1.public class Main {  
 public static void main(String[]args ){  
 Room room = new Room (115,"public",120.18,6);  
 room.display();  
 }  
}

public class Room {  
 public int roomno;  
 public String roomtype;  
 public double roomarea;  
 public int ACmachine;  
 public Room(int rn,String rt, double ra,int AC)  
 {  
 this.roomno=rn;  
 this.roomtype = rt;  
 this.roomarea=ra;  
 this.ACmachine=AC;  
 }  
 public void display()  
 {  
 System.out.println("Room No : "+roomno+"\n"+"Room type : "+roomtype+"\n"+"Room Area : "+ roomarea+ "Square fit"+"\n"+"AC machines : "+ACmachine);  
 }  
}

2.public class Main {  
 public static void main(String[] args) {  
 SimpleObject simple=new SimpleObject();  
 }  
}

public class SimpleObject {  
 public SimpleObject()  
 {  
 System.out.println("Hello, simple object is created");  
 }  
}

3.public class Main {  
 public static void main(String[] args) {  
 ExampleOfThis th=new ExampleOfThis(10);  
 th.display();  
 }  
}

public class ExampleOfThis{  
 public int x;  
 public ExampleOfThis(int x)  
 {  
 this.x=x;  
 }  
 public void display()  
 {  
 System.out.println("X = "+x);  
 }  
}

4.public class Animal {  
 Animal() {  
 System.out.println("animal is created");  
 }  
}  
 class Cat extends Animal{  
 Cat(){  
 super();  
 System.out.println("cat is created");  
 }  
 }   
 class Super {  
 public static void main (String arg []){  
 Cat c = new Cat ();  
   
 }  
 }

5.public class Adder {  
 static int add(int a, int b) {  
 return a + b;  
 }  
  
 static int add(int a, int b, int c) {  
 return a + b + c;  
 }  
}  
 class methodoverloading {  
 public static void main (String[] args){  
 System.out.println(Adder.add(9,9));  
 System.out.println(Adder.add(9,9,9));  
 }  
 }

6.public class A {  
 public void display()  
 {  
 System.out.println("Overriding : Parents Class");  
 }  
}  
  
  
public class B extends A{  
 @Override  
 public void display() {  
 super.display();  
 System.out.println("\n"+"Overriding : Child Class");  
 }  
}  
  
  
public class Main {  
 public static void main(String[] args) {  
 B b = new B();  
 b.display();  
 }  
}

7.public class Circle extends Shape{  
 @Override  
 public void draw() {  
 super.draw();  
 System.out.println("Draw a circle");  
 }  
  
 @Override  
 public void erase() {  
 super.erase();  
 System.out.println("Erase the circle");  
 }  
}  
  
  
public class Main {  
 public static void main(String[] args) {  
 Circle c=new Circle();  
 c.draw();  
 c.erase();  
 System.out.println("\n");  
 Triangle t= new Triangle();  
 t.draw();  
 t.erase();  
 System.out.println("\n");  
 Square s= new Square();  
 s.draw();  
 s.erase();  
 }  
}  
  
  
public class Shape {  
 public void draw()  
 {  
 System.out.println("Draw a shape");  
 }  
 public void erase()  
 {  
 System.out.println("Erase the shape");  
 }  
}  
  
  
public class Square extends Shape{  
  
 @Override  
 public void draw() {  
 super.draw();  
 System.out.println("Draw a square");  
 }  
  
 @Override  
 public void erase() {  
 super.erase();  
 System.out.println("Erase the square");  
 }  
}  
  
  
public class Square extends Shape{  
  
 @Override  
 public void draw() {  
 super.draw();  
 System.out.println("Draw a square");  
 }  
  
 @Override  
 public void erase() {  
 super.erase();  
 System.out.println("Erase the square");  
 }  
}

8.abstract class Bike{  
 abstract void run();  
}  
class Honda4 extends Bike {  
 void run() {  
 System.out.println("running safely");  
 }  
  
 public static void main(String args[]) {  
 Bike obj = new Honda4();  
 obj.run();  
 }  
}

