**Module 2**

**Topic**

**Public and Private access specifiers**

Question

An EmployeeDetail Class Which Take the Employ Detail

**Code**

class EmployeeDetail{

    private String eName;

    private int eId;

    private int Hour;

    private int eRate;

    static int count;

    public EmployeeDetail(String name, int id, int hour, int rate){

        set(name, id, hour, rate);

    }

    public void set(String name, int id, int hour, int rate){

        eName = name;

        eId = id + count++;

        Hour = hour;

        eRate = rate;

    }

    public void get(){

        System.out.println("Name: " + eName);

        System.out.println("Id: " + eId);

        System.out.println("Hour: " + Hour);

        System.out.println("Rate: " + eRate);

    }

}

public class Employee {

    public static void main(String[] args) {

        EmployeeDetail e1 = new EmployeeDetail("Dev", 1, 6, 20);

        EmployeeDetail e2 = new EmployeeDetail("Het", 2, 5, 10);

        EmployeeDetail e3 = new EmployeeDetail("Heet", 3, 6, 20);

        EmployeeDetail e4 = new EmployeeDetail("Mit", 4, 4, 15);

        e1.get();

        e2.get();

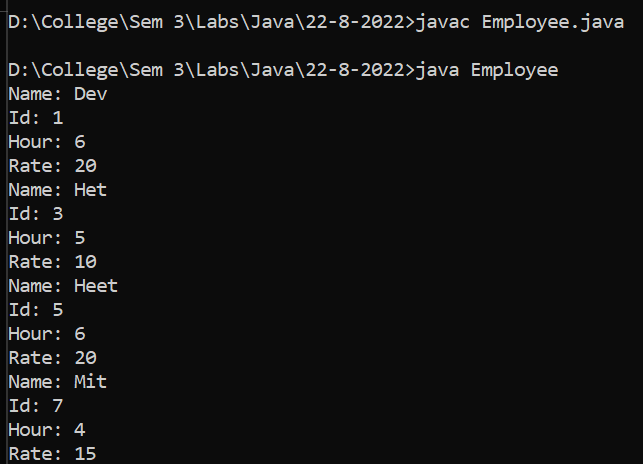
        e3.get();

        e4.get();

    }

}

**Input / Output**

****

**Topic**

**Constructer Overloading**

Question

A Product Class Which overload the constructor

**Code**

class ProductTest {

    int pID;

    String pName;

    int pRate;

    static int count;

    public ProductTest(String pName, int rate) {

        pID = 101 + count++;

        this.pName = pName;

        pRate = rate;

        System.out.println("Constructor called with " + pName + " and " + rate);

        System.out.println("Name: " + pName + " Rate: " + rate);

    }

    public ProductTest(String name) {

        pID = 101 + count++;

        pName = name;

        System.out.println("Constructor called with " + name);

        System.out.println("Name: " + name);

    }

    public ProductTest() {

        this("NoName", 0);

        System.out.println("Default constructor called");

        System.out.println("NoName");

    }

}

public class Product {

    public static void main(String[] args) {

        ProductTest p1 = new ProductTest("Product1", 2);

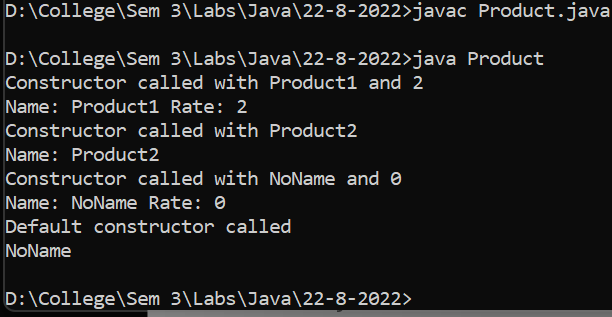
        ProductTest p2 = new ProductTest("Product2");

        ProductTest p3 = new ProductTest();

