

## EVEN SUBARRAY

```
import java.io.*;  
  
class GFG  
{  
    static int countEvenSum(int arr[],  
                           int n)  
    {  
        int result = 0;  
  
        // Find sum of all subarrays  
        // and increment result if  
        // sum is even  
        for (int i = 0; i <= n - 1; i++)  
        {  
            int sum = 0;  
            for (int j = i; j <= n - 1; j++)  
            {  
                sum = sum + arr[j];  
                if (sum % 2 == 0)  
                    result++;  
            }  
        }  
  
        return (result);  
    }  
  
    // Driver code  
    public static void main (String[] args)  
    {  
        int arr[] = {1, 2, 2,  
                     3, 4, 1};  
        int n = arr.length;  
  
        System.out.print("The Number of Subarrays"+  
                         " with even sum is ");  
  
        System.out.println(countEvenSum(arr, n));  
    }  
}
```

}

# DIVISIBILITY OF STRINGS

## Code is in c#. 12/14 test cases passing.

Given two strings,  $s$  and  $t$ , create a function that operates per the following rules:

- Find whether string  $s$  is divisible by string  $t$ . String  $s$  divisible by string  $t$  if string  $t$  can be concatenated some number of times to obtain the string  $s$ .
  - If  $s$  is divisible, find the smallest string  $u$ , such that it can be concatenated some number of times to obtain both  $s$  and  $t$ .
  - If it is not divisible, set the return value to -1.
Return the length of the string  $u$  or -1.

**Example 1**  
 $s = 'bcdcbcbcdcbcd'$   
 $t = 'bcdbcd'$

If string  $t$  is concatenated twice, the result is 'bcdcbcbcbcd' which is equal to the string  $s$ . The string  $s$  is divisible by string  $t$ . Since it passes the first test, look for the smallest string  $u$ , that can be concatenated.

Test Results   Custom Input   Run Code   Run Tests   Submit

```
public static int findSmallestDivisor(string s, string t)
{
    int ls=s.Length;
    int lt=t.Length;
    List<string> alist=new List<string>();
    int div=ls%lt;
    string temp=s.Substring(0,lt);
    if(div != 0 || temp != t)
        return -1;
    else{
        if(lt % 2 != 0)
            return lt;
        else
        {
            for(int i=1;i<lt;i++)
            {
                alist.Clear();
                if(lt%i == 0)
                {
                    for(int j=0;j<lt;)
                    {
                        alist.Add(t.Substring(j,i));
                        j+=i;
                    }
                    if( alist.All(o=> o == alist.First()))
                        return i;
                }
            }
        }
    }
    return lt;
}
```

Given two strings,  $s$  and  $t$ , create a function that operates per the following rules:

- Find whether string  $s$  is divisible by string  $t$ . String  $s$  divisible by string  $t$  if string  $t$  can be concatenated some number of times to obtain the string  $s$ .
  - If  $s$  is divisible, find the smallest string  $u$ , such that it can be concatenated some number of times to obtain both  $s$  and  $t$ .
  - If it is not divisible, set the return value to -1.
Return the length of the string  $u$  or -1.

**Example 1**  
 $s = 'bcdcbcbcdcbcd'$   
 $t = 'bcdbcd'$

If string  $t$  is concatenated twice, the result is 'bcdcbcbcbcd' which is equal to the string  $s$ . The string  $s$  is divisible by string  $t$ . Since it passes the first test, look for the smallest string  $u$ , that can be concatenated.

Test Results   Custom Input   Run Code   Run Tests   Submit

OR

```
function findSmallestDivisor(s, t) {

    if (!s || s.length == 0 || !t || t.length == 0) {

        return -1;

    }
```

```

if (Math.max(s.length, t.length) % Math.min(s.length, t.length) === 0 ) {

    if (s === t) {

        var sub = '';

        for (var i = 0; i < s.length; i++) {

            for(var j = 1; j < s.length; j++) {

                if (s[j] === s.charAt(i)) {

                    sub = s.slice(s,j);

                    var repeater = Math.floor(s.length/sub.length);

                    if (sub.repeat(repeater) === s) {

                        return sub.length;

                    }

                } else {

                    sub = s;

                }

            }

        }

        return sub.length;

    }

} else {

    if(s.indexOf(t.charAt(0)) == -1) {

        return -1;

    }

}

const count = Math.floor(s.length/t.length);

const getMinMaxStr = (s, t) => [s, t].sort((s, t) => s.length - t.length);

const isDivisible = (s, t) => s === t.repeat(s.length / t.length);

const gcdOfStrings = (s, t) => {

    const [min_str, max_str] = getMinMaxStr(s, t);

    if (min_str !== max_str.substring(0, min_str.length)) return '';

    let result= '';

    for (let i = 1; i <= min_str.length; i++) {

        if (max_str.length % i !== 0) continue;

        if (max_str.substring(0, i).repeat(Math.floor(max_str.length/i)) === max_str) result += i;
    }

    return result;
}

```

```
        const t = min_str.substring(0, i);

        if (isDivisible(max_str, t) && isDivisible(min_str, t)) {

            result = t;

        }

    }

    return result.length;
};

}

} else return -1
}
```

## [CostAndTimeEstimation.java](#)

```
package com.cts.conctes.client;
import com.cts.conctes.exception.ConstructionEstimationException;
import com.cts.conctes.service.ConstructionProjectEstimationService;
public class CostAndTimeEstimation {
    public static void main(String[] args) throws ConstructionEstimationException
    {
        ConstructionProjectEstimationService cpeService = new
        ConstructionProjectEstimationService();
        boolean isTrue=cpeService.addConstructionProjectDetails("inputfeed.txt");
        if(isTrue) {
            System.out.println("All are added successfully into the database");
        }
    }
}
```

## [CostAndTimeEstDAO.java](#)

```
package com.cts.conctes.dao;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Date;
import com.cts.conctes.exception.ConstructionEstimationException;
import com.cts.conctes.model.ConstructionProject;
import com.cts.conctes.util.ApplicationUtil;
public class CostAndTimeEstDAO {
    public static Connection connection = null;
    public boolean insertConstructionProject(ArrayList <ConstructionProject>
    constProjects)
    throws ConstructionEstimationException {
        boolean recordsAdded = false;
        int index=0;
        int size=constProjects.size();
        connection=DBConnectionManager.getInstance().getConnection();
```

```

String query="insert into constructionproject values(?, ?, ?, ?, ?, ?, ?);";
for(ConstructionProject cp: constProjects) {

    try {
        PreparedStatement ps=connection.prepareStatement(query);
        ps.setString(1, cp.getProjectId());
        ps.setDate(2,
ApplicationUtil.utilToSqlDateConverter(cp.getPlannedDOStart()));
        ps.setString(3, cp.getTypeOfProject());
        ps.setString(4, cp.getStructure());
        ps.setDouble(5, cp.getAreaInSqFt());
        ps.setDouble(6, cp.getEstimatedCostInlac());
        ps.setDouble(7, cp.getEstimatedTimeInMonths());
        int row=ps.executeUpdate();
        if(row>0)
            index+=1;
        if(index==size) {
            recordsAdded=true;
        }
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

return recordsAdded;
}

public ArrayList <ConstructionProject> getConstructionProjectsData() throws
ConstructionEstimationException
{
ArrayList <ConstructionProject> consApplicants = new
ArrayList<ConstructionProject>();
connection=DBConnectionManager.getInstance().getConnection();

```

```

String query="select * from constructionproject";
try {
    PreparedStatement ps=connection.prepareStatement(query);
    ResultSet rs=ps.executeQuery();
    while(rs.next()) {
        String id=rs.getString(1);
        Date d=rs.getDate(2);
        String typeProject=rs.getString(3);
        String structure=rs.getString(4);
        double areaSqFt=rs.getDouble(5);
        double costInLac=rs.getDouble(6);
        double timeInMonths=rs.getDouble(7);
        consApplicants.add(new ConstructionProject(id, d, typeProject,
structure, areaSqFt, costInLac, timeInMonths));
    }
} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

return consApplicants;
}
}

DBConnectionManager.java

package com.cts.conctes.dao;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;
import com.cts.conctes.exception.ConstructionEstimationException;
public class DBConnectionManager {

```

```
private static Connection con = null;
private static DBConnectionManager instance;
private DBConnectionManager() throws ConstructionEstimationException
{
//WRITE YOUR CODE HERE
//return con;
}
public static DBConnectionManager getInstance() throws
ConstructionEstimationException
{
    if(instance==null) {
        instance=new DBConnectionManager();
    }
    return instance;
}
public Connection getConnection()
{
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    Properties properties=new Properties();
    FileReader f=null;
    try {
        f = new FileReader("database.properties");
    } catch (FileNotFoundException e2) {
        // TODO Auto-generated catch block
        e2.printStackTrace();
    }
    try {
        properties.load(f);
    } catch (IOException e1) {
        // TODO Auto-generated catch block
    }
}
```

```
        e1.printStackTrace();
    }

    String url=properties.getProperty("url");
    String user=properties.getProperty("username");
    String password=properties.getProperty("password");
    try {
        con=DriverManager.getConnection(url, user, password);
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return con;
}
}
```

### ConstructionEstimationException.java

```
package com.cts.conctes.exception;
public class ConstructionEstimationException extends Exception{
String strMsg1;
Throwable strMsg2;
public ConstructionEstimationException() {
super();
}
}
```

### ConstructionProject.java

```
package com.cts.conctes.model;
import java.util.Date;
public class ConstructionProject {
    String projectId;
    Date plannedDOStart;
    String typeOfProject;
    String structure;
    double areaInSqFt;
    double estimatedCostInlac;
    double estimatedTimeInMonths;
public ConstructionProject() {
```

```
super();
}

public ConstructionProject(String projectId, Date plannedDOStart, String
typeOfProject, String structure, double areaInSqFt, double estimatedCostInlac,
double estimatedTimeInMonths) {
    super();
    this.projectId = projectId;
    this.plannedDOStart = plannedDOStart;
    this.typeOfProject = typeOfProject;
    this.structure = structure;
    this.areaInSqFt = areaInSqFt;
    this.estimatedCostInlac = estimatedCostInlac;
    this.estimatedTimeInMonths = estimatedTimeInMonths;
}

public String getprojectId() {
    return projectId;
}

public void setprojectId(String projectId) {
    this.projectId = projectId;
}

public Date getPlannedDOStart() {
    return plannedDOStart;
}

public void setPlannedDOStart(Date plannedDOStart) {
    this.plannedDOStart = plannedDOStart;
}

public String getTypeOfProject() {
    return typeOfProject;
}

public void setTypeOfProject(String typeOfProject) {
    this.typeOfProject = typeOfProject;
}

public String getStructure() {
    return structure;
}

public void setStructure(String structure) {
```

```

        this.structure = structure;
    }

    public double getAreaInSqFt() {
        return areaInSqFt;
    }

    public void setAreaInSqFt(double areaInSqFt) {
        this.areaInSqFt = areaInSqFt;
    }

    public double getEstimatedCostInlac() {
        return estimatedCostInlac;
    }

    public void setEstimatedCostInlac(double estimatedCostInlac) {
        this.estimatedCostInlac = estimatedCostInlac;
    }

    public double getEstimatedTimeInMonths() {
        return estimatedTimeInMonths;
    }

    public void setEstimatedTimeInMonths(double estimatedTimeInMonths)
    {this.estimatedTimeInMonths = estimatedTimeInMonths;
    }

    @Override

    public String toString() {
        return "ConstructionProject [projectId=" + projectId + ","
plannedDOStart=" +
plannedDOStart + ", typeOfProject="
+ typeOfProject + ", structure=" + structure + ", areaInSqFt="
+ areaInSqFt + ", estimatedCostInlac="
+ estimatedCostInlac + ", estimatedTimeInMonths="
+ estimatedTimeInMonths + "]";
    }
}

```

## [ConstructionProjectEstimationService.java](#)

```

package com.cts.conctes.service;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

```

```

import com.cts.conctes.dao.CostAndTimeEstDAO;
import com.cts.conctes.exception.ConstructionEstimationException;
import com.cts.conctes.model.ConstructionProject;
import com.cts.conctes.util.ApplicationUtil;
public class ConstructionProjectEstimationService {
    public static ArrayList <ConstructionProject> buildConstructionProjectList(List
<String>
consProjectRecords) {
        final String COMMADELIMITER = ",";
        ArrayList <ConstructionProject> consProjectRecordList = new
            ArrayList<ConstructionProject>();
        for(String s: consProjectRecords) {
            String[] s1=s.split(COMMADELIMITER);
            String id=s1[0];
            Date d=ApplicationUtil.stringToDateConverter(s1[1]);
            String typeproject=s1[2];
            String structure=s1[3];
            double areaInSqFt=Double.parseDouble(s1[4]);
            double costs[]=estimateTimeAndCostForConstruction(typeproject,
structure, areaInSqFt);
            double estimatedCostInLac=costs[0];
            double estimatedTimeInMon=costs[1];
            ConstructionProject p=new ConstructionProject(id, d, typeproject,
structure, areaInSqFt, estimatedCostInLac, estimatedTimeInMon);
            consProjectRecordList.add(p);
        }
        return consProjectRecordList;
    }
    public boolean addConstructionProjectDetails(String inputFeed) throws
        ConstructionEstimationException {
        ArrayList<ConstructionProject>
p=buildConstructionProjectList(ApplicationUtil.readFile(inputFeed));
        CostAndTimeEstDAO obj=new CostAndTimeEstDAO();
        if(obj.insertConstructionProject(p)) {
            ArrayList<ConstructionProject>
p1=obj.getConstructionProjectsData();
            for(ConstructionProject cp: p1) {

```

```

        System.out.println(cp);
    }
    return true;
}
return false;
}

public static double[] estimateTimeAndCostForConstruction(String
projectType,String
structure,double areaInSqFt)
{
    double costEstimateInRs=0.0,timeEstimateInMonths=0.0;
    double costs[] = {costEstimateInRs,timeEstimateInMonths};

/*
* The Cost Estimate and
*
Based on the type of the Project & the Structure , according to the required
area of Construction, the cost & time have to be calculated based on the base
data available in the table provided in the use case document:
For eg. If the Project Type is ◊Commercial◊ and the structure
is ◊Shopping Complex◊ the cost incurred for the construction of
per sq. ft is Rs.2600 and the time taken for the construction of
the 1000 sq ft of the same project is 0.23 Months,
calculation has to be performed on the similar basis
i.e Pro rata basis depending upon the type and the area of construction.
*/
    if(projectType.equals("Commercial")) {
        if(structure.equals("Shopping Complex")) {
            costs[0]=2600*areaInSqFt;
            costs[1]=0.23*areaInSqFt/1000;
        }
        else if(structure.equals("ResApartments")) {
            costs[0]=2750*areaInSqFt;
            costs[1]=0.24*areaInSqFt/1000;
        }
        else {
    
```

```

        costs[0]=2600*areaInSqFt;
        costs[1]=0.2*areaInSqFt/1000;
    }
}

else if(projectType.equals("Infrastructural")) {
    if(structure.equals("Bridge")) {
        costs[0]=10000*areaInSqFt;
        costs[1]=0.25*areaInSqFt/1000;
    }
    else if(structure.equals("FlyOver")) {
        costs[0]=14000*areaInSqFt;
        costs[1]=0.22*areaInSqFt/1000;
    }
    else {
        costs[0]=8000*areaInSqFt;
        costs[1]=(0.25/1000)*areaInSqFt;
    }
}
else {
    if(structure.equals("House")) {
        costs[0]=2250*areaInSqFt;
        costs[1]=0.26*areaInSqFt/1000;
    }
    else if(structure.equals("Apartments")) {
        costs[0]=2500*areaInSqFt;
        costs[1]=0.24*areaInSqFt/1000;
    }
    else {
        costs[0]=2750*areaInSqFt;
        costs[1]=0.23*areaInSqFt/1000;
    }
}
return costs;
}
}

```

## [ApplicationUtil.java](#)

```
package com.cts.conctes.util;
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.StringTokenizer;
import com.cts.conctes.exception.ConstructionEstimationException;
public class ApplicationUtil {
    public static List<String> readFile(String inputfeed) throws
ConstructionEstimationException {
        List<String> constructionProjects = new ArrayList<String>();
        BufferedReader br=null;
        try {
            br = new BufferedReader(new FileReader(inputfeed));
        } catch (FileNotFoundException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        String s=null;
        try {
            s = br.readLine();
            while(s!=null) {
                String s1[]=s.split(",");
                if(checkIfCurrentFinYearProject(stringToDateConverter(s1[1])))
                    constructionProjects.add(s);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```

        s=br.readLine();
    }
} catch (IOException e) {
    e.printStackTrace();
}
return constructionProjects;
}

public static java.sql.Date utilToSqlDateConverter(java.util.Date utDate) {
    java.sql.Date sqlDate;
    String s=new SimpleDateFormat("yyyy-MM-dd").format(utDate);
    sqlDate=java.sql.Date.valueOf(s);
    return sqlDate;
}

public static java.util.Date stringToDateConverter(String stringDate) {
    Date strDate = null;
    SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd");
    try {
        strDate=sdf.parse(stringDate);
    } catch (ParseException e) {
        e.printStackTrace();
    }
    return strDate;
}

public static boolean checkIfCurrentFinYearProject(Date dos)
{
    boolean flag = false;
    int givenYear,givenMonth;
    givenYear = (dos.getYear()+1900);
    givenMonth = dos.getMonth();
    Date curDate = new Date();
    int curYear,curMonth;
    curYear = (curDate.getYear()+1900);
    curMonth = curDate.getMonth();
    if( curYear == givenYear)
    {

```

```

if(((curMonth >=0)&&(curMonth <= 2)) && ((givenMonth
>=0)&&(givenMonth <= 2)))
{
    flag = true;
}
else if(((curMonth >=3)&&(curMonth <= 11)) && ((givenMonth
>=3)&&(givenMonth <= 11)))
{
    flag = true;
}
else
{
    flag = false;
}
}
else if(curYear > givenYear)
{
    int dif = curYear - givenYear;
    if(dif == 1)
{
    if(((curMonth >=0)&&(curMonth <= 2)) && ((givenMonth
>=3)&&(givenMonth <= 11)))
{
        flag = true;
}
    else if(((curMonth >=3)&&(curMonth <= 11)) &&
((givenMonth
>=3)&&(givenMonth <= 11)))
{
        flag = false;
}
    else{
        flag = false;
}
}
}

```

```

else
{
    flag = false;
}
}

else if(curYear < givenYear)
{
    int dif = givenYear-curYear;
    if(dif == 1)
    {
        if(((curMonth >=3)&&(curMonth <= 11)) &&
(givenMonth
        >=0)&&(givenMonth <= 2))
    {
        flag = true;
    }
    else if(((curMonth >=3)&&(curMonth <= 11)) &&
(givenMonth
        >=3)&&(givenMonth <= 11))
    {
        flag = false;
    }
    else
    {
        flag = false;
    }
}
else
{
    flag = false;
}
}
else
{
    flag = false;
}
}

```

```
    return flag;  
}  
}
```

1)Hello world

```
public class PrintMessage
{
    public static void main(String [] args)
    {
        System.out.println("Welcome to Java world");
    }
}
```

2) Student Details

```
import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        System.out.println("Applicant name");

        String str=sc.nextLine();

        System.out.println("Marks obtained in HSC");

        int num=sc.nextInt();

        System.out.println("Total possible marks in HSC");

        int num1=sc.nextInt();

        System.out.println("Engineering cutoff mark");

        float num2=sc.nextFloat();

        System.out.println("Marks obtained in SSLC");

        int num3=sc.nextInt();
```

```
System.out.println("Total possible marks in SSLC");

int num4=sc.nextInt();

System.out.println("Gender");

char ch=sc.next().charAt(0);

System.out.println("Your Application has been Submitted Successfully");

System.out.println("The name of the applicant: "+str);

System.out.println("Engineering Cutoff: "+num2);

System.out.println("Applicant gender: "+ch);

System.out.println("All the best for your Career");

//Fill code here

}

}

3)Runs Scored
```

```
import java.util.Scanner;

import java.util.*;

public class Main

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

//Fill code here

System.out.println("Runs scored");

double a=sc.nextDouble();

System.out.println("Boundaries scored");

double b=sc.nextDouble();
```

```

System.out.println("Sixers scored");

double c=sc.nextDouble();

int run =(int)(a-((b*4)+(c*6)));

double p= ((a-run)/a)*100;

double h=Math.round(p);

System.out.println("Runs scored running between wickets "+ run);

System.out.printf("Percentage of runs scored running between wickets %.2f",h);

}

}

```

#### 4)Population census

```

import java.util.Scanner;

public class Main

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

//Fill code here

System.out.println("Enter population before 10 years");

int p = sc.nextInt();

System.out.println("Enter population after 10 years");

int q = sc.nextInt();

System.out.println("Population before 10 years "+p+"\nPopulation after 10 years

"+q+"\nIncrease in population between 10 years "+(q-p)+"\nPercentage of population increase in

10 years "+(((q-p)*100/p))+"\nPercentage of population increase in a

year"+((((q-p)*100/p))/10));

```

```
}
```

```
}
```

5) find the word

```
import java.util.Scanner;
```

```
public class Main
```

```
{
```

```
public static void main(String[] args)
```

```
{
```

```
Scanner sc=new Scanner(System.in);
```

```
//Fill code here
```

```
System.out.println("Enter the alphabet");
```

```
String s = sc.nextLine();
```

```
switch(s)
```

```
{
```

```
case "A" :
```

```
System.out.println("A for Apple");
```

```
break;
```

```
case "B" :
```

```
System.out.println("B for Ball");
```

```
break;
```

```
case "C" :
```

```
System.out.println("C for Cat");
```

```
break;
```

```
case "D" :
```

```
System.out.println("D for Dog");
```

```
break;

case "E" :
    System.out.println("E for Elephant");

break;

default:
    System.out.println(s + " is an invalid input");
}

sc.close();
}

}
```

6)Ludo King

```
import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        //Fill code here

        System.out.println("Enter Alex points");

        int a = sc.nextInt();

        if(a < 0 || a > 50)

            System.out.println(a + " is an invalid number");

        System.out.println("Enter Nikil points");

        int b = sc.nextInt();

        if(b < 0 || b > 50)
```

```

System.out.println(b + " is an invalid number");

System.out.println("Enter Sam points");

int c = sc.nextInt();

if(c < 0 || c > 50)

System.out.println(c + " is an invalid number");

if(a > b && a > c )

System.out.println("Alex scored "+a+" points and won the game");

else if(b > c )

System.out.println("Nikil scored "+b+" points and won the game");

else

System.out.println("Sam scored "+c+" points and won the game");

sc.close();

}

}

}

```

## 7)Resort Booking

```

import java.util.Scanner;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

//Fill the code here

String s = sc.nextLine();

int i = 0;

while(s.charAt(i) != ':')

{

i++;

}

```

```

}

i++;

if((s.charAt(i)-'0') < 0)

System.out.println("Invalid input for number of adults");

else if((s.charAt(i+2)-'0') < 0)

System.out.println("Invalid input for number of children");

else if((s.charAt(i+4)-'0') <= 0)

System.out.println("Invalid input for number of days");

i = 0;

while(s.charAt(i) != ':')

{

System.out.print(s.charAt(i));

i++;

}

i++;

System.out.print(" your booking is confirmed and the total cost is Rs

"+(((s.charAt(i)-'0')1000 + (s.charAt(i+2)-'0')*650))(s.charAt(i+4)-'0')));

}
}

```

### 8) Find the Winner

```

import java.util.Scanner;

import java.util.*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

```

```
LinkedHashMap<Float,String> hm=new LinkedHashMap<Float,String>();

System.out.println("Enter the number of teams");

int t = sc.nextInt();

if(t<=1){

System.out.println("Invalid input");

System.exit(0);

}

float min_sum =Float.MAX_VALUE;

int m= t;

float sum=0;

System.out.println("Enter the details");

for(int k=0;k<t;k++){

sum=0;

String str=sc.next();

String arr1[]= str.split ":";

int n =arr1.length;

float arr[]=new float[4];

arr[0]=Float.parseFloat(arr1[1]);

arr[1]=Float.parseFloat(arr1[2]);

arr[2]=Float.parseFloat(arr1[3]);

arr[3]=Float.parseFloat(arr1[4]);

if(arr[0]<1.0 || arr[1]<1.0 || arr[2]<1.0 || arr[3]<1.0){

System.out.println("Invalid number");

return;

}
```

```

sum = arr[0]+arr[1]+arr[2]+arr[3];

hm.put(sum,arr1[0]);

for(int i=0;i<t;i++){

if(sum<min_sum)

min_sum=sum;

}

}

String minKey= hm.get(min_sum);

if(minKey.equals("Prince")){

System.out.printf("King"+" team wins the race in %.2f minutes",min_sum);

}else{

System.out.printf(minKey+" team wins the race in %.2f minutes",min_sum);

}

//Fill the code here

}

}

```

### 9)Alphabetical order

```

import java.util.*;

public class Main{

static String sortString(String str) {

char []arr = str.toCharArray();

Arrays.sort(arr);

return String.valueOf(arr);

}

public static void main(String[] args){

```

```
Scanner sc=new Scanner(System.in);

System.out.println("Enter the sentence");

String str= sc.nextLine();

String arr1[]= str.split(" ");

int n =arr1.length;

for(int j=0;j<n;j++){

char[] chars = arr1[j].toCharArray();

for(char c : chars){

if(Character.isDigit(c)){

System.out.print(str+" is an invalid input");

return;

}

}

}

String arr[]=new String[n];

for(int i=0;i<n;i++){

arr[i]=sortString(arr1[i]);

System.out.print(arr[i]+" ");

}

}

}

10)Sum of Max

import java.util.*;

public class Main{

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

```
Scanner sc=new Scanner(System.in);

System.out.println("Enter the array size");

int n = sc.nextInt();

if(n<=1 || n%2!=0){

System.out.println(n +" is an invalid number");

return;

}

int arr[] = new int[n];

System.out.println("Enter the number");

for(int i =0;i<n;i++){

arr[i]=sc.nextInt();

}

Arrays.sort(arr);

ArrayList<Integer> all= new ArrayList<Integer>();

for(int d=arr.length-1;d>=0;d--){

all.add(arr[d]);

}

ArrayList<Integer> al= new ArrayList<Integer>();

for(int j=0;j<n;j++){

al.add(arr[j] + all.get(j));

}

System.out.println("The maximum number is "+Collections.max(al));

}

}

11)Incredible toys
```

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class CustomerDetails {

    private String customerId;
    private String customerName;
    private long phoneNumber;
    private String emailId;
    private String toyType;
    private double price;

    public CustomerDetails(String customerId, String customerName, long phoneNumber, String
        emailId, String toyType, double price)
    {
        this.customerId=customerId;
        this.customerName=customerName;
        this.phoneNumber=phoneNumber;
        this.emailId=emailId;
        this.toyType=toyType;
        this.price=price;
    }

    public String getCustomerId()
    {
        return this.customerId;
    }

    public void setCustomerId(String customerId)
    {
```

```
this.customerId=customerId;  
}  
  
public String getCustomerName()  
{  
    return this.customerName;  
}  
  
public void setCustomerName(String customerName)  
{  
    this.customerName=customerName;  
}  
  
public long getPhoneNumber()  
{  
    return this.phoneNumber;  
}  
  
public void setPhoneNumber(long phoneNumber)  
{  
    this.phoneNumber=phoneNumber;  
}  
  
public String getEmailId()  
{  
    return this.emailId;  
}  
  
public void setEmailId(String emailId)  
{  
    this.emailId=emailId;
```

```
}

public String getToyType()

{

return toyType;

}

public void setToyType(String toyType)

{

this.toyType=toyType;

}

public double getPrice()

{

return this.price;

}

public void setPrice(double price)

{

this.price=price;

}

// Fill the code

public boolean validateCustomerId() {

String customerId=getCustomerId();

String pattern = "Incredible/[0-9]{3}/[0-9]{4}";

Pattern r = Pattern.compile(pattern);

Matcher m = r.matcher(customerId);

if(m.find())

{
```

```
return true;  
}  
  
return false;  
}  
  
public double calculateDiscount() {  
    double price = getPrice();  
    int dis=0;  
  
    toyType=toyType.toLowerCase();  
    switch(toyType)  
    {  
        case "softtoys":  
            dis=5;  
            break;  
        case "fidgettoys":  
            dis=10;  
            break;  
        case "sensorytoys":  
            dis=15;  
            break;  
        case "puzzles":  
            dis=20;  
            break;  
    }  
  
    return (price-(price*dis/100));  
}
```

```
}
```

---

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter Customer Id");

        String customerId=sc.next();

        System.out.println("Enter Customer Name");

        String customerName=sc.next();

        System.out.println("Enter Phone Number");

        long phoneNumber=sc.nextLong();

        System.out.println("Enter Email Id");

        String emailId=sc.next();

        System.out.println("Enter type");

        String toyType=sc.next();

        System.out.println("Enter Price");

        double price = Double.parseDouble(sc.next());

        CustomerDetails cd=new CustomerDetails(customerId,customerName,phoneNumber,emailId,toyType,price);

        if(cd.validateCustomerId())

        {

            double amount=cd.calculateDiscount();

            System.out.printf("Amount to be paid by the Customer %.2f",amount);

```

```

}

else{
    System.out.println("Provide a proper Customer Id");
}

}

}

}

12) car showroom

public class Car {

    // Fill the code

    private String carId;
    private String modelNumber;
    private String colour;
    private String transmissionType;
    private String fuelType;
    private String engineType;
    private String bodyType;
    private int engineDisplacement;
    private int bootSpace;
    private int seatCapacity;

    public Car(String carId, String modelNumber, String colour, String transmissionType, String
fuelType,
        String engineType, String bodyType, int engineDisplacement, int bootSpace, int seatCapacity)
    {
        this.carId=carId;
        this.modelNumber=modelNumber;
    }
}

```

```
    this.colour=colour;  
  
    this.transmissionType=transmissionType;  
  
    this.fuelType=fuelType;  
  
    this.engineType=engineType;  
  
    this.bodyType=bodyType;  
  
    this.engineDisplacement=engineDisplacement;  
  
    this.bootSpace=bootSpace;  
  
    this.seatCapacity=seatCapacity;  
  
}  
  
public String getCarId()  
{  
  
    return this.carId;  
  
}  
  
public void setCarId(String carId)  
{  
  
    this.carId=carId;  
  
}  
  
public String getModelNumber()  
{  
  
    return this.modelNumber;  
  
}  
  
public void setModelNumber(String modelNumber)  
{  
  
    this.modelNumber=modelNumber;  
  
}
```

```
public String getColour()
{
    return this.colour;
}

public void setColour(String colour)
{
    this.colour=colour;
}

public String getTransmissionType()
{
    return this.transmissionType;
}

public void setTransmissionType(String transmissionType)
{
    this.transmissionType=transmissionType;
}

public String getFuelType()
{
    return this.fuelType;
}

public void setFuelType(String fuelType)
{
    this.fuelType=fuelType;
}

public String getEngineType()
```

```
{  
    return this.engineType;  
}  
  
public void setEngineType(String engineType)  
{  
    this.engineType=engineType;  
}  
  
public String getBodyType()  
{  
    return this.bodyType;  
}  
  
public void setBodyType(String bodyType)  
{  
    this.bodyType=bodyType;  
}  
  
public int getEngineDisplacement()  
{  
    return this.engineDisplacement;  
}  
  
public void setEngineDisplacement(int engineDisplacement)  
{  
    this.engineDisplacement=engineDisplacement;  
}  
  
public int getBootSpace()  
{
```

```
        return this.bootSpace;  
    }  
  
    public void setBootSpace(int bootSpace)  
    {  
        this.bootSpace=bootSpace;  
    }  
  
    public int getSeatCapacity()  
    {  
        return this.seatCapacity;  
    }  
  
    public void setSeatCapacity(int seatCapacity)  
    {  
        this.seatCapacity=seatCapacity;  
    }  
  
}
```

---

### **Carutility.java**

```
import java.util.Scanner;  
  
public class CarUtility {  
  
    public static void main(String[] args) {  
        // Fill the code  
        Car[] carObj=createCarDetails();
```

```

int n=carObj.length;

System.out.println("No of Car Details "+n);

for(int i=0;i<n;i++)

{

    System.out.println("Car Details "+(i+1));

    System.out.println("Car Id "+carObj[i].getCarId());

    System.out.println("Model Number "+carObj[i].getModelNumber());

    System.out.println("Colour "+carObj[i].getColour());

    System.out.println("Transmission Type "+carObj[i].getTransmissionType());

    System.out.println("Fuel Type "+carObj[i].getFuelType());

    System.out.println("Engine Type "+carObj[i].getEngineType());

    System.out.println("Body Type "+carObj[i].getBodyType());

    System.out.println("Engine Displacement "+carObj[i].getEngineDisplacement());

    System.out.println("Boot Space "+carObj[i].getBootSpace());

    System.out.println("Seat Capacity "+carObj[i].getSeatCapacity());


}

}


```

```

public static Car[] createCarDetails(){

    Scanner sc = new Scanner(System.in);

    // Fill the code

    System.out.println("Enter the number of car details to created");

    int n=sc.nextInt();

```

```
Car[] carObj=new Car[n];

for(int i=0;i<n;i++)

{

    System.out.println("Enter Car Id");

    String carId=sc.nextLine();

    carId = sc.nextLine();

    System.out.println("Enter Model Number");

    String modelNumber=sc.nextLine();

    System.out.println("Enter Colour");

    String colour=sc.nextLine();

    System.out.println("Enter Transmission Type");

    String transmissionType=sc.nextLine();

    System.out.println("Enter Fuel Type");

    String fuelType=sc.nextLine();

    System.out.println("Enter Engine Type");

    String engineType=sc.nextLine();

    System.out.println("Enter Body Type");

    String bodyType=sc.nextLine();

    System.out.println("Enter Engine Displacement");

    int engineDisplacement=sc.nextInt();

    System.out.println("Enter Boot Space");

    int bootSpace=sc.nextInt();

    System.out.println("Enter Seat Capacity");

    int seatCapacity=sc.nextInt();

    carObj[i]=new Car(carId,modelNumber, colour, transmissionType, fuelType , engineType, bodyType, engineDisplacement, bootSpace, seatCapacity);
```

```
        }

        return carObj;
    }

}

13)Electricity board

public class CustomerDetails {

    private String customerId;

    private String customerName;

    private long phoneNumber;

    private String city;

    private double unitsConsumed;

    private double costPerUnit;

    public CustomerDetails(String customerId, String customerName, long
        phoneNumber, String city, double unitsConsumed, double costPerUnit) {
        this.customerId = customerId;
        this.customerName = customerName;
        this.phoneNumber = phoneNumber;
        this.city = city;
        this.unitsConsumed = unitsConsumed;
        this.costPerUnit = costPerUnit;
    }

    public String getCustomerId() {
        return customerId;
    }

    public void setCustomerId( String customerId){
```

```
this.customerId=customerId;  
}  
  
public String getCustomerName() {  
    return customerName;  
}  
  
public void setCustomerName( String customerName) {  
    this.customerName=customerName;  
}  
  
public long getPhoneNumber(){  
    return phoneNumber;  
}  
  
public void setPhoneNumber(long phoneNumber){  
    this.phoneNumber=phoneNumber;  
}  
  
public String getCity(){  
    return city;  
}  
  
public void setCity( String city){  
    this.city=city;  
}  
  
public double getUnitsConsumed(){  
    return unitsConsumed;  
}  
  
public void setUnitsConsumed(double unitsConsumed){  
    this.unitsConsumed=unitsConsumed;
```

```
}

public double getCostPerUnit(){

return costPerUnit;

}

public void setCustomerId(double costPerUnit){

this.costPerUnit=costPerUnit;

}

public double calculateElectricityBill(){

double amount = unitsConsumed * costPerUnit;

return amount;

}
```

---

```
import java.util.*;

public class Main {

public static void main(String[] args) {

Scanner scanner=new Scanner(System.in);

System.out.println("Enter Customer Id");

String str= scanner.nextLine();

System.out.println("Enter Customer Name");

String s2 = scanner.nextLine();

System.out.println("Enter Phone Number");

long phone = scanner.nextLong();