9.public interface A {  
 public void meth1();  
 public void meth2();  
}  
public class MyClass implements A{  
 @Override  
 public void meth1() {  
 System.out.println("Implements the method1");  
 }  
 @Override  
 public void meth2() {  
 System.out.println("Implements the method2");  
 }  
 public static void main(String[]args){  
 MyClass myc = new MyClass();  
 myc.meth1();  
 myc.meth2();  
 }  
}

10.interface My\_restaurents {  
 void eat();  
}  
interface My\_journey {  
 void travel();  
}  
class Holiday implements My\_restaurents, My\_journey {  
 public void eat() {  
 System.out.println("I am trying this food");  
 }  
 public void travel() {  
 System.out.println("I am trying this route");  
 }  
}  
public class My\_trip {  
 public static void main(String args[]) {  
 Holiday my\_schedule = new Holiday();  
 my\_schedule.eat();  
 my\_schedule.travel();  
 }  
}

11.public interface Test {  
 public int square(int number);  
}  
public class Arithmetic implements Test{  
 @Override  
 public int square(int number) {  
 return number\*number;  
 }  
}  
public class ToTestInt {  
 public static void main(String[]args){  
 Arithmetic arth1 = new Arithmetic();  
 int result = arth1.square(10);  
 System.out.println("The square of 10 is: "+result);  
 }  
}

12.import java.awt.Color;  
public class ColourChecking {  
 public static void main(String[] args) {  
 int red = 193;  
 int green = 255;  
 int blue = 183;  
 Color color = new Color(red, green, blue);  
 float[] hsbValues = Color.RGBtoHSB(red, green, blue, null);  
 float hue = hsbValues[0];  
 float saturation = hsbValues[1];  
 float brightness = hsbValues[2];  
  
 System.out.println("Hue: " + hue);  
 System.out.println("Saturation: " + saturation);  
 System.out.println("Brightness: " + brightness);  
 }  
}

13.public class Fibonacci {  
  
 public static void main(String[] args) {  
 int n = 20;  
 System.out.println("Fibonacci Series:");  
 for (int i = 0; i < n; i++) {  
 System.out.print(fibonacci(i) + " ");  
 }  
 }  
 public static int fibonacci(int n) {  
 if (n <= 1) {  
 return n;  
 } else {  
 return fibonacci(n - 1) + fibonacci(n - 2);  
 }  
 }  
}

14.package mypack;  
public class Point {  
 private int x;  
 private int y;  
 public Point() {  
 }  
 public Point(int x, int y) {  
 this.x = x;  
 this.y = y;  
 }  
 public void setX(int x) {  
 this.x = x;  
 }  
 public void setY(int y) {  
 this.y = y;  
 }  
 public void setXY(int x, int y) {  
 this.x = x;  
 this.y = y;  
 }  
}  
import mypack.Point;  
public class Main {  
 public static void main(String[] args) {  
 Point point = new Point(10, 20);  
 point.setXY(30, 40);  
 }  
}

15.public class Box {  
 protected double length;  
 protected double breadth;  
 public Box(double length, double breadth) {  
 this.length = length;  
 this.breadth = breadth;  
 }  
 public void setValues(double length, double breadth) {  
 this.length = length;  
 this.breadth = breadth;  
 }  
 public double calculateArea() {  
 return length \* breadth;  
 }  
}  
public class Box3D extends Box{  
 private double height;  
 public Box3D(double length, double breadth, double height) {  
 super(length, breadth);  
 this.height = height;  
 }  
 public void setValues(double length, double breadth, double height) {  
 super.setValues(length, breadth);  
 this.height = height;  
 }  
 public double calculateVolume() {  
 return length \* breadth \* height;  
 }  
}  
public class Main {  
 public static void main(String[] args) {  
 Box b1 = new Box(40, 50);  
 System.out.println("Box Area: " + b1.calculateArea());  
 Box3D b3D = new Box3D(40, 50, 60);  
 System.out.println("Box3D Area: " + b3D.calculateArea());  
 System.out.println("Box3D Volume: " + b3D.calculateVolume());  
 }  
}