    }

}

**Input / Output**

****

**Topic**

**Access specifiers and Constructor overloading**

Question

An Employee class which get all the detail and find salary

**Code**

class EmployeeDetails {

    String name;

    int eID;

    private int hours;

    private int rate;

    static int count;

    {

        eID = 101 + count++;

    }

    public EmployeeDetails(String name, int hours, int rate) {

        this.name = name;

        this.hours = hours;

        this.rate = rate;

    }

    public String getName() {

        return name;

    }

    public int getHours() {

        return hours;

    }

    public int getRate() {

        return rate;

    }

    public int findSalary() {

        int salary;

        if (getHours() > 180) {

            int extra = (getHours() - 180) \* getRate() / 2;

            salary = getHours() \* getRate() + extra;

        } else {

            salary = getHours() \* getRate();

        }

        return salary;

    }

    public void printInfo() {

        System.out.printf("EMP ID: %d\n", this.eID);

        System.out.printf("The name is : %s\n", getName());

        System.out.printf("The hours are : %d\n", getHours());

        System.out.printf("rate per hour is : %d\n", getRate());

        System.out.printf("The salary of the employee is :RS.%d\n", findSalary());

    }

}

public class Employee {

    public static void main(String[] args) {

        EmployeeDetails emp1 = new EmployeeDetails("Dev", 240, 50);

        emp1.printInfo();

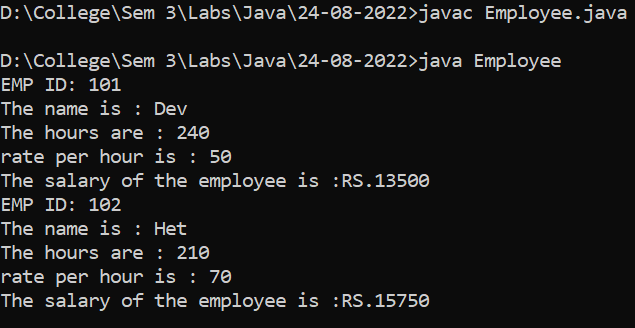
        EmployeeDetails emp2 = new EmployeeDetails("Het", 210, 70);

        emp2.printInfo();

    }

}

**Input / Output**

****

Question

A class of a Bank Account

**Code**

class AccountDetail {

    int AccNo;

    static int count = 1;

    {

        AccNo = 10000 + count++;

    }

    private int Balance;

    private String Name;

    static String BankName = "Sate Bank Of India";

    private long MoblieNo;

    public int getAccNo() {

        return AccNo;

    }

    public long getMoblieNo() {

        return MoblieNo;

    }

    public String getName() {

        return Name;

    }

    public int getBalance() {

        return Balance;

    }

    AccountDetail(String name, long mobile) {

        Name = name;

        MoblieNo = mobile;

    }

    AccountDetail(String name) {

        Name = name;

    }

    public String Deposite(int amount) {

        Balance = Balance + amount;

        return ("You Deposite: " + amount + " rs\n Current Bank Balance for Account Number '" + getAccNo() + "' is "

                + getBalance() + " rs");

    }

    public String Withdraw(int amount) {

        Balance = Balance - amount;

        return ("You Withdraw: " + amount + " rs\n Current Bank Balance for Account Number '" + getAccNo() + "' is "

                + getBalance() + " rs");

    }

}

public class Account {

    public static int Total(AccountDetail arr[]){

        int sum=0;

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

            sum+=arr[i].getBalance();

        }

        return sum;

    }

    public static void main(String[] args) {

        AccountDetail AccHolder1 = new AccountDetail("Dev", 9999999999l);

        AccountDetail AccHolder2 = new AccountDetail("Mit");

        AccountDetail AccHolder3 = new AccountDetail("Neel", 9898989898l);

        AccountDetail AccHolder4 = new AccountDetail("Ved", 98942544334l);

        AccountDetail AccHolder5 = new AccountDetail("Roy", 9845436654l);

        System.out.println(AccHolder1.Deposite(110));

        System.out.println(AccHolder2.Deposite(800));

        System.out.println(AccHolder3.Deposite(300));

        System.out.println(AccHolder4.Deposite(500));

        System.out.println(AccHolder5.Deposite(100));

        System.out.println(AccHolder1.Withdraw(10));

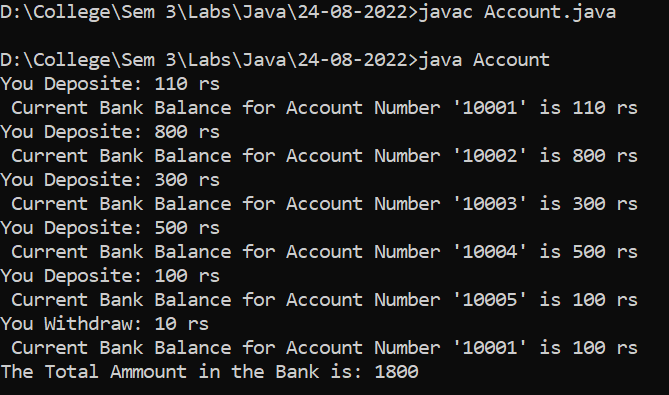
        AccountDetail[] dtl = {AccHolder1,AccHolder2,AccHolder3,AccHolder4,AccHolder5 };

        System.out.println("The Total Ammount in the Bank is: "+Account.Total(dtl));

    }

}

**Input / Output**

****

**Topic**

**Abstract Class, Inheritance, Runtime Polymorphism**

Question

A class of a Bank Account Using Abstract class

**Code**

class Account {

    int accNo;

    static int count;

    {

        accNo=10001+count++;

    }

    private int  balance;

    private long mobileNo;

    private String Name;

    static String bankname="HDFC";

    public Account(){

        System.out.println("Account created without any information");

    }

    public Account(String name){

        System.out.println("Account created but you have not given the mobile number So account can not be linked " +

                "mobile no.");

    }

    public Account(String name,long mobileNo) {

        this.Name = name;

        this.mobileNo = mobileNo;

    }

    public int getAccNo() {

        return accNo;

    }

    public String getName() {

        return Name;

    }

    public void setName(String name){

        this.Name=name;

    }

    public long getMobileNo(){

        return mobileNo;

    }

    public void setMobileNo(long mobileNo){

        this.mobileNo=mobileNo;

    }

    public int getBalance() {

        return balance;

    }

    public String Deposit(int dep\_money){

        this.balance+=dep\_money;

        return (dep\_money+"RS"+" debited to your account\n");

    }

    public String checkBalance(){

        return ("Your balance is : "+getBalance()+"\n");

    }

    public String withdraw(int money){

        this.balance-=money;

        return ("You have withdraw "+money+"RS from your account\n");

    }

    public void getAccinfo(){

        System.out.println("Bank name: "+bankname);

        System.out.println("Account holder name: "+getName());

        System.out.println("Mobile No."+getMobileNo());

        System.out.println("Account number :"+getAccNo());

        System.out.println("Balance : "+getBalance());

    }

}

class SavingAccount extends Account{

    int interestRate;

    public SavingAccount(){

        System.out.println("Savings Account created");

    }

    public SavingAccount(String name){

        System.out.println("Savings Account created ");

        name = this.getName();

        System.out.printf("Account holder name:%s", name);

    }

    public SavingAccount(String name,long number,int interestRate){

        super.setName(name);

        super.setMobileNo(number);

        this.interestRate=interestRate;

    }

    public String withdraw(int money){

        super.withdraw(money);

        if(super.getBalance()-money>1000){

            return ( money+"Rs has debited successfully from your account");

        }

        else{

            return ("Insufficient balance you can not withdraw money");

        }

    }

}

class Acc{

    public static void main(String[] args) {

        Account a1=new Account("Dev Sapariya", 999999999L);

        System.out.println(a1.checkBalance());

        System.out.println(a1.Deposit(10000));

        System.out.println(a1.checkBalance());

        System.out.println(a1.withdraw(500));

        a1.getAccinfo();

        SavingAccount s1=new SavingAccount("Mit",8888888888L,7);

        System.out.println(s1.checkBalance());

        System.out.println(s1.Deposit(10000));

        System.out.println(s1.checkBalance());

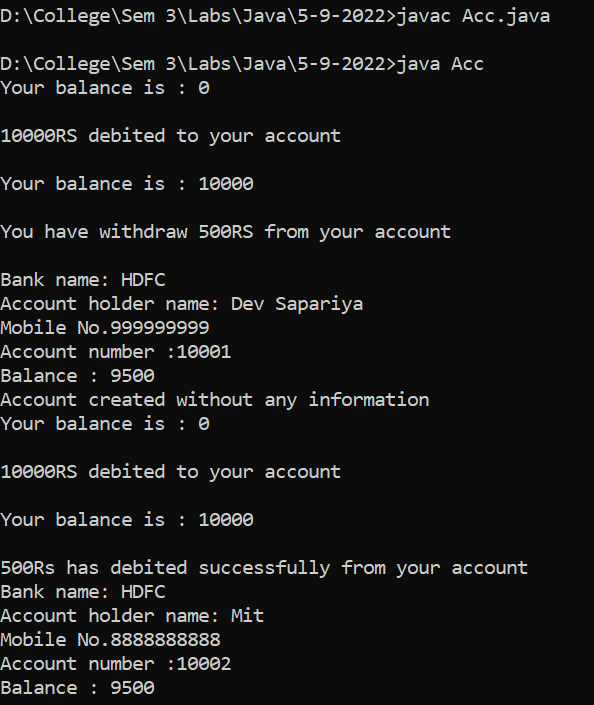
        System.out.println(s1.withdraw(500));

        s1.getAccinfo();

    }

}

**Input / Output**

****

Question

An Abstract Class To calculate the Area of Different 2D Shape

**Code**

abstract class TwoDShape {

    int length;

    int width;

    TwoDShape() {

        length = 0;

        width = 0;

    }

    TwoDShape(int length, int width) {

        this.length = length;

        this.width = width;

    }

    TwoDShape(TwoDShape t) {

        length = t.length;

        width = t.width;

    }

    public void setLength(int length) {

        this.length = length;

    }

    public void setWidth(int width) {

        this.width = width;

    }

    abstract float getArea();

}

class Rectangle extends TwoDShape {

    public Rectangle() {

        length = 10;

        width = 10;

    }

    public Rectangle(int l, int w) {

        super(l, w);

    }

    public Rectangle(TwoDShape r) {

        super(r);

    }

    public float getArea() {

        return length \* width;

    }

}

class Triangle extends TwoDShape {

    public Triangle(int l, int w) {

        super(l, w);

    }

    public float getArea() {

        return (float) (0.5 \* length \* width);

    }

}

public class Question\_area {

    public static void main(String[] args) {

        TwoDShape t = new Rectangle(5, 6);

        TwoDShape t1 = new Rectangle(t);

        Rectangle r = new Rectangle(4, 6);

        TwoDShape tri = new Triangle(3, 6);

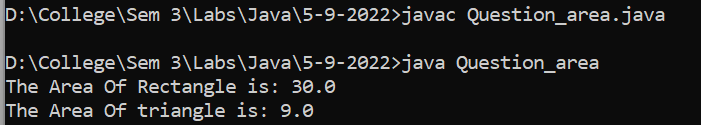
        System.out.println("The Area Of Rectangle is: " + t.getArea());

        System.out.println("The Area Of triangle is: " + tri.getArea());

    }

}

**Input/Output**

****

**Topic**

**Copy Constructor, Return the Object**

Question

A Measurement class which give the Difference of the measurement

**Code**

abstract class Measurement {

    protected int unit1;

    protected int unit2;

    protected int foot1;

    protected int foot2;

    protected int inch1;

    protected int inch2;

    Measurement() {

        unit1 = 10;

        unit2 = 20;