System.out.println("Enter City");
```

```
String city = scanner.next();

System.out.println("Enter Units Consumed");

double cu=scanner.nextDouble();

System.out.println("Enter Cost per Unit");

double cuu=scanner.nextDouble();

CustomerDetails c1= new CustomerDetails(str,s2,phone,city,cu,cuu);

System.out.println("Amount to be paid is Rs."+

String.format("%.2f",c1.calculateElectricityBill()));

}

}
```

#### 14)Movie Static

```
import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter movie name");

String str= scanner.nextLine();

System.out.println("Enter no of bookings");

int bookings = scanner.nextInt();

System.out.println("Enter the available tickets");

int availableTickets = scanner.nextInt();

Ticket ticket = new Ticket();

ticket.setAvailableTickets(availableTickets);

for (int i = 0; i < bookings; ++i) {

System.out.println("Enter the ticketid");
```

```
int ticketId = scanner.nextInt();

System.out.println("Enter the price");

int price = scanner.nextInt();

System.out.println("Enter the no of tickets");

int tickets = scanner.nextInt();

ticket.setTicketid(ticketId);

ticket.setPrice(price);

System.out.println("Available tickets: " + ticket.getAvailableTickets());

int cost = ticket.calculateTicketCost(tickets);

if(cost>0){

System.out.println("Total amount:" + cost);}

else{

System.out.println("Tickets are not available");

return;

}if(ticket.getAvailableTickets()==0){

System.out.println("House full");

}else{

System.out.println("Available ticket after booking:" + ticket.getAvailableTickets());

}

}

}
```

---

```
public class Ticket {

private int ticketid;
```

```
private int price;

private static int availableTickets;

public int getTicketid() {

    return ticketid;

}

public void setTicketid(int ticketid) {

    this.ticketid = ticketid;

}

public int getPrice() {

    return price;

}

public void setPrice(int price) {

    this.price = price;

}

public static int getAvailableTickets() {

    return availableTickets;

}

public static void setAvailableTickets(int availableTickets) {

    if (availableTickets > 0) {

        Ticket.availableTickets = availableTickets;

    }else{

        Ticket.availableTickets=-1;

    }

}

public int calculateTicketCost(int nooftickets) {
```

```
if (nooftickets <= availableTickets) {  
    availableTickets -= nooftickets;  
    return nooftickets * price;  
} else {  
    return -1;  
}  
}  
}  
}
```

15)game card

```
public class CardPoints {  
  
    private int cardId;  
    private String holderName;  
    private int balancePoints;  
  
    public CardPoints(int cardId, String holderName, int balancePoints) {  
        this.cardId = cardId;  
        this.holderName = holderName;  
        this.balancePoints = balancePoints;  
    }  
  
    public boolean withdrawPoints(int points) {  
        if(balancePoints >= points) {
```

```
balancePoints -= points;

System.out.println("Balance points after used:" + balancePoints);

return true;

} else {

    System.out.println("Sorry!!! No enough points");

    return false;

}

}

public int getCardId() {

    return cardId;

}

public void setCardId(int cardId) {

    this.cardId = cardId;

}

public String getHolderName() {

    return holderName;

}

public void setHolderName(String holderName) {

    this.holderName = holderName;

}
```

```
public int getBalancePoints() {  
    return balancePoints;  
}  
  
public void setBalancePoints(int balancePoints) {  
    this.balancePoints = balancePoints;  
}  


---



```
import java.util.Scanner;  
  
public class GameCardDetails {  
  
    public CardPoints getCardDetails()  
    {  
        Scanner sc = new Scanner(System.in);  
  
        System.out.println("Enter card id");  
        int cardId = sc.nextInt();  
  
        System.out.println("Enter card holder name");  
        String holderName = sc.next();  
  
        System.out.println("Enter balance points");  
        int balancePoints = sc.nextInt();  
  
        do{  
            if (balancePoints <= 0){
```


```

```
System.out.println("Balance points should be positive");

System.out.println("Enter balance points");

balancePoints = sc.nextInt();

continue;

}

else

{

    break;

}

}

while(true);

return new CardPoints(cardId, holderName, balancePoints);

}

public int getPointUsage()

{

Scanner sc = new Scanner(System.in);

int pointUsage = sc.nextInt();

if (pointUsage <= 0){

    System.out.println("Points should be positive");

    return getPointUsage();

}

return pointUsage;

}
```

```
public static void main(String[] arg)
{
    GameCardDetails gameCardDetails = new GameCardDetails();
    CardPoints cp = gameCardDetails.getCardDetails();
    System.out.println("Enter points should be used");
    int points = gameCardDetails.getPointUsage();
    cp.withdrawPoints(points);

}
}
```

#### 16) doctor details

```
public class Doctor {
    private String doctorId;
    private String doctorName;
    private String specialization;
    private Hospital hospital;
    public String getDoctorId() {
        return doctorId;
    }
    public void setDoctorId(String doctorId) {
        this.doctorId = doctorId;
    }
    public String getDoctorName() {
```

```
        return doctorName;  
    }  
  
    public void setDoctorName(String doctorName) {  
        this.doctorName = doctorName;  
    }  
  
    public String getSpecialization() {  
        return specialization;  
    }  
  
    public void setSpecialization(String specialization) {  
        this.specialization = specialization;  
    }  
  
    public Hospital getHospital() {  
        return hospital;  
    }  
  
    public void setHospital(Hospital hospital) {  
        this.hospital = hospital;  
    }  
  
    public Doctor(String doctorId, String doctorName, String specialization, Hospital hospital) {  
        this.doctorId = doctorId;  
        this.doctorName = doctorName;  
        this.specialization = specialization;  
        this.hospital = hospital;  
    }  
}  
  
-----  
public class Hospital {
```

```
private String hospitalName;  
private long contactNumber;  
private String city;  
  
public String getHospitalName() {  
    return hospitalName;  
}  
  
public long getContactNumber() {  
    return contactNumber;  
}  
  
public String getCity() {  
    return city;  
}  
  
public void setHospitalName(String hospitalName) {  
    this.hospitalName = hospitalName;  
}  
  
public void setContactNumber(long contactNumber) {  
    this.contactNumber = contactNumber;  
}  
  
public void setCity(String city) {  
    this.city = city;  
}  
  
public Hospital(String hospitalName, long contactNumber, String city) {  
    this.hospitalName = hospitalName;  
    this.contactNumber = contactNumber;  
    this.city = city;
```

```
 }  
 }  
  
-----
```

```
import java.util.Scanner;  
  
public class Main {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        Doctor doc=createDoctorDetails();  
  
        System.out.println("Doctor id:"+doc.getDoctorId());  
  
        System.out.println("Doctor name:"+doc.getDoctorName());  
  
        System.out.println("Specialization:"+doc.getSpecialization());  
  
        System.out.println("Hospital name:"+doc.getHospital().getHospitalName());  
  
        System.out.println("Contact Number:"+doc.getHospital().getContactNumber());  
  
        System.out.println("City:"+doc.getHospital().getCity());  
    }  
  
    public static Doctor createDoctorDetails() {  
        Scanner sc = new Scanner(System.in);  
  
        System.out.println("Enter Hospital Name");  
  
        String hName=sc.nextLine();  
  
        System.out.println("Enter Contact Number");  
  
        long cNo=sc.nextLong();  
  
        sc.nextLine();  
  
        System.out.println("Enter City");  
    }  
}
```

```
String city=sc.nextLine();

Hospital h=new Hospital(hName,cNo,city);

System.out.println("Enter Doctor Id");

String id=sc.nextLine();

System.out.println("Enter Doctor Name");

String dName=sc.nextLine();

System.out.println("Enter Specialization");

String specialization = sc.nextLine();

Doctor doc=new Doctor(id,dName,specialization,h);

return doc;

}

}

17)Disney
```

```
public class BoatHouseBooking extends Booking{
```

```
// Fill the code
```

```
private int noOfDays;
```

```
private String foodType;
```

```
public int getNoOfDays() {
```

```
    return noOfDays;
```

```
}
```

```
public void setNoOfDays(int noOfDays) {
```

```
this.noOfDays = noOfDays;  
}  
  
  
public String getFoodType() {  
    return foodType;  
}  
  
  
public void setFoodType(String foodType) {  
    this.foodType = foodType;  
}  
  
  
public BoatHouseBooking(String customerName, String cityName, String phoneNumber, int noOfPeople,  
int noOfDays,String foodType) {  
    super(customerName, cityName, phoneNumber, noOfPeople);  
    this.noOfDays=noOfDays;  
    this.foodType=foodType;  
}  
  
  
public double calculateTotalAmount() {  
  
  
    // Fill the code  
  
  
    if(this.foodType.equals("NonVeg")){  
        return (double)(this.noOfPeople*800) + (this.noOfDays*3000)+500;  
    }else if(this.foodType.equals("Veg") || this.foodType.equals("VEG")){  
        return (double)((this.noOfPeople*800) + (this.noOfDays*3000)+250);  
    }  
}
```

```
}
```

```
return 0.0;
```

```
}
```

```
}
```

---

```
public class BoatRideBooking extends Booking{
```

```
// Fill the code
```

```
private float noOfHours;
```

```
private String guide;
```

```
public BoatRideBooking(String customerName, String cityName, String phoneNumber, int noOfPeople,  
float noOfHours, String guide) {
```

```
super(customerName, cityName, phoneNumber, noOfPeople);
```

```
this.noOfHours=noOfHours;
```

```
this.guide=guide;
```

```
}
```

```
public double calculateTotalAmount() {
```

```
// Fill the code

if(this.guide.equals("Yes") || this.guide.equals("yes")){
    return (double)(this.noOfPeople*80) + (this.noOfHours*300) + 150;
} else if(this.guide.equals("No") || this.guide.equals("no")){
    return (double)(this.noOfPeople*80) + (this.noOfHours*300);
}
return 0.0;
}

}
```

---

```
public abstract class Booking {

// Fill the code

protected String customerName;
protected String cityName;
protected String phoneNumber;
protected int noOfPeople;

public Booking(String customerName, String cityName, String phoneNumber, int noOfPeople) {
    this.customerName = customerName;
    this.cityName = cityName;
    this.phoneNumber = phoneNumber;
}
```

```
this.noOfPeople = noOfPeople;  
}  
  
  
public String getCustomerName() {  
    return customerName;  
}  
  
  
public void setCustomerName(String customerName) {  
    this.customerName = customerName;  
}  
  
  
public String getCityName() {  
    return cityName;  
}  
  
  
public void setCityName(String cityName) {  
    this.cityName = cityName;  
}  
  
  
public String getPhoneNumber() {  
    return phoneNumber;  
}  
  
  
public void setPhoneNumber(String phoneNumber) {  
    this.phoneNumber = phoneNumber;
```

```
}
```

```
public int getNoOfPeople() {
```

```
    return noOfPeople;
```

```
}
```

```
public void setNoOfPeople(int noOfPeople) {
```

```
    this.noOfPeople = noOfPeople;
```

```
}
```

```
public abstract double calculateTotalAmount();
```

```
}
```

---

```
import java.util.Scanner;
```

```
public class UserInterface {
```

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

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

```
        // Fill the code
```

```
        System.out.println("Enter the Customer Name");
```

```
        String customerName = s.next();
```

```
        System.out.println("Enter the City name");
```

```
String cityName = s.next();  
System.out.println("Enter the phone number");  
String phoneNumber = s.next();  
System.out.println("Enter number of people");  
int noOfPeople = s.nextInt();  
System.out.println("Enter the option\n" +  
"1. Boat House Booking\n" +  
"2. Boat Ride Booking"  
);  
int option = s.nextInt();  
  
switch (option){  
case 1:  
System.out.println("Enter number of days");  
int noOfDays = s.nextInt();  
System.out.println("Enter food type (Veg/NonVeg)");  
String foodType = s.next();  
  
BoatHouseBooking BHB = new BoatHouseBooking(  
customerName,  
cityName,  
phoneNumber,  
noOfPeople,  
noOfDays,  
foodType);
```

```
System.out.println("Your booking has been confirmed pay Rs."+BHB.calculateTotalAmount());  
break;
```

case 2:

```
System.out.println("Enter number of hours");  
float noOfHours = s.nextFloat();  
System.out.println("Do you want guide (Yes/No)");  
String guide = s.next();
```

```
BoatRideBooking BRB = new BoatRideBooking(  
customerName,  
cityName,  
phoneNumber,  
noOfPeople,  
noOfHours,  
guide  
);
```

```
System.out.println("Your booking has been confirmed pay Rs."+BRB.calculateTotalAmount());
```

```
break;
```

```
}
```

```
}
```

```
}
```

## 18)College Fees

```
public class DayScholar extends Student{
```

```
// Fill the code
```

```
private int busNumber;
```

```
private float distance;
```

```
public DayScholar(int studentId, String studentName, String department, String gender, String category,  
double collegeFee,int busNumber,float distance) {
```

```
super(studentId, studentName, department, gender, category, collegeFee);
```

```
this.busNumber=busNumber;
```

```
this.distance=distance;
```

```
}
```

```
public int getBusNumber() {
```

```
return busNumber;
```

```
}
```

```
public void setBusNumber(int busNumber) {
```

```
this.busNumber = busNumber;  
}
```

```
public float getDistance() {  
    return distance;  
}
```

```
public void setDistance(float distance) {  
    this.distance = distance;  
}
```

```
public double calculateTotalFee() {
```

// Fill the code

```
int busFees=0;
```

```
if(this.distance > 30 && this.distance <= 40){
```

busFees=28000;

```
}else if(this.distance > 20 && this.distance <= 30){
```

busFees=20000;

```
}else if(this.distance > 10 && this.distance <= 20){
```

```
busFees=12000;  
}else if(this.distance <= 10){  
busFees=6000;  
}  
if(this.collegeFee>0){  
return (double)(this.collegeFee+busFees);  
}  
  
return 0;  
}  
  
}
```

---

```
public abstract class Student {
```

```
// Fill the code  
  
protected int studentId;  
protected String studentName;  
protected String department;  
protected String gender;
```



```
public String getStudentName() {  
    return studentName;  
}
```

```
public void setStudentName(String studentName) {  
    this.studentName = studentName;  
}
```

```
public String getDepartment() {  
    return department;  
}
```

```
public void setDepartment(String department) {  
    this.department = department;  
}
```

```
public String getGender() {
```

```
    return gender;
```

```
}
```

```
public void setGender(String gender) {
```

```
    this.gender = gender;
```

```
}
```

```
public String getCategory() {
```

```
    return category;
```

```
}
```

```
public void setCategory(String category) {
```

```
    this.category = category;
```

```
}
```

```
public double getCollegeFee() {  
    return collegeFee;  
}  
  
public void setCollegeFee(double collegeFee) {  
    this.collegeFee = collegeFee;  
}  
  
public abstract double calculateTotalFee();  
}  
  
-----  
import java.util.Scanner;
```

```
public class UserInterface {
```

```
public static void main(String[] args) {  
  
    Scanner s = new Scanner(System.in);  
    // Fill the code  
  
    System.out.println("Enter Student Id");  
    int studentId = s.nextInt();  
  
    System.out.println("Enter Student name");  
    String studentName = s.next();  
  
    System.out.println("Enter Department name");  
    String department = s.next();  
  
    System.out.println("Enter gender");  
    String gender = s.next();  
  
    System.out.println("Enter category");  
    String category = s.next();  
  
    System.out.println("Enter College fee");  
    double collegeFee = s.nextDouble();  
  
    switch(category){  
        case "DayScholar":  
            System.out.println("Enter Bus number");  
            int busNumber = s.nextInt();  
            System.out.println("Enter the distance");  
    }  
}
```

```
float distance = s.nextFloat();

DayScholar dayScholar = new DayScholar(
    studentId,
    studentName,
    department,
    gender,
    category,
    collegeFee,
    busNumber,
    distance
);

System.out.println("Total College fee is "+dayScholar.calculateTotalFee());
break;

case "Hosteller":
    System.out.println("Enter the room number");
    int roomNumber = s.nextInt();
```

```
System.out.println("Enter the Block name");
char blockName = s.next().charAt(0);
System.out.println("Enter the room type");
String roomType = s.next();

Hosteller hosteller = new Hosteller(
    studentId,
    studentName,
    department,
    gender,
    category,
    collegeFee,
    roomNumber,
    blockName,
    roomType
);

System.out.println("Total College fee is "+hosteller.calculateTotalFee());
break;
}
}
```

```
}
```

---

```
public class Hosteller extends Student{
```

```
// Fill the code
```

```
private int roomNumber;  
private char blockName;  
private String roomType;
```

```
public Hosteller(int studentId, String studentName, String department, String gender, String category,  
double collegeFee, int roomNumber, char blockName, String roomType) {  
super(studentId, studentName, department, gender, category, collegeFee);  
this.roomNumber=roomNumber;  
this.blockName=blockName;  
this.roomType=roomType;  
}
```

```
@Override  
public double calculateTotalFee() {  
    int hostelFee=0,roomTypeAmount=0;  
    switch (this.blockName){  
        case 'A':  
            hostelFee=60000;  
            if(this.roomType.equals("AC")){  
                roomTypeAmount=8000;  
            }  
            break;  
        case 'B':  
            hostelFee=50000;  
            if(this.roomType.equals("AC")){  
                roomTypeAmount=5000;  
            }  
            break;  
        case 'C':  
            hostelFee=40000;  
            if(this.roomType.equals("AC")){  
                roomTypeAmount=2500;  
            }  
            break;  
    }  
}
```

```
if(this.roomType.equals("AC")){
    return (double)(this.collegeFee+hostelFee+roomTypeAmount);
} else if(!this.roomType.equals("AC")){
    return (double)(this.collegeFee+hostelFee);
}
return 0.0;
}
```

## 19)Departmental store

```
public interface BonusPoints {
    // Fill the code
    public abstract double calculateBonusPoints();
}
```

---

```
public class CustomerDetails implements BonusPoints, DoorDelivery {
```

```
// Fill the code

private String customerName;

private String phoneNumber;

private String streetName;

private double billAmount;

private int distance;

public String getCustomerName() {

    return customerName;

}

public void setCustomerName(String customerName) {

    this.customerName = customerName;

}

public String getPhoneNumber() {

    return phoneNumber;

}

public void setPhoneNumber(String phoneNumber) {

    this.phoneNumber = phoneNumber;

}

public String getStreetName() {

    return streetName;

}

public void setStreetName(String streetName) {

    this.streetName = streetName;

}

public double getBillAmount() {
```

```
        return billAmount;  
    }  
  
    public void setBillAmount(double billAmount) {  
        this.billAmount = billAmount;  
    }  
  
    public int getDistance() {  
        return distance;  
    }  
  
    public void setDistance(int distance) {  
        this.distance = distance;  
    }  
  
    public CustomerDetails(String customerName, String phoneNumber, String streetName, double  
billAmount,  
                           int distance) {  
        this.customerName = customerName;  
        this.phoneNumber = phoneNumber;  
        this.streetName = streetName;  
        this.billAmount = billAmount;  
        this.distance = distance;  
    }  
  
  
    public double calculateBonusPoints() {  
        // Fill the code  
        double billAmount=this.getBillAmount();  
        if(billAmount<250)  
    {
```



```
}
```

---

```
public interface DoorDelivery {  
  
    // Fill the code  
  
    public abstract double deliveryCharge();  
  
}
```

---

```
import java.util.Scanner;
```

```
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner sc = new Scanner (System.in);  
  
        // Fill the code  
  
        System.out.println("Enter the customer name");  
  
        String name=sc.nextLine();  
  
        System.out.println("Enter the phone number");  
  
        String phone=sc.nextLine();  
  
        System.out.println("Enter the street name");  
  
        String street=sc.nextLine();  
  
        System.out.println("Enter the bill amount");  
  
        double amount=sc.nextDouble();  
  
        sc.nextLine();  
  
        System.out.println("Enter the distance");  
  
        int distance=sc.nextInt();  
  
        CustomerDetails obj=new CustomerDetails(name,phone,street,amount,distance);
```

```
        double bonus=obj.calculateBonusPoints();

        double delivery=obj.deliveryCharge();

        System.out.println("Customer name "+obj.getCustomerName());

        System.out.println("Phone number "+obj.getPhoneNumber());

        System.out.println("Street name "+obj.getStreetName());

        System.out.println("Bonus points "+bonus);

        System.out.println("Delivery charge "+delivery);

    }

}

20)Viveks
```

```
public abstract class Bero {

    protected String beroType;

    protected String beroColour;

    protected double price;

    public String getBeroType() {

        return beroType;

    }

    public void setBeroType(String beroType) {

        this.beroType = beroType;

    }

    public String getBeroColour() {

        return beroColour;

    }

    public void setBeroColour(String beroColour) {

        this.beroColour = beroColour;

    }

}
```

```
}

public double getPrice() {

    return price;

}

public void setPrice(double price) {

    this.price = price;

}

public Bero(String beroType, String beroColour) {

    this.beroType = beroType;

    this.beroColour = beroColour;

}

public abstract void calculatePrice();

}
```

---

```
public class CustomerDetails {

    private String customerName;

    private long phoneNumber;

    private String address;

    public String getCustomerName() {

        return customerName;

    }

    public void setCustomerName(String customerName) {

        this.customerName = customerName;

    }

    public long getPhoneNumber() {
```

```
        return phoneNumber;  
    }  
  
    public void setPhoneNumber(long phoneNumber) {  
        this.phoneNumber = phoneNumber;  
    }  
  
    public String getAddress() {  
        return address;  
    }  
  
    public void setAddress(String address) {  
        this.address = address;  
    }  
  
    public CustomerDetails(String customerName, long phoneNumber, String address) {  
        super();  
        this.customerName = customerName;  
        this.phoneNumber = phoneNumber;  
        this.address = address;  
    }  
}
```

---

```
public class Discount {  
    public double calculateDiscount(Bero bObj)  
    {  
        double price=bObj.getPrice();  
        double discount=0;  
        if(bObj instanceof SteelBero)
```

```
{  
    discount=price*0.1;  
}  
  
else if(bObj instanceof WoodenBero)  
{  
    discount=price*0.15;  
}  
  
return discount;  
}  
}
```

---

```
public class SteelBero extends Bero {  
  
    private int beroHeight;  
  
    public int getBeroHeight() {  
        return beroHeight;  
    }  
  
    public void setBeroHeight(int beroHeight) {  
        this.beroHeight = beroHeight;  
    }  
  
    public SteelBero(String type, String colour, int height) {  
        // TODO Auto-generated constructor stub  
    }  
}
```

```
super(type, colour);

this.beroHeight = height;

}

@Override

public void calculatePrice() {

    int height=this.getBeroHeight();

    if(height==3)

    {

        this.setPrice(5000);

    }

    else if(height==5)

    {

        this.setPrice(8000);

    }

    else if(height==7)

    {

        this.setPrice(10000);

    }

}

}

-----
```

```
public class WoodenBero extends Bero {

    private String woodType;
```

```
public String getWoodType() {  
    return woodType;  
}  
  
public void setWoodType(String woodType) {  
    this.woodType = woodType;  
}  
  
public WoodenBero(String beroType, String beroColour, String woodType) {  
    super(beroType, beroColour);  
    this.woodType = woodType;  
}  
  
@Override  
public void calculatePrice() {  
    // TODO Auto-generated method stub  
    String type=this.getWoodType();  
    if(type.equals("Ply Wood"))  
    {  
        this.setPrice(15000);  
    }  
    else if(type.equals("Teak Wood"))  
    {  
        this.setPrice(12000);  
    }  
}
```

```
        else if(type.equals("Engineered Wood"))

    {

        this.setPrice(10000);

    }

}

-----
```

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        System.out.println("Enter Customer Name");

        String name=sc.nextLine();

        System.out.println("Enter Phone Number");

        long phone=sc.nextLong();

        sc.nextLine();

        System.out.println("Enter Address");

        String address=sc.nextLine();

        System.out.println("Enter Bero Type");

        String type=sc.nextLine();

        System.out.println("Enter Bero Colour");

        String colour=sc.nextLine();

        double price=0;
```

```
Discount d=new Discount();

Bero obj;

if(type.equals("Steel Bero"))

{

    System.out.println("Enter Bero Height");

    int height=sc.nextInt();

    sc.nextLine();

    obj=new SteelBero(type,colour,height);

    obj.calculatePrice();

    price=obj.getPrice()- d.calculateDiscount(obj);

}

else if(type.equals("Wooden Bero"))

{

    System.out.println("Enter Wood Type");

    String wtype=sc.nextLine();

    obj=new WoodenBero(type,colour,wtype);

    obj.calculatePrice();

    price=obj.getPrice()- d.calculateDiscount(obj);

}

System.out.print("Amount needs to be paid Rs.");

System.out.printf("%.2f",price);

}

}
```

21)Campus radio

```
//make the necessary change to make this class a Exception  
public class StationNotAvailableException extends Exception{  
    public StationNotAvailableException(String message){  
        super(message);  
    }  
}
```

---

```
import java.util.Scanner;
```

```
public class UserInterface {  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        System.out.println("Scan the radio station");  
        float freq = sc.nextFloat();  
  
        try{  
            Validator vd=new Validator();  
            vd.validateStation(freq);  
            System.out.println("Radio Station on!");  
        }catch(StationNotAvailableException e){  
            System.out.println("Radio Station not available");  
        }  
        // fill in the code  
    }  
}
```

```
    }

}

-----
```

```
public class Validator {

    public static boolean validateStation(float freq) throws StationNotAvailableException {
        boolean flag = false;

        if (freq == 91.2f || freq == 93.5f || freq == 98.9f || freq == 109.4f)
            flag = true;
        else{
            throw new StationNotAvailableException("Radio Station not available");
            // fill the logic
        }
        return flag;
    }
}

22)String extract

import java.util.Scanner;
```

```
public class Main {  
  
    public static void main(String args[])  
    {  
        Main stringExtraction = new Main();  
  
        Scanner sc=new Scanner(System.in);  
  
        // Fill the code  
  
        System.out.println("Enter the String");  
  
        String sentence = sc.nextLine();  
  
        System.out.println("Enter First Index");  
  
        int s = sc.nextInt();  
  
        System.out.println("Enter Second Index");  
  
        int e = sc.nextInt();  
  
        System.out.println(stringExtraction.extractString(sentence,s,e));  
  
    }  
  
    public String extractString(String sentence,int number1,int number2)  
    {  
        // Fill the code  
    }  
}
```

```
StringBuilder stringBuilder = new StringBuilder();

try{
    if(number1<0 || number1 > sentence.length()){
        throw new StringIndexOutOfBoundsException();
    }else if(number2<0 || number2 > sentence.length()){
        throw new StringIndexOutOfBoundsException();
    }else if(number1>number2){
        throw new StringIndexOutOfBoundsException();
    }
    else{
        for (int i = number1; i < number2; i++) {
            stringBuilder.append(sentence.charAt(i));
        }
        stringBuilder.append(".");
    }
} catch (StringIndexOutOfBoundsException s){
    stringBuilder.append("Extraction of String using the given index is not possible.Thanks for using the application. ");
}finally {
    stringBuilder.append("Thanks for using the application.");
}

return stringBuilder.toString();
}
```

}

23)Vega Cricket

```
import java.util.*;
import java.util.stream.*;

public class Main {
    public static void main(String args[]){
        Scanner sc=new Scanner(System.in);
        ManagementUtility M=new ManagementUtility();
        Map<String,Integer> playerMap=new HashMap<String,Integer>();
        M.setPlayerMap(playerMap);
        int z=0;
        while(z!=3)
        {
            System.out.println("Select an option:");
            System.out.println("1.Add player score");
            System.out.println("2.Display");
            System.out.println("3.Exit");
            z=sc.nextInt();
            String name="";
            int score=0;
            if(z==1)
            {
```

```
System.out.println("Enter the player name");

name=sc.next();

System.out.println("Enter the score");

score=sc.nextInt();

M.addPlayerScore(name,score);

}

else if(z==2)

{

Map<String,Integer> playermap=M.getPlayerMap();

Stream<Map.Entry<String,Integer>> playerStream=playermap.entrySet().stream();

int max_score=M.maximumScore(playerStream);

if(max_score==0)

{

System.out.println("No players found");

}

else

{

System.out.println("The maximum score of an individual player for these match is

"+max_score);

}

}

else{

System.out.println("Thank you for using the application.");

break;

}
```

```
    }  
}  
  
-----
```

```
import java.util.Map;  
import java.util.*;  
import java.util.stream.Stream;  
import java.util.stream.Collectors;  
  
  
public class ManagementUtility {  
    private Map<String, Integer> playerMap;  
  
  
    public Map<String, Integer> getPlayerMap() {  
        return playerMap;  
    }  
  
  
    public void setPlayerMap(Map<String, Integer> playerMap) {  
        this.playerMap = playerMap;  
    }  
  
  
    public void addPlayerScore(String playerName, int score) {  
        Map<String, Integer> playerMap = getPlayerMap();  
        playerMap.put(playerName, score);  
  
  
        //Fill the code here  
    }  
}
```

```
}
```

```
public static int maximumScore(Stream<Map.Entry<String, Integer>> playerStream){  
  
    //List list = playerStream.collect(Collectors.toList());  
  
    //Collections.sort(hashList, Collections.reverseOrder());  
  
    List<Integer> l=playerStream.map(Map.Entry::getValue).collect(Collectors.toList());  
  
    if(l.isEmpty())  
  
    {  
  
        return 0;  
  
    }  
  
    else{  
  
        int m=Collections.max(l);  
  
        System.out.println(m);  
  
        // Fill the code here  
  
        return m;  
    }  
}
```

```
}
```

24)Top tier

```
import java.util.*;
```

```
public class Main {
```

```

public static void main(String args[]){
    Scanner sc=new Scanner(System.in);

    System.out.println("Enter the number of vehicles");

    int n = Integer.parseInt(sc.nextLine());

    VehicleUtility vh = new VehicleUtility();

    vh.setVehicleMap(new HashMap<String,Double>());

    for(int i=0;i<n;i++){

        System.out.println("Enter the vehicle name and price of Vehicle "+(i+1));

        String veh=sc.nextLine();

        double price=Double.parseDouble(sc.nextLine());

        vh.addVehiclePriceDetails(veh,price);

    }

    Map<String,Double> polmap=vh.getVehicleMap();

    char choice;

    do

    {

        System.out.println("Enter the vehicle name to be searched");

        String search = sc.nextLine();

        if(vh.calculateCostAfterDiscount(search)!=0){

            System.out.println("Price after discount for "+ search +" is

"+vh.calculateCostAfterDiscount(search));

        }else{

            System.out.println( "TVS Apache RTR 180 is not available currently");

        }

        System.out.println("Do you want to continue (Y/N)");
    }
}

```

```
choice=sc.nextLine().charAt(0);

if((choice=='n')||(choice=='N')){

    System.out.println("Thank you for using the Application");

    break;

}

}

while((choice=='y')||(choice=='Y'));

}

-----
```

```
import java.util.HashMap;

import java.util.Map;

public class VehicleUtility {

    private Map<String, Double> vehicleMap = new HashMap<String, Double>();

    public Map<String, Double> getVehicleMap() {

        return vehicleMap;

    }

    public void setVehicleMap(Map<String, Double> vehicleMap) {

        this.vehicleMap = vehicleMap;

    }

    // This method should add the vehicleName as key and the price of the
```

```

// vehicle as value into a Map

public void addVehiclePriceDetails(String vehicleName, double price) {

    // fill the code

    vehicleMap.put(vehicleName,price);

}

// This method should calculate the discount and return the selling price

// after the discount for the vehicle name passed as an argument.

public double calculateCostAfterDiscount(String vehicleName) {

    if(vehicleMap.containsKey(vehicleName)){

        double discount=0;

        // fill the code

        double sp=0;

        double h=vehicleMap.get(vehicleName);

        if(vehicleName.contains("TVS")){
            discount=10;
        }else if(vehicleName.contains("Honda")){
            discount=5;
        }else if(vehicleName.contains("Yamaha")){
            discount=7;
        }

        sp= h*(100-discount)/100;

        return sp;
    }else{

```

```
    return 0;
```

```
}
```

```
}
```

```
}
```

25)Global Hospital

```
import java.util.Scanner;
```

```
import java.util.stream.Stream;
```

```
import java.util.List;
```

```
public class Main {
```

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

```
        Scanner sc = new Scanner(System.in);
```

```
        // Fill the code here
```

```
        ManagementUtility mu = new ManagementUtility();
```

```
        System.out.println("Enter the number of donors");
```

```
        int n = Integer.parseInt(sc.nextLine());
```

```
        System.out.println("Enter the donor details");
```

```
        String[] Strings = new String[n];
```

```
        for(int i=0;i<n;i++)
```

```
{
```

```
        String donor = sc.nextLine();
```

```
        Strings[i] = donor;
```

```
}

List<Donor> res = mu.getDonorDetails(Strings);

Stream<Donor> result = mu.getStreamOfDonor(res);

System.out.println("Enter the blood group to search");

String bloodGroup = sc.nextLine();

List<String> finalResult = mu.shortlistedDonor(result,bloodGroup);

if(finalResult.size() != 0)

{

    System.out.println("List of shortlisted donors");

    finalResult.forEach(s ->{System.out.println(s);});

}

else

{

    System.out.println("No donor found");

}

}

-----
```

```
import java.util.ArrayList;

import java.util.Collections;

import java.util.stream.Stream;

import java.util.List;

public class ManagementUtility {
```

```
public List<Donor> getDonorDetails(String [] details) {  
  
    // Fill the code here  
  
    List<Donor> temp = new ArrayList<Donor>();  
  
    for(int i=0;i<details.length;i++)  
    {  
        String newTemp = details[i];  
        String[] splitString = newTemp.split(":");  
        long mb = Long.parseLong(splitString[3]);  
        temp.add(new Donor(splitString[0],splitString[1],splitString[2],mb));  
    }  
    return temp;  
}  
  
public Stream<Donor> getStreamOfDonor(List<Donor> donorDetails) {  
  
    // Fill the code here  
  
    return donorDetails.stream();  
}  
  
public List<String> shortlistedDonor(Stream<Donor> donorStream, String bloodGroup) {  
  
    // Fill the code here
```

```
        List<String> result = new ArrayList<String>();

        donorStream.filter(s-> s.getBloodGroup().equals(bloodGroup)).forEach(s ->
{result.add(s.getDonorName());});

        Collections.sort(result);

        return result;

    }

}
```

---

```
//Don't change the skeleton

public class Donor {

    private String donorId;

    private String donorName;

    private String bloodGroup;

    private long mobileNumber;

    public Donor(String id,String name,String group,long mn)

    {

        this.donorId = id;

        this.donorName = name;

        this.bloodGroup = group;

        this.mobileNumber = mn;

    }

    public String getDonorId() {

        return donorId;

    }
```

```
public void setDonorId(String donorId) {  
    this.donorId = donorId;  
}  
  
public String getDonorName() {  
    return donorName;  
}  
  
public void setDonorName(String donorName) {  
    this.donorName = donorName;  
}  
  
public String getBloodGroup() {  
    return bloodGroup;  
}  
  
public void setBloodGroup(String bloodGroup) {  
    this.bloodGroup = bloodGroup;  
}  
  
public long getMobileNumber() {  
    return mobileNumber;  
}  
  
public void setMobileNumber(long mobileNumber) {  
    this.mobileNumber = mobileNumber;  
}  
}
```

## 26)Hyper super market

```
import java.util.Scanner;  
  
import java.util.stream.Stream;  
  
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner sc = new Scanner(System.in);  
  
        int num = 0;  
  
        ManagementUtility m=new ManagementUtility();  
  
        while(num != 3){  
  
            System.out.println("Select an option:\n1.Add Vegetable to Basket\n2.Calculate  
Bill\n3.Exit");  
  
            num = sc.nextInt();  
  
            if(num == 1){  
  
                VegetableBasket vege=new VegetableBasket();  
  
                System.out.println("Enter the vegetable name");  
  
                String name=sc.next();  
  
                System.out.println("Enter weight in Kgs");  
  
                int weight = sc.nextInt();  
  
                System.out.println("Enter price per Kg");  
  
                int price = sc.nextInt();  
  
                vege.setVegetableName(name);  
            }  
        }  
    }  
}
```

```
vege.setWeightInKgs(weight);

vege.setPricePerKg(price);

m.addToBasket(vege);

}

else if(num==2)

{

List<VegetableBasket>vegeBasketList =m.getVegetableBasketList();

if(vegeBasketList.size()==0)

{

System.out.println("Customer basket is empty. Please add vegetables.");

}

else{

    Stream<VegetableBasket>vegetableBasketStream = vegeBasketList.stream();

    int amount = m.calculateBill(vegetableBasketStream);

    System.out.println("The estimated bill amount is Rs "+amount);

}

}

System.out.println("Thank you for using the application.");

}



---



//Don't change the skeleton


```

```
public class VegetableBasket {  
    private String vegetableName;  
    private int weightInKgs;  
    private int pricePerKg;  
  
    public String getVegetableName() {  
        return vegetableName;  
    }  
  
    public void setVegetableName(String vegetableName) {  
        this.vegetableName = vegetableName;  
    }  
  
    public int getWeightInKgs() {  
        return weightInKgs;  
    }  
  
    public void setWeightInKgs(int weightInKgs) {  
        this.weightInKgs = weightInKgs;  
    }  
  
    public int getPricePerKg() {  
        return pricePerKg;  
    }  
  
    public void setPricePerKg(int pricePerKg) {  
        this.pricePerKg = pricePerKg;  
    }  
}
```

```
}
```

---

```
import java.util.List;  
  
import java.util.stream.Stream;  
  
import java.util.*;  
  
import java.util.stream.Collectors;  
  
import java.util.function.Function;  
  
public class ManagementUtility {  
  
    private List<VegetableBasket> vegetableBasketList = new ArrayList<VegetableBasket>();  
  
  
    public List<VegetableBasket> getVegetableBasketList() {  
  
        return this.vegetableBasketList;  
    }  
  
  
    public void setVegetableBasketList(List<VegetableBasket> vegetableBasketList) {  
  
        this.vegetableBasketList = vegetableBasketList;  
    }  
  
  
    public void addToBasket(VegetableBasket fbObj) {  
  
        List<VegetableBasket> vge= getVegetableBasketList();  
  
        vge.add(fbObj);  
  
        setVegetableBasketList(vge);  
  
        //Fill the code here  
    }  
}
```

```

public static int calculateBill(Stream<VegetableBasket> vegetableBasketStream) {
    Map<Integer, Integer> V = vegetableBasketStream.collect(Collectors.toMap(VegetableBasket->
VegetableBasket.getWeightInKgs(), VegetableBasket -> VegetableBasket.getPricePerKg()));

    int amount=0;

    for(Map.Entry<Integer, Integer>entry : V.entrySet()) {
        amount=amount+(entry.getKey()*entry.getValue());
    }

    return amount;
}

}

```

## 27)Food cart

```

import java.io.File; // Import the File class

import java.io.FileNotFoundException; // Import this class to handle errors

import java.util.Scanner;

public class Main

{

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        int flag=1;

        try {

            File myObj = new File("customer.txt");

            Scanner myReader = new Scanner(myObj);

```

```
String data = myReader.nextLine();

System.out.println("Enter the mobile number");

String Value = sc.next();

while (myReader.hasNextLine())

{

    final String lineFromFile = myReader.nextLine();

    if(lineFromFile.contains(Value))

    {

        // System.out.println("customer.txt");

        System.out.println(lineFromFile);

        flag=0;

        break;

    }

}

if (flag==1)

{

    System.out.println("No customer found");

}

myReader.close();

} catch (FileNotFoundException e) {
```

```
        System.out.println("An error occurred.");
        e.printStackTrace();
    }
}
```

---

Robert:7329102839:Pizza:2:2000

Advaith:8329103940:BBQ:1:350

Thanos:9762453210:Burger:4:570

Lewi:9036524133:Pizza:2:633

Albert:6852416398:BBQ:6:2000

Rahul:6663410275:Burger:4:451

Robert:8763957410:Pizza:2:1026

Jerold:9096741023:BBQ:2:457

Virat:8078451296:Burger:4:600

Dhoni:9363021245:Pizza:2:974

Praga:7685963210:BBQ:4:540

Aravind:6241053689:Burger:4:570

Praveen:8956451203:Pizza:2:3201

Issac:6410258963:BBQ:1:2000

Manish:9512023654:Burger:4:941

Ranjith:7456899874:BBQ:1:720

Akash:6352987451:Burger:4:570

Rakesh:8090461375:Pizza:8:610

Anees:9080708090:BBQ:7:350

Alex:6501200332:Burger:4:340

28)Fuel

```
@FunctionalInterface

public interface FuelCalc {

    public double calculateFuel(String fuelName,double litres);

}

import java.util.*;

public class UserInterface {

    public static FuelCalc generateRate()

    {

        return (String fuelName, double litres)->

        { double price=0;

            fuelName=fuelName.toLowerCase();

            if(fuelName.equals("petrol"))

                price=litres*90.4;

            else if(fuelName.equals("diesel"))

                price=litres*86.8;

            return price;

        };

    }

    public static void main(String [] args)

    {

        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the type of fuel");

        String s=sc.next();

        System.out.println("Enter litres");
```

```

        double l=sc.nextDouble();

        FuelCalc f=generateRate();

        double p=f.calculateFuel(s,l);

System.out.printf("Total Cost of the fuel =Rs. %.2f",p);

    }

}

```

## 29)College account

```

public interface TuitionFee {

    int calculateTuitionFees(String courseType, int basicFee, int noOfSemesters);

}

```

---

```

import java.util.Scanner;

public class UserInterface {

    public static TuitionFee generateFeeReceipt() {

        return (String courseType, int basicFee, int noOfSemesters) -> {

            int regularFee = basicFee * noOfSemesters;

            if (courseType.equalsIgnoreCase("regular")) {

                return regularFee;

            } else if (courseType.equalsIgnoreCase("selfFinance")) {

                return regularFee + 50000;

            } else {

                return 0;
            }
        }
    }
}

```

```
        }

    };

}

public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);

    System.out.println("Enter registration number");

    int registrationNumber = scanner.nextInt();

    System.out.println("Enter student name");

    String studentName = scanner.next();

    System.out.println("Enter no of semesters");

    int noOfSemesters = scanner.nextInt();

    System.out.println("Enter basic fee");

    int basicFee = scanner.nextInt();

    System.out.println("Course type");

    String courseType = scanner.next();

    TuitionFee tuitionFee = generateFeeReceipt();

    int tuitionFeeAmount = tuitionFee.calculateTuitionFees(courseType, basicFee, noOfSemesters);

    System.out.println("Fees Receipt");

    System.out.println("Registration number: "+ registrationNumber);

    System.out.println("Student name: " + studentName);

    System.out.println(String.format("Tuition fee for %s student: %d", courseType, tuitionFeeAmount));

}

}
```

29)Water distributor

```
public class Container {  
    public String distributorName;  
    public int volume;  
    public int count;  
  
    public String getDistributorName() {  
        return distributorName;  
    }  
  
    public void setDistributorName(String distributorName) {  
        this.distributorName = distributorName;  
    }  
  
    public int getVolume() {  
        return volume;  
    }  
  
    public void setVolume(int volume) {  
        this.volume = volume;  
    }  
  
    public int getCount() {  
        return count;  
    }  
  
    public void setCount(int count) {  
        this.count = count;  
    }  
  
    public Container(String distributorName, int volume, int count) {
```

```
super();  
  
this.distributorName = distributorName;  
  
this.volume = volume;  
  
this.count = count;  
  
}  
  
}
```

---

```
public interface DiscountInfo {  
  
    // Fill the code  
  
    public double calculatePayableAmount (Container containerObj);  
  
}
```

---

```
import java.util.Scanner;  
  
public class UserInterface {  
  
    public static DiscountInfo generateBillAmount() {  
  
        return(Container containerobj)->  
  
        { double dp=0;  
  
            if(containerobj.count>=100){  
  
                if(containerobj.volume==10)  
  
                    dp=(20*containerobj.count)-(1*containerobj.count*20);  
  
                else if(containerobj.volume==25)  
  
                    dp=50*containerobj.count-(.15*containerobj.count*50);  
  
            }  
  
            else{  
  
                if(containerobj.volume==10)
```

```

dp=(containerobj.count*20);

else if(containerobj.volume==25)

dp= (containerobj.count*50);

}

return dp;

};

}

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

System.out.println("Enter the name of the distributor");

String s=sc.next();

System.out.println("Enter the volume of the container(in litre)");

int d=sc.nextInt();

System.out.println("Enter the no of containers");

int h=sc.nextInt();




if(d==25 || d==10)

{

Container obj=new Container(s,d,h);

DiscountInfo k=generateBillAmount();

double g=k.calculatePayableAmount (obj);

System.out.println("Generated Bill Amount");

System.out.println("Distributor name: "+s);

System.out.printf("Amount to be paid: Rs.%,.2f",g);

}

```

```
        else
        {
            System.out.println("There is no Discount");
        }

    }

}

31)Book sale

// DO NOT EDIT OR ADD ANY CODE

public class Book {

    private String bookName;

    private String authorName;

    private double bookCost;

    //Setters and Getters

    public String getBookName() {
        return bookName;
    }

    public void setBookName(String bookName) {
        this.bookName = bookName;
    }

    public String getAuthorName() {
        return authorName;
    }

    public void setAuthorName(String authorName) {
```

```
        this.authorName = authorName;  
    }  
  
    public double getBookCost() {  
        return bookCost;  
    }  
  
    public void setBookCost(double bookCost) {  
        this.bookCost = bookCost;  
    }  
  
    @Override  
  
    public String toString() {  
        return "Book [bookName=" + bookName + ", authorName=" + authorName + ",  
bookCost=" + bookCost + "]";  
    }  
  
}
```

---

```
import java.util.List;  
  
import java.util.stream.Stream;  
  
import java.util.*;  
  
import java.util.Map;  
  
import java.util.stream.Collectors;  
  
import java.util.Collections;  
  
public class BookUtility {  
  
    private List<Book> bookList;
```

```

public List<Book> getBookList(){
    return this.bookList;
}

public void setBookList(List<Book> bookList){
    this.bookList = bookList;
}

//FILL THE CODE HERE

public Stream<Book> listToStream() {
    //FILL THE CODE HERE
    return getBookList().stream();
}

public List<Double> calculateBookDiscount(Stream<Book> stream1) {
    List<Double> ans = stream1.map(Book::getBookCost).collect(Collectors.toList());
    Double MUTIPLY_ELEMENT=0.90;
    List<Double> answer = ans.parallelStream().map(in -> in *
        MUTIPLY_ELEMENT).collect(Collectors.toList());
    return answer;
}
}

```

---

```

import java.util.*;
import java.util.stream.Stream;

public class UserInterface {

    public static void main(String [] args)
    {

```

```
Scanner sc = new Scanner(System.in);

System.out.println("Enter the number of books");

int N=sc.nextInt();

if(N>0)

{

List<Book> list = new ArrayList<Book>();

System.out.println("Enter book details");

sc.nextLine();

String arr[]=new String[N];

for(int i=0;i<N;i++)

{

arr[i]=sc.nextLine();

}

for(int i=0;i<N;i++)

{

Book b=new Book();

String[] word=arr[i].split ":";

b.setAuthorName(word[1]);

b.setBookName(word[0]);

double v=Double.parseDouble(word[2]);

b.setBookCost(v);

list.add(b);

}

BookUtility B=new BookUtility();

B.setBookList(list);
```

```
Stream<Book> stream=B.listToStream();

List<Double> answer=B.calculateBookDiscount(stream);

System.out.println("Updated Book Cost:");

for(int i=0;i<N;i++)

{

System.out.println(answer.get(i));

}

}

else{

System.out.println("Invalid Entry");

}

}

}

}

32)Association contest
```

//DO NOT EDIT OR ADD ANY CODE

```
public class Contestant {

    private String id;

    private String name;

    private String address;
```

//setters & getters

```
public String getId() {

    return id;
```

```
}

public void setId(String id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}

}
```

---

```
import java.util.List;
import java.util.Map;
import java.util.*;
```

```
import java.util.stream.Stream;
import java.util.stream.Collectors;

public class ContestUtility {

    private List<Contestant> ContestList;

    public List<Contestant> getContestList(){
        return this.ContestList;
    }

    public void setContestList(List<Contestant> ContestList)
    {
        this.ContestList=ContestList;
    }

    public Stream<Contestant> changeToStream() {
        List<Contestant> list = getContestList();

        return list.stream();
    }

    public List searchFromDetails(Stream<Contestant> stream1,char varString) {
        Map<String,String>file = stream1.collect(Collectors.toMap(Contestant->Contestant.getId(),Contestant->Contestant.getName()));

        List<String> l = new ArrayList<String>();
        for (Map.Entry<String,String> entry : file.entrySet())
        {
            String s=Character.toString(varString);
            if(entry.getValue().toLowerCase().startsWith(s.toLowerCase()))

```

```
{  
    l.add(entry.getKey());  
}  
  
}  
  
return l;  
  
}  


---



```
import java.util.*;  
import java.util.stream.Stream;  
public class UserInterface {
```


```

```
public static void main(String[] args)
{
    Scanner sc = new Scanner(System.in);

    System.out.println("Enter the number of contestant details to enter:");

    int N=sc.nextInt();

    if(N>0)

    {

        System.out.println("Enter the contestant details:");

        List<Contestant> ContestList = new ArrayList<Contestant>();

        ContestUtility C = new ContestUtility();

        for(int i=0;i<N;i++)

        {

            String arr=sc.next();

            String[] word=arr.split("/");

            String id=word[0];

            String name = word[1];

            String place=word[2];

            Contestant c=new Contestant();

            c.setId(id);

            c.setName(name);

            c.setAddress(place);

            ContestList.add(c);

        }

        C.setContestList(ContestList);
    }
}
```

```
Stream<Contestant>stream = C.changeToStream();

System.out.println("Enter character to search");

char alpha = sc.next().charAt(0);

List list=C.searchFromDetails(stream,alpha);

List<Contestant> Contest = C.getContestList();

if(list.size()!=0)

{

    System.out.println("Filter the names which starts with "+alpha+":");

    for(int i=0;i<N;i++)

    {

        if(list.contains(Contest.get(i).getId()))

        {

            System.out.println("Id: "+Contest.get(i).getId()+" , Name:

"+Contest.get(i).getName());

        }

    }

}

else{

    System.out.println("No record found");

}

}

else{

    System.out.println("Invalid");
}
```

```
    }

}

}

33) vehicle registration

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class UserInterface {

    public static void main(String [] args){

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the number of vehicle details:");

        int a = Integer.parseInt(sc.nextLine());

        if (a<=0){

            System.out.println("Invalid");

        }

    }

    else{

        System.out.println("Enter the vehicle details:");

        String [] x = new String[a];

        List<Vehicle> clm=new ArrayList<Vehicle>();

        for (int j=0;j<a;j++){

            x[j] =sc.nextLine();

            String[] p = x[j].split("/");

            Vehicle cd = new Vehicle();

            cd.setVtype(p[0]);

            cd.setVno(p[1]);

            cd.setCname(p[2]);

            cd.setCnic(p[3]);

            cd.setCadd(p[4]);

            cd.setCtel(p[5]);

            clm.add(cd);

        }

        System.out.println("Total "+clm.size()+" vehicles registered");

        System.out.println("Enter the vehicle number to search");

        String vno = sc.nextLine();

        Vehicle v = null;

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

            if (x[i].contains(vno)) v = clm.get(i);

        }

        if (v==null) System.out.println("Vehicle not found");

        else System.out.println("Vehicle found");

    }

}
```

```
        cd.setVehicleNo(Integer.parseInt(p[0]));
        cd.setVehicleHolderName(p[1]);
        cd.setVehicleHolderAddress(p[2]);
        clm.add(cd);

    }

VehicleUtility obj = new VehicleUtility();
obj.setVehicleList(clm);
System.out.println("Enter length of the number to search");
int uh = Integer.parseInt(sc.nextLine());
List<Vehicle> ans = obj.filterVehicleDetails(obj.convertToStream(),uh);
if (ans.isEmpty()){
    System.out.println("No vehicle found");

}

else{
    System.out.println("Filter the vehicle numbers which has a length of "+uh+":");
    for (int g=0;g<ans.size();g++){
        System.out.println("VehicleNumber: "+ans.get(g).getVehicleNo()+", Name:
"+ans.get(g).getVehicleHolderName());
    }
}
}
```

```
}
```

```
}
```

---

```
// DO NOT EDIT OR ADD ANY CODE
```

```
public class Vehicle {
```

```
    private int vehicleNo;
```

```
    private String vehicleHolderName;
```

```
    private String vehicleHolderAddress;
```

```
//Setters and Getterss
```

```
    public int getVehicleNo() {
```

```
        return vehicleNo;
```

```
}
```

```
    public void setVehicleNo(int vehicleNo) {
```

```
        this.vehicleNo = vehicleNo;
```

```
}
```

```
    public String getVehicleHolderName() {
```

```
        return vehicleHolderName;
```

```
}
```

```
    public void setVehicleHolderName(String vehicleHolderName) {
```

```
        this.vehicleHolderName = vehicleHolderName;
```

```
}
```

```
    public String getVehicleHolderAddress() {
```

```
        return vehicleHolderAddress;  
    }  
  
    public void setVehicleHolderAddress(String vehicleHolderAddress) {  
        this.vehicleHolderAddress = vehicleHolderAddress;  
    }  
  
    @Override  
    public String toString() {  
        return "Vehicle [vehicleNo=" + vehicleNo + ", vehicleHolderName=" +  
vehicleHolderName  
                + ", vehicleHolderAddress=" + vehicleHolderAddress + "]";  
    }  
  
}  
  
-----  
  
import java.util.Comparator;  
  
import java.util.List;  
  
import java.util.stream.Collectors;  
  
import java.util.stream.Stream;public class VehicleUtility {  
  
    private List<Vehicle> vehicleList;  
  
    //FILL THE CODE HERE  
  
    public List<Vehicle> getVehicleList(){  
  
        return vehicleList;  
    }  
  
    public void setVehicleList(List<Vehicle> vehicleList){
```

```
this.vehicleList=vehicleList;

}

public Stream<Vehicle> convertToStream() {

    Stream<Vehicle> w =vehicleList.stream();

    return w;

}

public List filterVehicleDetails(Stream<Vehicle> stream1,int length){

    List<Vehicle> shot=stream1.filter(x->String.valueOf(x.getVehicleNo()).length()>=length).sorted(Comparator.comparingInt(Vehicle::getVehicleNo)).collect(Collectors.toList());

    return shot;

}

}

}

34) Library management
```

```
//DO NOT EDIT OR ADD ANY CODE HERE

public class Book {

    private String bookId;

    private String bookName;

    private String bookType;

    private String bookCategory;

    private String bookEdition;
```

```
public Book() {
    super();
}

public Book(String bookId, String bookName, String bookType, String bookCategory, String
bookEdition) {
    super();
    this.bookId = bookId;
    this.bookName = bookName;
    this.bookType = bookType;
    this.bookCategory = bookCategory;
    this.bookEdition = bookEdition;
}

public String getBookId() {
    return bookId;
}

public void setBookId(String bookId) {
    this.bookId = bookId;
}

public String getBookName() {
    return bookName;
}
```

```
public void setBookName(String bookName) {  
    this.bookName = bookName;  
}  
  
public String getBookType() {  
    return bookType;  
}  
  
public void setBookType(String bookType) {  
    this.bookType = bookType;  
}  
  
public String getBookEdition() {  
    return bookEdition;  
}  
  
public void setBookEdition(String bookEdition) {  
    this.bookEdition = bookEdition;  
}  
  
public String getBookCategory() {  
    return bookCategory;  
}  
  
public void setBookCategory(String bookCategory) {  
    this.bookCategory = bookCategory;  
}  
  
}
```

---

```
import java.util.ArrayList;
import java.util.List;
public class BookUtility implements Runnable{
    private List<Book> bookList = new ArrayList<Book>();
    private String searchbookName;
    private int counter;
    //FILL THE CODE HERE
    public List<Book> getBookList(){
        return bookList;
    }
    public String getSearchbookName(){
        return searchbookName;
    }
    public void setBookList(List<Book> a){
        this.bookList=a;
    }
    public void setSearchbookName(String x){
        this.searchbookName=x;
    }
    public int getCounter(){
```

```
    return counter;

}

public void setCounter(int counter){

    this.counter=counter;

}

public void toValidateBookType(Book obj) throws InvalidBookException{

    //FILL THE CODE

    if (obj.getBookType().compareToIgnoreCase("Engineering")==0){

        bookList.add(obj);

    }

    else{

        throw new InvalidBookException("Book type Invalid");

    }

}

public void run(){

    //FILL THE CODE HERE

    int count=0;

    for(int j=0;j<bookList.size();j++){

        if (bookList.get(j).getBookName().compareToIgnoreCase(searchbookName)==0){

            count=count+1;

        }

    }

}
```

```
    }

}

if (count==0) {

    System.out.println("No Books found");

}

else {

    System.out.println("Count of books in the library with the book name
"+searchbookName.toUpperCase()+" is "+count);

}

}

-----  
public class InvalidBookException extends Exception{

//FILL THE CODE

String s;

public InvalidBookException (String s){

    super(s);

}

}
```

```
}
```

---

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;public class UserInterface {

    public static void main(String [] args){

        Scanner sc=new Scanner(System.in);

        BookUtility ans = new BookUtility();

        // FILL THE CODE HERE

        System.out.println("Enter the number of entries");

        int a = Integer.parseInt(sc.nextLine());

        System.out.println("Enter the bookId/bookName/bookType/bookCategory/bookEdition");

        String [] x = new String[a];

        List<Book> clm=new ArrayList<Book>();

        boolean flag = false;

        for (int j =0;j<a;j++){

            x[j] =sc.nextLine();

            String[] p = x[j].split("/");

            Book cd= new Book();

            cd.setBookId(p[0]);

            cd.setBookName(p[1]);

            cd.setBookType(p[2]);

            cd.setBookCategory(p[3]);

            cd.setBookEdition(p[4]);

            try{
```

```
ans.toValidateBookType(cd);

clm.add(cd);

}

catch(InvalidBookException e){

    flag=true;

    System.out.println(e.getMessage());

}

}

if (flag==false){

    ans.setBookList(clm);

    System.out.println("Print the book details");

    for(int y=0;y<clm.size();y++){

        System.out.println("bookId="+clm.get(y).getBookId()+
bookName="+clm.get(y).getBookName()+" , bookType="+clm.get(y).getBookType());

    }

    System.out.println("Enter book name to search");

    String hh = sc.nextLine();

    ans.setSearchbookName(hh);

    Thread myThread = new Thread(ans);

    myThread.run();

}

}
```

}

}

<https://allhackerranksolutionsbykaira.blogspot.com/search/label/HackerRank>

<https://awesomelivesource.com/project/Java-aid/Hackerrank-Solutions#data-structures>

## MEANDERING ARRAY

```
// Java program to print the array in given order

import java.util.Arrays;

public class GFG {

    // Function which arrange the array.

    static void rearrangeArray(int arr[], int n)

    {

        // Sorting the array elements

        Arrays.sort(arr);

        int[] tempArr = new int[n]; // To store modified array

        // Adding numbers from sorted array to

        // new array accordingly

        int ArrIndex = 0;

        // Traverse from begin and end simultaneously

        for (int i = 0, j = n-1; i <= n / 2 || j > n / 2;

             i++, j--) {

            if(ArrIndex < n)

            {

                tempArr[ArrIndex] = arr[i];

                ArrIndex++;

            }

            if(ArrIndex < n)

            {

                tempArr[ArrIndex] = arr[j];

                ArrIndex++;

            }

        }

    }

}
```

```

}

// Modifying original array

for (int i = 0; i < n; i++)
    arr[i] = tempArr[i];

}

// Driver Code

public static void main(String args[])
{
    int arr[] = { 5, 8, 1, 4, 2, 9, 3, 7, 6 };
    int n = arr.length;
    rearrangeArray(arr, n);
    for (int i = 0; i < n; i++)
        System.out.print(arr[i]+" ");
}
}

```

### **DISTINCT DIGIT NUMBER**

```

// Java implementation of brute
// force solution.

import java.util.LinkedHashSet;

class GFG
{
    // Function to check if the given
    // number has repeated digit or not
    static int repeated_digit(int n)
    {
        LinkedHashSet<Integer> s = new LinkedHashSet<>();
        // Traversing through each digit
        while (n != 0)
        {
            int d = n % 10;

```

```
// if the digit is present
// more than once in the
// number
if (s.contains(d))
{
    // return 0 if the number
    // has repeated digit
    return 0;
}
s.add(d);
n = n / 10;
}

// return 1 if the number has
// no repeated digit
return 1;
}

// Function to find total number
// in the given range which has
// no repeated digit
static int calculate(int L, int R)
{
    int answer = 0;
    // Traversing through the range
    for (int i = L; i < R + 1; ++i)
    {
        // Add 1 to the answer if i has
        // no repeated digit else 0
        answer = answer + repeated_digit(i);
    }
    return answer;
}
```

```

// Driver Code

public static void main(String[] args)
{
    int L = 80, R = 120;

    // Calling the calculate

    System.out.println(calculate(L, R));
}
}

```

### TWIN STRING PROBLEM

```

import java.io.IOException;
import java.util.Scanner;

public class TwinStringProblem {
    //creating method for checking two string
    static boolean[] twins(String[] a, String[] b) {
        boolean[] result = new boolean[a.length];

        for (int i = 0; i < a.length; i++) {
            String aVal = a[i].toLowerCase();
            String bVal = b[i].toLowerCase();
            String[] aValArray = aVal.split("");
            String[] bValArray = bVal.split("");

            for (String s : aValArray) {
                for (int index = 0; index < aValArray.length; index++) {
                    if (bValArray[index].equals(s)) { // checking whether the
index match or not
                        if ((s.indexOf(s) % 2 == 0 && index % 2 == 0) || // ch
ecking even values
                            (s.indexOf(s) % 2 != 0 && index % 2 != 0)) {
                            result[i] = false;
                        } else if ((s.indexOf(s) % 2 == 0 && index % 2 != 0)
                            || (s.indexOf(s) % 2 != 0 && index % 2 == 0))
{
                            result[i] = true;
                            break;
                        }
                    }
                }
            }
        }
        return result;
    }
}

```

```
}

//main Method
public static void main(String[] args) throws IOException {
    Scanner in = new Scanner(System.in);

    int n = Integer.parseInt(in.nextLine().trim());
    String[] a = new String[n];
    for (int i = 0; i < n; i++) {
        a[i] = in.nextLine();
    }

    int m = Integer.parseInt(in.nextLine().trim());
    String[] b = new String[m];
    for (int i = 0; i < m; i++) {
        b[i] = in.nextLine();
    }

    // call twins function
    boolean[] results = twins(a, b);

    for (int i = 0; i < results.length; i++) {
        System.out.println(results[i] ? "Yes" : "No");
    }
}
```

## ARRAY GAME

```
20     * The function accepts INTEGER_ARRAY numbers as parameter.  
21     */  
22  
23     public static long countMoves(List<Integer> numbers) {  
24         // Write your code here  
25         long arrsum,smallest,arr_size=numbers.size();  
26         arrsum=0;  
27         smallest=(long) numbers.get(0);  
28         for(int i=0;i<arr_size;i++){  
29             long temp=(long) numbers.get(i);  
30             if(temp<smallest){  
31                 smallest=temp;  
32             }  
33             arrsum+=temp;  
34         }  
35         long minoperation = arrsum-arr_size*smallest;  
36         return minoperation;  
37     }  
38  
39 }  
40 > public class Solution { ...
```

## PRODUCT SORT

```
10  
11     class FreqComparator implements Comparator<Integer> {  
12         private final Map<Integer, Integer> freq;  
13         FreqComparator(Map<Integer, Integer> tempFreqMap)  
14         {  
15             this.freq = tempFreqMap;  
16         }  
17         @Override  
18         public int compare(Integer k1, Integer k2)  
19         {  
20             int freqCompare = freq.get(k1).compareTo(freq.get(k2));  
21             int valueCompare = k1.compareTo(k2);  
22             if (freqCompare == 0)  
23                 return valueCompare;  
24             else  
25                 return freqCompare;  
26         }  
27     }  
28  
29     class Result {  
30         /* Complete the "lenshort" function below.  
31         ... */  
32     }
```

```
56     * The function accepts INTEGER_ARRAY,
57     */
58
59     /* public static List<Integer> itemsSort(List<Integer> items) {
60     // Write your code here
61     }*/
62
63     public static List<Integer> itemsSort(List<Integer> items){
64         HashMap<Integer, Integer> myMap = new HashMap<>();
65         List<Integer> output = new ArrayList<>();
66         Integer temp=0;
67         for(Integer curr:items){
68             //Integer count = map.getOrDefault(curr, temp);
69             // map.put(curr, count + 1);
70             //output.add(curr);
71             Integer res=myMap.get(curr);
72             if(res==null) myMap.put(curr, myMap.get(curr) + 1);
73             else myMap.put(curr,1);
74         }
75         FreqComparator comp=new FreqComparator(myMap);
76         Collections.sort(items,comp);
77         return items;
78     }
79
80     /**
81      * This method is used to sort the array based on frequency of elements.
82      * @param arr
83      * @return
84     */
85     public static int[] sortBasedOnFrequency(int[] arr) {
86         List<Integer> items = new ArrayList<>(Arrays.asList(arr));
87         List<Integer> sortedItems = itemsSort(items);
88         int[] result = new int[sortedItems.size()];
89         for (int i = 0; i < sortedItems.size(); i++) {
90             result[i] = sortedItems.get(i);
91         }
92         return result;
93     }
94 }
```

Test Results      Custom Input

Run Code      Run Tests      Submit

## ANOTHER SOLUTION

```
49m left
1. Prc
public static List<int> itemsSort(List<int> items)
{
    Console.WriteLine(items.Count);
    Dictionary<int,int> myMap=new Dictionary<int,int>();
    List<int> output=new List<int>();
    foreach(int item in items)
    {
        int count=0;
        if(!myMap.ContainsKey(item))
        {
            myMap.Add(item,1);
        }
        else if(myMap.ContainsKey(item))
        {
            int values=myMap[item];
            values=values+1;
            myMap[item]=values;
        }
    }
    var keysorted=from entry in myMap orderby entry.Key,entry.Value descending select entry;
    var sorted=from entry in keysorted orderby entry.Key,entry.Value select entry;
    foreach(var k in sorted)
    {
        The occ
        The occ
    }
}
```

```
tail YouCompleteMe Language: C# Run Tests Run Code
37     }
38     else if(myMap.ContainsKey(item))
39     {
40         int values=myMap[item];
41         values=values+1;
42         myMap[item]=values;
43     }
44     var keysorted=from entry in myMap orderby entry.Key,entry.Value descending select entry;
45     var sorted=from entry in keysorted orderby entry.Value select entry;
46     foreach(var k in sorted)
47     {
48         for(int i=0;i<k.Value;i++)
49         {
50             output.Add(k.Key);
51         }
52     }
53     return output;
54 }
55 }
56 The
57 occ
58 }
59 > class Solution ...
The
occ
```

**VOWELS**

**RESTRUCTURED ARRAY**

(another solution also available)

The screenshot shows a Java code editor with the following code:

```
language Java 8
● Autocomplete Ready ⓘ
```

```
1 > import java.io.*;
2 > class Result {
3
4     /*
5      * Complete the 'getElements' function below.
6      *
7      * The function is expected to return an INTEGER_ARRAY.
8      * The function accepts following parameters:
9      * 1. INTEGER_ARRAY arr
10     * 2. 2D_INTEGER_ARRAY queries
11     */
12
13     public static List<Integer> getElements(List<Integer> arr, List<List<Integer>> queries) {
14         // Write your code here
15         int s;
16         List<Integer> row = new ArrayList<Integer>();
17         for(int i=0;i<queries.size();i++) {
18             s=((queries.get(i).get(0)-1)*arr.get(0))+queries.get(i).get(1);
19             row.add(arr.get(s));
20         }
21         return (row);
22     }
23 }
24
25 > public class Solution { ... }
```

## IS IT POSSIBLE

```
static LinkedList<Pair<Integer, Integer>> pairs = new LinkedList<Pair<Integer, Integer>>();
```

```
public static String isItPossible(Integer a, Integer b, Integer c, Integer d){
    pairs.addLast(new Pair<Integer, Integer>(a,b));
    while (!pairs.isEmpty()){
        Pair<Integer, Integer> pair = pairs.poll();
        Integer key = pair.getKey();
        Integer value = pair.getValue();
        if(key.equals(c) &&
```

```
    value.equals(d)){  
        return "YES";  
    }  
  
    int sum=key+value;  
  
    if (sum<=c){  
        pairs.addLast(new Pair<Integer, Integer>(sum,value));  
    }  
  
    if (sum<=d){  
        pairs.addLast(new Pair<Integer, Integer>(key,sum));  
    }  
  
}  
  
return "NO";  
}
```

## VOWELS

Language Java 8    Autocomplete Ready

```
1 > import java.io.*;
16
17     // Complete the hasVowels function below.
18     static List<Integer> hasVowels(List<String> strArr, List<String> query) {
19         List<Character> vowels = new ArrayList<>();
20         List<Integer> result = new ArrayList<>();
21         vowels.add('a');
22         vowels.add('e');
23         vowels.add('i');
24         vowels.add('o');
25         vowels.add('u');
26         int count,l,r;
27         Character firstChar, lastChar;
28         int[] vowelArray = new int[strArr.size()];
29         for(int i=0; i < strArr.size(); i++){
30             String str = strArr.get(i);
31             firstChar = str.charAt(0);
32             lastChar = str.charAt(str.length()-1);
33             if(vowels.contains(firstChar) && vowels.contains(lastChar)){
34                 vowelArray[i] = 1;
35             }
36             else
37                 vowelArray[i] = 0;
38         }
39         for(String qry: query) {
40             count = 0;
41             String[] lrStringArray = qry.split("-");
42             l = Integer.parseInt(lrStringArray[0]) -1;
43             r = Integer.parseInt(lrStringArray[1]) - 1;
44             // System.out.println(l+" "+ r)
45             for(int i =l;i<= r;i++){
46                 count += vowelArray[i];
47             }
48             result.add(count);
49         }
50     }
51     return result;
52 }
53 }
54
55 > public static void main(String[] args) throws IOException { ... }
```

Test Results    Custom Input

## ANOTHER SOLUTION

```
Java 8 A code editor ready to use
```

```
1 > import java.io.*;
16
17 // Complete the hasVowels function below.
18 static List<Integer> hasVowels(List<String> strArr, List<String> query) {
19 List<Character> vowels=new ArrayList<Character>();
20 List<Integer> result=new ArrayList<Integer>();
21 vowels.add('a');
22 vowels.add('e');
23 vowels.add('i');
24 vowels.add('o');
25 vowels.add('u');
26 int count,l,r;
27 Character firstChar, lastChar;
28 int[] vowelArray=new int[strArr.size()];
29 for(int i=0;i<strArr.size(); i++) {
30 String str=strArr.get(i);
31 firstChar=str.charAt(0);
32 lastChar=str.charAt(str.length()-1);
33 if(vowels.contains(firstChar)&& vowels.contains(lastChar)) {
34 vowelArray[i]=1;
35 }
36 else
37 vowelArray[i]=0;
38 }
39 for(String qry: query) {
40 count=0;
41 for(int i=0,i<strArr.size(), i++) {
42 String str=strArr.get(i);
43 firstChar=str.charAt(0);
44 lastChar=str.charAt(str.length()-1);
45 if(vowels.contains(firstChar)&& vowels.contains(lastChar)) {
46 vowelArray[i]=1;
47 }
48 else
49 vowelArray[i]=0;
50 }
51 for(String qry: query) {
52 count=0;
53 String[] lrStringArray=qry.split("-");
54 l=Integer.parseInt(lrStringArray[0])-1;
55 r=Integer.parseInt(lrStringArray[1])-1;
56 for(int i=l;i<=r;i++) {
57 count +=vowelArray[i];
58 }
59 result.add(count);
60 }
61 return result;
62 }
63 > public static void main(String[] args) throws IOException {
```

## AUTOSCALE POLICY

The screenshot shows a Java code editor with a dark theme. The code is a static method named `finalInstances` that takes two parameters: `int instances` and `List<Integer> averageUtil`. The method iterates through the `averageUtil` list to calculate a new value for `instances` based on specific conditions. The code uses nested if statements and arithmetic operations like division and multiplication.

```
29
30
31     public static int finalInstances(int instances, List<Integer> averageUtil){
32
33         for(int i=0; i<averageUtil.size(); i++){
34
35             if(averageUtil.get(i) < 25){
36
37                 if(instances != 1 && (instances & 1) == 0){
38
39                     instances = instances / 2;
40                     i = i + 10;
41
42                 }else if(instances != 1 && (instances & 1) == 1){
43
44                     instances = (instances / 2) + 1;
45                     i = i + 10;
46
47                 }else if(averageUtil.get(i) > 60){
48
49                     if( instances <= 100000000 ){
50                         instances = 2 * instances;
51                     }
52
53                 }
54             }
55         }
56         return instances;
57     }
58
59
60 }
61
62 }
```

## MINIMUM SWAPS

```
1 > import java.io.*;
2 class Result {
3
4
5
6     /*
7      * Complete the 'minimumSwaps' function below.
8      *
9      * The function is expected to return an INTEGER.
10     * The function accepts INTEGER_ARRAY popularity as parameter.
11     */
12
13
14     public static int minimumSwaps(List<Integer> popularity) {
15         // Write your code here
16         Collections.reverse(popularity);
17         int swapCounter=0;
18         for(int i=0; i<popularity.size(); i++){
19             if(popularity.get(i)==(i+1)){
20                 continue;
21             }
22             Collections.swap(popularity, i, popularity.get(i)-1);
23             swapCounter++;
24             i--;
25         }
26         return swapCounter;
27     }
28
29 }
30
31
32
33
34
35 }
```

```
40 > public class Solution {...
```

Test Results

Custom Input



## ANOTHER SOLUTION

The screenshot shows a code editor window with the following details:

- Language:** C#
- Autocomplete Ready:** Indicated by a circular icon.

```
1 > using System.CodeDom.Compiler;...
16 class Result
17 {
18
19     /*
20      * Complete the 'minimumSwaps' function below.
21      *
22      * The function is expected to return an INTEGER.
23      * The function accepts INTEGER_ARRAY popularity as parameter.
24      */
25
26     public static int minimumSwaps(List<int> popularity)
27     {
28         popularity.Reverse();
29         int swapCounter=0;
30         for(int i=0;i<popularity.Count;i++)
31         {
32             if(popularity[i]==(i+1))
33             {
34                 continue;
35             }
36             int temp = popularity[popularity[i]-1];
37             popularity[popularity[i]-1]=popularity[i];
38             popularity[i]=temp;
39             swapCounter++;
40             i--;
41         }
42     }
43     return swapCounter;
44 }
45 }
46
47 }
48 > class Solution...
```

## ARE ALMOST EQUIVALENT

The screenshot shows a Java code editor with the following code:

```
public static List<String> areAlmostEquivalent(List<String> s, List<String> t) {  
    // Write your code here  
    List<String> result = new ArrayList<String>();  
    for(int j=0; j<s.size();j++){  
        String x=s.get(j);  
        String y=t.get(j);  
        if(x.length() != y.length()){  
            result.add("NO");  
        }  
        else{  
            Map<Character, Integer> mapx= new HashMap<Character, Integer>();  
            Map<Character, Integer> mapy= new HashMap<Character, Integer>();  
            for(int i = 0;i<x.length();i++){  
                if(mapx.containsKey(x.charAt(i))){  
                    mapx.put(x.charAt(i),mapx.get(x.charAt(i))+1);  
                }  
                else{  
                    mapx.put(x.charAt(i), 1);  
                }  
            }  
            for(int i=0;i<y.length();i++){  
                if(mapy.containsKey(y.charAt(i))){  
                    mapy.put(y.charAt(i),mapy.get(y.charAt(i))+1);  
                }  
                else{  
                    mapy.put(y.charAt(i), 1);  
                }  
            }  
            int count=0;  
            for(char p : mapx.keySet()){  
                if(mapy.containsKey(p)){  
                    if(Math.abs(mapx.get(p)-mapy.get(p)) >= 4){  
                        count++;  
                    }  
                }  
            }  
            for(char q : mapy.keySet()){  
                if(mapx.containsKey(q)){  
                    if(Math.abs(mapx.get(q)-mapy.get(q)) >= 4){  
                        count++;  
                    }  
                }  
                else if(mapy.get(q)>=4){  
                    count++;  
                }  
            }  
            result.add(count==0? "YES" : "NO");  
        }  
    }  
    return result;  
}
```

Below the code editor are tabs for "Test Results" and "Custom Input". At the bottom right are buttons for "Run Code", "Run Tests", and "Submit". The status bar at the bottom indicates "Line: 14 Col: 3".

The screenshot shows a Java code editor with the following code:

```
int count=0;  
for(char p : mapx.keySet()){  
    if(mapy.containsKey(p)){  
        if(Math.abs(mapx.get(p)-mapy.get(p)) >= 4){  
            count++;  
        }  
    }  
    else if(mapx.get(p)>=4){  
        count++;  
    }  
}  
for(char q : mapy.keySet()){  
    if(mapx.containsKey(q)){  
        if(Math.abs(mapx.get(q)-mapy.get(q)) >= 4){  
            count++;  
        }  
    }  
    else if(mapy.get(q)>=4){  
        count++;  
    }  
}  
result.add(count==0? "YES" : "NO");  
return result;
```

Below the code editor are tabs for "Test Results" and "Custom Input". At the bottom right are buttons for "Run Code", "Run Tests", and "Submit". The status bar at the bottom indicates "Line: 14 Col: 3".

## ARRANGING COINS

Language Java 8 Autocomplete Ready ⓘ

```
24     public static void arrangeCoins(List<Long> coins) {  
25         // Write your code here  
26         for(Long a: coins){  
27             int res = (int)(Math.sqrt(8*a+1));  
28             Integer res = (int)((temp-1)/2);  
29             System.out.println(res);  
30         }  
31     }  
32 }
```

Test Results Custom Input ⌂ R

## COUNT DUPLICATE ELEMENTS

The screenshot shows a Java code editor interface. At the top, there are tabs for "Activity Details - SumTotal" and "External Site". Below the tabs, the code language is set to "Java 8" and the status bar indicates "Autocomplete Ready". The code itself is a Java function named "countDuplicate" that takes a list of integers as input. It uses a nested loop to compare each element with every other element in the list, adding them to a list if they are found to be equal. Finally, it returns the size of this list, which represents the count of duplicate elements. The code is annotated with Javadoc-style comments at the top.

```
/*
 * The function is expected to return an INTEGER.
 * The function accepts INTEGER_ARRAY numbers as parameter.
 */

public static int countDuplicate(List<Integer> numbers) {
    // Write your code here
    int[] arr=new int[numbers.size()];
    for(int i=0;i<numbers.size();i++){
        arr[i]=numbers.get(i);
    }
    List<Integer> li=new ArrayList<Integer>();
    for(int i=0;i<arr.length;i++){
        for(int j=i+1;j<arr.length;j++){
            if(arr[i]==arr[j]){
                li.add(arr[i]);
            }
        }
    }
    List<Integer> listwithout=new ArrayList<>(new HashSet<>(li));
    return listwithout.size();
}
}

> public class Solution { ... }
```

Test Results    Custom Input    Run Code    Run Test

## ANOTHER SOLUTION

```
public static int countDuplicate(List<int> numbers)
{
    Dictionary<int,int> frequency = new Dictionary<int, int>();
    for(int i=0; i<numbers.Count;i++){
        if(frequency.ContainsKey(numbers[i])){
```

```
if(frequency.ContainsKey(numbers[i])){
    frequency[numbers[i]] = frequency[numbers[i]]+1;
}
else{
    frequency.Add(numbers[i],1);
}
}

List<int> result = new List<int>();
foreach(var keys in frequency.Keys){
    if(frequency[keys]>1){
        result.Add(keys);
    }
}
return result.Count;
}
```

## BINARY NUMBER IN A LINKED LIST

```
75     * }
76     *
77     */
78
79     public static long getNumber(SinglyLinkedListNode binary) {
80         // Write your code here
81         long ans = 0;
82         SinglyLinkedListNode a = binary;
83         while(a!=null){
84             ans = (ans<<1)+(a.data);
85             a = a.next;
86         }
87         return ans;
88
89     }
90
91 }
```

## DISTINCT NUMBERS

```
static void countNumbers(List<List<Integer>> arr)
{
    int A[][]=new int[arr.size()][2];
    int x=0;
    for(List<Integer> item:arr)
    {
        for(int i=0;i<item.size();i++)
        {
            A[x][i]=item.get(i);
        }
        x++;
    }
    for(int i=0;i<arr.size();i++)
    {
        int l=A[i][0];
        int r=A[i][1];
        long count=0;
        for(int j=l;j<=r;j++)
        {
            int num=j;
            boolean visited[]=new boolean[10];
            while(num!=0)
            {
                if(visited[num%10])
                    break;
                visited[num%10]=true;
                num=num/10;
            }
            if(num==0)
                count++;
        }
        System.out.println(count);
    }
}
public static void main(String[] args) throws IOException {
```

```

25
26     public static int numPlayers(int k, List<Integer> scores) {
27         // Write your code here
28         if(k<=0)
29             return 0;
30         Collections.sort(scores, Collections.reverseOrder());
31         int rank=1;
32         int res=0;
33         for(int i=0;i<scores.size();i++){
34             if(i==0){
35                 rank=1;
36             }
37             else if(scores.get(i)!=scores.get(i-1)){
38                 rank=i+1;
39             }
40             if(rank<=k && scores.get(i)>0)
41                 res++;
42             else break;
43         }
44         return res;
45     }
46 }
47 }
48 > public class Solution {

```

## competitive gaming soln 2

3. Competitive Gaming

A group of friends are playing a video game together. During the game, each player earns a number of points. At the end of a round, players who achieve at least a certain rank get to "level up" their characters to gain increased abilities.

Given the scores of the players at the end of a round, how many players will be able to level up?

**Note:** Players with equal scores will have equal ranks, but the player with the next lower score will be ranked based on the position within the list of all players' scores. For example, if there are four players, and three players tie for first place, their ranks are 1, 1, 1, and 4.

**Note:** No player with a score of 0 can level up, regardless of rank.

**Example**

n = 4

```

21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45

```

Language: Java 7    Autocomplete Ready

Test Results    Custom Input    Run Code    Run Tests    2020-03-22 16:53

## Consolidated partitions

The screenshot shows a code editor window with the following details:

- Title Bar:** One Cognizant, Email - Das, Payel L..., iPRIMED LVDS: Log..., Cognizant, Chat | Microsoft Tea..., Attachments - One..., MyPractice, Other bookmarks, Reading list.
- Language:** Java 8
- Code Content:**

```
21  * The function accepts following parameters
22  * 1. INTEGER_ARRAY used
23  * 2. INTEGER_ARRAY totalCapacity
24
25  public static int minPartitions(List<Integer> used, List<Integer> totalCapacity) {
26      // Write your code here
27      int siz = used.size();
28      int sum=0;
29      int count=0;
30      Collections.sort(totalCapacity);
31      Collections.reverse(totalCapacity);
32      int[] u= new int[siz];
33      int[] t=new int[siz];
34      for(int i=0; i<siz; i++){
35          u[i] = used.get(i);
36          t[i] = totalCapacity.get(i);
37          sum=sum+u[i];
38      }
39      for(int j=0; j<siz; j++){
40          sum=sum-t[j];
41          count++;
42          if(sum <= 0){
43              break;
44          }
45      }
46      return count;
47  }
48
49 }
50 > public class Solution { ... }
```
- Bottom Bar:** Run Code, Run Tests, Submit, Line: 14 Col: 1, Custom Input, ENG 16:12.

consolidate partitions soln 2

7m left

1. Consolidating Partitions

ALL

1 There is a computer with a hard drive that is partitioned into several partitions. The data needs to be moved around so that the minimum number of partitions are being used. Given how much space is currently being used on each partition, as well as the total capacity of each, what is the minimum number of partitions needed to hold all the data if it is moved around optimally?

2

3 Example  
used = [3, 2, 1, 3, 1]  
totalCapacity = [3, 5, 3, 5, 5]

The data can be moved around like so:

- Move all the data (3 units of space) from the first partition to the second. This would empty the first partition, and the second partition would then be at full capacity.
- Move the data from the third partition (1 unit) and fifth partition (1 unit) into the

Language Java 7 Autocomplete Ready

```
15  /*Integer sum = used.stream()
16  .mapToInt(Integer::intValue).sum();*/
17  int partitions = used.size();
18  int counterTotal = 0;
19  int countTotal=0;
20  Collections.sort(totalCapacity);
21  Collections.reverse(totalCapacity);
22  Int[] arr1 = new int[partitions];
23  Int[] arr2 = new int[partitions];
24  for(int i=0;i<partitions;i++) {
25      arr1[i] = used.get(i);
26      arr2[i] = totalCapacity.get(i);
27      counterTotal=counterTotal+arr1[i];
28  }
29  for(int j=0;j<partitions;j++) {
30      counterTotal=counterTotal-arr2[j];
31      countTotal++;
32      if(counterTotal<=0) {
33          break;
34      }
35  }
36  return countTotal;
37 }
38 }
```

Test Results Custom Input Run Code Run Tests Submit Line: 16 Col: 57 4522 2021/03/22 16:52 22-Mar-21

Construction management

Hackerrank

rs/4629/tabc2h/questions/59554n38

### Management

In building a new neighborhood, and they to the design. Each house will be built using trials (e.g., wood, brick, or concrete), but no n be made of the same material. Because varying size and complexity, the cost of the se varies. Given the cost of using each h. What is the minimum cost needed to..

there are  $n = 3$  houses to be built. Also,  $\text{cost} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$ , denoting the cost of materials for each of murn cost to build all the houses is 4, as seen

The cheapest material is the first one, which st house, the materials cost the same as with the same material can't be used because the de. The next best option is the second material, the cheapest material for the third house is with costs 1. Therefore, the total cost to build all  $t = 4$ .

minCost in the editor below.

swing-parameter

Dimensions: 3 x 3 x 3

Language: Java 8    Autocomplete Ready

```
1 > import java.io.*;
2
3 class Result {
4
5     /*
6      * Complete the 'minCost' function below.
7      *
8      * The function is expected to return an INTEGER.
9      *
10     * The function accepts 3D_INTEGER_ARRAY cost as parameter.
11     */
12
13
14     public static int minCost(List<List<Integer>> cost) {
15         int len,min_cost;
16         Integer[][] array=new Integer[cost.size()]();
17         Integer[] array1=new Integer[0];
18         for(int i=0;i<cost.size();i++){
19             array[i]=cost.get(i).toArray(array1);
20         }
21         for(int i=1;i<array.length;i++){
22             array[i][0]=Math.min(array[i-1][1],array[i-1][2]);
23             array[i][1]=Math.min(array[i-1][0],array[i-1][2]);
24             array[i][2]=Math.min(array[i-1][0],array[i-1][1]);
25         }
26         len=array.length;
27         min_cost=Math.min(array[len-1][0],Math.min(array[len-1][1],array[len-1][2]));
28         return min_cost;
29     }
30
31     public class Solution {
32
33     }
34 }
```

Test Results: Custom Input

SLOWEST KEY PRESS

**2. Slowest Key Press**

Engineers have redesigned a keypad used by ambulance drivers in urban areas. In order to determine which key takes the longest time to press, the keypad is tested by a driver. Given the results of that test, determine which key takes the longest to press.

**Example**

`keyTimes = [[0, 2], [1, 5], [0, 9], [2, 15]]`

Elements in `keyTimes[i][0]` represent encoded characters in the range `ascii[a-z]` where `a = 0, b = 1, ..., z = 25`. The second element, `keyTimes[i][1]` represents the time the key is pressed since the start of the test. The elements will be given in ascending time order. In the example, keys pressed, in order are `0102`, encoded = `abac` at times `2, 5, 9, 15`. From the start time, it took `2 - 0 = 2` to press the first key, `5 - 2 = 3` to press the second, and so on. The longest time it

```
public static char slowestKey(List<List<Integer>> keyTimes) {
    int arr[][] = new int[keyTimes.size()][2];
    int arr2[][] = new int[keyTimes.size()][2];

    for(int i=0;i<keyTimes.size();i++) {
        List<Integer> list = keyTimes.get(i);
        for(int j=0;j<2;j++) {
            arr[i][j] = list.get(j);
            arr2[i][j] = list.get(j);
        }
    }

    int prev = 0;
    for(int i=0;i<keyTimes.size();i++) {
        arr2[i][1] = arr[i][1]-prev;
        prev = arr[i][1];
    }

    int max = Integer.MIN_VALUE;
    int ans = 0;
    for(int i=0;i<keyTimes.size();i++) {
        int value = arr2[i][1];
        if(value>max) {
            ans = arr2[i][0];
            max=arr2[i][1];
        }
    }

    int result = 97+ans;
    char c= (char)result;
    return c;
}
```

Line: 18 Col: 15  
Test Results Custom Input Run Code Run Tests Submit 2021/03/22 16:53 453 PM 22-Mar-21

**2. Slowest Key Press**

Engineers have redesigned a keypad used by ambulance drivers in urban areas. In order to determine which key takes the longest time to press, the keypad is tested by a driver. Given the results of that test, determine which key takes the longest to press.

**Example**

`keyTimes = [[0, 2], [1, 5], [0, 9], [2, 15]]`

Elements in `keyTimes[i][0]` represent encoded characters in the range `ascii[a-z]` where `a = 0, b = 1, ..., z = 25`. The second element, `keyTimes[i][1]` represents the time the key is pressed since the start of the test. The elements will be given in ascending time order. In the example, keys pressed, in order are `0102`, encoded = `abac` at times `2, 5, 9, 15`. From the start time, it took `2 - 0 = 2` to press the first key, `5 - 2 = 3` to press the second, and so on. The longest time it

```
int prev = 0;
for(int i=0;i<keyTimes.size();i++) {
    arr2[i][1] = arr[i][1]-prev;
    prev = arr[i][1];
}

int max = Integer.MIN_VALUE;
int ans = 0;
for(int i=0;i<keyTimes.size();i++) {
    int value = arr2[i][1];
    if(value>max) {
        ans = arr2[i][0];
        max=arr2[i][1];
    }
}

int result = 97+ans;
char c= (char)result;
return c;
```

Line: 10 Col: 15  
Test Results Custom Input Run Code Run Tests Submit 2021/03/22 16:53 453 PM 22-Mar-21

AREA OF THE BOX

The screenshot shows a Java code editor with the following code:

```
// write your code here
int n, m ;
int min ;
long res=0;
List<Long> fin = new ArrayList<>();

for(List<Integer> lst : queries) {
    n = lst.get(0);
    m = lst.get(1);
    min = Math.min(m,n);
    res = 0 ;

    for(int i=1; i<=min; ++i) {
        res += (n-i+1) * (m-i+1);
    }
    fin.add(res);
}

return fin;
}
```

The code is intended to calculate the sum of products for each query. It iterates over a list of lists (queries), where each inner list contains two integers (n and m). It calculates the minimum of n and m, initializes a result variable (res) to 0, and then loops from 1 to the minimum value, adding the product of (n-i+1) and (m-i+1) to res. Finally, it adds the result to a list (fin).

## CAR INHERITANCE

```
class WagonR extends Car{  
    int mileage;  
    public WagonR(Integer mileage){  
        super(false, "4");  
        this.mileage = mileage;  
    }  
  
    @Override  
    public String getMileage(){  
        String mil=Integer.toString(mileage);  
        return mileage+" kmpl";  
    }  
}
```

```
class HondaCity extends Car{  
    int mileage;  
    public HondaCity(Integer mileage){  
        super(true, "4");  
        this.mileage = mileage;  
    }  
  
    @Override  
    public String getMileage(){  
        String mil=Integer.toString(mileage);  
        return mil+" kmpl";  
    }  
}
```

```
class InnovaCrysta extends Car{  
    int mileage;  
    public InnovaCrysta(Integer mileage){  
  
        super(false, "6");  
        this.mileage = mileage;  
  
    }  
  
    @Override  
    public String getMileage(){  
        String mil=Integer.toString(mileage);  
        return mil+" kmpl";  
    }  
}
```

## PRISON BREAK

The screenshot shows a Java code editor with the following details:

- Language: Java 8
- Autocomplete Ready
- Code content (lines 1-57):

```
1  PUBLIC static long prison(int n, int m, List<Integer> h, List<Integer> v){\n2      // Write your code here\n3      boolean[] xBol=new boolean[n+1];\n4      Arrays.fill(xBol,true);\n5      boolean[] yBol=new boolean[m+1];\n6      Arrays.fill(yBol,true);\n7      for(int x:h){\n8          xBol[x]=false;\n9      }\n10     for(int y:v){\n11         yBol[y]=false;\n12     }\n13     int cx=0,xMax=Integer.MIN_VALUE,cy=0,yMax=Integer.MIN_VALUE;\n14     for(int i=1;i<=n;i++){\n15         if(xBol[i]){\n16             cx=0;\n17         }else{\n18             cx++;\n19             xMax=Math.max(xMax,cx);\n20         }\n21     }\n22     for(int i=1;i<=m;i++){\n23         if(yBol[i]){\n24             cy=0;\n25         }else{\n26             cy++;\n27             yMax=Math.max(yMax,cy);\n28         }\n29     }\n30     return (xMax+1)*(yMax+1);\n31 }
```

- Test Results and Custom Input tabs are visible at the bottom.
- Run button is visible on the right.
- DELL logo is visible at the bottom center of the screen.

Check this also: <https://leetcode.com/discuss/interview-question/1002082/twillio-qa-prison-break>

## COUNT STRING PERMUTATIONS

The screenshot shows a Java code editor with the following code:

```
1 > import java.io.*;...
14
15 class Result {
16
17     /*
18      * Complete the 'countPerms' function below.
19      *
20      * The function is expected to return an INTEGER.
21      * The function accepts INTEGER n as parameter.
22      */
23
24     public static int countPerms(int n) {
25         // Write your code here
26         long[][] per=new long[n+1][5];
27         int MOD=(int)(1e9+7);
28         for(int i=0;i<5;i++){
29             per[1][i]=1;
30         }
31     }
32
33     int[][] relation=new int[][]{
34         {1},{0,2},{0,1,3,4},{2,4},{0}
35     };
36
37     for(int i=1;i<n;i++){
38         for(int u=0;u<5;u++){
39             per[i+1][u]=0;
40             for(int v:relation[u]){
41                 per[i+1][u]+=per[i][v]%MOD;
42             }
43         }
44     }
45     long ans=0;
46     for(int i=0;i<5;i++){
47         ans=(ans+per[n][i])%MOD;
48     }
49     return (int)ans;
50 }
51
52 }
53
54 > public class Solution { ... }
```

At the bottom of the editor, there are two tabs: "Test Results" and "Custom Input".

### **LIST INHERITANCE**

<https://allhackerranksolutionsbykaira.blogspot.com/2020/08/list-inheritance-hackerrank-solution.html>

### **DEVICE NAME SYSTEM**

<https://www.homeworklib.com/question/1486956/in-java-6-device-name-system-suggested-problem>

### **SPORT INHERITANCE**

<https://www.goeduhub.com/2885/program-inherit-cricketplayer-footballplayer-hockeyplayer>

## MERGE 2 ARRAYS

Language: Java 8      Autocomplete Ready ⓘ

```
22     * 1. INTEGER_ARRAY a
23     * 2. INTEGER_ARRAY b
24     */
25
26     public static List<Integer> mergeArrays(List<Integer>
27         // Write your code here
28         List<Integer> out=new ArrayList<Integer>();
29         // List<integer> out1=new ArrayList<Integer>();
30         out.addAll(a);
31         out.addAll(b);
32         out.sort(Comparator.naturalOrder());
33         return out;
34     }
35
36 }
37
38 > public class Solution { ...
```

---

**Test Results**      **Custom Input**

Compiled successfully. All available test cases passed

- ✓ Test case 10 □
- ✓ Test case 11 □  
3 5  
4 7
- ✓ Test case 12 □  
5 7  
6 4
- ✓ Test case 13 □  
7 0  
8 1
- ✓ Test case 14 □  
9 2  
10 3
- ✓ Test case 15 □  
Your Output (stdout)
- ✓ Test case 16 □  
1 0  
2 1

## CONDENSED LIST

```
77     */
78
79     public static SinglyLinkedListNode condense(SinglyLinkedListNode head) {
80         // Write your code here
81         HashSet<Integer> hSet = new HashSet<>();
82         SinglyLinkedListNode ptr1;
83         SinglyLinkedListNode ptr2;
84         ptr1 = head;
85         ptr2 = null;
86         while(ptr1!=null){
87             int val = ptr1.data;
88             if(hSet.contains(val))ptr2.next = ptr1.next;
89             else{
90                 hSet.add(val);
91                 ptr2 = ptr1;
92             }
93             ptr1 = ptr1.next;
94         }
95         return head;
96     }
```

## WAYS TO SUM

The screenshot shows a Java code editor with the following details:

- Title:** Ways to Sum
- Language:** Java 8
- Code:**

```
22     * 1. INTEGER total
23     * 2. INTEGER k
24     */
25
26     public static int ways(int total, int k) {
27         // Write your code here
28         int[][] b=new int[k+1][total+1];
29         for(int i=1;i<=total;i++){
30             b[1][i]=1;
31         }
32         for(int i=1;i<=k;i++){
33             b[i][0]=1;
34         }
35         for(int i=2;i<=k;i++){
36             for(int j=1;j<=total;j++){
37                 if(j==i){
38                     b[i][j]=b[i][j-i]+b[i-1][j];
39                 }else{
40                     b[i][j]=b[i-1][j];
41                 }
42             }
43         }
44         return Math.abs( b[k][total]);
45     }
46
47
48 }
```
- Test Results:** Pending
- Custom Input:** Pending
- Line:** 14 Col: 1

## BALANCED ARRAY

The screenshot shows a Java code editor interface. At the top, there are tabs for 'Problem', 'Editorial', 'Submission...', and 'Doubt Support...'. The main area displays Java code for a 'Solution' class. The code defines a static method 'minValueToBalance' that calculates the minimum value needed to be added to the array to make it balanced. It uses two pointers to sum the left and right halves of the array. The code is annotated with comments explaining the logic. Below the code editor are three buttons: 'Compile & Run', 'Submit', and a 'Save' icon.

```
14 //User function Template for Java
15
16
17 class Solution
18 {
19     long minValueToBalance(long a[], int n)
20     {
21         long sum1=0;
22         long sum2=0;
23         for(int i=0;i<n/2;i++){
24             sum1+=a[i];
25
26         }
27         for(int j=n/2;j<n;j++){
28             sum2+=a[j];
29         }
30     }
31     return Math.abs(sum1-sum2);
32
33
34 }
35
36
37
38 // } Driver Code Ends
```

Example 2:

The screenshot shows a browser window displaying a programming challenge titled '4. Balanced Array'. The challenge description states: 'Given an array of numbers, find the index of the smallest array element (the pivot), for which the sums of all elements to the left and to the right are equal. The array may not be reordered.' Below the description are examples and a function description. The function 'balancedSum' is expected to return an integer and accept an array of integers as input. The constraints are:  $3 \leq n \leq 10^5$ ,  $1 \leq arr[i] \leq 2 \times 10^4$ , where  $0 \leq i < n$ , and it is guaranteed that a solution always exists. The code editor on the right contains the implementation of the 'balancedSum' function in Java, which uses two arrays to calculate prefix and suffix sums and then iterates through the array to find the pivot index where the sums are equal.

```
18
19
20     * Complete the 'balancedSum' function below.
21     *
22     * The function is expected to return an INTEGER.
23     * The function accepts INTEGER_ARRAY arr as parameter.
24     */
25
26     public static int balancedSum(List<Integer> arr)
27     {
28         int n=arr.Count;
29         int[] fsum=new int[n];
30         int[] bsum=new int[n];
31         fsum[0]=arr[0];
32         for(int i=1;i<n;i++)
33         {
34             fsum[i]=fsum[i-1]+arr[i];
35         }
36         bsum[n-1]=arr[n-1];
37         for(int j=n-2;j>=0;j--)
38         {
39             bsum[j]=bsum[j+1]+arr[j];
40         }
41         for(int k=1;k<n-1;k++)
42         {
43             if(fsum[k]==bsum[k])
44                 return k;
45         }
46         return -1;
47     }
48
49
50 }
```

## EMPLOYEE IMPLEMENTATION

<https://github.com/vadym-usatiuk/Hackerrank-Employee-Implementation/blob/master/src/Solution.java>

## **WORK SCHEDULE**

```
1 > import java.io.*;
14 class Result {
15
16     /*
17      * Complete the 'findSchedules' function below.
18      *
19      * The function is expected to return a STRING_ARRAY.
20      * The function accepts following parameters:
21      * 1. INTEGER workHours
22      * 2. INTEGER dayHours
23      * 3. STRING pattern
24      */
25
26     static List<String> timeTable = new ArrayList<>();
27     public static List<String> createTable(String[] listPatterns, int idx, int variation, int
dayHours){
28         if(idx==listPatterns.length){
29             if(variation==0){
30                 timeTable.add(String.join("", listPatterns));
31             }
32             return timeTable;
33         }
34         if(listPatterns[idx].equals(String.valueOf("?"))){
35             for(int i =0;i<=dayHours;i++){
36                 String buffer = listPatterns[idx];
37
38                 public static List<String> construct(int workHours, int dayHours, String pattern){
39
40                     Integer worked_hrs = 0;
41                     int variation = 0;
42                     String[] list_pattern = pattern.split("");
43                     for(String s: list_pattern){
44                         if(!String.valueOf("?").equals(s)){
45                             worked_hrs+=Integer.valueOf(s);
46                         }
47                     }
48                     variation = workHours - worked_hrs;
49                     createTable(list_pattern, 0, variation, dayHours);
50                     return timeTable;
51
52                 }
53                 public static List<String> findSchedules(int workHours, int dayHours, String pattern) {
54                     // Write your code here
55                     return construct(workHours, dayHours, pattern);
56                 }
57             }
58         }
59     }
60
61     public class Solution {
```

Test Results   Custom Input   Run Code   Run Tests   Submit

```
Language | Java 8 | Autocomplete Ready | ⚙️
```

```
many hours  
duling no more  
of hours. On  
rked will be  
the  
he given  
exactly 7 digits  
ch day's  
similar to the  
e of the digits  
(63 decimal).  
that can be  
n marks with  
ed hours is  
ed in a week.  
ust work 24 -  
of the possible
```

```
34     if(listPatterns[idx].equals(String.valueOf("?"))){  
35         for(int i =0;i<=dayHours;i++){  
36             String buffer = listPatterns[idx];  
37             listPatterns[idx] = String.valueOf(i);  
38             createTable(listPatterns, idx+1, variation-i, dayHours);  
39             listPatterns[idx] = buffer;  
40         }  
41     }  
42     else{  
43         createTable(listPatterns, idx+1, variation, dayHours);  
44     }  
45     return timeTable;  
46 }  
47 public static List<String> construct(int workHours, int dayHours, String pattern){  
48     Integer worked_hrs = 0;  
49     int variation = 0;  
50     String[] list_pattern = pattern.split("");  
51     for(String s: list_pattern){  
52         if(!String.valueOf("?").equals(s)){  
53             worked_hrs+=Integer.valueOf(s);  
54         }  
55     }  
56     variation = workHours - worked_hrs;  
57     createTable(list_pattern, 0, variation, dayHours);  
58 }
```

Line: 65 Col: 12

## COUNT OPTIONS

```
*/  
  
public static long countOptions(int n, int k) {  
    // Write your code here  
    if(n<k) return 0;  
    int[][] dp=new int[n+1][k+1];  
    for(int i=0;i<=n;i++)  
        dp[i][1]=1;  
    dp[0][0]=1;  
    for(int i=1;i<=n;i++)  
        for(int j=2;j<=k;j++){  
            dp[i][j]=dp[i-1][j-1];  
            if(i>=2*j) dp[i][j]+=dp[i-j][j];  
        }  
    return dp[n][k];
```

Count the number of ways to divide N in k groups incrementally

<https://www.geeksforgeeks.org/count-the-number-of-ways-to-divide-n-in-k-groups-incrementally/>

## PERFECT SUBSTRING

kerrank.com/test/25iher3t6te/questions/7o8lell38fr

Language Java 8 Autocomplete Ready

```
27     return false;
28 }
29 }
30 return true;
31 }
32 public static int perfectSubstring(String s, int k) {
33 // Write your code here
34 int res = 0;
35 for(int i = 0; i<s.length();i++) {
36     int[] arr = new int[10];
37     for(int j = i; j<s.length(); j++) {
38         if(j > i + (10*k))
39             break;
40         char ch = s.charAt(j);
41         arr[ch-'0']++;
42         if(check(arr, k)) res++;
43     }
44 }
45 return res;
46 }
47 }
48 }
49 }
50
51 > public class Solution { ...
```

Line: 25

kerrank.com/test/25iher3t6te/questions/7o8lell38fr

Language Java 8 Autocomplete Ready

```
1 > import java.io.*;
14
15 class Result {
16
17     /*
18     * Complete the 'perfectSubstring' function below.
19     *
20     * The function is expected to return an INTEGER.
21     * The function accepts following parameters:
22     * 1. STRING s
23     * 2. INTEGER k
24     */
25     public static boolean check(int[] arr, int k) {
26         for(int val : arr) {
27             if(val != 0 && val != k)
28                 return false;
29         }
30         return true;
31     }
32     public static int perfectSubstring(String s, int k) {
33 // Write your code here
34     int res = 0;
35     for(int i = 0; i<s.length();i++) {
36         int[] arr = new int[10];
37         for(int j = i; j<s.length(); j++) {
38             if(j > i + (10*k))
39                 break;
40             char ch = s.charAt(j);
41             arr[ch-'0']++;
42             if(check(arr, k)) res++;
43         }
44     }
45     return res;
46 }
```

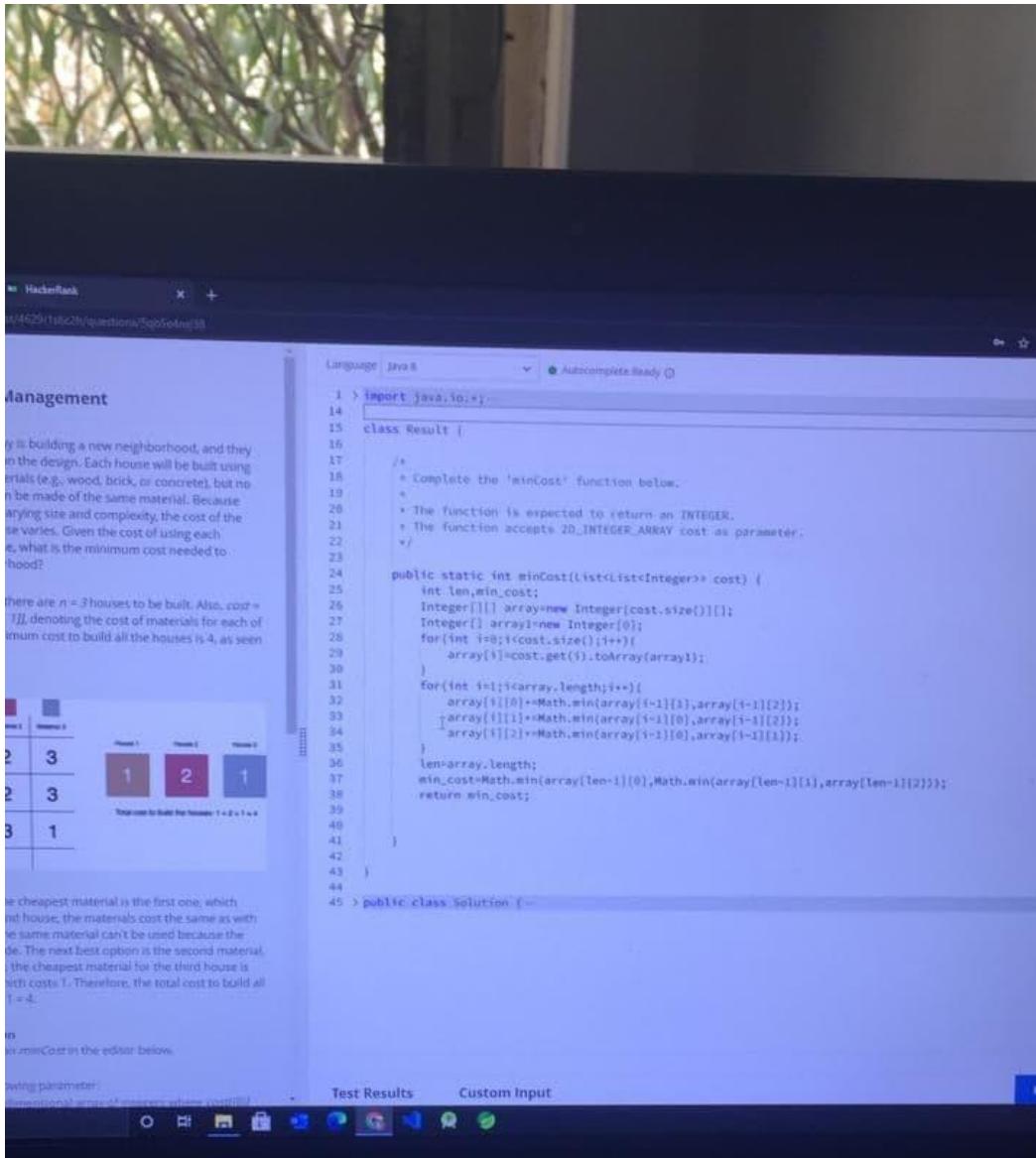
Line: 25

Test Results Custom Input Run Code Run Tests Submit

## LOAD BALANCING

<https://leetcode.com/discuss/interview-question/719447/wish-online-assessment-question>

## CONSTRUCTION MANAGEMENT



## IS POSSIBLE

The screenshot shows a Java code editor interface. At the top, there is a dropdown menu labeled "Language" with "Java 8" selected, and a status bar indicating "Autocomplete Ready". The main area contains Java code:

```
language Java 8
Autocomplete Ready

import java.io.*;
class Result {

    /**
     * Complete the 'isPossible' function below.
     *
     * The function is expected to return a STRING.
     * The function accepts following parameters:
     * 1. INTEGER a
     * 2. INTEGER b
     * 3. INTEGER c
     * 4. INTEGER d
     */
    public static String isPossible(int a, int b, int c, int d) {
        // Write your code here
        return gcd(a,b)==gcd(c,d) ? "Yes": "No";
    }

    public static long gcd(long p,long q){
        return q==0 ? p : gcd(q,p%q);
    }
}

> public class Solution {..}
```

## ANOTHER SOLUTION

```
static LinkedList<Pair<Integer,Integer>> pairs = new LinkedList<Pair<Integer, Integer>>();
```

```
public static String isItPossible(Integer a, Integer b, Integer c, Integer d){

    pairs.addLast(new Pair<Integer, Integer>(a,b));

    while (!pairs.isEmpty()){

        Pair<Integer, Integer> pair = pairs.poll();

        Integer key = pair.getKey();

        Integer value = pair.getValue();
```

```
if(key.equals(c) &&
   value.equals(d)){
    return "YES";
}

int sum=key+value;
if (sum<=c){
    pairs.addLast(new Pair<Integer, Integer>(sum,value));
}
if (sum<=d){
    pairs.addLast(new Pair<Integer, Integer>(key,sum));
}

}

return "NO";
}
```

## GROUP DIVISION

The screenshot shows a programming challenge titled "2. Group Division" on the HackerRank platform. The challenge involves grouping students based on their skill levels. The code editor contains a C# solution for the "groupDivision" function. The constraints and example input are also visible.

**Challenge Details:**

**ALL**

**①**

**2. Group Division**

A university has admitted a group of  $n$  students with varying skill levels. To better accommodate the students, the university has decided to create classes tailored to the skill levels. A placement examination will return a skill level that will be used to group the students, where  $levels[i]$  represents the skill level of student  $i$ . All students within a group must have a skill level within  $maxSpread$ , a specified range of one another. Determine the minimum number of classes that must be formed.

**Example**

$n = 5$   
 $levels = [1, 4, 7, 3, 4]$   
 $maxSpread = 2$

**4.** The students in any group must be within  $maxSpread - 2$  levels of each other. In this case, one optimal grouping is  $(1, 3), (4, 4)$ , and  $(7)$ . Another possible grouping is  $(1, 3, 4), (7)$ . There is no way to form fewer than 3 groups.

**Function Description**

Complete the function `groupDivision` in the editor below.

`groupDivision` has the following parameter(s):

- `int levels[n]`: the skill level for each student
- `int maxSpread`: the maximum allowed skill level difference between any two members of a group

**Returns**

- `int`: the minimum number of groups that can be formed

**Constraints**

<https://www.hackerrank.com/test/fesebq3k19o/questions/sfr5842b28>

**Test Results**    **Custom Input**

```
1 > using System.CodeDom.Compiler; ...
16 Class Result
17 {
18
19     /*
20      * Complete the 'groupDivision' function below.
21      *
22      * The function is expected to return an INTEGER.
23      * The function accepts following parameters:
24      * 1. INTEGER_ARRAY levels
25      * 2. INTEGER maxSpread
26
27
28     public static int groupDivision(List<int> levels, int maxSpread)
29     {
30         levels.Sort();
31         int result=1;
32         int previousOne=levels[0];
33         for(int i=1;i<levels.Count;i++)
34         {
35             if(previousOne+maxSpread<levels[i])
36             {
37                 previousOne=levels[i];
38                 result++;
39             }
40         }
41         return result;
42     }
43
44 }
45 > Close Solution...
```

## SORT AN ARRAY

**1. Sort an Array**

Given an array of integers, any array element can be moved to the end in one move. Determine the minimum number of moves required to sort the array, ascending.

**Example:**  
 $arr = \{5, 1, 3, 2\}$

- Move the value  $arr[2] = 3$  to the end to get  $arr = \{5, 1, 2, 3\}$ .
- Move  $arr[0] = 5$  to the end to achieve the sorted array,  $arr = \{1, 2, 3, 5\}$ .
- The minimum number of moves required to sort the array is 2.

**Function Description**  
Complete the function `getMinimumMoves` in the editor below.

`getMinimumMoves` has the following parameter:

- $int arr[n]$ : an array of integers

Returns:

- $int$ : the minimum number of moves needed to sort the array in ascending order

**Constraints**

- $1 \leq n \leq 10^5$
- $0 \leq arr[i] \leq 10^9$
- array elements are distinct

```

28     List<int> sortedArray=new List<int>();
29     int temp=0;
30     int check=0;
31     int output=0;
32     foreach(int a in arr)
33     {
34         sortedArray.Add(a);
35     }
36     sortedArray.Sort();
37     if(arr.SequenceEqual(sortedArray))
38     {
39         return 0;
40     }
41     for(int i=0;i<sortedArray.Count;i++)
42     {
43         temp=sortedArray[i];
44         while(check<sortedArray.Count && arr[check]!=temp)
45         {
46             check=check+1;
47             output=output+1;
48         }
49         check=check+1;
50         if(check==sortedArray.Count)
51         {
52             break;
53         }
54     }
55     }
56     return output;
57 }
58 }
59 ]
60 ]
61 > class Solution{
62 > 
```

## EFFICIENT SHIPPING

The screenshot shows a C# code editor interface. At the top, there's a toolbar with tabs for 'Page' and 'C#' (selected), and an 'Autocomplete Ready' status indicator. Below the toolbar is a code editor area containing the following code:

```
* Complete the 'getMaxUnits' function below.  
*  
* The function is expected to return a LONG_INTEGER.  
* The function accepts following parameters:  
* 1. LONG_INTEGER_ARRAY boxes  
* 2. LONG_INTEGER_ARRAY unitsPerBox  
* 3. LONG_INTEGER truckSize  
*/  
  
private static long maxvalue=long.MinValue;  
public static long getMaxUnits(List<long> boxes, List<long> unitsPerBox, long truckSize)  
{  
    long capt=0;  
  
    int boxsize=boxes.Count();  
    int j;  
    long endingmaxvalue=0;  
    for(j=0;j<boxsize;j++)  
    {  
        capt+=boxes[j];  
        if(capt<=truckSize)  
        {  
            endingmaxvalue+=boxes[j]*unitsPerBox[j];  
            if(maxvalue<endingmaxvalue)  
                maxvalue=endingmaxvalue;  
        }  
    }  
    return maxvalue;  
}
```

At the bottom of the code editor, there are tabs for 'Results' and 'Custom Input', and a blue 'Run Code' button.

## THE RESTRUCTURED ARRAY

```
23     * The function is expected to return an INTEGER_ARRAY.
24     * The function accepts following parameters:
25     * 1. INTEGER_ARRAY arr
26     * 2. 2D_INTEGER_ARRAY queries
27 */
28
29     public static List<int> getElements(List<int> arr, List<List<int>> queries)
30     {
31
32         List<int> new_array = new List<int>();
33         int n;
34         for(int i =0;i<queries.Count;i++){
35             n = ((queries[i][0]-1)*arr[0])+queries[i][1];
36             new_array.Add(arr[n]);
37         }
38         return new_array;
39     }
40
41 }
```

### PRODUCT OF MAXIMUM AND MINIMUM IN A DATA SET

```
28
29     public static List<long> maxMin(List<string> operations, List<
30     {
31         List<long> parray=new List<long>();
32         List<long> elements=new List<long>();
33         long max=long.MinValue;
34         long min=long.MaxValue;
35
36         for(int i=0;i<operations.Count;i++)
37         {
38             if(operations[i].Equals("push"))
39             {
40                 elements.Add((long)x[i]);
41                 if(max<x[i])
42                 {
43                     max=x[i];
44                 }
45                 if(min>x[i])
46                 {
47                     min=x[i];
48                 }
49                 parray.Add(min*max);
50             }
51             else
52             {
53                 elements.Remove((long)x[i]);
54                 min=elements.Min();
55                 max=elements.Max();
56                 parray.Add(min*max);
57             }
58         }
59     }
60 }
```

```
for(int i=0;i<operations.Count;i++)
{
    if(operations[i].Equals("push"))
    {
        elements.Add((long)x[i]);
        if(max<x[i])
        {
            max=x[i];
        }
        if(min>x[i])
        {
            min=x[i];
        }
        parray.Add(min*max);
    }
    else
    {
        elements.Remove((long)x[i]);
        min=elements.Min();
        max=elements.Max();
        parray.Add(min*max);
    }
}
return parray;
}
```

65 > class Solution ...

# DollarCityThemePark

**UI FOLDER → USER INTERFACE . JAVA**

### **UTILITY-->ThemeParkBO.Java**

```
package com.utility;

import com.ui.UserInterface;
import java.util.*;
import java.util.List;

public class ThemeParkBO {

    private List<Integer> bookingList = new ArrayList<>();

    public List<Integer> getBookingList() {
        return bookingList;
    }

    public void setBookingList(List<Integer> bookingList) {
        this.bookingList = bookingList;
    }

    // This Method should add the customerCount passed as argument into
the
    // bookingList

    public void addBookingDetails(int customerCount) {

        // Fill the Code here
        bookingList.add(customerCount);

    }

    /*
     * This method should return the average customer booked based on
the
     * customerCount values available in the bookingList.
     */
}

public double findAverageCustomerBooked() {
    double avg;

    // Fill the Code here
    double count = 0;
    double counter = 0;
    for(int i=0;i<bookingList.size();++i){
        count+=bookingList.get(i);
        counter++;
    }

    if(counter==0) return 0;
    avg = count/counter;
    return avg;
}
}
```



**Cognizant Academy**

**Electricity Bill Calculation**

**C#, ADO .Net Knock Out Challenge**

**Version 2.0**

	<b>Prepared By / Last Updated By</b>	<b>Reviewed By</b>	<b>Approved By</b>
<b>Name</b>	Senthil Kumar P		
<b>Role</b>			
<b>Signature</b>			
<b>Date</b>			

## Table of Contents

<b>1.0 Introduction</b>	<b>1</b>
<b>1.1 Purpose of this document</b>	<b>1</b>
<b>1.2 Definitions &amp; Acronyms</b>	<b>1</b>
<b>1.3 Project Overview</b>	<b>1</b>
<b>1.4 Scope</b>	<b>2</b>
<b>1.5 Target Audience</b>	<b>2</b>
<b>1.6 Hardware and Software Requirement</b>	<b>2</b>
1.6.1     Hardware Requirements	2
1.6.2     Software Requirements	2
<b>2.0 Functional Requirements</b>	<b>3</b>
<b>2.1 Functional Requirements</b>	<b>3</b>
<b>2.2 Use case Diagram</b>	<b>5</b>
<b>2.3 System Architecture Diagram</b>	<b>6</b>
<b>3.0 Design Specification</b>	<b>7</b>
<b>3.1 Data Design</b>	<b>7</b>
<b>3.2 Component Details for identified Use Cases</b>	<b>8</b>
3.2.1 Get data and calculate bill amount for each customer	8
3.2.2 Add Electricity Bill Details	12
3.2.3 Retrieve 'N' number of Bill Details	12
<b>3.3 Component Specification</b>	<b>12</b>
<b>Class Name : ElectricityBoard (utility class)</b>	<b>14</b>
3.3.1 AddBill Method	14
3.3.2 CalculateBill Method	14
3.3.3 Generate_N_BillDetails Method	15
3.3.4 GetConnection Method	15
<b>3.4 General Design Constraints</b>	<b>15</b>
<b>4.0 Submission</b>	<b>16</b>
<b>4.1 Code submission instructions</b>	<b>16</b>
<b>5.0 Change Log</b>	<b>16</b>
<b>6.0 Evaluation Areas</b>	<b>16</b>

## 1.0 Introduction

### 1.1 Purpose of this document

The Electricity Board of the State was facing a huge work load on the billing process of their domestic consumers. Validating the consumer number, getting the respective units consumed for the users, Calculating the bill amount based on the units consumed, and more process were done manually which consumes more working time. So, the Electricity Board decides to outsource the billing process of their domestic consumers to Global Tek Software Company. Help the Global Tek to automate the above task.

Electricity Board has the following business activities that must be automated.

1. Get data and calculate the bill for 'n' number of customers.
2. Store the ElectricityBill of each customer to the database.

### 1.2 Definitions & Acronyms

Definition / Acronym	Description
Req	Requirement

### 1.3 Project Overview

This project captures the various concepts, techniques and skills learnt and help to put them into practice using C# with ADO.NET that a software engineer must solve. Admittedly, this would be at a scaled-down level since the purpose is to let the associate experience the various concepts learned in C# as an individual. The individual associate is expected to create a console based application within the specified time.

## 1.4 Scope

The scope of the system is explained through its following modules

1. Get input data related to electricity bill and calculate the bill amount.
2. Store the ElectricityBill of each customer.
3. Retrieve the last 'N' number of bill details from the database.

## 1.5 Target Audience

Learner Level

## 1.6 Hardware and Software Requirement

### 1.6.1 Hardware Requirements

#	Item	Specification/Version

### 1.6.2 Software Requirements

#	Item	Specification/Version
1.	C#	6
2.	ADO.NET	4.5
3	ORACLE SERVER	18C

**Note:** All the required hardware and software will be provided in the Tekstac platform

## 2.0 Functional Requirements

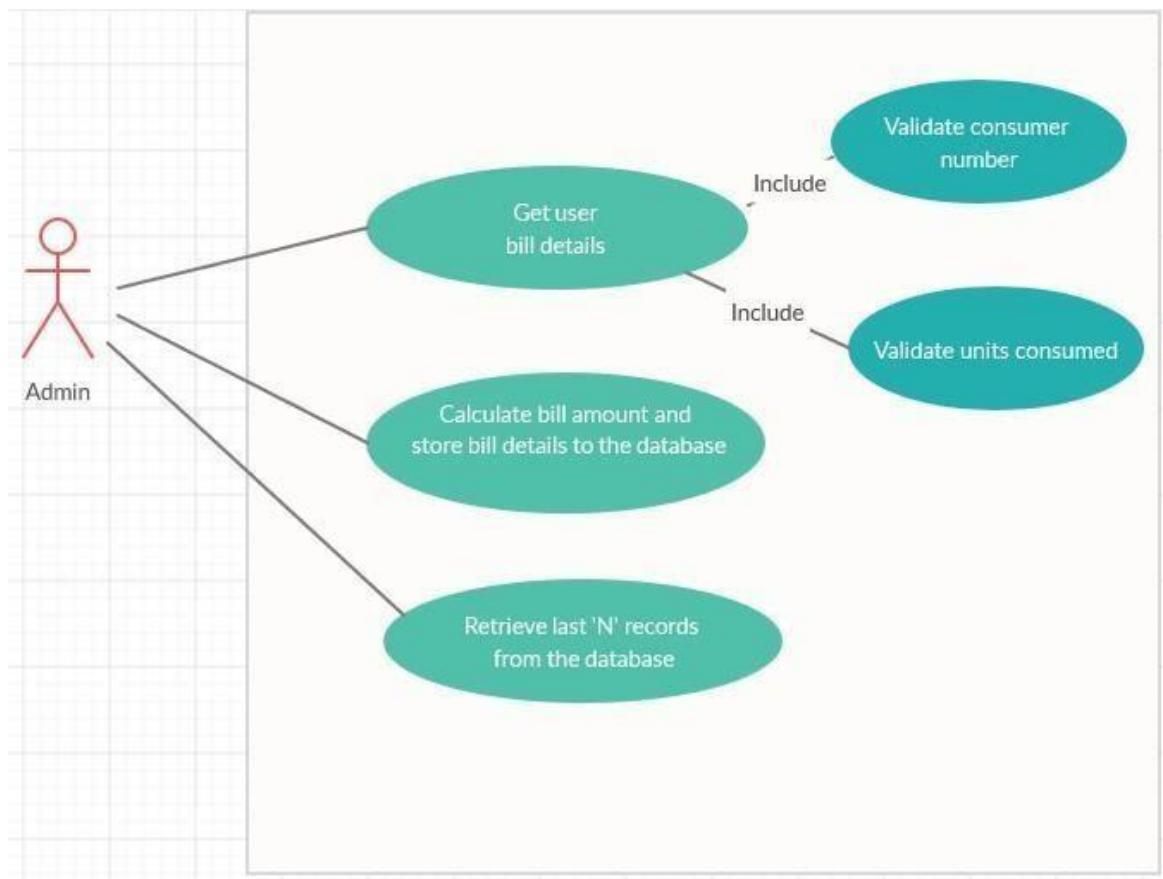
### 2.1 Functional Requirements

<b>Req. #</b>	1
<b>Req. Name</b>	Get the data from the user and calculate the bill amount for the customer
<b>Req. Description</b>	The units consumed by the customer and the other details of the customer are entered through the Main method. Assign the values to ElectricityBill object and calculate the bill amount for each customer based on the units consumed.
<b>Actors/ Users</b>	Admin
<b>Comments</b>	The admin of the electricity board is responsible for entering the data and calculating the bill amount for all the customers

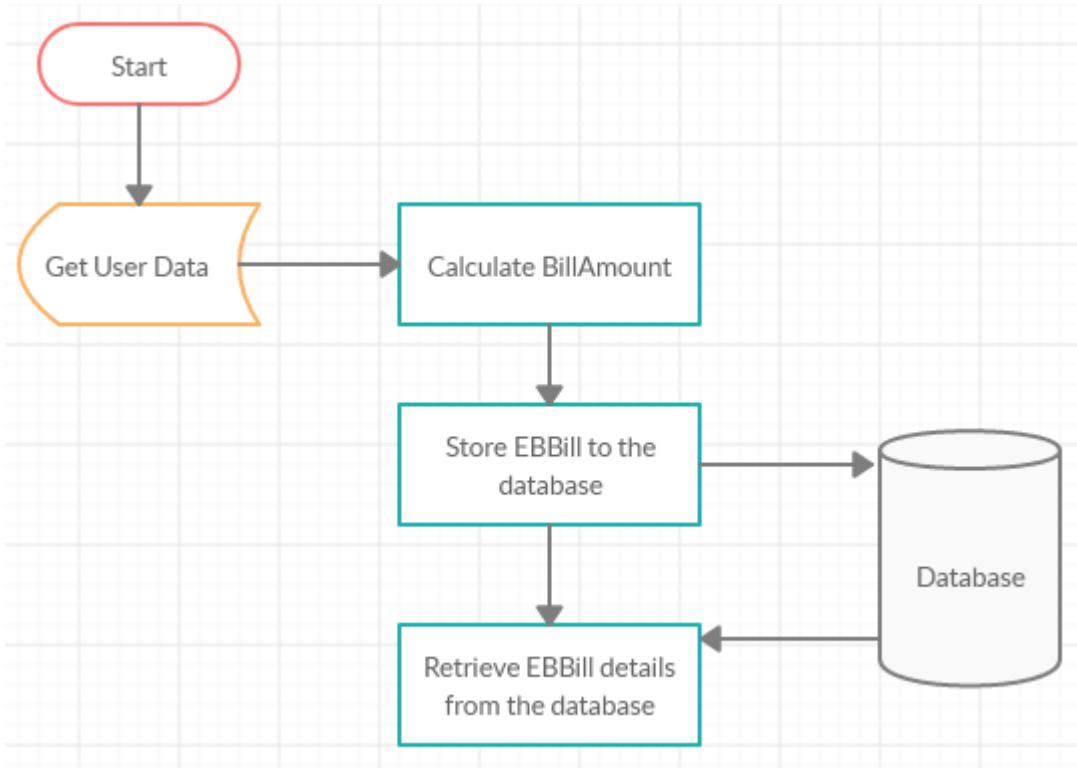
<b>Req. #</b>	2
<b>Req. Name</b>	Store the electricity bill of each customer.
<b>Req. Description</b>	After calculating the bill amount the electricity board will store the electricity bill of each customer into the database.
<b>Actors/ Users</b>	Admin
<b>Comments</b>	The admin is responsible for adding the EB bill details of each customer into the database.

<b>Req. #</b>	3
<b>Req. Name</b>	Retrieve the last 'N' number of bill details from the database.
<b>Req. Description</b>	In the main, get the number of records to be retrieved from the user. Retrieve those last 'N' number of records from the database. Using the main, display the bill details as shown in the sample input/output (refer section 2.4).
<b>Actors/ Users</b>	Admin
<b>Comments</b>	The admin is responsible for retrieving and displaying the EB bill details from the database.

## 2.2 Use case Diagram



### 2.3 System Architecture Diagram



### 2.4 Sample Input/Output

#### SAMPLE INPUT :

```

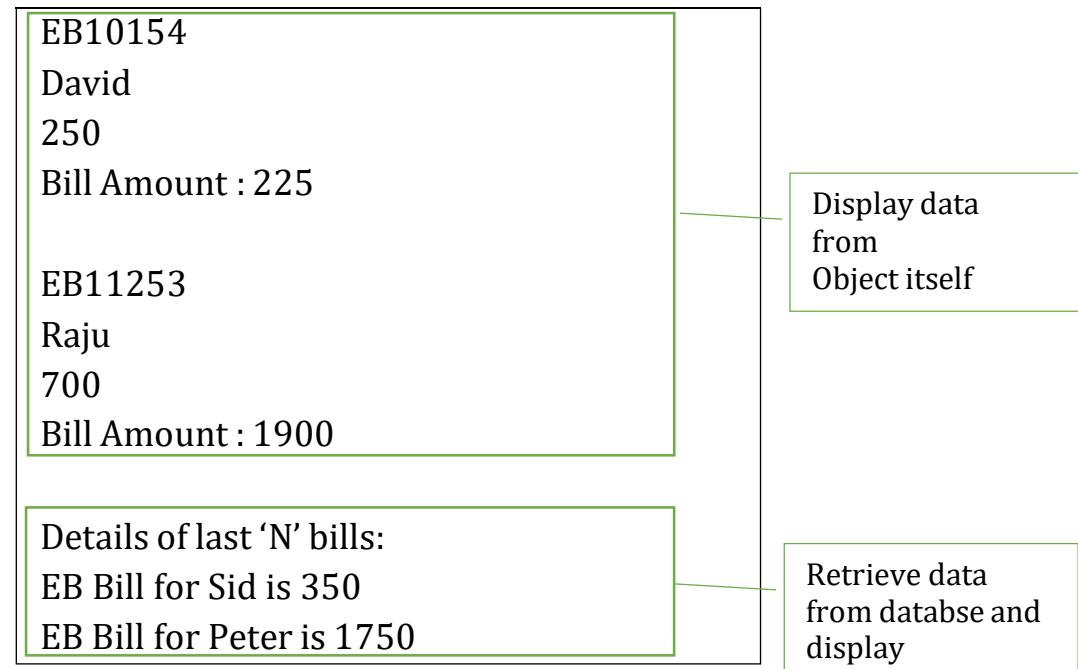
"Enter Number of Bills To Be Added : "
2

"Enter Consumer Number:"
EB10154
"Enter Consumer Name:"
David
"Enter Units Consumed:"
250

"Enter Consumer Number:"
EB11253
"Enter Consumer Name:"
Raju
"Enter Units Consumed:"
  
```

700
Enter Last 'N' Number of Bills To Generate:
2

### SAMPLE OUTPUT :



## 3.0 Design Specification

### 3.1 Data Design

#### Table Structure:

Table name: ElectricityBill	
Column Name	Data type
consumer_number	VARCHAR2(100)
consumer_name	VARCHAR2(100)
units_consumed	NUMBER
bill_amount	FLOAT

**Design Constraints:**

- Use ORACLE SERVER database to store the data. The database name is “ElectricityBillDB”. This is already created for you in Tekstac.
- The table ‘ElectricityBill’ has been created already in Tekstac.
- The table names and the column names should be the same as specified in the table structure. Consumer Number size is given as 20. Consumer Name size is given as 50.
- Connection string is given in **DBConnection.cs**, Should use the '**connStr**' static variable for DB connections, which is also provided as part of code skeleton. THIS IS GIVEN ONLY FOR YOUR REFERENCE. **You need NOT change this.**

**Note:** The code skeleton will be available in the Tekstac platform

### 3.2 Component Details for identified Use Cases

#### 3.2.1 Get data and calculate bill amount for each customer

In the ‘Main’ get the following values,

Number of bills to be added
Consumer Number
Consumer Name
Units Consumed
Enter Last 'N' Number of Bills To Generate

Construct the Electricity Bill object and assign the values. Calculate the bill amount based on the units consumed and display the details as shown in the sample input/output.

Give option to the admin to get multiple user data.

For sample input/output, refer section 2.4.

Calculate the bill amount for each Electricity Bill based on the below condition:

Units Consumed	Rate per unit in rupees
<b>&lt;=100</b>	Free(0)
<b>&gt;100 and &lt;=300</b>	1.5
<b>&gt;300 and &lt;=600</b>	3.5
<b>&gt;600 and &lt;=1000</b>	5.5
<b>&gt;1000</b>	7.5

**For example 1:** If the units Consumed is 650,

- First 100 units are free
  - Next 200 units the charges are Rs. 1.50/unit
  - Next 300 units the charges are Rs. 3.50/unit
  - Remaining 50 units the charges are Rs. 5.5 /unit
  - The total bill amount is =>
- $$100*0 + 200*1.50 + 300*3.50 + 50*5.5 \\ = 1625$$

**For example 2:** If the units Consumed is 1300,

- First 100 units are free
  - Next 200 units the charges are Rs. 1.50/unit
  - Next 300 units the charges are Rs. 3.50/unit
  - Next 400 units the charges are Rs. 5.5 /unit
  - Remaining 300 units the charges are Rs. 7.5 /unit
  - The total bill amount is =>
- $$100*0 + 200*1.50 + 300*3.50 + 400*5.5 + 300*7.5 \\ = 5800$$

After calculating the bill amount store the values in the ElectricityBill objects and store into the database.

Repeat this for the number of users the admin have

opted. As per our sample input, it is 2.

### **Validation 1:**

*The **consumerNumber** should start with the characters "EB" and it should contain 5 numbers (eg: EB11389).*

*If the Consumer Number is valid then assign the data and calculate the bill amount.*

*If Consumer Number is NOT valid then throw a built-in Exception named 'FormatException' with a message "**Invalid Consumer Number**".*

*On printing the exception object "itself" this message must be displayed.*

**Validation 2:**

***The Units Consumed*** should NOT be less than 0.

***If the given units consumed is less than 0 then display a message “Given units is invalid”.***

***In case of invalid units consumed is given, then prompt the user to enter the units again until a valid(greater or equal to 0) is entered.***

**SAMPLE INPUT / OUTPUT:**

"Enter Number of Bills To Be Added : "

1

"Enter Consumer Number:"

EB10323

"Enter Consumer Name:"

Peter

"Enter Units Consumed:"

-200

Given units is invalid

"Enter Units Consumed:"

-200

Given units is invalid

"Enter Units Consumed:"

200

EB10323

Peter

200

Bill Amount : 150

### 3.2.2 Add Electricity Bill Details

Add the bill details along with the calculated bill amount to the database.

### 3.2.3 Retrieve ‘N’ number of Bill Details

In the main, get the **last** ‘n’ number of bill details to be retrieved from database. Retrieve the ‘n’ records. Store each record in Electricity Bill object and add the objects to a ‘List’ . Using main, display the bill details added to the ‘List’.

## 3.3 Component Specification

### Class Name : BillValidator

#### Responsibility:

To validate the given units consumed. Units consumed must not be less than zero.

Type(Class)	Fields	Methods
BillValidator	N/A	public String ValidateUnitsConsumed(int UnitsConsumed)

Method ‘ValidateUnitsConsumed’ must accept units consumed as parameter. Validate to units consumed. If units consumed is less than zero then **return** a message, “**Given units is invalid**”.

### **Class Name : ElectricityBill (model class)**

#### **Responsibility:**

This model object holds the state of the electricity bill at all point-in-time.

Type(Class)	Fields	Properties
ElectricityBill	<b>String</b> consumerNumber <b>String</b> consumerName <b>int</b> unitsConsumed <b>double</b> billAmount	<b>String</b> ConsumerNumber <b>String</b> ConsumerName <b>int</b> UnitsConsumed <b>double</b> BillAmount

**Note:** Keep all the fields ‘private’ and properties ‘public’.

#### **Exception:**

Throw a built-in exception “FormatException”, if the consumer number is invalid based on the constraint mentioned above. *Implement in such a way that it returns “Invalid Consumer Number” message in the exception*

### **Class Name : ElectricityBoard (utility class)**

#### **Responsibility:**

This class has the supporting methods to automate the billing process given in our scope. This class includes the methods to calculate bill amount and add the bill details to database

### 3.3.1 AddBill Method

Type(Class)	Method	Responsibilities
ElectricityBoard	void AddBill(ElectricityBill ebill)	This method should accept an Electricity Bill object and <b>execute a sql query</b> to insert the consumer details into the database

### 3.3.2 CalculateBill Method

Type(Class)	Method	Responsibilities
ElectricityBoard	void CalculateBill (ElectricityBill ebill)	This method should accept an ElectricityBill object, calculate and <b>set the bill amount</b> based on the units consumed.

### 3.3.3 Generate\_N\_BillDetails Method

Type(Class)	Method	Responsibilities
ElectricityBoard	List<ElectricityBill> Generate_N_BillDetails(int num)	<p>This method should accept number of records to be retrieved from database. Store each record in an ElectricityBill object. Add each object to a 'List' and return it.</p> <p><b>NOTE : Retrieve the last 'N' records.</b></p>

### Class Name : DBHandler(DAO class)

#### 3.3.4 GetConnection Method

#### Responsibility:

This method should connect to the database by reading the database details from the **DBConnection.cs** file and it should return the connection object.

Type(Class)	Method	Resources
DBHandler	OracleConnection GetConnection()	DBConnection.cs file contains the database connection details.

### 3.4 General Design Constraints

1. The fields/properties/method/class name should be correctly specified as given in the document.
2. Keep all the classes as '**public**'

3. Do not change the **DBConnection.cs** file.

4. Do not change the namespace name.

## 4.0 Submission

### 4.1 Code submission instructions

1. Do not change the code skeleton given, as your code will be auto evaluated.
2. You can validate your solution against sample test cases during the assessment duration.
3. Your last submitted solution will be considered for detailed evaluation.
4. Make sure to submit the solution before the time limit. After the assessment duration you will not be allowed to submit the solution.

## 5.0 Change Log

<b>Changes Made</b>				
V1.0.0	Initial baseline created on <dd-Mon-yy> by <Name of Author>			
	<b>Section No.</b>	<b>Changed By</b>	<b>Effective Date</b>	<b>Changes Effected</b>
Vx.y.z	<Please refer the configuration control tool / change item status form if the details of changes are maintained separately. If not, the template given below needs to be followed>			

## 6.0 Evaluation Areas

<b>S.No</b>	<b>Description</b>
1.	Declaration of attributes, properties and methods in the class ElectricityBill and ElectricityBoard

2.	Implementation to validate units consumed
3.	Implementation to add bill details to database
4.	Implementation to retrieve bill details from database
5.	Implementation to throw “ParseException” for invalid Consumer Number
6.	Calculation of EB bill units less than 100
7.	Calculation of EB bill units between 100 and 300
8.	Calculation of EB bill units between 300 and 600
9.	Calculation of EB bill units between 600 and 1000
10.	Calculation of EB bill units greater than 1000

# **Group -1**

## **1. AirVoice - Registration**

Grade settings: Maximum grade: 100

Run: Yes Evaluate: Yes

Automatic grade: Yes Maximum execution time: 16 s

SmartBuy is a leading mobile shop in the town. After buying a product, the customer needs to provide a few personal details for the invoice to be generated.

You being their software consultant have been approached to develop software to retrieve the personal details of the customers, which will help them to generate the invoice faster.

Component Specification: Customer

Type(Class)	Attributes	Methods	Responsibilities
Customer	String customerName  long contactNumber  String emailId  int age	Include the getters and setters method for all the attributes.	

In the Main class, create an object for the Customer class.

Get the details as shown in the sample input and assign the value for its attributes using the setters.

Display the details as shown in the sample output using the getters method.

All classes and methods should be public, Attributes should be private.

Note:

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

Ensure to follow the object oriented specifications provided in the question.

Ensure to provide the names for classes, attributes and methods as specified in the question.

Adhere to the code template, if provided.

Sample Input 1:

Enter the Name:

john

Enter the ContactNumber:

9874561230

Enter the EmailId:

john@gmail.com

Enter the Age:

32

Sample Output 1:

Name:john

ContactNumber:9874561230

EmailId:john@gmail.com

Age:32

Automatic evaluation[\[+\]](#)

Customer.java

```
1 public class Customer {  
2     private String customerName;
```

```

3
4     private long contactNumber;
5
6     private String emailId;
7
8     private int age;
9
10    public String getCustomerName() {
11        return customerName;
12    }
13
14    public void setCustomerName(String customerName) {
15        this.customerName = customerName;
16    }
17
18    public long getContactNumber() {
19        return contactNumber;
20    }
21
22    public void setContactNumber(long contactNumber) {
23        this.contactNumber = contactNumber;
24    }
25
26    public String getEmailId() {
27        return emailId;
28    }
29
30    public void setEmailId(String emailId) {
31        this.emailId = emailId;
32    }
33
34    public int getAge() {
35        return age;
36    }
37
38    public void setAge(int age) {
39        this.age = age;
40    }
41
42
43
44}
45

```

## Main.java

```

1 import java.util.Scanner;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         Scanner sc=new Scanner(System.in);
8         Customer c=new Customer();
9         System.out.println("Enter the Name:");
10        String name=(sc.nextLine());
11        System.out.println("Enter the ContactNumber:");
12        long no=sc.nextLong();
13        sc.nextLine();
14        System.out.println("Enter the EmailId:");
15        String mail=sc.nextLine();
16
17        System.out.println("Enter the Age:");

```

```

18     int age=sc.nextInt();
19     c.setCustomerName(name);
20     c.setContactNumber(no);
21     c.setEmailId(mail);
22     c.setAge(age);
23     System.out.println("Name:"+c.getCustomerName());
24     System.out.println("ContactNumber:"+c.getContactNumber());
25     System.out.println("EmailId:"+c.getEmailId());
26     System.out.println("Age:"+c.getAge());
27
28
29
30 }
31
32 }
```

## Grade

Reviewed on Monday, 7 February 2022, 4:45 PM by Automatic grade

**Grade** 100 / 100

**Assessment report**

[\[+\]Grading and Feedback](#)

---

## 2. Payment - Inheritance

**Grade settings:** Maximum grade: 100

**Run:** Yes **Evaluate:** Yes

**Automatic grade:** Yes **Maximum execution time:** 16 s  
**Payment Status**

Roy is a wholesale cloth dealer who sells cloth material to the local tailors on monthly installments. At the end of each month, he collects the installment amount from all his customers. Some of his customers pay by Cheque, some pay by Cash and some by Credit Card. He wants to automate this payment process.

Help him to do this by writing a java program.

### Requirement 1: Make Payment

The application needs to verify the payment process and display the status report of payment by getting the inputs like due amount, payment mode and data specific to the payment mode from the user and calculate the balance amount.

**Component Specification: Payment Class (Parent Class)**

<b>Component Name</b>	<b>Type(Class)</b>	<b>Attributes</b>	<b>Methods</b>	<b>Responsibilities</b>
Make payment for EMI amount	Payment	int dueAmount	Include a public getter and setter method	
Make payment for EMI amount	Payment		public boolean payAmount()	The boolean payAmount() method should return true if there is no due to be paid, else return false.

**Note:**

- The attributes of Payment class should be private.
- The payment can be of three types: Cheque, Cash, Credit Card.

**Component Specification: Cheque class** (Needs to be a child of Payment class)

<b>Component Name</b>	<b>Type(Class)</b>	<b>Attributes</b>	<b>Methods</b>	<b>Responsibilities</b>
	Cheque	String chequeNo int chequeAmount Date dateOfIssue	Include a public getter and setter method for all the attributes.	
Make payment for EMI amount	Cheque		public boolean payAmount()	This is an overridden method of the parent class. It should return true if the cheque is valid and the amount is valid. Else return false.

**Note:**

- The cheque is valid for 6 months from the date of issue.
- Assume the current date is 01-01-2020 in dd-MM-yyyy format.
- The chequeAmount is valid if it is greater than or equal to the dueAmount.

**Component Specification: Cash class** (Needs to be a child of Payment class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Make payment for EMI amount	Cash	int cashAmount	Include a public getter and setter method for the attribute.	
Make payment for EMI amount	Cash		public boolean payAmount()	This is an overridden method of the parent class. It should return true if the cashAmount is greater than or equal to the dueAmount. Else return false.

**Component Specification: Credit class** (Needs to be a child of Payment class)

Component Name	Type (Class)	Attributes	Methods	Responsibilities
Make payment for EMI amount	Credit	int creditCardNo String cardType int creditCardAmount	Include a public getter and setter method for all the attributes.	
Make payment for EMI amount	Credit		public boolean payAmount()	This is an overridden method of the parent class. It should deduct the dueAmount and service tax from the creditCardAmount and return true if the credit card payment was done successfully. Else return false.

**Note:**

- The payment can be done if the credit card amount is greater than or equal to the sum of due amount and service tax. Else payment cannot be made.
- The cardType can be “silver” or “gold” or “platinum”. Set the creditCardAmount based on the cardType.
- Also service tax is calculated on dueAmount based on cardType.

Credit Card Type	Credit Card Amount	Service Tax
silver	10000	2% of the due amount
gold	50000	5% of the due amount
platinum	100000	10% of the due amount

- The boolean payAmount() method should deduct the due amount and the service tax amount from a credit card. If the creditCardAmount is less than the dueAmount+serviceTax, then the payment cannot be made.
- The balance in credit card amount after a successful payment should be updated in the creditCardAmount by deducting the sum of dueAmount and serviceTax from creditCardAmount itself.

#### Component Specification: Bill class

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Payment Status Report	Bill		public String processPayment (Payment obj)	This method should return a message based on the status of the payment made.

**Note:**

- If the payment is successful, processPayment method should return a message “Payment done successfully via cash” or “Payment done successfully via cheque” or “Payment done successfully via creditcard. Remaining amount in your <<cardType>> card is <<balance in CreditCardAmount>>”
- If the payment is a failure, then return a message “Payment not done and your due amount is <<dueAmount>>”

Create a **public class Main** with the main method to test the application.

**Note:**

- Assume the current date as 01-01-2020 in dd-MM-yyyy format.
- In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

- Ensure to follow the object oriented specifications provided in the question.
- Ensure to provide the names for classes, attributes and methods as specified in the question.
- Adhere to the code template, if provided.
- Adhere to the sample input and output.

### **Sample Input 1:**

Enter the due amount:

**3000**

Enter the mode of payment(cheque/cash/credit):

**cash**

Enter the cash amount:

**2000**

### **Sample Output 1:**

Payment not done and your due amount is 3000

### **Sample Input 2:**

Enter the due amount:

**3000**

Enter the mode of payment(cheque/cash/credit):

**cash**

Enter the cash amount:

**3000**

### **Sample Output 2:**

Payment done successfully via cash

### **Sample Input 3:**

Enter the due amount:

**3000**

Enter the mode of payment(cheque/cash/credit):

**cheque**

Enter the cheque number:

**123**

Enter the cheque amount:

**3000**

Enter the date of issue:

**21-08-2019**

### **Sample Output 3:**

Payment done successfully via cheque

### **Sample Input 4:**

Enter the due amount:

**3000**

Enter the mode of payment(cheque/cash/credit):

**credit**

Enter the credit card number:

**234**

Enter the card type(silver,gold,platinum):

**silver**

### **Sample Output 4:**

Payment done successfully via credit card. Remaining amount in your silver card is 6940

## Automatic evaluation[\[+\]](#)

### Main.java

```
1 import java.text.ParseException;
2 import java.text.SimpleDateFormat;
3 import java.util.Date;
4 import java.util.Scanner;
5 public class Main {
6
7     public static void main(String[] args) {
8
9         Scanner sc=new Scanner(System.in);
10        System.out.println("Enter the due amount:");
11        int dueAmount=sc.nextInt();
```

```

12 System.out.println("Enter the mode of payment(cheque/cash/credit):");
13 String mode=sc.next();
14 Bill b = new Bill();
15 if(mode.equals("cheque"))
16 {
17     System.out.println("enter the cheque number:");
18     String chequeNumber=sc.next();
19     System.out.println("enter the cheque amount:");
20     int chequeAmount=sc.nextInt();
21     System.out.println("enter the date of issue:");
22     String date=sc.next();
23     SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");
24     Date dateOfIssue=null;
25     try
26     {
27         dateOfIssue = dateFormat.parse(date);
28     }
29     catch (ParseException e)
30     {
31     }
32     }
33     Cheque cheque= new Cheque();
34     cheque.setChequeNo(chequeNumber);
35     cheque.setChequeAmount(chequeAmount);
36     cheque.setDateOfIssue(dateOfIssue);
37     cheque.setDueAmount(dueAmount);
38     System.out.println(b.processPayment(cheque));
39 }
40 }
41 else if(mode.equals("cash"))
42 {
43     System.out.println("enter the cash amount:");
44     int CashAmount=sc.nextInt();
45     Cash cash=new Cash();
46     cash.setCashAmount(CashAmount);
47     cash.setDueAmount(dueAmount);
48     System.out.println(b.processPayment(cash));
49 }
50 else if(mode.equals("credit"))
51 {
52     System.out.println("enter the credit card number:");
53     int creditCardNumber=sc.nextInt();
54     System.out.println("enter the card type:");
55     String cardType=sc.next();
56     Credit credit=new Credit();
57     credit.setCreditCardNo(creditCardNumber);
58     credit.setCardType(cardType);
59     credit.setDueAmount(dueAmount);
60     System.out.println(b.processPayment(credit));
61 }
62 }
63 }
64 }
```

## Payment.java

```

1 public class Payment {
2     private int dueAmount;
3
4     public boolean payAmount()
5     {
6         if(dueAmount == 0)
7             return true;
8         else
9             return false;
10    }
11 }
```

```

12     public int getDueAmount() {
13         return dueAmount;
14     }
15
16     public void setDueAmount(int dueAmount) {
17         this.dueAmount = dueAmount;
18     }
19 }
```

## Cheque.java

```

1 import java.text.ParseException;
2 import java.text.SimpleDateFormat;
3 import java.util.Date;
4 public class Cheque extends Payment {
5     String chequeNo;
6     int chequeAmount;
7     Date dateOfIssue;
8     public String getChequeNo() {
9         return chequeNo;
10    }
11    public void setChequeNo(String chequeNo) {
12        this.chequeNo = chequeNo;
13    }
14    public int getChequeAmount() {
15        return chequeAmount;
16    }
17    public void setChequeAmount(int chequeAmount) {
18        this.chequeAmount = chequeAmount;
19    }
20    public Date getDateOfIssue() {
21        return dateOfIssue;
22    }
23    public void setDateOfIssue(Date dateOfIssue) {
24        this.dateOfIssue = dateOfIssue;
25    }
26
27    @Override
28    public boolean payAmount()
29    {
30        SimpleDateFormat format = new SimpleDateFormat("dd-MM-yyyy");
31        Date today = new Date();
32        try
33        {
34            today = format.parse("01-01-2020");
35        }
36        catch (ParseException e)
37        {
38            return false;
39        }
40        long diff = today.getTime()-dateOfIssue.getTime();
41        int day = (int) Math.abs(diff/(1000*60*60*24));
42        int month = day/30;
43        if(month <=6)
44        {
45            if(chequeAmount>=getDueAmount())
46            {
47                return true;
48            }
49        }
50        else
51            return false;
52    }
53    else
54        return false;
55    }
56 }
```

```
57  
58  
59 }
```

## Cash.java

```
1 public class Cash extends Payment {  
2     int cashAmount;  
3  
4     public int getCashAmount() {  
5         return cashAmount;  
6     }  
7  
8     public void setCashAmount(int cashAmount) {  
9         this.cashAmount = cashAmount;  
10    }  
11  
12    @Override  
13    public boolean payAmount()  
14    {  
15        if(cashAmount>=getDueAmount())  
16            return true;  
17        else  
18            return false;  
19    }  
20  
21  
22 }
```

## Credit.java

```
1 public class Credit extends Payment {  
2     int creditCardNo;  
3     String cardType;  
4     int creditCardAmount;  
5     public int getCreditCardNo() {  
6         return creditCardNo;  
7     }  
8     public void setCreditCardNo(int creditCardNo) {  
9         this.creditCardNo = creditCardNo;  
10    }  
11    public String getCardType() {  
12        return cardType;  
13    }  
14    public void setCardType(String cardType) {  
15        this.cardType = cardType;  
16    }  
17    public int getCreditCardAmount() {  
18        return creditCardAmount;  
19    }  
20    public void setCreditCardAmount(int creditCardAmount) {  
21        this.creditCardAmount = creditCardAmount;  
22    }  
23  
24  
25    @Override  
26    public boolean payAmount()  
27    {  
28        int netAmount = 0;  
29        if(cardType.equals("silver"))  
30        {  
31            netAmount = (int) (getDueAmount()*1.02);  
32            creditCardAmount = 10000;  
33        }  
34        else if(cardType.equals("gold"))  
35        {  
36            netAmount = (int) (getDueAmount()*1.05);  
37        }  
38    }  
39}
```

```

37             creditCardAmount = 50000;
38         }
39     else if(cardType.equals("platinum"))
40     {
41         netAmount = (int) (int) (getDueAmount())*1.1);
42         creditCardAmount = 100000;
43     }
44
45     if(creditCardAmount>=netAmount)
46     {
47         creditCardAmount = creditCardAmount - netAmount;
48         return true;
49     }
50     else
51         return false;
52     }
53
54
55 }
```

## Bill.java

```

1 public class Bill {
2     public String processPayment(Payment obj)
3     {
4         String res="";
5         if(obj instanceof Cheque)
6         {
7             if(obj.payAmount())
8                 res = "Payment done successfully via cheque";
9             else
10                res = "Payment not done and your due amount is
"+obj.getDueAmount();
11        }
12        else if(obj instanceof Cash)
13        {
14            if(obj.payAmount())
15                res = "Payment done successfully via cash";
16            else
17                res = "Payment not done and your due amount is
"+obj.getDueAmount();
18        }
19        else if(obj instanceof Credit)
20        {
21            Credit c = (Credit) obj;
22            if(obj.payAmount())
23                res = "Payment done successfully via credit card. Remaining
amount in your "+c.getCardType()+" card is "+c.getCreditCardAmount();
24            else
25                res = "Payment not done and your due amount is
"+obj.getDueAmount();
26        }
27        return res;
28    }
29 }
```

## Grade

Reviewed on Wednesday, 1 December 2021, 10:08 PM by Automatic grade

**Grade** 100 / 100

**Assessment report**

TEST CASE PASSED

[\[+\]Grading and Feedback](#)

---

## 3.Power Progress

**Grade settings:** Maximum grade: 100

**Run:** Yes **Evaluate:** Yes

**Automatic grade:** Yes

Andrews taught exponential multiplication to his daughter and gave her two inputs.

Assume, the first input as M and the second input as N. He asked her to find the sequential power of M until N times. For Instance, consider M as 3 and N as 5. Therefore, 5 times the power is incremented gradually from 1 to 5 such that,  $3^1=3$ ,  $3^2=9$ ,  $3^3=27$ ,  $3^4=81$ ,  $3^5=243$ . The input numbers should be greater than zero Else print "<Input> is an invalid". The first Input must be less than the second Input, Else print "<first input> is not less than <second input>".

Write a Java program to implement this process programmatically and display the output in sequential order. (  $3^3$  means  $3*3*3$  ).

**Note:**

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

Adhere to the code template, if provided.

Kindly do not use System.exit() in the code.

**Sample Input 1:**

3

5

**Sample Output 1:**

3 9 27 81 243

**Explanation:** Assume the first input as 3 and second input as 5. The output is to be displayed are based on the sequential power incrementation. i.e.,  $3(3)$   $9(3*3)$   $27(3*3*3)$   $81(3*3*3*3)$   $243(3*3*3*3*3)$

**Sample Input 2:**

-3

**Sample Output 2:**

-3 is an invalid

**Sample Input 3:**

3

0

**Sample Output 3:**

0 is an invalid

**Sample Input 4:**

4

2

**Sample Output 4:**

4 is not less than 2

## Automatic evaluation[\[+\]](#)

### Main.java

```
1 import java.util.*;
2 public class Main
3 {
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         //Fill the code
8         int m=sc.nextInt();
9         if(m<=0){
10             System.out.println(""+m+" is an invalid");
11             return;
12         }
13         int n=sc.nextInt();
14         if(n<=0){
15             System.out.println(""+n+" is an invalid");
16             return;
17         }
18         if(m>=n){
19             System.out.println(""+m+" is not less than "+n);
20             return;
21         }
22         for(int i=1;i<=n;i++){
23             System.out.print((int)Math.pow(m,i)+");
24         }
25     }
26 }
```

## Grade

Reviewed on Monday, 7 February 2022, 4:46 PM by Automatic grade

**Grade** 100 / 100

**Assessment report**

TEST CASE PASSED

[\[+\]Grading and Feedback](#)

---

## 4. ZeeZee bank

**Grade settings:** Maximum grade: 100

**Run:** Yes **Evaluate:** Yes

**Automatic grade:** Yes **Maximum execution time:** 16 s

ZeeZee is a leading private sector bank. In the last Annual meeting, they decided to give their customer a 24/7 banking facility. As an initiative, the bank outlined to develop a stand-alone device that would offer deposit and withdrawal of money to the customers anytime.

You being their software consultant have been approached to develop software to implement the functionality of deposit and withdrawal anytime.

#### **Component Specification: Account**

Type(Class)	Attributes	Methods	Responsibilities
<b>Account</b>	long accountNumber  double balanceAmount	Include the getters and setters method for all the attributes.  Include a parametrized constructor of two arguments in the order – accountNumber,balanceAmount to initialize the values for the account object	

#### **Requirement 1: Being able to deposit money into an account anytime**

As per this requirement, the customer should be able to deposit money into his account at any time and the deposited amount should reflect in his account balance.

#### **Component Specification: Account**

Component Name	Type(Class)	Methods	Responsibilities
Deposit amount to an account	Account	public void deposit(double depositAmt)	This method takes the amount to be deposited as an argument  This method should perform the deposit, by adding the deposited amount to the balanceAmount

#### **Requirement 2: Being able to withdraw money from the account anytime**

As per this requirement, the customer should be able to withdraw money from his account anytime he wants. The amount to be withdrawn should be less than or equal to the balance in the account. After the withdrawal, the account should reflect the balance amount

### Component Specification: Account

Component Name	Type(Class)	Methods	Responsibilities
Withdraw amount from an account	Account	public boolean withdraw(double withdrawAmt)	This method should take the amount to be withdrawn as an argument.  This method should check the balanceAmount and deduct the withdraw amount from the balanceAmount and return true. If there is insufficient balance then return false.

In the **Main** class, Get the details as shown in the sample input.

Create an object for the Account class and invoke the deposit method to deposit the amount and withdraw method to withdraw the amount from the account.

All classes and methods should be public, Attributes should be private.

#### Note:

Balance amount should be displayed corrected to 2 decimal places.

Order of the transactions to be performed (Display,Deposit,Withdraw).

If the balance amount is insufficient then display the message as shown in the Sample Input / Output.

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

Ensure to follow the object-oriented specifications provided in the question.

Ensure to provide the names for classes, attributes, and methods as specified in the question.

Adhere to the code template, if provided.

#### Sample Input/Output 1:

Enter the account number:

**1234567890**

Enter the available amount in the account:

**15000**

Enter the amount to be deposited:

**1500**

Available balance is:16500.00

Enter the amount to be withdrawn:

**500**

Available balance is:16000.00

### **Sample Input/Output 2:**

Enter the account number:

**1234567890**

Enter the available amount in the account:

**15000**

Enter the amount to be deposited:

**1500**

Available balance is:16500.00

Enter the amount to be withdrawn:

**18500**

Insufficient balance

Available balance is:16500.00

### **Automatic evaluation**[\[+\]](#)

Main.java

```
1 import java.text.DecimalFormat;
```

```

2 import java.util.Scanner;
3 import java.util.Scanner;
4
5
6 public class Main{
7     static Account ac=new Account(0, 0);
8     public static void main (String[] args) {
9         Scanner sc=new Scanner(System.in);
10        System.out.println("Enter the account number:");
11        ac.setAccountNumber(sc.nextLong());
12        System.out.println("Enter the available amount in the account:");
13        ac.setBalanceAmount(sc.nextDouble());
14        System.out.println("Enter the amount to be deposited:");
15        ac.deposit(sc.nextDouble());
16        System.out.printf("Available balance is:%.2f",ac.getBalanceAmount());
17        System.out.println();
18        System.out.println("Enter the amount to be withdrawn:");
19        ac.withdraw(sc.nextDouble());
20        System.out.printf("Available balance is:%.2f",ac.getBalanceAmount());
21        //Fill the code
22    }
23}
24
25
26

```

## Account.java

```

1
2 public class Account {
3     long accountNumber;
4     double balanceAmount;
5
6
7     public Account(long accno, double bal){
8         super();
9         this.accountNumber=accno;
10        this.balanceAmount=bal;
11    }
12    public long getAccountNumber(){
13        return accountNumber;
14    }
15    public void setAccountNumber(long accno){
16        this.accountNumber=accno;
17    }
18    public double getBalanceAmount(){
19        return balanceAmount;
20    }
21    public void setBalanceAmount(double bal) {
22        this.balanceAmount=bal;
23    }
24    public void deposit(double depositAmt){
25        float total=(float)(balanceAmount+depositAmt);
26        balanceAmount=total;
27    }
28    public boolean withdraw(double withdrawAmt){
29        float total;
30        if(withdrawAmt>balanceAmount){
31            System.out.println("Insufficient balance");
32
33        return false;
34    }else{
35        total=(float)(balanceAmount-withdrawAmt);

```

```
36     setBalanceAmount(total);
37     return true;
38   }
39 }
40 }
```

## Grade

Reviewed on Monday, 7 February 2022, 4:47 PM by Automatic grade

**Grade** 100 / 100

**Assessment report**

[\[+\]Grading and Feedback](#)

---

## 5. Reverse a word

**Grade settings:** Maximum grade: 100

**Run:** Yes **Evaluate:** Yes

**Automatic grade:** Yes

**Reverse a word**

Rita and Brigitha want to play a game. That game is to check the first letter of each word in a given sentence (Case Insensitive). If it is equal, then reverse the last word and concatenate the first word. Else reverse the first word and concatenate the last word. Create a Java application and help them to play the game

**Note:**

- Sentence must contain at least 3 words else print "Invalid Sentence" and terminate the program
- Each word must contain alphabet only else print "Invalid Word" and terminate the program
- Check the first letter of each word in a given sentence (Case Insensitive). If it is equal, then reverse the last word and concatenate the first word and print. Else reverse the first word and concatenate the last word and print.
- Print the output without any space.

Please do not use System.exit(0) to terminate the program

**Sample Input 1:**

Sea sells seashells

**Sample Output 1:**

sllehsaesSea

**Sample Input 2:**

Sam is away from Australia for a couple of days

**Sample Output 2:**

maSdays

**Sample Input 3:**

Welcome home

**Sample Output 3:**

Invalid Sentence

**Sample Input 4:**

Friendly fire fighting fr@gs.

**Sample Output 4:**

Invalid Word

---

**Automatic evaluation**[\[+\]](#)**Main.java**

```
1 import java.util.Scanner;
2 import java.lang.String.*;
3 import java.util.*;
4 public class Main{
5     public static void main(String[] args){
6         String[] words;
7         Scanner read =new Scanner(System.in);
8         String sentence=read.nextLine();
9         words=sentence.split(" ");
10        if(words.length<3)
11            System.out.println("Invalid Sentence");
12        else{
13            String a=words[0].substring(0,1);
14            String b=words[1].substring(0,1);
15            String c=words[2].substring(0,1);
```

```

16     if(a.equalsIgnoreCase(b)&&b.equalsIgnoreCase(c))
17     {
18         StringBuilder k= new StringBuilder();
19         k.append(words[words.length-1]);
20         k=k.reverse();
21         k.append(words[0]);
22         System.out.println(k);
23     }
24     else{
25         StringBuilder k = new StringBuilder();
26         k.append(words[0]);
27         k=k.reverse();
28         k.append(words[words.length-1]);
29         System.out.println(k);
30     }
31 }
32 }
33 }
```

## Grade

Reviewed on Monday, 7 February 2022, 5:12 PM by Automatic grade

**Grade** 90 / 100

**Assessment report**

---

*Fail 1 -- test5\_CheckForTheSentenceContainsOtherThanAlphabets::  
\$Expected output: "[Invalid Word]" Actual output: "[tahWme]"\$*  
[\[+\]Grading and Feedback](#)

## 6. Dominion cinemas

**Grade settings:** Maximum grade: 100

**Run:** Yes **Evaluate:** Yes

**Automatic grade:** Yes

Dominion cinema is a famous theatre in the city. It has different types of seat tiers – Platinum, Gold and Silver. So far the management was manually calculating the ticket cost for all their customers which proved very hectic and time consuming. Going forward they want to calculate ticket cost using their main computer. Assist them in calculating and retrieving the amount to be paid by the Customer.

### Requirements 1: Calculation of Ticket Cost

The application needs to calculate the ticket cost to be paid by the Customer according to the seat tier.

**Component Specification:** BookAMovieTicket Class (Parent Class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Calculation of Ticket cost	BookAMovieTicket	String ticketId String customerName long mobileNumber String emailId String movieName	Public getter and setter method for all the attributes and 5 argument constructor in the given order - ticketId, customerName, mobileNumber, emailId, movieName are provided as a part of the code skeleton.	

**Note:**

- The attributes of the BookAMovieTicket class should be protected.

**Component Specification: GoldTicket class** (Needs to be a child of BookAMovieTicket class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Calculation of Ticket cost	GoldTicket		Include a public 5 argument constructor in the given order - ticketId, customerName, mobileNumber, emailId, movieName.	
Validate Ticket Id	GoldTicket		public boolean validateTicketId ()	This method should validate the Ticket Id, Ticket Id should contain a string “GOLD” followed by 3 digits. If the ticket id is valid this method should return true else it should return false.
Calculation of Ticket cost	GoldTicket		public double calculateTicketCost (int	This method should calculate the ticket cost according to the seat tier and return the same.

			numberoftickets, String ACFacility)	
--	--	--	--	--

**Component Specification: PlatinumTicket class** (Needs to be a child of the BookAMovieTicket class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Calculation of Ticket cost	PlatinumTicket		Include a public 5 argument constructor in the given order - ticketId, customerName, mobileNumber, emailId, movieName.	
Validate Ticket Id	PlatinumTicket		public boolean validateTicketId()	This method should validate the Ticket Id, Ticket Id should contain a string “PLATINUM” followed by 3 digits. If the ticket id is valid this method should return true else it should return false.
Calculation of Ticket cost	PlatinumTicket		calculateTicketCost(int numberoftickets, String ACFacility)	This method should calculate the ticket cost according to the seat tier and return the same.

**Component Specification: SilverTicket class** (Needs to be a child of the BookAMovieTicket class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Calculation of Ticket cost	SilverTicket		Include a public 5 argument constructor in the given order - ticketId, customerName,	

			mobileNumber, emailId, movieName.	
Validate Ticket Id	SilverTicket		public boolean validateTicketId()	This method should validate the Ticket Id, Ticket Id should contain a string “SILVER” followed by 3 digits. If the ticket id is valid this method should return true else it should return false.
Calculation of Ticket cost	SilverTicket		calculateTicketCost(int numberOfTickets, String ACFfacility)	This method should calculate the ticket cost according to the seat tier and return the same.

**Note:**

- The classes GoldTicket, PlatinumTicket and SilverTicket should be concrete classes.

Ticket cost according to the seat tier without AC facilities.

Seat Tier	Silver	Gold	Platinum
Without AC Facility	100	350	600
With AC Facility	250	500	750

Amount is calculated based on the seat tier,

Amount = ticketCost \* numberOfTickets

Use a **public class UserInterface** with the main method to test the application. In the main method call the validateTicketId() method, if the method returns true display the amount else display "Provide valid Ticket Id".

**Note:**

- **Display the amount to be paid to 2 decimal places.**
- **Use the System.out.printf method.**
- In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.
- Ensure to follow the object oriented specifications provided in the question.

- Ensure to provide the names for classes, attributes and methods as specified in the question.
- Adhere to the code template, if provided.

**Sample Input 1:**

Enter Ticket Id

**SILVER490**

Enter Customer name

**Venkat**

Enter Mobile number

**9012894578**

Enter Email Id

**venkat@gmail.com**

Enter Movie name

**Avengers**

Enter number of tickets

**8**

Do you want AC or not

**yes** // Case insensitive

Ticket cost is 2000.00

**Sample Input 2:**

Enter Ticket Id

**ACN450**

Enter Customer name

**Kamal**

Enter Mobile number

**9078561093**

Enter Email Id

**kamal@gmail.com**

Enter Movie name

**Tangled**

Enter number of tickets

**9**

Provide valid Ticket Id

## Automatic evaluation[\[+\]](#)

### BookAMovieTicket.java

```
1  public class BookAMovieTicket {  
2      protected String ticketId;  
3      protected String customerName;  
4      protected long mobileNumber;  
5      protected String emailld;  
6      protected String movieName;  
7  
8      public String getTicketId() {  
9          return ticketId;  
10     }  
11     public void setTicketId(String ticketId) {  
12         this.ticketId = ticketId;  
13     }  
14     public String getCustomerName() {  
15         return customerName;  
16     }  
17     public void setCustomerName(String customerName) {  
18         this.customerName = customerName;  
19     }  
20     public long getMobileNumber() {  
21         return mobileNumber;  
22     }  
23     public void setMobileNumber(long mobileNumber) {  
24         this.mobileNumber = mobileNumber;  
25     }  
26     public String getEmailId() {  
27         return emailld;  
28     }  
29     public void setEmailId(String emailld) {  
30         this.emailld = emailld;  
31     }  
32     public String getMovieName() {  
33         return movieName;  
34     }  
35     public void setMovieName(String movieName) {  
36         this.movieName = movieName;  
37     }  
38 }
```

```

41     public BookAMovieTicket(String ticketId, String customerName, long mobileNumber, String emailId,
42     String movieName) {
43         this.ticketId = ticketId;
44         this.customerName = customerName;
45         this.mobileNumber = mobileNumber;
46         this.emailId = emailId;
47         this.movieName = movieName;
48     }
49
50
51
52 }
53

```

## GoldTicket.java

```

1
2 public class GoldTicket extends BookAMovieTicket{
3     public GoldTicket(String ticketId, String customerName, long mobileNumber,
4     String emailId, String movieName){
5         super(ticketId, customerName, mobileNumber, emailId, movieName);
6     }
7
8     public boolean validateTicketId(){
9         int count=0;
10        if(ticketId.contains("GOLD")){
11            count++;
12            char[] cha=ticketId.toCharArray();
13            for(int i=4;i<7;i++){
14                if(cha[i]>='1'&& cha[i]<='9')
15                    count++;
16            }
17            if(count==4)
18                return true;
19            else
20                return false;
21        }
22
23
24 // Include Constructor
25
26     public double calculateTicketCost(int numberOfTickets, String ACFacility){
27         double amount;
28         if(ACFacility.equals("yes")){
29             amount=500*numberOfTickets;
30         }
31         else{
32             amount=350*numberOfTickets;
33         }
34
35         return amount;
36     }
37
38 }

```

## PlatinumTicket.java

```

1 public class PlatinumTicket extends BookAMovieTicket{
2     public PlatinumTicket(String ticketId, String customerName, long mobileNumber,
3     String emailId, String movieName){
4         super(ticketId, customerName, mobileNumber, emailId, movieName);

```

```

5 }
6
7     public boolean validateTicketId(){
8         int count=0;
9         if(ticketId.contains("PLATINUM")){
10            count++;
11            char[] cha=ticketId.toCharArray();
12            for(int i=8;i<11;i++){
13                if(cha[i]>='1'&& cha[i]<='9')
14                    count++;
15            }
16            if(count==4)
17                return true;
18            else
19                return false;
20        }
21
22 // Include Constructor
23
24     public double calculateTicketCost(int numberOfTickets, String ACFacility){
25         double amount;
26         if(ACFacility.equalsIgnoreCase("yes")){
27             amount=750*numberOfTickets;
28         }
29         else{
30             amount=600*numberOfTickets;
31         }
32
33         return amount;
34     }
35
36 }
37

```

## SilverTicket.java

```

1
2     public class SilverTicket extends BookAMovieTicket{
3         public SilverTicket(String ticketId, String customerName, long mobileNumber,
4             String emaiId, String movieName){
5             super(ticketId, customerName, mobileNumber, emaiId, movieName);
6         }
7
8         public boolean validateTicketId(){
9             int count=0;
10            if(ticketId.contains("SILVER")){
11                count++;
12                char[] cha=ticketId.toCharArray();
13                for(int i=6;i<9;i++){
14                    if(cha[i]>='1'&& cha[i]<='9')
15                        count++;
16                }
17                if(count==4)
18                    return true;
19                else
20                    return false;
21            }
22
23 // Include Constructor
24
25     public double calculateTicketCost(int numberOfTickets, String ACFacility){
26         double amount;
27         if(ACFacility.equals("yes")){

```

```

28         amount=250*numberOfTickets;
29     }
30     else{
31         amount=100*numberOfTickets;
32     }
33
34     return amount;
35 }
36
37 }
38

```

## UserInterface.java

```

1 import java.util.*;
2
3 public class UserInterface {
4
5     public static void main(String[] args){
6         Scanner sc=new Scanner(System.in);
7         System.out.println("Enter Ticket Id");
8         String tid=sc.next();
9         System.out.println("Enter Customer name");
10        String cnm=sc.next();
11        System.out.println("Enter Mobile number");
12        long mno=sc.nextLong();
13        System.out.println("Enter Email id");
14        String email=sc.next();
15        System.out.println("Enter Movie name");
16        String mnmm=sc.next();
17        System.out.println("Enter number of tickets");
18        int tno=sc.nextInt();
19        System.out.println("Do you want AC or not");
20        String choice =sc.next();
21        if(tid.contains("PLATINUM")){
22            PlatinumTicket PT= new PlatinumTicket(tid,cnm,mno,email,mnmm);
23            boolean b1=PT.validateTicketId();
24            if(b1==true){
25                double cost=PT.calculateTicketCost(tno, choice);
26                System.out.println("Ticket cost is "+String.format("%.2f",cost));
27            }
28            else if(b1==false){
29                System.out.println("Provide valid Ticket Id");
30                System.exit(0);
31            }
32        }
33        else if(tid.contains("GOLD")){
34            GoldTicket GT= new GoldTicket(tid,cnm,mno,email,mnmm);
35            boolean b2=GT.validateTicketId();
36            if(b2==true){
37                double cost=GT.calculateTicketCost(tno,choice);
38                System.out.println("Ticket cost is "+String.format("%.2f",cost));
39            }
40            else if (b2==false){
41                System.out.println("Provide valid Ticket Id");
42                System.exit(0);
43            }
44        }
45        else if(tid.contains("SILVER")){
46            SilverTicket ST= new SilverTicket(tid,cnm,mno,email,mnmm);
47            boolean b3=ST.validateTicketId();
48            if(b3==true){
49                double cost=ST.calculateTicketCost(tno,choice);

```

```
50         System.out.println("Ticket cost is "+String.format("%.2f",cost));
51     }
52     else if (b3==false){
53         System.out.println("Provide valid Ticket Id");
54         System.exit(0);
55     }
56 }
57 }
58 }
59
60
```

## Grade

Reviewed on Monday, 7 February 2022, 4:18 PM by Automatic grade

**Grade** 100 / 100

**Assessment report**

[\[+\]Grading and Feedback](#)

---

```
1  
2  
3 public class InvalidPatientIdException extends Exception{  
4     public InvalidPatientIdException(String message) {  
5         super(message);  
6     }  
7 }  
8 }
```

```
>Main.java  SkeletonValidator.java  PatientUtility.java  InvalidPatientIdException.java  *Patient.java X
3  public class Patient {
4      private String patientId;
5      private String patientName;
6      private String contactNumber;
7      private String dateOfVisit;
8      private String patientAddress;
9
10     public String getPatientId() {
11         return patientId;
12     }
13     public void setPatientId(String patientId) {
14         this.patientId = patientId;
15     }
16     public String getPatientName() {
17         return patientName;
18     }
19     public void setPatientName(String patientName) {
20         this.patientName = patientName;
21     }
22     public String getContactNumber() {
23         return contactNumber;
24     }
25     public void setContactNumber(String contactNumber) {
26         this.contactNumber = contactNumber;
27     }
28     public String getDateOfVisit() {
29         return dateOfVisit;
30     }
31     public void setDateOfVisit(String dateOfVisit) {
32         this.dateOfVisit = dateOfVisit;
33     }
34     public String getPatientAddress() {
35         return patientAddress;
36     }
37     public void setPatientAddress(String patientAddress) {
38         this.patientAddress = patientAddress;
39     }
40
41 }
```

```
>Main.java  SkeletonValidator.java  PatientUtility.java  InvalidPatientIdException.java  *Patient.java  module-info.java
3@ import java.util.List;
4 import java.util.ArrayList;
5 import java.util.stream.Stream;
6 import java.io.File;
7 import java.io.FileNotFoundException;
8 import java.util.Scanner;
9 import java.util.regex.*;
10 import java.util.stream.Collectors;
11 import java.text.ParseException;
12 import java.text.SimpleDateFormat;
13 import java.util.Date;
14
15 public class PatientUtility {
16@     public List <Patient> fetchPatient(String filePath) {
17         List<Patient> patients = new ArrayList<>();
18         try {
19             File register = new File(filePath);
20             Scanner reader = new Scanner(register);
21             while(reader.hasNextLine()) {
22                 Patient p = new Patient();
23                 String[] infos = reader.nextLine().split(",");
24                 try {
25                     if (isValidPatientId(infos[0])) {
26                         p.setPatientId(infos[0]);
27                         p.setPatientName(infos[1]);
28                         p.setContactNumber(infos[2]);
29                         p.setDateOfVisit(infos[3]);
30                         p.setPatientAddress(infos[4]);
31                         patients.add(p);
32                     }
33                 }
34                 catch(InvalidPatientIdException e1) {
35                     System.out.println(e1.getMessage());
36                 }
37             }
38             reader.close();
39         }
40         catch(FileNotFoundException e) {}
41         return patients;
```

```
33         }
34     } catch(InvalidPatientIdException e1) {
35         System.out.println(e1.getMessage());
36     }
37     reader.close();
38 }
39 catch(FileNotFoundException e) {} |
40 return patients;
41 }
42 }
43
44 public boolean isValidPatientId (String patientId) throws InvalidPatientIdException {
45     Pattern p = Pattern.compile("^WM_[A-Z][0-9]{2}$");
46     Matcher m = p.matcher(patientId);
47     boolean isValid = m.matches();
48     if (!isValid) {
49         throw new InvalidPatientIdException(patientId + " is an Invalid Patient Id.");
50     }
51     return isValid;
52 }
53 public List<Patient> retrievePatientRecords_ByDateOfVisit(Stream<Patient> patientStream, String fromDate, String toDate) {
54     SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd-MM-yyyy");
55     return patientStream
56         .filter((p) -> {
57             try {
58                 Date start = simpleDateFormat.parse(fromDate);
59                 Date end = simpleDateFormat.parse(toDate);
60                 Date current = simpleDateFormat.parse(p.getDateOfVisit());
61                 return start.compareTo(current) * current.compareTo(end) >= 0;
62             }
63             catch(ParseException e) {}
64             return false;
65         }).collect(Collectors.toList());
66 }
67 public Stream<Patient> retrievePatientRecords_ByAddress(Stream<Patient> patientStream, String address) {
68     return patientStream.filter(p -> address.equals(p.getPatientAddress()));
69 }
70 }
71 }
```

```
Main.java *SkeletonValidator.java X PatientUtility.java InvalidPatientIdException.java *Patient.java module-info.java

20 import java.lang.reflect.Method;
3 import java.util.logging.Logger;
4 import java.util.logging.Level;
5 /**
6 * @author TJ
7 * This class is used to verify if the Code Skeleton is intact and not modified by participants thereby ensuring smooth auto evaluation*
8 */
9 public class SkeletonValidator {
10     public SkeletonValidator() {
11         validateClassName("Patient");
12         validateClassName("PatientUtility");
13         validateClassName("InvalidPatientIdException");
14         validateMethodSignature("fetchPatient:java.util.List,isValidPatientId:boolean,retrievePatientRecords_ByDateOfVisit:java.util.List,"
15             + "retrievePatientRecords_ByAddress:java.util.stream.Stream","PatientUtility");
16     }
17
18     private static final Logger LOG = Logger.getLogger("SkeletonValidator");
19
20     protected final boolean validateClassName(String className) {
21         boolean iscorrect = false;
22         try {
23             Class.forName(className);
24             iscorrect = true;
25             LOG.info("Class Name " + className + " is correct");
26         }
27         catch (ClassNotFoundException e) {
28             LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use the correct package "
29                 + "and class name as provided in the skeleton");
30         }
31         catch (Exception e) {
32             LOG.log(Level.SEVERE, "There is an error in validating the " + "Class Name. Please manually verify that the "
33                 + "Class name is same as skeleton before uploading");
34         }
35         return iscorrect;
36     }
37
38     protected final void validateMethodSignature(String methodWithExcptn, String className) {
39         Class cls = null;
40         try {
```

```
Main.java *SkeletonValidator.java X PatientUtility.java InvalidPatientIdException.java *Patient.java module-info.java
38+     protected final void validateMethodSignature(String methodWithExcptn, String className) {
39         Class cls = null;
40         try {
41             String[] actualmethods = methodWithExcptn.split(",");
42             boolean errorFlag = false;
43             String[] methodSignature;
44             String methodName = null;
45             String returnType = null;
46
47             for(String singleMethod : actualmethods) {
48                 boolean foundMethod = false;
49                 methodSignature = singleMethod.split(":");
50
51                 methodName = methodSignature[0];
52                 returnType = methodSignature[1];
53                 cls = Class.forName(className);
54                 Method[] methods = cls.getMethods();
55                 for (Method findMethod : methods) {
56                     if (methodName.equals(findMethod.getName())) {
57                         foundMethod = true;
58                         if (!((findMethod.getReturnType().getName()).equals(returnType))) {
59                             errorFlag = true;
60                             LOG.log(Level.SEVERE, " You have changed the " + "return type in '" + methodName + "' method. Please stick to the " + "skeleton provided");
61                         }
62                         else {
63                             LOG.info("Method signature of " + methodName + " is valid");
64                         }
65                     }
66                 }
67                 if (!foundMethod) {
68                     errorFlag = true;
69                     LOG.log(Level.SEVERE, " Unable to find the given public method " + methodName + ". Do not change the " + "given public method name. " + "Verify it with the ske
70
71
72
73                 }
74                 if (!errorFlag) {
75                     LOG.info("Method signature is valid");
76                 }
77             }
78         }
79     }
80 }
```

```
Main.java *SkeletonValidator.java × PatientUtility.java InvalidPatientIdException.java *Patient.java module-info.java
```

```
45 String returnType = null;
46
47 for(String singleMethod : actualmethods) {
48     boolean foundMethod = false;
49     methodSignature = singleMethod.split(":");
50
51     methodName = methodSignature[0];
52     returnType = methodSignature[1];
53     cls = Class.forName(className);
54     Method[] methods = cls.getMethods();
55     for (Method findMethod : methods) {
56         if (methodName.equals(findMethod.getName())) {
57             foundMethod = true;
58             if (!(findMethod.getReturnType().getName().equals(returnType))) {
59                 errorFlag = true;
60                 LOG.log(Level.SEVERE, " You have changed the " + "return type in '" + methodName + "' method. Please stick to the " + "skeleton provided");
61             }
62             else {
63                 LOG.info("Method signature of " + methodName + " is valid");
64             }
65         }
66     }
67     if (!foundMethod) {
68         errorFlag = true;
69         LOG.log(Level.SEVERE, " Unable to find the given public method " + methodName + ". Do not change the " + "given public method name. " + "Verify it with the ske
70     }
71 }
72 if (!errorFlag) {
73     LOG.info("Method signature is valid");
74 }
75
76
77 }
78 catch (Exception e) {
79     LOG.log(Level.SEVERE, " There is an error in validating the " + "method structure. Please manually verify that the " + "Method signature is same as the skeleton before
80 }
81 }
82 }
83 }
```

before uploading"); ↵

```
Main.java X *SkeletonValidator.java PatientUtility.java InvalidPatientIdException.java *Patient.java module-info.java
2 import java.util.Scanner;
3 import java.util.List;
4 import java.util.stream.Stream;
5
6 public class Main {
7
8     public static void main(String[] args) {
9
10        // CODE SKELETON - VALIDATION STARTS
11        // DO NOT CHANGE THIS CODE
12
13        new SkeletonValidator();
14
15        // CODE SKELETON - VALIDATION ENDS
16
17        Scanner sc = new Scanner(System.in);
18        PatientUtility pUtil = new PatientUtility();
19        System.out.println("Invalid Patient Id are:");
20
21        List<Patient> patients = pUtil.fetchPatient("PatientRegister.txt");
22        System.out.println("Retrieve Patient Details\n1. By Date of Visit\n2. By Address\nEnter your choice:");
23        int choice = sc.nextInt();
24        sc.nextLine();
25
26        if (choice == 1) {
27            System.out.println("Enter the start date");
28            String fromDate = sc.nextLine();
29            System.out.println("Enter the end date");
30            String toDate = sc.nextLine();
31            patients = pUtil.retrievePatientRecords_ByDateOfVisit(patients.stream(), fromDate, toDate);
32            if (!patients.isEmpty()) {
33                for (Patient p : patients) {
34                    System.out.printf("%s %s %s %s %s\n", p.getPatientId(), p.getPatientName(), p.getContactNumber(), p.getDateOfVisit(), p.getPatientAddress());
35                }
36            } else {
37                System.out.println("No patient records available during this interval");
38            }
39        }
40    }
}
```

```
>Main.java X *SkeletonValidator.java PatientUtility.java InvalidPatientIdException.java *Patient.java module-info.java

16
17     Scanner sc = new Scanner(System.in);
18     PatientUtility pUtil = new PatientUtility();
19     System.out.println("Invalid Patient Id are:");
20
21     List<Patient> patients = pUtil.fetchPatient("PatientRegister.txt");
22     System.out.println("Retrieve Patient Details\n1. By Date of Visit\n2. By Address\nEnter your choice:");
23     int choice = sc.nextInt();
24     sc.nextLine();
25
26     if (choice == 1) {
27         System.out.println("Enter the start date");
28         String fromDate = sc.nextLine();
29         System.out.println("Enter the end date");
30         String toDate = sc.nextLine();
31         patients = pUtil.retrievePatientRecords_ByDateOfVisit(patients.stream(), fromDate, toDate);
32         if (!patients.isEmpty()) {
33             for (Patient p : patients) {
34                 System.out.printf("%s %s %s %s %s\n", p.getPatientId(), p.getPatientName(), p.getContactNumber(), p.getDateOfVisit(), p.getPatientAddress());
35             }
36         } else {
37             System.out.println("No patient records available during this interval");
38         }
39     }
40
41     else if(choice == 2) {
42         System.out.println("Enter the address");
43         String address = sc.nextLine();
44         Stream<Patient> pStream = pUtil.retrievePatientRecords_ByAddress(patients.stream(), address);
45         pStream.forEach(p -> System.out.printf("%s %s %s %s %s\n", p.getPatientId(), p.getPatientName(), p.getContactNumber(), p.getDateOfVisit(), p.getPatientAddress()));
46     }
47     else {
48         System.out.println("Invalid Option");
49     }
50     sc.close();
51 }
52
53 }
```

```
driver - Notepad
File Edit Format View Help
package com.spring.main;

import java.util.Scanner;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Driver {
    public static void main(String[] args) {
        // fill the code
        Scanner sc=new Scanner(System.in);
        ApplicationContext context= new AnnotationConfigApplicationContext(ApplicationConfig.class);

        System.out.println("Enter the student name:");
        String name=sc.nextLine();
        System.out.println("Enter the admission number:");
        String admissionNo=sc.nextLine();
        System.out.println("Enter the grade:");
        char grade= sc.nextChar();

        System.out.println("Enter the hostel details");
        System.out.println("Enter the hostel name:");
        String hostelName=sc.nextLine();
        System.out.println("Enter the room rent:");
        float rent=sc.nextFloat();

        StudentService ss=new StudentService();
        ss.calculateFee (name, admissionNo, grade, hostelName, rent);
    }
}
```

```
studentBO - Notepad
File Edit Format View Help
package com.spring.bo;

import com.spring.model.Student;

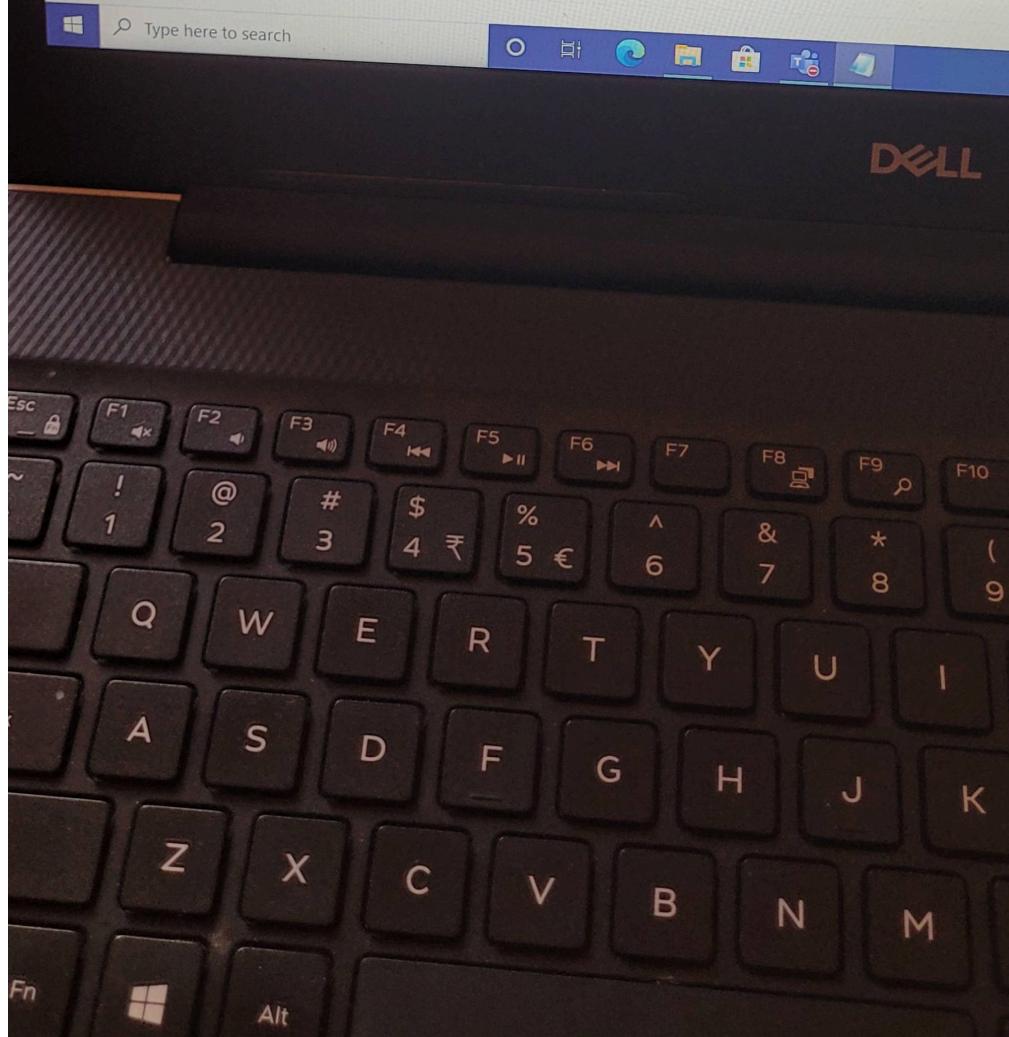
import org.springframework.stereotype.Component;
//use appropriate annotation to make this class as component class
@Component
public class StudentBO {

    public float calculateFee(Student obj) {
        float fee=0;
        //fill the code

        if(Student.getGrade()=='A'){
            fee = (Student.getHostel.getRent()-(Student.getHostel.getRent()*30/100));
        }else if(Student.getGrade()=='O'){
            fee= (Student.getHostel.getRent()-(Student.getHostel.getRent()*50/100));
        }else if(Student.getGrade()=='D'){
            fee=(Student.getHostel.getRent()-(Student.getHostel.getRent()*40/100));
        }else if(Student.getGrade()=='B'){
            fee=(Student.getHostel.getRent()-(Student.getHostel.getRent()*20/100));
        }else if(Student.getGrade()=='C' || Student.getGrade()=='F'){
            fee=Student.getHostel.getRent();
        }

        return fee;
    }

}
```



## StudentService - Notepad

File Edit Format View Help

```
package com.spring.service;
```

```
import com.spring.exception.InvalidGradeException;  
  
import com.spring.bo.StudentBO ;  
import org.springframework.stereotype.Component;  
import org.springframework.beans.factory.annotation.Autowired;  
  
//use appropriate annotation to make this class as component class  
@Component  
public class StudentService {  
  
    public StudentBO getStudentBOObj() {  
        return studentBOObj;  
    }  
  
    public void setStudentBOObj(StudentBO studentBOObj) {  
        this.studentBOObj = studentBOObj;  
    }  
  
    private StudentBO studentBOObj;  
  
    //fill the code  
    @Autowired  
    public StudentService(StudentBO studentBOObj) {  
        super();  
        this.studentBOObj = studentBOObj;  
    }  
  
    public float calculateFee(String name, String admissionNo, char grade, String hostelName, float rent) {  
        //fill the code  
  
        if(grade=='O' || grade=='A' || grade=='D' || grade=='B' || grade=='C' || grade=='F'){  
            Student st=new Student();  
            st.setName(name);  
            st.setAdmissionNo(admissionNo);  
            st.setGrade(grade);  
            st.setHostelName(hostelName);  
            st.setRent(rent);  
  
            studentBOObj.calculateFee(st);  
        }  
        else{  
            throws new InvalidGradeException("Invalid Grade");  
        }  
  
        return 0;  
    }  
}
```



Type here to search



& Run   Evaluate   Full screen   Description   Assessment Home

FeesDiscountData... Hostel.java Student.java StudentService.java

```
1 package com.spring.model;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Component;
5 //pojo class with required attributes, getters and setters
6 //use appropriate annotation to make this class as component class
7 @Component
8 public class Student {
9
10     private String name;
11     private String admissionNo;
12     private char grade;
13     private Hostel hostel;
14     //fill the code
15     @Autowired
16     public Student(Hostel hostel) {
17         super();
18         this.hostel = hostel;
19     }
20     public Hostel getHostel() {
21         return hostel;
22     }
23     public void setHostel(Hostel hostel) {
24         this.hostel = hostel;
25     }
}
```

Note: Ensure to provide all your program's inputs in the input field.

the Student object and set the name, admissionNo, hostelName, rent in that object. In case the grade is user-defined Exception InvalidGradeException should be thrown with message "Invalid Grade".

If the validation is done call the method calculateFee in StudentBO class and perform the implementation.

Please editor before Compile & Run.  
Assessment.

**Limitations and Constraints:****Full screen****Description**

Driver.java

FeesDiscountData...

Hostel.java

Student.java

StudentSer...

```
1 package com.spring.model;
2
3 import org.springframework.stereotype.Component;
4 //pojo class with required attributes, getters and setters
5 //use appropriate annotation to make this class as component class
6 @Component
7 public class Hostel {
8
9     private String hostelName;
10    private float rent;
11
12    public String getHostelName() {
13        return hostelName;
14    }
15    public void setHostelName(String hostelName) {
16        this.hostelName = hostelName;
17    }
18    public float getRent() {
19        return rent;
20    }
21    public void setRent(float rent) {
22        this.rent = rent;
23    }
24 }
```

53123

Note: Ensure to provide all your program's inputs at a single stretch before clicking "Compile and Run".  
Eg: Program expects an input of ID, Name and Phone which displays all these details in the console as output, then the inputs must be given as:



DELL

d=53123#

**Cognizant |**

the Student object and set the name, admission hostelName, rent in that object. In case the grade user-defined Exception InvalidGradeException is thrown with message "Invalid Grade".

If the validation is done call the method calculateFees in StudentBO class and perform the implementation.

**Limitations and Constraints:**

code editor before Compile & Run.  
ie assessment.

te

Full screen

Description

InvalidGradeExce... ▾ Driver.java ▾ FeesDiscountData... ▾ Hostel.java ▾ Studen...

```
1 package com.spring.model;
2
3 import java.util.Map;
4
5 import org.springframework.stereotype.Component;
6 //pojo class with required attributes, getters and setters
7 //use appropriate annotation to make this class as component class
8 @Component
9 public class FeesDiscountDetails {
10
11
12     //fill the code
13     @Value("#{${feesdiscount.map}}")
14     private Map<String, Integer> feesDiscount;
15
16     public Map<String, Integer> getfeesDiscount() {
17         return feesDiscount;
18     }
19
20     public void setFeesDiscount(Map<String, Integer> feesDiscount) {
21         this.feesDiscount = feesDiscount;
22     }
23
24 }
```

Note: Ensure to provide all your program's inputs at a single stretch before clicking "Compile and Run".  
Eg: Program expects an input of ID Name and Phone, which displays all these data in the console as output, then the inputs must be given as

DELL

**Question Description**

Getter and setter method for the above attribute part of code skeleton. One argument constructor part of code skeleton. The **StudentBO** object be autowired above the constructor via annotation.

Spring Framework / Hostel Management V1

Please edit before Compile & Run.  
assessment.

In this **StudentService** class, the method **public void calculateFee ( String name, String admissionGrade, String hostelName, float rent )** accepts name, admissionGrade, hostelName, rent as the arguments .Validation grade values should be 'O', 'D', 'A', 'B', 'C', 'F' if the Student object and set the name, admissionGrade, hostelName, rent in that object. In case the grade is user-defined Exception **InvalidGradeException** is thrown with message "Invalid Grade".

If the validation is done call the method **calculateFee()** in **StudentBO** class and perform the implementation.

**Limitations and Constraints:**

Full screen

Description

StudentBO.java   ApplicationConfig...   InvalidGradeException.java   ★ Driver.java   Fees

```
1 package com.spring.exception;
2
3 public class InvalidGradeException extends Exception {
4     public InvalidGradeException(string msg) {
5         // fill the code
6         super(msg);
7     }
8 }
```



DELL

j=53123#

**Cognizant |**

Spring Framework / Hostel Management V1

code editor before Compile & Run.  
he assessment.

Full screen      Description

pom.xml   StudentBO.java   ApplicationConfig...   InvalidGradeExce...   feesDiscountData...

```
1 package com.spring.config;
2
3 import org.springframework.context.annotation.ComponentScan;
4 import org.springframework.context.annotation.Configuration;
5 //Use appropriate annotation
6
7 @Configuration
8 @ComponentScan
9 public class ApplicationConfig {
10
11
12 }
13
```

DELL

**Question Description**

Getter and setter method for the above attribute part of code skeleton. One argument constructor part of code skeleton. The **StudentBO** object should be **autowired** above the **constructor** via annotation.

In this **StudentService** class, the method **public float calculateFee ( String name, String admissionNo, String hostelName, float rent )** accepts name, admissionNo, grade, hostelName, rent as the arguments .Valid grade values should be 'O', 'D', 'A', 'B', 'C', 'F' if the grade is invalid then set the grade in the Student object and set the name, admissionNo, hostelName, rent in that object. In case the grade is invalid then throw user-defined Exception **InvalidGradeException** should be thrown with message "**Invalid Grade**".

If the validation is done call the method **calculateFee()** in **StudentBO** class and perform the implementation.

**Limitations and Constraints:**

```
driver - Notepad
File Edit Format View Help
package com.spring.main;

import java.util.Scanner;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Driver {
    public static void main(String[] args) {
        // fill the code
        Scanner sc=new Scanner(System.in);
        ApplicationContext context= new AnnotationConfigApplicationContext(ApplicationConfig.class);

        System.out.println("Enter the student name:");
        String name=sc.nextLine();
        System.out.println("Enter the admission number:");
        String admissionNo=sc.nextLine();
        System.out.println("Enter the grade:");
        char grade= sc.nextChar();

        System.out.println("Enter the hostel details");
        System.out.println("Enter the hostel name:");
        String hostelName=sc.nextLine();
        System.out.println("Enter the room rent:");
        float rent=sc.nextFloat();

        StudentService ss=new StudentService();
        ss.calculateFee (name, admissionNo, grade, hostelName, rent);
    }
}
```

```
studentBO - Notepad
File Edit Format View Help
package com.spring.bo;

import com.spring.model.Student;

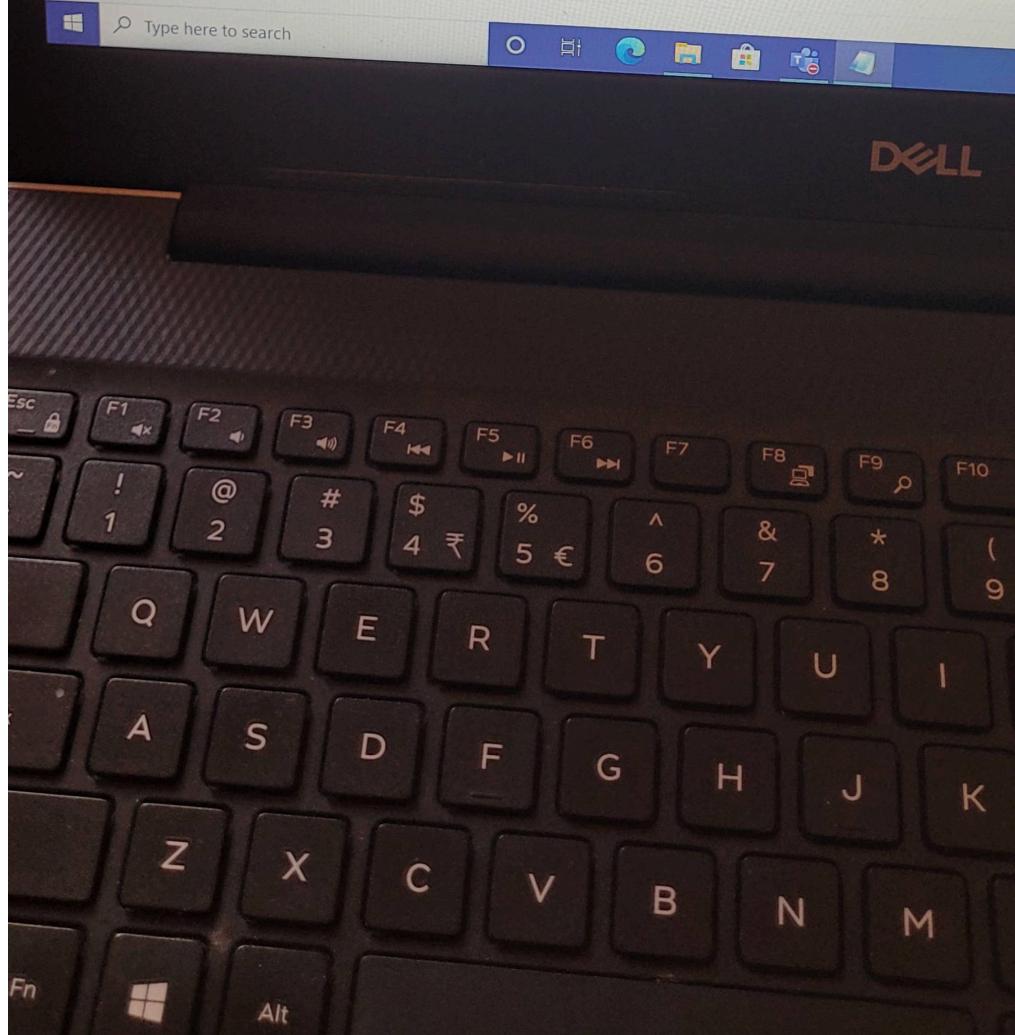
import org.springframework.stereotype.Component;
//use appropriate annotation to make this class as component class
@Component
public class StudentBO {

    public float calculateFee(Student obj) {
        float fee=0;
        //fill the code

        if(Student.getGrade()=='A'){
            fee = (Student.getHostel.getRent()-(Student.getHostel.getRent()*30/100));
        }else if(Student.getGrade()=='O'){
            fee= (Student.getHostel.getRent()-(Student.getHostel.getRent()*50/100));
        }else if(Student.getGrade()=='D'){
            fee=(Student.getHostel.getRent()-(Student.getHostel.getRent()*40/100));
        }else if(Student.getGrade()=='B'){
            fee=(Student.getHostel.getRent()-(Student.getHostel.getRent()*20/100));
        }else if(Student.getGrade()=='C' || Student.getGrade()=='F'){
            fee=Student.getHostel.getRent();
        }

        return fee;
    }

}
```



## StudentService - Notepad

File Edit Format View Help

```
package com.spring.service;
```

```
import com.spring.exception.InvalidGradeException;  
  
import com.spring.bo.StudentBO ;  
import org.springframework.stereotype.Component;  
import org.springframework.beans.factory.annotation.Autowired;  
  
//use appropriate annotation to make this class as component class  
@Component  
public class StudentService {  
  
    public StudentBO getStudentBOObj() {  
        return studentBOObj;  
    }  
  
    public void setStudentBOObj(StudentBO studentBOObj) {  
        this.studentBOObj = studentBOObj;  
    }  
  
    private StudentBO studentBOObj;  
  
    //fill the code  
    @Autowired  
    public StudentService(StudentBO studentBOObj) {  
        super();  
        this.studentBOObj = studentBOObj;  
    }  
  
    public float calculateFee(String name, String admissionNo, char grade, String hostelName, float rent) {  
        //fill the code  
  
        if(grade=='O' || grade=='A' || grade=='D' || grade=='B' || grade=='C' || grade=='F'){  
            Student st=new Student();  
            st.setName(name);  
            st.setAdmissionNo(admissionNo);  
            st.setGrade(grade);  
            st.setHostelName(hostelName);  
            st.setRent(rent);  
  
            studentBOObj.calculateFee(st);  
        }  
        else{  
            throws new InvalidGradeException("Invalid Grade");  
        }  
  
        return 0;  
    }  
}
```



Type here to search



& Run   Evaluate   Full screen   Description   Assessment Home

FeesDiscountData... Hostel.java Student.java StudentService.java

```
1 package com.spring.model;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Component;
5 //pojo class with required attributes, getters and setters
6 //use appropriate annotation to make this class as component class
7 @Component
8 public class Student {
9
10     private String name;
11     private String admissionNo;
12     private char grade;
13     private Hostel hostel;
14     //fill the code
15     @Autowired
16     public Student(Hostel hostel) {
17         super();
18         this.hostel = hostel;
19     }
20     public Hostel getHostel() {
21         return hostel;
22     }
23     public void setHostel(Hostel hostel) {
24         this.hostel = hostel;
25     }
}
```

Note: Ensure to provide all your program's inputs in the input field.

the Student object and set the name, admissionNo, hostelName, rent in that object. In case the grade is user-defined Exception InvalidGradeException should be thrown with message "Invalid Grade".

If the validation is done call the method calculateFee in StudentBO class and perform the implementation.

Please editor before Compile & Run.  
Assessment.

**Limitations and Constraints:****Full screen****Description**

Driver.java

FeesDiscountData...

Hostel.java

Student.java

StudentSer...

```
1 package com.spring.model;
2
3 import org.springframework.stereotype.Component;
4 //pojo class with required attributes, getters and setters
5 //use appropriate annotation to make this class as component class
6 @Component
7 public class Hostel {
8
9     private String hostelName;
10    private float rent;
11
12    public String getHostelName() {
13        return hostelName;
14    }
15    public void setHostelName(String hostelName) {
16        this.hostelName = hostelName;
17    }
18    public float getRent() {
19        return rent;
20    }
21    public void setRent(float rent) {
22        this.rent = rent;
23    }
24 }
```

53123

Note: Ensure to provide all your program's inputs at a single stretch before clicking "Compile and Run".  
Eg: Program expects an input of ID, Name and Phone which displays all these details in the console as output, then the inputs must be given as:



DELL

d=53123#

**Cognizant |**

the Student object and set the name, admission hostelName, rent in that object. In case the grade user-defined Exception InvalidGradeException is thrown with message "Invalid Grade".

If the validation is done call the method calculateFees in StudentBO class and perform the implementation.

**Limitations and Constraints:**

code editor before Compile & Run.  
ie assessment.

te

Full screen

Description

InvalidGradeExce... ▾ Driver.java ▾ FeesDiscountData... ▾ Hostel.java ▾ Studen...

```
1 package com.spring.model;
2
3 import java.util.Map;
4
5 import org.springframework.stereotype.Component;
6 //pojo class with required attributes, getters and setters
7 //use appropriate annotation to make this class as component class
8 @Component
9 public class FeesDiscountDetails {
10
11
12     //fill the code
13     @Value("#{${feesdiscount.map}}")
14     private Map<String, Integer> feesDiscount;
15
16     public Map<String, Integer> getfeesDiscount() {
17         return feesDiscount;
18     }
19
20     public void setFeesDiscount(Map<String, Integer> feesDiscount) {
21         this.feesDiscount = feesDiscount;
22     }
23
24 }
```

Note: Ensure to provide all your program's inputs at a single stretch before clicking "Compile and Run".  
Eg: Program expects an input of ID Name and Phone, which displays all these data in the console as output, then the inputs must be given as

DELL

**Question Description**

Getter and setter method for the above attribute part of code skeleton. One argument constructor part of code skeleton. The **StudentBO** object be autowired above the constructor via annotation.

Spring Framework / Hostel Management V1

Please edit before Compile & Run.  
assessment.

In this **StudentService** class, the method **public void calculateFee ( String name, String admissionGrade, String hostelName, float rent )** accepts name, admissionGrade, hostelName, rent as the arguments .Validation grade values should be 'O', 'D', 'A', 'B', 'C', 'F' if the Student object and set the name, admissionGrade, hostelName, rent in that object. In case the grade is user-defined Exception **InvalidGradeException** is thrown with message "Invalid Grade".

If the validation is done call the method **calculateFee()** in **StudentBO** class and perform the implementation.

**Limitations and Constraints:**

Full screen

Description

StudentBO.java

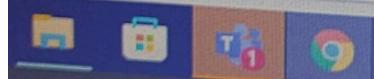
ApplicationConfig...

InvalidGradeExce...

Driver.java

Fees

```
1 package com.spring.exception;
2
3 public class InvalidGradeException extends Exception {
4     public InvalidGradeException(string msg) {
5         // fill the code
6         super(msg);
7     }
8 }
```



DELL

j=53123#

**Cognizant |**

Spring Framework / Hostel Management V1

code editor before Compile & Run.  
he assessment.

**Full screen** **Description**

pom.xml StudentBO.java ApplicationConfig... InvalidGradeExc... feesDiscountData...

