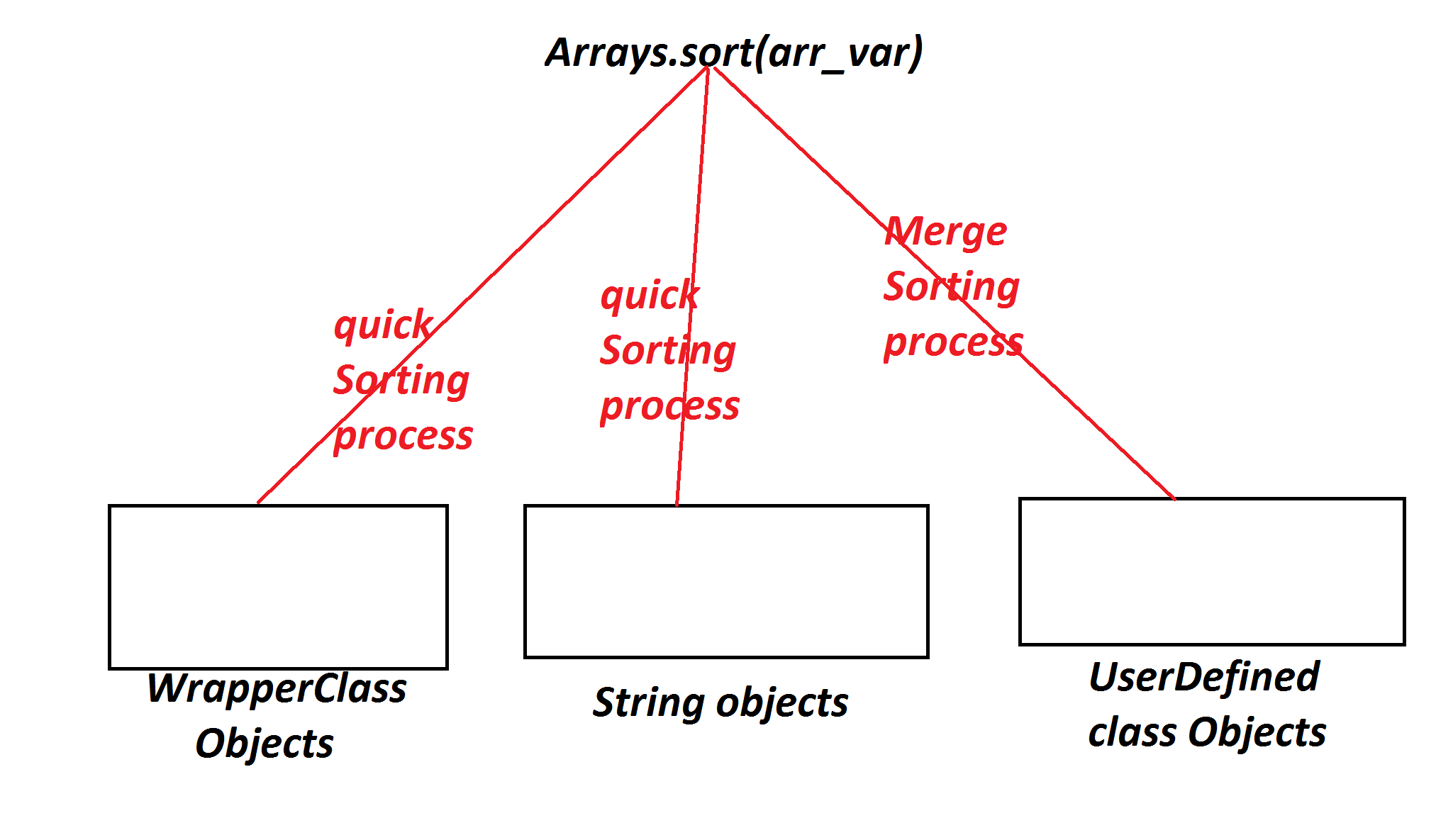
***===Display after Sorting====***

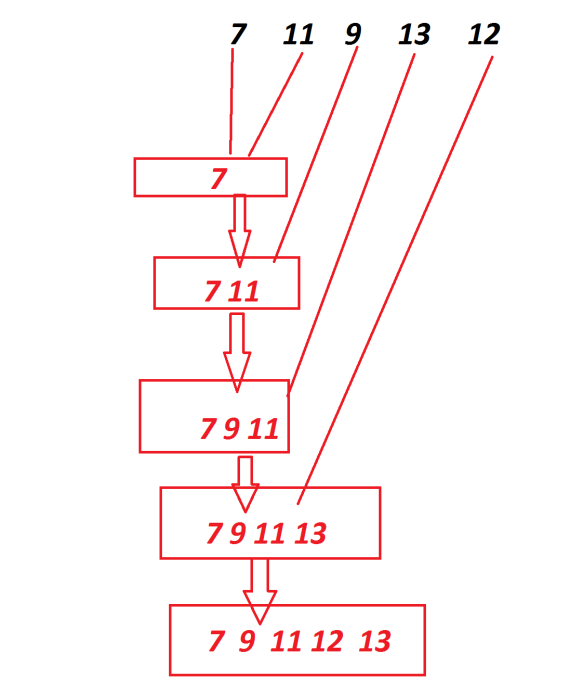
***A001 CDR 98.67 10***

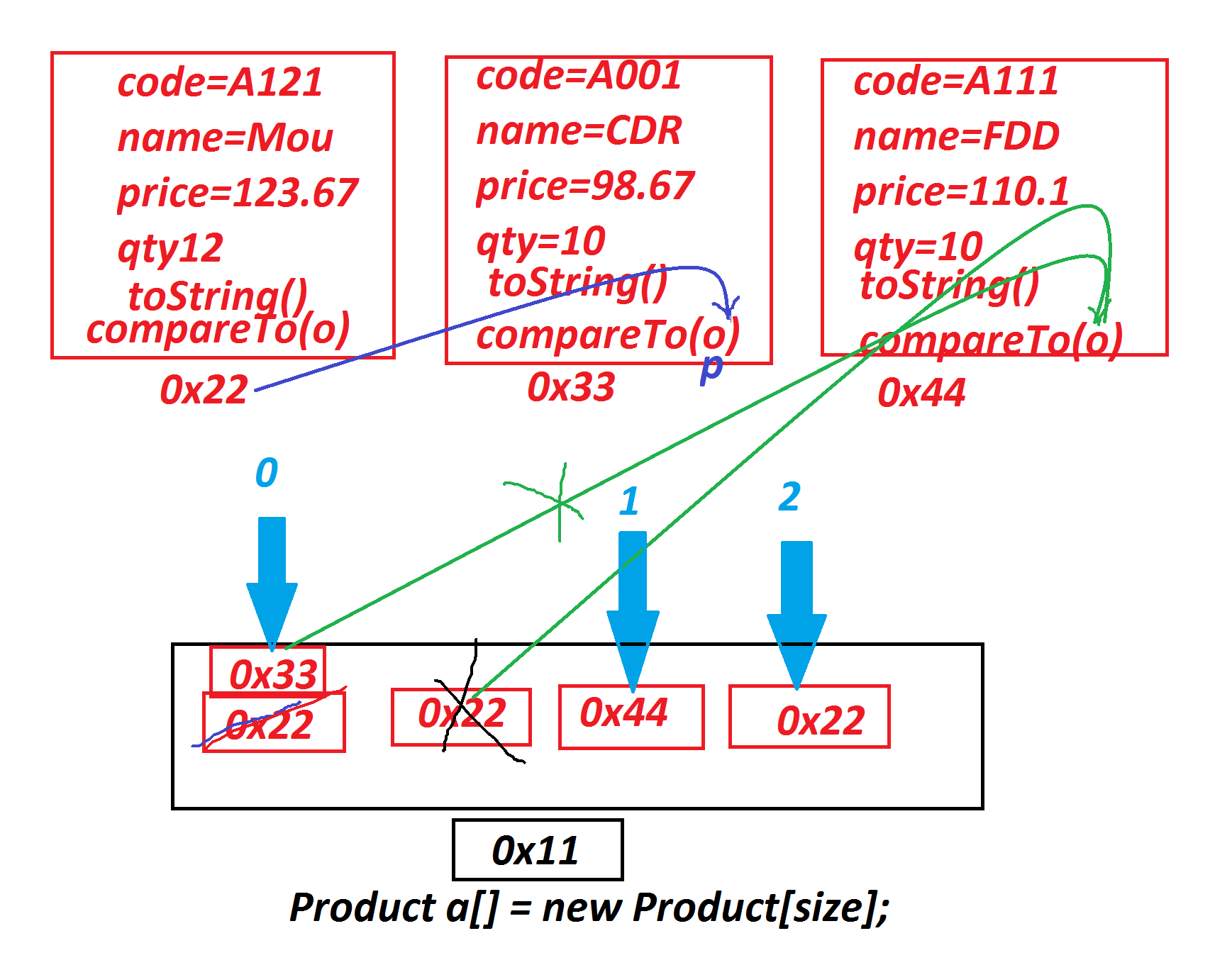
***A111 FDD 110.1 10***

***A121 Mou 123.67 12***

***Diagram:***

******

******

******

***==============================================================***

***Assignment:***

***Contruct application to display Multiple Student details in Sorting***

***order based on percentages.***

***==============================================================***

***\*imp***

***Abstract Classes in Java:***

***=>The classes which are declared with "abstract" keyword are***

***known as Abstract Classes.***

***=>AbstractClasses can hold Variables,abstract methods,Concrete***

***methods,blocks,constructors and features.***

***=>AbstractClasses cannot be instantiated,which means we cannot***

***create object for abtract classes.***

***=>These abstract classes are extended to classes using "extends"***

***keyword and the classes are known as extention classes or***

***implemented classes.***

***=>These extention classes must construct body for all the abstract***

***methods of Abstract classes.***

***Ex:***

***AClass.java***

***package test;***

***public abstract class AClass {***

***public void m1(int x) {***

***System.out.println("====Concrete method m1(x)====");***

***System.out.println("The value x:"+x);***

***}***

***public abstract void m2(int y);***

***}***

***EClass.java***

***package test;***

***public class EClass extends AClass{***

***public void m2(int y) {***

***System.out.println("====abstract m2(y)====");***

***System.out.println("The value y:"+y);***

***}***

***}***

***DemoAbstractClass.java(MainClass)***

***package maccess;***

***import test.\*;***

***public class DemoAbstractClass {***

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

***EClass ob = new EClass();***

***ob.m1(12);***

***ob.m2(13);***

***}***

***}***

***o/p:***

***====Concrete method m1(x)====***

***The value x:12***

***====abstract m2(y)====***

***The value y:13***

***==========================================================***

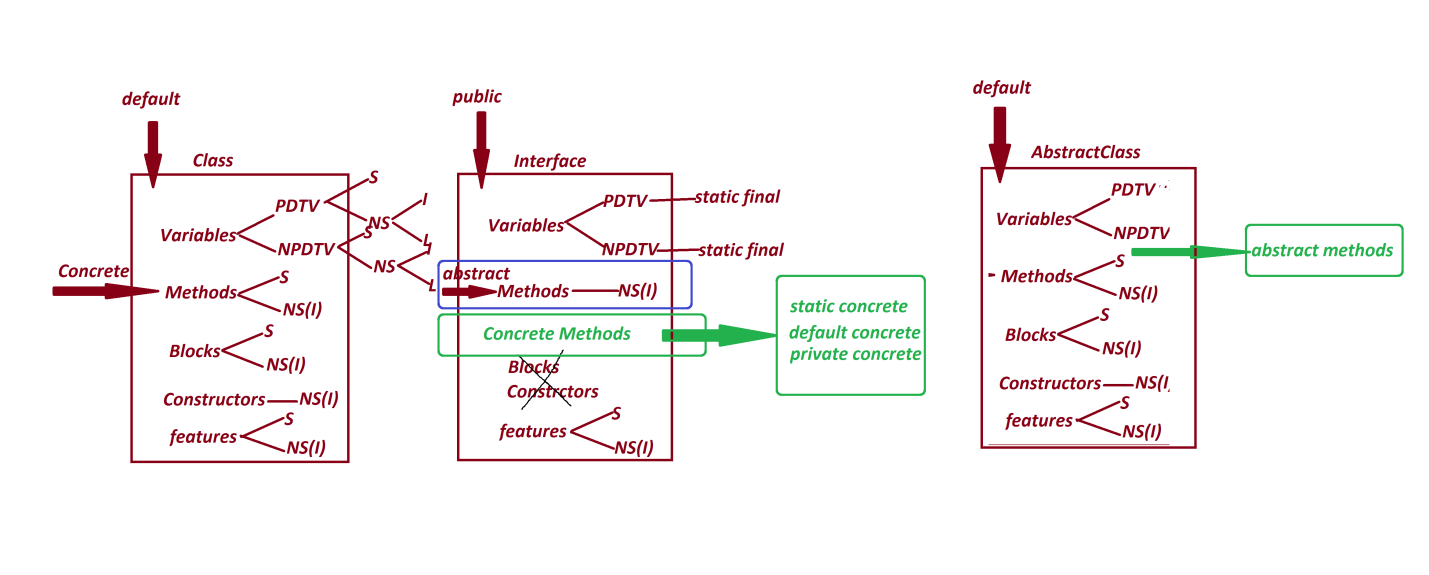
***Note:***

***=>The abstract methods within the abstract classes must be declared***

***with "abstract" keyword.***

***============================================================***

***Comparasion Diagram:***

******

***===========================================================***

***faq:***

***wt is the diff b/w***

***(i)Class***

***(ii)AbstractClass***

***=>Class can hold only concrete methods,but abstract classes can hold***

***both abstract methods and Concrete methods.***

***=>Class can be instantiated,but AbstractClasses cannot be***

***instantiated.***

***faq:***

***wt is the diff b/w***

***(i)Interface***

***(ii)AbstractClass***

***=>Interface cannot hold blocks and Constructors,but abstract class***

***can hold both blocks and constructors.***

***=>In Interfaces "abstract" keyword is not manditory to declare***

***abstract methods,but "abstract" keyword is manditory in abstract***

***classes to declare abstract methods.***

***=>methods in interfaces are automatically abstract,but methods in***

***abstract classes are automatically concrete methods.***

***=>Variables in interfaces are automatically static and final,but***

***Variables in AbstractClasses are user choice.***

***===============================================================***