    }

    Measurement(int foot1, int inch1, int foot2, int inch2) {

        this.foot1 = foot1;

        this.foot2 = foot2;

        this.inch1 = inch1;

        this.inch2 = inch2;

        unit1 = (foot1 \* 12) + inch1;

        unit2 = (foot2 \* 12) + inch2;

    }

    Measurement(Measurement m) {

        foot1 = m.foot1;

        foot2 = m.foot2;

        inch1 = m.inch1;

        inch2 = m.inch2;

        unit1 = (foot1 \* 12) + inch1;

        unit2 = (foot2 \* 12) + inch2;

    }

    public int getUnit1() {

        return unit1;

    }

    public int getUnit2() {

        return unit2;

    }

    public void setUnit1(int unit1) {

        this.unit1 = unit1;

    }

    public void setUnit2(int unit2) {

        this.unit2 = unit2;

    }

    abstract Measurement getDifference(Measurement m);

}

class MeasurementPrint extends Measurement {

    int foot;

    int inch;

    MeasurementPrint(int foot, int inch) {

        this.foot = foot;

        this.inch = inch;

    }

    void printfootinch() {

        System.out.println("foot :" + foot + " inch: " + inch);

    }

    @Override

    Measurement getDifference(Measurement m) {

        return null;

    }

}

class HeightMeasurement extends Measurement {

    HeightMeasurement(int foot1, int inch1, int foot2, int inch2) {

        super(foot1, inch1, foot2, inch2);

    }

    HeightMeasurement(Measurement m) {

        super(m);

    }

    @Override

    Measurement getDifference(Measurement m) {

        int diff = m.unit2 - m.unit1;

        int foot = (diff) / 12;

        int inch = diff % 12;

        Measurement m6 = new MeasurementPrint(foot, inch);

        return m6;

    }

}

public class Question\_measurement {

    public static void main(String[] args) {

        Measurement m = new HeightMeasurement(6, 2, 9, 3);

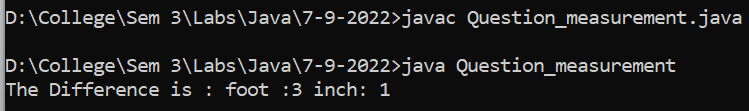
        System.out.print("The Difference is : ");

        ((MeasurementPrint) m.getDifference(m)).printfootinch();

    }

}

**Input / Output**

****

**Topic**

**Interface**

Question

Interface Example

**Code**

interface Interface1{

    public abstract  void printMethod1();

}

class Testclass1 implements Interface1{

    private  int number;

    public  Testclass1(){

        number=20;

    }

    public  Testclass1(int n){

        number=n;

    }

    public  int getNumber(){

        return  number;

    }

    public  void  setNumber(int n){

        number=n;

    }

    public void printMethod1(){

        System.out.println("TestCalss1 : "+number);

    }

}

public class InterfaceTesting {

    public static void main(String[] args) {

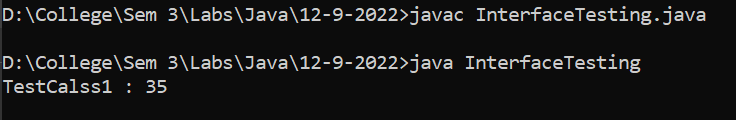
            Interface1 i1= new Testclass1(35);

            i1.printMethod1();

    }

}

**Input / Output**

****

Question

A Class Describe As Below

abstact class Residency

    instance fields :name, rNumber, area, unit\_rate

    provide all overloaded constructor including copy constructor

    provide necessary getter/setter method.

    abstract long getPriceOfResidency();

   override toString() and equals() method from Object class.

create abstract class naming Semi\_FurnishedResidency extends Residency.

   instance field - furnitureCharge, parkingCharge

   create class LuxuriousResidency extends Residency.

   instance field - amenityCharge

   override long getPriceOfResidency().

   create interface naming Rentable -

   abstract public int getRent();

   create TwoBHKResidency extends Semi\_FurnishedResidency implements Rentable.

   override long getPriceOfResidency().

   override int getRent().