16.public class Car {  
 private String model;  
 public Car(String model) {  
 this.model = model;  
 }  
 public String getModel() {  
 return model;  
 }  
}  
public class Driver {  
 private String name;  
 private Car car;  
 public Driver(String name) {  
 this.name = name;  
 }  
 public void assignCar(Car car) {  
 this.car = car;  
 }  
 public void driveCar() {  
 if (car != null) {  
 System.out.println(name + " is driving a " + car.getModel());  
 } else {  
 System.out.println(name + " does not have a car assigned");  
 }  
 }  
}  
public class Department {  
 private String name;  
 private List<Employee> employees;  
 public Department(String name) {  
 this.name = name;  
 employees = new ArrayList<>();  
 }  
 public void addEmployee(Employee employee) {  
 employees.add(employee);  
 }  
 public void printEmployees() {  
 System.out.println("Employees in " + name + " department:");  
 for (Employee employee : employees) {  
 System.out.println(employee.getName());  
 }  
 }  
}  
public class Employee {  
 private String name;  
 public Employee(String name) {  
 this.name = name;  
 }  
 public String getName() {  
 return name;  
 }  
}  
public class House {  
 private Room room;  
 public House() {  
 room = new Room();  
 }  
 public void enterHouse() {  
 room.enterRoom();  
 }  
 private class Room {  
 public void enterRoom() {  
 System.out.println("You entered the room in the house");  
 }  
 }  
}  
public class Main {  
 public static void main(String[] args) {  
  
 Car car = new Car("Toyota");  
 Driver driver = new Driver("John");  
 driver.assignCar(car);  
 driver.driveCar();  
  
 Department department = new Department("HR");  
 Employee employee1 = new Employee("Alice");  
 Employee employee2 = new Employee("Bob");  
 department.addEmployee(employee1);  
 department.addEmployee(employee2);  
 department.printEmployees();  
  
 House house = new House();  
 house.enterHouse();  
 }  
}

17.import java.util.Scanner;  
public class TryCatchFinallyExample {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 try {  
 System.out.print("Enter the size of the array: ");  
 int size = scanner.nextInt();  
 if (size < 0) {  
 throw new NegativeArraySizeException();  
 }  
 int[] array = new int[size];  
 System.out.println("Array created successfully!");  
 } catch (NegativeArraySizeException e) {  
 System.out.println("Error: Negative array size is not allowed.");  
 } finally {  
 System.out.println("Finally block executed.");  
 scanner.close();  
 }  
 }  
}

18.import java.io.\*;  
class M{  
 void method()throws IOException{  
 System.out.println("device operation performed");  
 }  
}  
class Testthrows3{  
 public static void main(String args[])throws IOException{//declare exception   
 M m=new M();  
 m.method();  
  
 System.out.println("normal flow...");  
 }  
}

19.class JavaException{  
 public static void main(String args[]){  
 try{  
 throw new MyException(2);  
 }  
 catch(MyException e){  
 System.out.println(e) ;  
 }  
 }  
}  
class MyException extends Exception{  
 int a;  
 MyException(int b) {  
 a=b;  
 }  
 public String toString(){  
 return ("Exception Number = "+a) ;  
 }  
}

20. import java.util.\*;  
public class SortArrayListExample1  
{  
 public static void main(String args[])  
 {  
 ArrayList<String> list = new ArrayList<String>();  
// adding elements to the ArrayList   
 list.add("Volkswagen");  
 list.add("Toyota");  
 list.add("Porsche");  
 list.add("Ferrari");  
 list.add("Mercedes-Benz");  
 list.add("Audi");  
 list.add("Rolls-Royce");  
 list.add("BMW");  
// printing the unsorted ArrayList   
 System.out.println("Before Sorting: "+ list);  
// Sorting ArrayList in ascending Order   
 Collections.sort(list);  
// printing the sorted ArrayList   
 System.out.println("After Sorting: "+ list);  
 }  
}

|  |
| --- |
|  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |