**Method Overloading**

**public** **class** Eg1 {

/\*public static int max(int num1, int num2)

{

System.out.println("I");

if (num1 > num2)

return num1;

else

return num2;

}\*/

**public** **static** **double** max(**double** num1,**double** num2)

{

System.***out***.println("II");

**if** (num1 > num2)

**return** num1;

**else**

**return** num2;

}

**static** **int** max (**int** x, **int** y, **int** z) // The function (module) accepts x,y,z

{ // The function returns the largest number

System.***out***.println("III");

**if** (( x < y) && ( z < y))

**return** (y) ; // y is the largest number

**else** **if** ((y > z) && (z < x))

**return** (x); // x is the largest number

**else**

**return** (z); // z is the largest number

}

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

System.***out***.println(*max*(10,20));

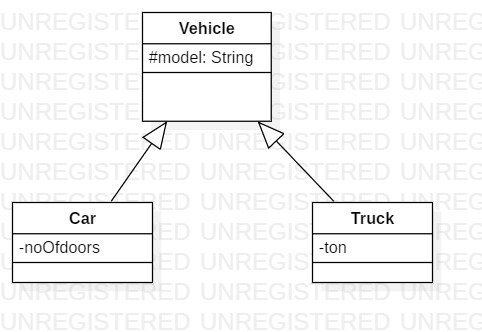
System.***out***.println(*max*(10,12,13));

//Error --> System.out.println(max(10.5,12.6,13.3));

}

}

**Inheritance**

****

**class** Vehicle

{

String model;

**public** Vehicle(String m) {

model=m;

}

}

**class** Car **extends** Vehicle

{ **int** noOfdoors;

Car(**int** d,String m)

{

**super**(m);

noOfdoors=d;

}

}

**class** Truck **extends** Vehicle

{

**int** ton;

**public** Truck(**int** t,String m) {

**super**(m);

ton=t;

}

}

**public** **class** Test {

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

Car LX=**new** Car(4,"2014");

Truck T=**new** Truck(10,"2015");

System.***out***.println("Car model:"+LX.model);

System.***out***.println("Number of doors:"+LX.noOfdoors);

System.***out***.println("Truck model:"+T.model);

System.***out***.println("Number of tons:"+T.ton);

System.***out***.println();

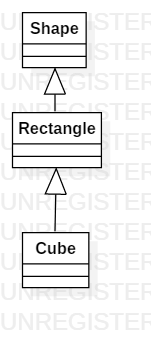
}

}

**Multilevel Inheritance**

class Shape {

   public void display() {



      System.out.println("Inside display");

   }

}

class Rectangle extends Shape {

   public void area() {

      System.out.println("Inside area");

   }

}

class Cube extends Rectangle {

   public void volume() {

      System.out.println("Inside volume");

   }

}

public class Tester {

   public static void main(String[] arguments) {

      Cube cube = new Cube();

      cube.display();

      cube.area();

      cube.volume();

   }

}

Output

Inside display

Inside area

Inside volume

**Multiple Inheritance**

|  |
| --- |
| // First Parent class  **class** Parent1  {  **void** fun()      {          System.out.println("Parent1");      }  }    // Second Parent Class  **class** Parent2  {  **void** fun()      {          System.out.println("Parent2");      }  }    // Error : Test is inheriting from multiple  // classes  **class** Test **extends** Parent1, Parent2  {  **public** **static** **void** main(String args[])     {         Test t = **new** Test();         t.fun();     }  } |

Output : Compile Error

|  |
| --- |
| interface PI1  {     // default method      default void show()      {          System.out.println("Default PI1");      }  }    interface PI2  {      // Default method      default void show()      {          System.out.println("Default PI2");      }  }    // Implementation class code  class TestClass implements PI1, PI2  {      // Overriding default show method      public void show()      {          // use super keyword to call the show          // method of PI1 interface          PI1.super.show();            // use super keyword to call the show          // method of PI2 interface          PI2.super.show();      }      public static void main(String args[])      {          TestClass d = new TestClass();          d.show();      }  } |

Output:

Default PI1

Default PI2

**public** **abstract** **class** Animal {

**private** String nameOfAnimal;

**public** **abstract** **void** sound();

**public** String getAnimalName(){

**return** nameOfAnimal;

}

**public** **void** setAnimalName(String name){

nameOfAnimal = name;

}

}

**public** **class** Dog **extends** Animal{

**public** **void** sound() {

System.*out*.println("Woof!");

}

}

**public** **class** Cow **extends** Animal {

**public** **void** sound() {

System.*out*.println("Moo!");

}

}

**public** **class** Snake **extends** Animal{

**public** **void** sound() {

System.*out*.println("Ssss!");

}

}

**public** **class** Cat **extends** Animal {

**public** **void** sound() {

System.*out*.println("Mewn!");

}

}

**public** **class** UseAnimals{

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

Dog myDog = **new** Dog();

Cow myCow = **new** Cow();

Snake mySnake = **new** Snake();

Cat myCat=**new** Cat();

myDog.setAnimalName ("My dog Murphy");

myCow.setAnimalName ("My cow Elsie");

mySnake.setAnimalName ("My snake Sammy");

myCat.setAnimalName("My cat ShweWah");

System.*out*.print(myDog.getAnimalName() + "says " );

myDog.sound();

System.*out*.print(myCow.getAnimalName() + "says ");

myCow.sound();

System.*out*.print(mySnake.getAnimalName() + "says " );

mySnake.sound();

System.*out*.print(myCat.getAnimalName() + "says " );

myCat.sound();

}

}

**public** **abstract** **class** GeoShape {

**private** String color="red";

**public** GeoShape(String color2) {

**this**.color=color2;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

**public** String getColor() { **return** color; }

**public** **abstract** **double** calculateArea();

**public** **abstract** **double** calculatePerimeter();

**public** **abstract** String Identity();

}

**public** **class** Triangle **extends** GeoShape {

**private** **double** base;

**private** **double** height;

**private** **double** side1;

**private** **double** side2;

**public** Triangle() {

**this**(1.0,1.0,1.0,1.0);

}

**public** Triangle(**double** base,**double** side1,**double** side2,**double** height){

**this**(base,side1,side2,height,"red");

}

**public** Triangle(**double** base,**double** side1,**double** side2,**double** height,String color){

**super**(color);

**this**.base=base;

**this**.side1=side1;

**this**.side2=side2;

**this**.height=height;

}

**public** **void** setBase(**double** base) {

**this**.base = base;

}

**public** **double** getBase() {

**return** base;

}

**public** **void** setHeight(**double** height) {

**this**.height = height;

}

**public** **double** getHeight() {

**return** height;

}

**public** **void** setSide1(**double** side1) {

**this**.side1 = side1;

}

**public** **double** getSide1() {

**return** side1;

}

**public** **void** setSide2(**double** side2) {

**this**.side2 = side2;

}

**public** **double** getSide2() {

**return** side2;

}

@Override

**public** **double** calculateArea() {

r**eturn** 0.5\*base\*height;

}

@Override

**public** **double** calculatePerimeter() {

**return** base+side1+side2;

}

@Override

**public** String Identity() {

r**eturn** **super**.getColor()+"Triangle";

}

**public** String toString()

{

**return** "Triangle->base="+base+"height="+height+"side1="+side1+"side2="+side2;

}

}

**public** **class** Rectangle **extends** GeoShape {

**private** **double** length;

**private** **double** breadth;

**public** Rectangle() {

**this**(1.0,1.0);

}

**public** Rectangle(**double** breadth,**double** length){

**this**(breadth,length,"blue");

}

**public** Rectangle(**double** breadth,**double** length,String color){

**super**(color);

**this**.breadth=breadth;

**this**.length=length;

}

**public** **void** setLength(**double** length) {

**this**.length = length;

}

**public** **double** getLength() {

**return** length;

}

**public** **void** setBreadth(**double** breadth) {

**this**.breadth = breadth;

}

**public** **double** getBreadth() {

**return** breadth;

}

**public** **double** calculateArea() {

**return** breadth\*length;

}

**public** **double** calculatePerimeter() {

**return** 2\*(breadth\*length);

}

@Override

**public** String Identity() {

**return** **super**.getColor()+"Rectangle";

}

**public** String toString()

{

**return** "Rectangle->breadth"+breadth+"and length="+length;

}

}

**import** java.util.LinkedList;

**public** **class** Exercise {

**static** **void** display(GeoShape shape){

System.*out*.println();

System.*out*.println(shape.toString());

System.*out*.println("The area of"+shape.Identity()+"is"+shape.calculateArea());

System.*out*.println("The perimeter is"+shape.calculatePerimeter());

}

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

GeoShape s1=**new** Rectangle(9.0,4.0);

GeoShape s2=**new** Triangle(6.0,4.0,5.0,5.0);

LinkedList sl=**new** LinkedList();

sl.add(0,s1);

sl.add(1,s2);

**for**(**int** i=0;i<sl.size();i++)

*display*((GeoShape)sl.get(i));

}

}

**import** java.util.Comparator;

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

**abstract** **class** Employee {

**private** String name;

**private** **int** age;

**private** **int** basicSalary;

**public** Employee() {

}

**public** Employee(String name, **int** age, **int** basicSalary) {

**this**.name = name;

**this**.age = age;

**this**.basicSalary = basicSalary;

}

**public** String getName() {

**return** **this**.name;

}

**public** **int** getAge() {

**return** **this**.age;

}

**public** **int** getBasicSalary() {

**return** **this**.basicSalary;

}

}

**class** SalesPerson **extends** Employee **implements** Comparable {

**private** **int** commission;

**public** SalesPerson() {

}

**public** SalesPerson(String name, **int** age, **int** basicSalary, **int** commission) {

**super**(name, age, basicSalary);

**this**.commission = commission;

}

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

**if** (**this**.getAge() < ((SalesPerson) o).getAge())

**return** -1;

**else** **if** (**this**.getAge() > ((SalesPerson) o).getAge())

**return** 1;

**return** 0;

}

**public** **int** getCommission() {

**return** **this**.commission;

}

}

**class** SortPersonByName **implements** Comparator {

**public** **int** compare(Object o1, Object o2) {

SalesPerson p1 = (SalesPerson) o1;

SalesPerson p2 = (SalesPerson) o2;

**return** p1.getName().toLowerCase().compareTo(p2.getName().toLowerCase());

}

}

**class** SortPersonByWage **implements** Comparator {

**public** **int** compare(Object o1, Object o2) {

SalesPerson p1 = (SalesPerson) o1;

SalesPerson p2 = (SalesPerson) o2;

**int** wage1 = p1.getBasicSalary() + p1.getCommission();

**int** wage2 = p2.getBasicSalary() + p2.getCommission();

**return** ((wage1 < wage2) ? -1 : (wage1 > wage2) ? 1 : 0);

}

}

**public** **class** Chap10InterfaceExampleSerial {

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

SalesPerson personArr[] = **new** SalesPerson[5];

personArr[0] = **new** SalesPerson("John", 25, 1500, 2300);

personArr[1] = **new** SalesPerson("Maru", 18, 3000, 3000);

personArr[2] = **new** SalesPerson("Jack", 15, 600, 6000);

personArr[3] = **new** SalesPerson("Billy", 40, 4000, 1500);

personArr[4] = **new** SalesPerson("Kit", 32, 6800, 400);

System.out.println("Before sorting");

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

System.out.println(personArr[i].getName() + "\t:"

+ personArr[i].getAge() + "\t:"

+ personArr[i].getBasicSalary() + "\t:"

+ personArr[i].getCommission());

}

Arrays.sort(personArr);

System.out.println("after sorting by age");

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

System.out.println(personArr[i].getName() + "\t:"

+ personArr[i].getAge() + "\t:"

+ personArr[i].getBasicSalary() + "\t:"

+ personArr[i].getCommission());

}

System.out.println("sort by name \n");

Arrays.sort(personArr, **new** SortPersonByName());

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

System.out.println(personArr[i].getName() + "\t:"

+ personArr[i].getAge() + "\t:"

+ personArr[i].getBasicSalary() + "\t:"

+ personArr[i].getCommission());

}

System.out.println("sort by wages \n");

Arrays.sort(personArr, **new** SortPersonByWage());

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

System.out.println(personArr[i].getName() + "\t:"

+ personArr[i].getAge() + "\t:"

+ personArr[i].getBasicSalary() + "\t:"

+ personArr[i].getCommission());

}

}

}