**Code**

abstract class Residency {

    public String name;

    private int rNumber;

    private int area;

    private int unit\_rate;

    public Residency() {

        name = null;

        rNumber = 0;

        area = 0;

        unit\_rate = 0;

    }

    public Residency(String name, int rNumber, int area, int unit\_rate) {

        this.name = name;

        this.rNumber = rNumber;

        this.area = area;

        this.unit\_rate = unit\_rate;

    }

    Residency(Residency r) {

        this(r.name, r.rNumber, r.area, r.unit\_rate);

    }

    public int getArea() {

        return area;

    }

    public int getrNumber() {

        return rNumber;

    }

    public int getUnit\_rate() {

        return unit\_rate;

    }

    public String getName() {

        return name;

    }

    public void setArea(int area) {

        this.area = area;

    }

    public void setName(String name) {

        this.name = name;

    }

    public void setUnit\_rate(int unit\_rate) {

        this.unit\_rate = unit\_rate;

    }

    public void setrNumber(int rNumber) {

        this.rNumber = rNumber;

    }

    abstract long getPriceOfResidency();

}

abstract class Semi\_FurnishedResidency extends Residency {

    int furnitureCharge = 1000;

    int parkingCharge = 99;

    int price;

    Semi\_FurnishedResidency() {

        super();

    }

    Semi\_FurnishedResidency(String name, int rNumber, int area, int unit\_rate) {

        super.setName(name);

        super.setArea(area);

        super.setrNumber(rNumber);

        super.setUnit\_rate(unit\_rate);

    }

}

class LuxuriousResidency extends Residency {

    int amenityCharge = 66600;

    int price;

    @Override

    long getPriceOfResidency() {

        price = (getUnit\_rate() \* getArea()) + amenityCharge;

        return price;

    }

    LuxuriousResidency() {

        super();

    }

    LuxuriousResidency(String name, int rNumber, int area, int unit\_rate) {

        super.setName(name);

        super.setArea(area);

        super.setrNumber(rNumber);

        super.setUnit\_rate(unit\_rate);

    }

}

interface Rentable {

    abstract public long getRent();

}

class TwoBHKResidency extends Semi\_FurnishedResidency implements Rentable {

    long rent;

    public TwoBHKResidency(String name, int rNumber, int area, int unit\_rate) {

        super(name, rNumber, area, unit\_rate);

    }

    @Override

    public long getRent() {

        rent = (getPriceOfResidency() \* 5) / 100;

        return rent;

    }

    @Override

    long getPriceOfResidency() {

        price = (getUnit\_rate() \* getArea()) + furnitureCharge + parkingCharge;

        return price;

    }

}

public class Residency\_main {

    public static void main(String[] args) {

        LuxuriousResidency r1 = new LuxuriousResidency("PYTHON", 1,2000,63);

        System.out.println("The price of Luxurious Residency is: "+r1.getPriceOfResidency());

        Semi\_FurnishedResidency r2 = new TwoBHKResidency("JAVA", 2, 1000, 50);

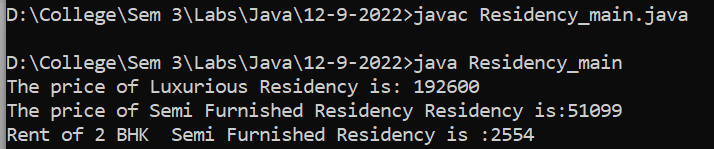
        System.out.println("The price of Semi Furnished Residency Residency is:"+r2.getPriceOfResidency());

        System.out.println("Rent of 2 BHK  Semi Furnished Residency is :"+((Rentable) r2).getRent());

    }

}

**Input / Output**

****

**Topic**

**Pass by Value and Variable Argument**

Pass By Value Example

class Account {

    private int accNo;

    private double balance;

    public Account(int an, double am) {

        accNo = an;

        balance = am;

    }

    public void deposit(double amount) {

        balance = balance + amount;

    }

    public void transfer(Account ac1, Account ac2) {

    }

    public double getBalance() {

        return balance;

    }

}

public class JavaPassByValue {

    public static void testPassByValue(Account ac, int amount) {

        amount = amount + 1000;

        ac.deposit(7000);

        System.out.println("Balance in method :" + ac.getBalance());

        ac = new Account(102, 5000);

        ac.deposit(2000);

        System.out.println("New Acc Balance in method :" + ac.getBalance());

        System.out.println("Amount Value in function : " + amount);

    }

    public static void main(String[] args) {

        int amount = 2000;

        Account ac = new Account(101, amount);

        ac.deposit(4000);

        testPassByValue(ac, amount);

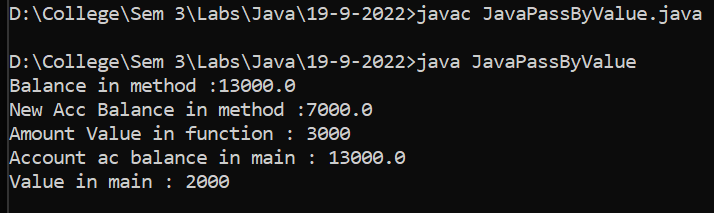
        System.out.println("Account ac balance in main : " + ac.getBalance());

        System.out.println("Value in main : " + amount);

    }

}

**Input / Output**

****

Variable Argument Example

**Code**

class VarArgTest {

    public static double average(double first, double second, double... remaining) {

        double total = first + second;

        for (double value : remaining)

            total += value;

        return total / (remaining.length + 2);

    }

    public static void main(String[] args) {

        System.out.printf("Average = %.3f%n", average(12.3, 13.7, 11.9, 19.8, 14.1, 17.7));

    }

    public static int maximum(String SubName, int DivisionNo, int... marks) {

        int max = marks[0];

        for (int num : marks) {

            if (num > max) {

                max = num;

            }

        }

        return max;

    }

}

public class VarArgs {

    public static void main(String[] args) {

        double avg = VarArgTest.average(10, 20, 30, 40, 50, 60, 70);

        System.out.println(avg);

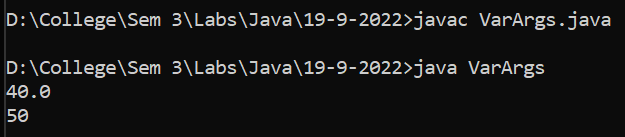
        int max = VarArgTest.maximum("Maths", 5, 20, 30, 40, 50);

        System.out.println(max);

    }

}

**Input / Output**

****

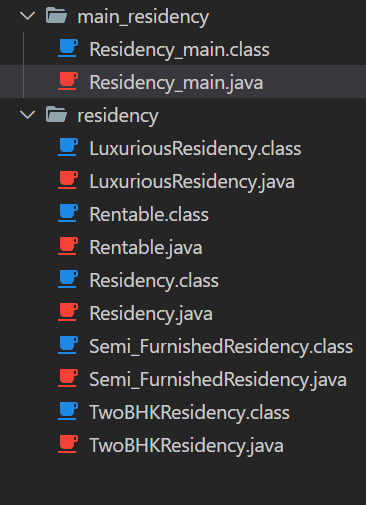
**Topic**

**Package**

Question

Example of using package in Residency Class

**File Structure**

****

**Code**

**Residency\_main.java**

package main\_residency;

import residency.\*;

public class Residency\_main {

    public static void main(String[] args) {

        LuxuriousResidency r1 = new LuxuriousResidency("PYTHON", 1,2000,63);

        System.out.println("The price of Luxurious Residency is: "+r1.getPriceOfResidency());

        Semi\_FurnishedResidency r2 = new TwoBHKResidency("JAVA", 2, 1000, 50);

        System.out.println("The price of Semi Furnished Residency Residency is:"+r2.getPriceOfResidency());

        System.out.println("Rent of 2 BHK  Semi Furnished Residency is :"+((Rentable) r2).getRent());

    }

}

**LuxuriousResidency.java**

package residency;

public class LuxuriousResidency extends Residency {

    int amenityCharge = 66600;

    int price;

    @Override

    public

    long getPriceOfResidency() {

        price = (getUnit\_rate() \* getArea()) + amenityCharge;

        return price;

    }

    LuxuriousResidency() {

        super();

    }

    public LuxuriousResidency(String name, int rNumber, int area, int unit\_rate) {

        super.setName(name);

        super.setArea(area);

        super.setrNumber(rNumber);

        super.setUnit\_rate(unit\_rate);

    }

}

**Rentable.java**

package residency;

public interface Rentable {

    abstract public long getRent();

}

**Residency.java**

package residency;

abstract class Residency {

    public String name;

    private int rNumber;

    private int area;

    private int unit\_rate;

    public Residency() {

        name = null;

        rNumber = 0;

        area = 0;

        unit\_rate = 0;

    }

    public Residency(String name, int rNumber, int area, int unit\_rate) {

        this.name = name;

        this.rNumber = rNumber;

        this.area = area;

        this.unit\_rate = unit\_rate;

    }

    Residency(Residency r) {

        this(r.name, r.rNumber, r.area, r.unit\_rate);

    }

    public int getArea() {

        return area;

    }

    public int getrNumber() {

        return rNumber;

    }

    public int getUnit\_rate() {

        return unit\_rate;

    }

    public String getName() {

        return name;

    }

    public void setArea(int area) {

        this.area = area;

    }

    public void setName(String name) {

        this.name = name;

    }

    public void setUnit\_rate(int unit\_rate) {

        this.unit\_rate = unit\_rate;

    }

    public void setrNumber(int rNumber) {

        this.rNumber = rNumber;

    }

    abstract long getPriceOfResidency();

}

**Semi\_FurnishedResidency.java**

package residency;

public class Semi\_FurnishedResidency extends Residency{

    int furnitureCharge = 1000;

    int parkingCharge = 99;

    int price;

    @Override

    public

    long getPriceOfResidency() {

        price = (getUnit\_rate() \* getArea()) + furnitureCharge + parkingCharge;

        return price;

    }

    Semi\_FurnishedResidency() {

        super();

    }

    public Semi\_FurnishedResidency(String name, int rNumber, int area, int unit\_rate) {

        super.setName(name);

        super.setArea(area);

        super.setrNumber(rNumber);

        super.setUnit\_rate(unit\_rate);

    }

}

**TwoBHKResidency.java**

package residency;

public class TwoBHKResidency extends Semi\_FurnishedResidency implements Rentable {

    long rent;

    public TwoBHKResidency(String name, int rNumber, int area, int unit\_rate) {

        super(name, rNumber, area, unit\_rate);

    }

    @Override

    public long getRent() {

        rent = (getPriceOfResidency() \* 5) / 100;

        return rent;

    }

    @Override

    public long getPriceOfResidency() {

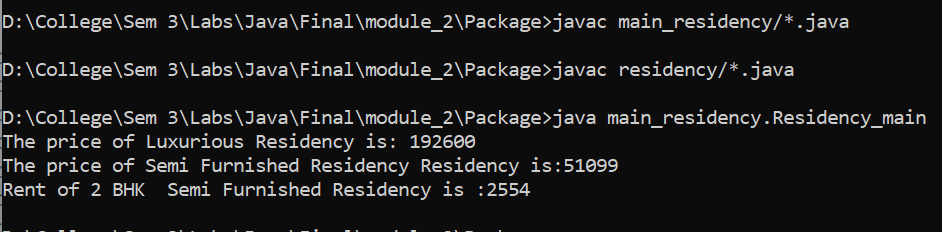
        price = (getUnit\_rate() \* getArea()) + furnitureCharge + parkingCharge;

        return price;

    }

}

**Input / Output**

****