```
1 package com.spring.config;
2
3 import org.springframework.context.annotation.ComponentScan;
4 import org.springframework.context.annotation.Configuration;
5 //Use appropriate annotation
6
7 @Configuration
8 @ComponentScan
9 public class ApplicationConfig {
10
11
12 }
13
```

**Question Description**

Getter and setter method for the above attribute part of code skeleton. One argument constructor part of code skeleton. The **StudentBO** object should be **autowired** above the **constructor** via annotation.

In this **StudentService** class, the method **public float calculateFee ( String name, String admissionNo, String hostelName, float rent )** accepts name, admissionNo, grade, hostelName, rent as the arguments. Validate the grade values should be 'O', 'D', 'A', 'B', 'C', 'F' if the grade is invalid then create a new **Student** object and set the name, admissionNo, hostelName, rent in that object. In case the grade is invalid then throw user-defined Exception **InvalidGradeException** should be thrown with message "**Invalid Grade**".

If the validation is done call the method **calculateFee()** in **StudentBO** class and perform the implementation.

**Limitations and Constraints:**

DELL

## #1-ADD CUSTOMER- PACKAGE:

```
create PACKAGE pkg_customer
as
  PROCEDURE addCustomer (c_id t_customer.cust_id%type, c_name t_customer.cust_
name%type, c_addr t_customer.cust_address%type);
end pkg_customer;
/

create PACKAGE body pkg_customer
as
  PROCEDURE addCustomer(c_id t_customer.cust_id%type, c_name t_customer.cust_n
ame%type, c_addr t_customer.cust address%type)
  as
    begin
      insert into t_customer(cust_id, cust_name, cust_address)
      values (c_id, c_name,c_addr);
    end addCustomer;
end pkg_customer;
/
```

## #2-CHARCOUNT FUNCTION:

```
CREATE OR REPLACE FUNCTION CHARCOUNT(str VARCHAR2)
RETURN NUMBER
AS
  r NUMBER:=0;
  x NUMBER;
BEGIN
  X := REGEXP_COUNT(str, ' ');
  r := LENGTH (str)-x;
  RETURN r;
END;
/
```

-----OR-----

```
create or replace FUNCTION characterCount(str IN VARCHAR2)
return number
as
  character_val number :=0;
begin
  for i in 1..length(str) loop
    character_val:=character_val+1;
  end loop;
  return character_val;
end;
```

/

### #3-DELETE CUSTOMER- PACKAGE:

```
create PACKAGE pkg_delcustomer
as
  PROCEDURE deleteCustomer (c_id t_customer.cust_id%type);
end pkg_delcustomer;
/

create PACKAGE body pkg_delcustomer
as
  PROCEDURE deleteCustomer (c_id t_customer.cust_id%type)
  as
  begin
    delete from t_customer where cust_id=c_id;
    exception
      when no_data_found then
        dbms_output.put_line( 'NO CUSTOMER AVAILABLE');
    end deleteCustomer;
  end pkg_delcustomer;
/
```

### #4-GET ORDER DETAILS- PROCEDURE:

```
set serveroutput on;
create or replace PROCEDURE getorderDetails(p_order_id IN t_order_details.order_id%TYPE)
as
  cursor cur_tod is select item_id, quantity from t_order_details where order_id=p_order_id;
  a_item_id t_order_details.item_id%type;
  a_quantity t_order_details.quantity%type;
begin
  dbms_output.put_line('Order ID : ' || p_order_id);
  dbms_output.put_line('Item ID' || ' ' || 'Quantity');
  dbms_output.put_line ('-----');
  open cur_tod;
  loop
    fetch cur_tod into a_item_id, a_quantity;
    exit when cur_tod%notfound;
    dbms_output.put_line (a_item_id || ' ' || a_quantity);
  end loop;
  close cur_tod;
```

```

exception
when no_data_found then
dbms_output.put_line('NA');
end;
/

```

## #5-PALINDROME:

```

create or replace function palindrome(str IN varchar2)
return boolean
as
c varchar2(50);
rv varchar2(50);
begin
for i in reverse 1..length(str) loop
c:=substr(str,i,1);
rv:= rv || c;
end loop;
if rv=str then
return true;
else
return false;
end if;
end;
/

```

## #6-WORDCOUNT FUNCTION:

```

CREATE OR REPLACE FUNCTION wordcount(str VARCHAR2)
RETURN NUMBER
AS
r number;
BEGIN
str := LTRIM(str, ' ');
R:= REGEXP_COUNT(STR, ' ') +1;
RETURN R;
END;
/

```

-----OR-----

```

CREATE OR REPLACE FUNCTION wordcount(str VARCHAR2)
RETURN NUMBER
AS
r number:=1;

```

```
c CHAR;
BEGIN
  FOR i IN 1..LENGTH(STR) LOOP
    C:= SUBSTR(str, i, 1);
    IF C=' ' THEN
      R:= R+1;
    END IF;
  END LOOP;
  RETURN R;
END;
/
```

-----OR-----

```
create or replace FUNCTION wordCount(str IN VARCHAR2)
return number
as
  val_words number:=1;
  z char;
begin
  for i in 1..length(str) loop
    z:=substr(trim(str,' '),i,1);
    if z=' ' then
      val_words:=val_words+1;
    end if;
  end loop;
  return val_words;
end;
/
```

## #1-ADD CUSTOMER- PACKAGE:

```
create PACKAGE pkg_customer
as
  PROCEDURE addCustomer (c_id t_customer.cust_id%type, c_name t_customer.cust_
name%type, c_addr t_customer.cust_address%type);
end pkg_customer;
/

create PACKAGE body pkg_customer
as
  PROCEDURE addCustomer(c_id t_customer.cust_id%type, c_name t_customer.cust_n
ame%type, c_addr t_customer.cust address%type)
  as
    begin
      insert into t_customer(cust_id, cust_name, cust_address)
      values (c_id, c_name,c_addr);
    end addCustomer;
end pkg_customer;
/
```

## #2-CHARCOUNT FUNCTION:

```
CREATE OR REPLACE FUNCTION CHARCOUNT(str VARCHAR2)
RETURN NUMBER
AS
  r NUMBER:=0;
  x NUMBER;
BEGIN
  X := REGEXP_COUNT(str, ' ');
  r := LENGTH (str)-x;
  RETURN r;
END;
/
```

-----OR-----

```
create or replace FUNCTION characterCount(str IN VARCHAR2)
return number
as
  character_val number :=0;
begin
  for i in 1..length(str) loop
    character_val:=character_val+1;
  end loop;
  return character_val;
end;
```

/

### #3-DELETE CUSTOMER- PACKAGE:

```
create PACKAGE pkg_delcustomer
as
  PROCEDURE deleteCustomer (c_id t_customer.cust_id%type);
end pkg_delcustomer;
/

create PACKAGE body pkg_delcustomer
as
  PROCEDURE deleteCustomer (c_id t_customer.cust_id%type)
  as
  begin
    delete from t_customer where cust_id=c_id;
    exception
      when no_data_found then
        dbms_output.put_line( 'NO CUSTOMER AVAILABLE');
    end deleteCustomer;
  end pkg_delcustomer;
/
```

### #4-GET ORDER DETAILS- PROCEDURE:

```
set serveroutput on;
create or replace PROCEDURE getorderDetails(p_order_id IN t_order_details.order_id%TYPE)
as
  cursor cur_tod is select item_id, quantity from t_order_details where order_id=p_order_id;
  a_item_id t_order_details.item_id%type;
  a_quantity t_order_details.quantity%type;
begin
  dbms_output.put_line('Order ID : ' || p_order_id);
  dbms_output.put_line('Item ID' || ' ' || 'Quantity');
  dbms_output.put_line ('-----');
  open cur_tod;
  loop
    fetch cur_tod into a_item_id, a_quantity;
    exit when cur_tod%notfound;
    dbms_output.put_line (a_item_id || ' ' || a_quantity);
  end loop;
  close cur_tod;
```

```

exception
when no_data_found then
dbms_output.put_line('NA');
end;
/

```

## #5-PALINDROME:

```

create or replace function palindrome(str IN varchar2)
return boolean
as
c varchar2(50);
rv varchar2(50);
begin
for i in reverse 1..length(str) loop
c:=substr(str,i,1);
rv:= rv || c;
end loop;
if rv=str then
return true;
else
return false;
end if;
end;
/

```

## #6-WORDCOUNT FUNCTION:

```

CREATE OR REPLACE FUNCTION wordcount(str VARCHAR2)
RETURN NUMBER
AS
r number;
BEGIN
str := LTRIM(str, ' ');
R:= REGEXP_COUNT(STR, ' ') +1;
RETURN R;
END;
/

```

-----OR-----

```

CREATE OR REPLACE FUNCTION wordcount(str VARCHAR2)
RETURN NUMBER
AS
r number:=1;

```

```
c CHAR;
BEGIN
  FOR i IN 1..LENGTH(STR) LOOP
    C:= SUBSTR(str, i, 1);
    IF C=' ' THEN
      R:= R+1;
    END IF;
  END LOOP;
  RETURN R;
END;
/
```

-----OR-----

```
create or replace FUNCTION wordCount(str IN VARCHAR2)
return number
as
  val_words number:=1;
  z char;
begin
  for i in 1..length(str) loop
    z:=substr(trim(str,' '),i,1);
    if z=' ' then
      val_words:=val_words+1;
    end if;
  end loop;
  return val_words;
end;
/
```

## #1-ADD CUSTOMER- PACKAGE:

```
create PACKAGE pkg_customer
as
  PROCEDURE addCustomer (c_id t_customer.cust_id%type, c_name t_customer.cust_
name%type, c_addr t_customer.cust_address%type);
end pkg_customer;
/

create PACKAGE body pkg_customer
as
  PROCEDURE addCustomer(c_id t_customer.cust_id%type, c_name t_customer.cust_n
ame%type, c_addr t_customer.cust address%type)
  as
    begin
      insert into t_customer(cust_id, cust_name, cust_address)
      values (c_id, c_name,c_addr);
    end addCustomer;
end pkg_customer;
/
```

## #2-CHARCOUNT FUNCTION:

```
CREATE OR REPLACE FUNCTION CHARCOUNT(str VARCHAR2)
RETURN NUMBER
AS
  r NUMBER:=0;
  x NUMBER;
BEGIN
  X := REGEXP_COUNT(str, ' ');
  r := LENGTH (str)-x;
  RETURN r;
END;
/
```

-----OR-----

```
create or replace FUNCTION characterCount(str IN VARCHAR2)
return number
as
  character_val number :=0;
begin
  for i in 1..length(str) loop
    character_val:=character_val+1;
  end loop;
  return character_val;
end;
```

/

### #3-DELETE CUSTOMER- PACKAGE:

```
create PACKAGE pkg_delcustomer
as
  PROCEDURE deleteCustomer (c_id t_customer.cust_id%type);
end pkg_delcustomer;
/

create PACKAGE body pkg_delcustomer
as
  PROCEDURE deleteCustomer (c_id t_customer.cust_id%type)
  as
  begin
    delete from t_customer where cust_id=c_id;
    exception
      when no_data_found then
        dbms_output.put_line( 'NO CUSTOMER AVAILABLE');
    end deleteCustomer;
  end pkg_delcustomer;
/
```

### #4-GET ORDER DETAILS- PROCEDURE:

```
set serveroutput on;
create or replace PROCEDURE getorderDetails(p_order_id IN t_order_details.order_id%TYPE)
as
  cursor cur_tod is select item_id, quantity from t_order_details where order_id=p_order_id;
  a_item_id t_order_details.item_id%type;
  a_quantity t_order_details.quantity%type;
begin
  dbms_output.put_line('Order ID : ' || p_order_id);
  dbms_output.put_line('Item ID' || ' ' || 'Quantity');
  dbms_output.put_line ('-----');
  open cur_tod;
  loop
    fetch cur_tod into a_item_id, a_quantity;
    exit when cur_tod%notfound;
    dbms_output.put_line (a_item_id || ' ' || a_quantity);
  end loop;
  close cur_tod;
```

```

exception
when no_data_found then
dbms_output.put_line('NA');
end;
/

```

## #5-PALINDROME:

```

create or replace function palindrome(str IN varchar2)
return boolean
as
c varchar2(50);
rv varchar2(50);
begin
for i in reverse 1..length(str) loop
c:=substr(str,i,1);
rv:= rv || c;
end loop;
if rv=str then
return true;
else
return false;
end if;
end;
/

```

## #6-WORDCOUNT FUNCTION:

```

CREATE OR REPLACE FUNCTION wordcount(str VARCHAR2)
RETURN NUMBER
AS
r number;
BEGIN
str := LTRIM(str, ' ');
R:= REGEXP_COUNT(STR, ' ') +1;
RETURN R;
END;
/

```

-----OR-----

```

CREATE OR REPLACE FUNCTION wordcount(str VARCHAR2)
RETURN NUMBER
AS
r number:=1;

```

```
c CHAR;
BEGIN
  FOR i IN 1..LENGTH(STR) LOOP
    C:= SUBSTR(str, i, 1);
    IF C=' ' THEN
      R:= R+1;
    END IF;
  END LOOP;
  RETURN R;
END;
/
```

-----OR-----

```
create or replace FUNCTION wordCount(str IN VARCHAR2)
return number
as
  val_words number:=1;
  z char;
begin
  for i in 1..length(str) loop
    z:=substr(trim(str,' '),i,1);
    if z=' ' then
      val_words:=val_words+1;
    end if;
  end loop;
  return val_words;
end;
/
```

# ICT Electricity Bill

## Automatic evaluation[+]

### EBBill/src/db.properties

```
1 db.classname=com.mysql.jdbc.Driver
2 db.url=jdbc:mysql://localhost:3306/ElectricityBill
3 db.username=
4 db.password=
```

### EBBill/src/DBHandler.java

```
1 import java.sql.Connection;
2 import java.io.FileInputStream;
3 import java.io.IOException;
4 import java.sql.*;
5 import java.util.Properties;
6 import java.io.FileNotFoundException;
7
8 public class DBHandler {
9
10    public Connection establishConnection() throws ClassNotFoundException, SQLException,
FileNotFoundException {
11
12        Connection con = null;
13        Properties props = new Properties();
14        // this try block reads the db Properties file and establishConnection.
15        try{
16            FileInputStream fis = new FileInputStream("src/db.properties");
17            props.load(fis);
18
19            Class.forName(props.getProperty("db.classname"));
20
21            con =
DriverManager.getConnection(props.getProperty("db.url"),props.getProperty("db.username"),props.getProperty("db.password"));
22        }
23        catch(IOException e){
24            e.printStackTrace();
25        }
26
27        return con;
28
29
30        //fill code here
31
32    }
33 }
34
```

### EBBill/src/ElectricityBill.java

```
1
2 //This is the POJO/model class
3
4 public class ElectricityBill {
5
6     private String consumerNumber;
7     private String consumerName;
8     private String consumerAddress;
```

```

9  private int unitsConsumed;
10 private double billAmount;
11
12 public String getConsumerNumber() {
13     return consumerNumber;
14 }
15
16 public void setConsumerNumber(String consumerNumber) {
17     this.consumerNumber = consumerNumber;
18 }
19
20 public String getConsumerName() {
21     return consumerName;
22 }
23
24 public void setConsumerName(String consumerName) {
25     this.consumerName = consumerName;
26 }
27
28 public String getConsumerAddress() {
29     return consumerAddress;
30 }
31
32 public void setConsumerAddress(String consumerAddress) {
33     this.consumerAddress = consumerAddress;
34 }
35
36 public int getUnitsConsumed() {
37     return unitsConsumed;
38 }
39
40 public void setUnitsConsumed(int unitsConsumed) {
41     this.unitsConsumed = unitsConsumed;
42 }
43
44 public double getBillAmount() {
45     return billAmount;
46 }
47
48 public void setBillAmount(double billAmount) {
49     this.billAmount = billAmount;
50 }
51
52
53 //Write the required business logic as expected in the question description
54 public void calculateBillAmount() {
55     // method for calculating the bill amount.
56     int units = unitsConsumed;
57     double bill = 0;
58
59     if(units <= 100){
60         bill = 0;
61     }
62     if(units > 100 && units <= 300){
63         bill = (units-100) * 1.5;
64     }
65     if(units > 300 && units <= 600){
66         bill = 200 * 1.5 + (units-300) * 3.5;
67     }
68     if(units > 600 && units <= 1000){
69         bill = 200 * 1.5 + 300 * 3.5 + (units-600) * 5.5;
70     }
71     if(units > 1000){
72         bill = 200 * 1.5 + 300 * 3.5 + 400 * 5.5 + (units-1000) * 7.5;
73     }
74
75     setBillAmount(bill);

```

```

76
77
78     //fill the code
79
80 }
81
82 }
```

## EBBill/src/ElectricityBill.txt

```

1 0191919191,John,Chennai,650
2 0191919192,Peter,Mumbai,1100
3 1919191919,Rose,Mumbai,453
4 0191919193,Tom,Hyderabad,750
5 01919191945,Raj,Chennai,120
6 0191919194,Sam,Chennai,250
7 0191919195,Anya,Chennai,34
```

## EBBill/src/ElectricityBoard.java

```

1 import java.util.List;
2 import java.util.*;
3 import java.io.FileReader;
4 import java.io.File;
5 import java.io.BufferedReader;
6 import java.io.FileNotFoundException;
7 import java.io.IOException;
8 import java.util.regex.Pattern;
9 import java.sql.SQLException;
10 import java.sql.Connection;
11 import java.sql.PreparedStatement;
12
13 public class ElectricityBoard {
14
15     //write the required business logic methods as expected in the question description
16     public void addBill(List<ElectricityBill> billList) {
17
18         DBHandler db = new DBHandler();
19
20         try(Connection con = db.establishConnection()){
21
22             PreparedStatement stmt = con.prepareStatement("insert into ElectricityBill values(?,?,?,?,?)");
23
24             // for loop to insert the values into the table
25             for(ElectricityBill obj : billList){
26                 stmt.setString(1,obj.getConsumerNumber());
27                 stmt.setString(2,obj.getConsumerName());
28                 stmt.setString(3,obj.getConsumerAddress());
29                 stmt.setInt(4,obj.getUnitsConsumed());
30                 stmt.setDouble(5,obj.getBillAmount());
31
32                 stmt.execute();
33             }
34         }
35         catch(ClassNotFoundException e){
36             e.printStackTrace();
37         }
38
39         catch(FileNotFoundException e){
40             e.printStackTrace();
41         }
42
43         catch(SQLException e){
44             e.printStackTrace();
45         }
46
47 }
```

```

48
49
50     //fill the code
51
52 }
53
54 public List<ElectricityBill> generateBill(String filePath) {
55
56     List <ElectricityBill> list = new ArrayList<>();
57     File f = new File (filePath);
58
59     // this try block is for opening and reading the file
60     try(BufferedReader br = new BufferedReader(new FileReader(f)))
61     {
62         String line = null;
63
64         while((line = br.readLine())!= null)
65         {
66             String records[] = null;
67             String consumerNumber = "";
68             String consumerName = "";
69             String consumerAddress = "";
70             int unitsConsumed = 0;
71
72             records = line.split(",");
73             consumerNumber = records[0];
74             consumerName = records[1];
75             consumerAddress = records[2];
76             unitsConsumed = Integer.parseInt(records[3]);
77
78
79         //this try block checks for the validated consumerNumber
80         try{
81             if(validate(consumerNumber)){
82                 ElectricityBill obj = new ElectricityBill();
83                 obj.setConsumerNumber(consumerNumber);
84                 obj.setConsumerName(consumerName);
85                 obj.setConsumerAddress(consumerAddress);
86                 obj.setUnitsConsumed(unitsConsumed);
87                 obj.calculateBillAmount();
88
89                 list.add(obj);
90             }
91         }
92         catch(InvalidConsumerNumberException e){
93             System.out.println(e.getMessage());
94         }
95     }
96
97     catch(FileNotFoundException e){
98         e.printStackTrace();
99     }
100    catch(IOException e){
101        e.printStackTrace();
102    }
103
104    return list;
105    //fill the code
106
107 }
108
109 public boolean validate(String consumerNumber) throws InvalidConsumerNumberException {
110
111     // method for validating the consumerNumber
112     boolean isValid = Pattern.matches("^\\d{10}$", consumerNumber);
113
114     if(!isValid){

```

```

115         throw new InvalidConsumerNumberException("Invalid Consumer Number");
116     }
117
118     return true;
119
120     //fill the code
121
122 }
123
124 }
125

```

## EBBill/src/InvalidConsumerNumberException.java

```

1
2 //make the required changes to this class so that InvalidConsumerNumberException is of type exception.
3
4 public class InvalidConsumerNumberException extends Exception{
5
6     public InvalidConsumerNumberException(String message)
7     {
8         super(message);
9     }
10
11    //fill the code
12
13 }

```

## EBBill/src/Main.java

```

1 import java.util.*;
2 import java.util.List;
3 import java.util.ArrayList;
4 public class Main {
5
6     public static void main(String[] args) {
7         Scanner sc= new Scanner(System.in);
8
9         String filePath = "src/ElectricityBill.txt";
10
11        List<ElectricityBill> list = new ArrayList<>();
12
13        ElectricityBoard eb = new ElectricityBoard();
14        list = eb.generateBill(filePath);
15
16        for(ElectricityBill obj: list){
17            System.out.println(obj.getConsumerNumber() + " " + obj.getConsumerName() + " " +
obj.getBillAmount());
18
19        }
20
21        eb.addBill(list);
22
23        System.out.println("Successfully Inserted");
24
25        sc.close();
26        //fill your code here
27
28    }
29
30 }
31

```

## script.sql

```

1 drop database if exists EBBill;
2

```

```
3 create database EBBill;
4
5 use EBBill;
6
7 create table ElectricityBill(consumer_number varchar(15) primary key,name varchar(2
8 5),address varchar(30),units_consumed int(5),bill_amount float(6,2));
9
10 COMMIT;
```

## Grade

Reviewed on Tuesday, 19 January 2021, 8:00 PM by Automatic grade

**Grade** 98.75 / 100

**Assessment report**

[\[+\]Grading and Feedback](#)

## 1. AirVoice - Registration

**Grade settings:** Maximum grade: 100

**Disable external file upload, paste and drop external content:** Yes

**Run:** Yes **Evaluate:** Yes

**Automatic grade:** Yes **Maximum execution time:** 16 s

SmartBuy is a leading mobile shop in the town. After buying a product, the customer needs to provide a few personal details for the invoice to be generated.

You being their software consultant have been approached to develop software to retrieve the personal details of the customers, which will help them to generate the invoice faster.

### Component Specification: Customer

Type(Class)	Attributes	Methods	Responsibilities
Customer	String customerName long contactNumber String emailId int age	Include the getters and setters method for all the attributes.	

In the **Main** class, create an object for the Customer class.

Get the details as shown in the sample input and assign the value for its attributes using the setters.

Display the details as shown in the sample output using the getters method.

All classes and methods should be public, Attributes should be private.

#### Note:

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

Ensure to follow the object oriented specifications provided in the question.

Ensure to provide the names for classes, attributes and methods as specified in the question.

Adhere to the code template, if provided.

#### Sample Input 1:

Enter the Name:

john

Enter the ContactNumber:

**9874561230**

Enter the EmailId:

**john@gmail.com**

Enter the Age:

**32**

### **Sample Output 1:**

Name:john

ContactNumber:9874561230

EmailId:john@gmail.com

Age:32

---

### [Automatic evaluation\[+\]](#)

*Customer.java*

```
1 public class Customer {  
2     private String customerName;  
3  
4     private long contactNumber;  
5  
6     private String emailId;  
7  
8     private int age;  
9  
10    public String getCustomerName() {  
11        return customerName;  
12    }  
13  
14    public void setCustomerName(String customerName) {  
15        this.customerName = customerName;  
16    }  
17  
18    public long getContactNumber() {  
19        return contactNumber;  
20    }  
21  
22    public void setContactNumber(long contactNumber) {  
23        this.contactNumber = contactNumber;  
24    }  
25  
26    public String getEmailId() {  
27        return emailId;  
28    }  
29  
30    public void setEmailId(String emailId) {  
31        this.emailId = emailId;  
32    }
```

```
33
34     public int getAge() {
35         return age;
36     }
37
38     public void setAge(int age){
39         this.age = age;
40     }
41
42
43
44 }
```

### Main.java

```
1 import java.util.Scanner;
2
3 public class Main {
4
5     public static void main (String[] args) {
6         Scanner sc=new Scanner(System.in);
7
8         //Fill the code
9         Customer c=new Customer();
10        System.out.println("Enter the Name:");
11        String name=(sc.nextLine());
12        System.out.println("Enter the ContactNumber:");
13        long no=sc.nextLong();
14        sc.nextLine();
15        System.out.println("Enter the EmailId:");
16        String mail=sc.nextLine();
17
18        System.out.println("Enter the Age:");
19        int age=sc.nextInt();
20        c.setCustomerName(name);
21        c.setContactNumber(no);
22        c.setEmailId(mail);
23        c.setAge(age);
24        System.out.println("Name:"+c.getCustomerName());
25        System.out.println("ContactNumber:"+c.getContactNumber());
26        System.out.println("EmailId:"+c.getEmailId());
27        System.out.println("Age:"+c.getAge());
28
29
30
31    }
32
33 }
```

## 2. Grade

Reviewed on Friday, 10 December 2021, 6:14 PM by Automatic grade

**Grade** 100 / 100

**Assessment report**

[\[+\]Grading and Feedback](#)

---

### 3. ZeeZee bank

**Grade settings:** Maximum grade: 100

**Disable external file upload, paste and drop external content:** Yes

**Run:** Yes **Evaluate:** Yes

**Automatic grade:** Yes **Maximum execution time:** 16 s

ZeeZee is a leading private sector bank. In the last Annual meeting, they decided to give their customer a 24/7 banking facility. As an initiative, the bank outlined to develop a stand-alone device that would offer deposit and withdrawal of money to the customers anytime.

You being their software consultant have been approached to develop software to implement the functionality of deposit and withdrawal anytime.

#### Component Specification: Account

Type(Class)	Attributes	Methods	Responsibilities
<b>Account</b>	long accountNumber  double balanceAmount	Include the getters and setters method for all the attributes.  Include a parametrized constructor of two arguments in the order – accountNumber,balanceAmount to initialize the values for the account object	

#### Requirement 1: Being able to deposit money into an account anytime

As per this requirement, the customer should be able to deposit money into his account at any time and the deposited amount should reflect in his account balance.

#### Component Specification: Account

Component Name	Type(Class)	Methods	Responsibilities
Deposit amount to an account	Account	public void deposit(double depositAmt)	This method takes the amount to be deposited as an argument  This method should perform the deposit, by adding the deposited amount to the balanceAmount

#### Requirement 2: Being able to withdraw money from the account anytime

As per this requirement, the customer should be able to withdraw money from his account anytime he wants. The amount to be withdrawn should be less than or equal to the balance in the account. After the withdrawal, the account should reflect the balance amount

## Component Specification: Account

Component Name	Type(Class)	Methods	Responsibilities
Withdraw amount from an account	Account	public boolean withdraw(double withdrawAmt)	This method should take the amount to be withdrawn as an argument.  This method should check the balanceAmount and deduct the withdraw amount from the balanceAmount and return true. If there is insufficient balance then return false.

In the **Main** class, Get the details as shown in the sample input.

Create an object for the Account class and invoke the deposit method to deposit the amount and withdraw method to withdraw the amount from the account.

All classes and methods should be public, Attributes should be private.

### Note:

Balance amount should be displayed corrected to 2 decimal places.

Order of the transactions to be performed (Display,Deposit,Withdraw).

If the balance amount is insufficient then display the message as shown in the Sample Input / Output.

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

Ensure to follow the object-oriented specifications provided in the question.

Ensure to provide the names for classes, attributes, and methods as specified in the question.

Adhere to the code template, if provided.

### Sample Input/Output 1:

Enter the account number:

**1234567890**

Enter the available amount in the account:

**15000**

Enter the amount to be deposited:

**1500**

Available balance is:16500.00

Enter the amount to be withdrawn:

**500**

Available balance is:16000.00

### **Sample Input/Output 2:**

Enter the account number:

**1234567890**

Enter the available amount in the account:

**15000**

Enter the amount to be deposited:

**1500**

Available balance is:16500.00

Enter the amount to be withdrawn:

**18500**

Insufficient balance

Available balance is:16500.00

---

---

## **Automatic evaluation** [\[+\]](#)

### **Main.java**

```
1 import java.util.Scanner;
2 import java.text.DecimalFormat;
3
4 public class Main{
5
6     public static void main (String[] args) {
7         Scanner sc=new Scanner(System.in);
8         DecimalFormat decimalFormat=new DecimalFormat("0.00");
9         System.out.println("Enter the account number:");
```

```

10    long accountNumber= sc.nextLong();
11    System.out.println("Enter the available amount in the account:");
12    double balanceAmount= sc.nextDouble();
13    Account account=new Account(accountNumber,balanceAmount);
14    System.out.println("Enter the amount to be deposited:");
15    double depositAmount=sc.nextDouble();
16    account.deposit(depositAmount);
17    double availableBalance=account.getBalanceAmount();
18    System.out.println("Available balance is:"+decimalFormat.format(availableBalance));
19    System.out.println("Enter the amount to be withdrawn:");
20    double withdrawAmount= sc.nextDouble();
21    boolean isWithdrawn=account.withdraw(withdrawAmount);
22    availableBalance=account.getBalanceAmount();
23    if(isWithdrawn){
24        System.out.println("Insufficient balance");
25    }
26    System.out.println("Available balance is:"+decimalFormat.format(availableBalance));
27
28    //Fill the code
29}
30}

```

## Account.java

```

1
2 public class Account {
3     private long accountNumber;
4     private double balanceAmount;
5     public Account(long accountNumber,double balanceAmount){
6         this.accountNumber=accountNumber;
7         this.balanceAmount=balanceAmount;
8     }
9     public long getAccountNumber(){
10        return accountNumber;
11    }
12    public void setAccountNumber(long accountNumber){
13        this.accountNumber=accountNumber;
14    }
15    public double getBalanceAmount(){
16        return balanceAmount;
17    }
18    public void setBalanceAmount(double balanceAmount){
19        this.balanceAmount=balanceAmount;
20    }
21    public void deposit(double depositAmount){
22        balanceAmount+=depositAmount;
23    }
24    public boolean withdraw(double withdrawAmount){
25        if(withdrawAmount<=balanceAmount){
26            balanceAmount-=withdrawAmount;
27            return true;
28        }
29        return false;
30    }
31}

```

## Grade

Reviewed on Thursday, 27 May 2021, 3:28 AM by Automatic grade

**Grade** 100 / 100

**Assessment report**

[\[+\]Grading and Feedback](#)

### 3.Call Details

**Grade settings:** Maximum grade: 100

**Disable external file upload, paste and drop external content:** Yes

**Run:** Yes **Evaluate:** Yes

**Automatic grade:** Yes **Maximum execution time:** 16 s

AirCarrier is a leading mobile network provider. They maintain a record of all the calls made by their postpaid customers. The details are stored in a particular format

[callId:calledNumber:noOfMinutes] . At the end of every month, the network provider wants to extract the information from the file and populate it to the Call object for calculating the bill.

You being their software consultant have been approached to develop software to implement the functionality of extracting the data from the given format.

#### Component Specification: Call

Type(Class)	Attributes	Methods	Responsibilities
Call	int callId long calledNumber float duration	Include the getters and setters method for all the attributes.	

#### Requirement 1: Extracting the data from the callDetails

This requirement is responsible for extracting the customer's callId, calledNumber and duration from the callDetails. After the extraction set the callId, calledNumber and duration to the call object.

#### Component Specification: Call

Component Name	Type(Class)	Methods	Responsibilities
Extraction from file	Call	public void parseData(String callDetails)	This method takes the callDetails as an argument  This method should perform the extraction process, to set the callId, calledNumber and duration of the call object.

In the **Main** class, Get the details as shown in the sample input.

Create an object for the Call and invoke the **parseData** method to set the callId, calledNumber and duration for each customer.

Invoke the corresponding getters to display the call details as shown in the Sample Output

All classes and methods should be public, Attributes should be private.

**Note:**

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

Ensure to follow the object-oriented specifications provided in the question.

Ensure to provide the names for classes, attributes and methods as specified in the question.

Adhere to the code template, if provided.

**Sample Input 1:**

Enter the call details:

**102:6547891230:2.15**

**Sample Output 1:**

Call id:102

Called number:6547891230

Duration:2.15

**Automatic evaluation[+]**

*Main.java*

```
1 import java.util.Scanner;
2
3 public class Main {
4
5     public static void main (String[] args) {
6         Scanner sc=new Scanner(System.in);
7         System.out.println("Enter the call details:");
8         String a=sc.nextLine();
9         Call obj=new Call();
10        obj.parseData(a);
11        System.out.println("Call id:"+obj.getCallId());
12        System.out.println("Called number:"+obj.getCalledNumber());
13        System.out.println("Duration:"+obj.getDuration());
14        //Fill the code
15
16    }
17 }
```

## Call.java

```
1  public class Call {  
2      private int callId;  
3      private long calledNumber;  
4      private float duration;  
5      public Call(){  
6      }  
7      public int getCallId(){  
8          return callId;  
9      }  
10     public long getCalledNumber(){  
11         return calledNumber;  
12     }  
13     public float getDuration(){  
14         return duration;  
15     }  
16     public void setCallId(int callId){  
17         this.callId=callId;  
18     }  
19     public void setCalledNumber(long calledNumber){  
20         this.calledNumber=calledNumber;  
21     }  
22     public void setDuration(float duration){  
23         this.duration=duration;  
24     }  
25     public void parseData(String calld){  
26         callId=Integer.parseInt(calld.split(":")[0]);  
27         setCallId(callId);  
28         calledNumber=Long.parseLong(calld.split(":")[1]);  
29         setCalledNumber(calledNumber);  
30         duration=Float.parseFloat(calld.split(":")[2]);  
31         setDuration(duration);  
32     }  
33 }  
34 }
```

## Grade

Reviewed on Tuesday, 4 May 2021, 4:58 AM by Automatic grade

**Grade** 100 / 100

**Assessment report**

[\[+\]Grading and Feedback](#)

---

## 4. Pair of Two digits

**Grade settings:** Maximum grade: 100

**Disable external file upload, paste and drop external content:** Yes

**Run:** Yes **Evaluate:** Yes

**Automatic grade:** Yes **Maximum execution time:** 32 s

Jerold teacher assigned a task to him. The task is to find the pair of two-digit numbers.

The pair is found by checking whether the product of the numbers is same as the product of the reversed numbers. If it is same, then print "Correct pair found". If not print, "Correct pair not found".

Write a Java program to find the correct pair of two-digit numbers.

Assume both the inputs are 2-digit values.

**Note:**

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the remaining text represents the output.

Adhere to the code template, if provided.

**Hint:** [13\*62=31\*26]

**Sample Input 1:**

**13**

**62**

**Sample Output 1:**

13 and 62 are correct pair

**Sample Input 2:**

**10**

**56**

**Sample Output 2:**

10 and 56 are not correct pair

---

**Automatic evaluation[+]**

*Main.java*

```
1 import java.util.*;
2
3 public class Main{
4
5     public static void main (String[] args) {
6         Scanner sc=new Scanner(System.in);
7
8         int a=sc.nextInt();
9         int b=sc.nextInt();
10        if(a>99||a<10||b>99||b<10){
11            System.out.println("No");
12
13        }
14        Main obj=new Main();
15        int ra=obj.rvs(a);
16        int rb=obj.rvs(b);
17        if(a*b==ra*rb){
18            System.out.println(a+" and "+b+" are correct pair");
19        }
}
```

```
20     else{
21         System.out.println(a+" and "+b+" are not correct pair");
22     }
23 }
24 int rvs(int num){
25     int r,rnum=0;
26     while(num>0)
27     {
28         r=num%10;
29         rnum=rnum*10+r;
30         num/=10;
31     }
32     return(rnum);
33 }
34 }
35
36
37
38
39
40
```

## Grade

Reviewed on Thursday, 27 May 2021, 3:38 AM by Automatic grade

**Grade** 100 / 100

**Assessment report**

TEST CASE PASSED

[\[+\]Grading and Feedback](#)

---

-----End-----

# Group-2

## 1. Find MemberShip Category Count

**Grade settings:** Maximum grade: 100

**Disable external file upload, paste and drop external content:** Yes

**Run:** Yes **Evaluate:** Yes

**Automatic grade:** Yes **Maximum execution time:** 32 s

### Find Membership Category Count

**ZEE Shopping mall** wanted to know how many Members are available for each of their Membership category. The member ship category is of three types (Gold, Silver and Platinum).

You being their software consultant have been approached by them to develop an application which can be used for managing their business. You need to implement a java program using thread to find out the count of members in each membership category. Membership details should be obtained from the user in the console.

#### Component Specification: Member (Model Class)

Type(Class)	Attributes	Methods	Responsibilities
Member	String memberId String memberName String category	Include getters and setter method for all the attributes.  Include a three argument constructor in the given order – memberId, memberName and category.	Set the values for all the attributes via constructor.

**Note:** The class and methods should be declared as public and all the attributes should be declared as private.

#### Requirement 1: Count the number of members

Count the number of members available in the memberList based on the membership category to be searched and set the value to count attribute.

#### Component Specification: ZEEShop (Thread Class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
	ZEEShop	String memberCategory int count List<Member>memberList	Include getters and setter method for all the attributes.	Set the values for all the attributes via constructor.

			Include a two argument constructor with arguments – memberCategory and memberList.	
Count the number of members	ZEEShop		void run()	Count the number of members based on the Membership category and set the value to the countattribute.

**Note:** The class and methods should be declared as public and all the attributes should be declared as private.

Create a class called **Main** with the main method and get the inputs like **number of members**, **member details**, **number of times Membership category needs to be searched** and **Membership category to be searched** from the user.

The member details will be in the form of String in the following format **memberId:memberName:category**.

Parse the member details and set the values for all attributes in **Member** class using **constructor**.

Invoke the ZEEShop thread class for each memberCategory and count the number of members in that category and display the count as shown in the sample input and output.

**Assumption:** The **memberCategory** is **case –sensitive** and will be of only three values – **Platinum or Gold or Silver**.

#### **Note:**

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the remaining text represents the output.

Ensure to follow object oriented specifications provided in the question description.  
Ensure to provide the names for classes, attributes and methods as specified in the question description.

Adhere to the code template, if provided.

#### **Sample Input / Output 1:**

Enter the number of Members:

**5**

Enter the Member Details:

**101:Tom:Gold**

Enter the Member Details:

**102:Sam:Gold**

Enter the Member Details:

**103:John:Silver**

Enter the Member Details:

**104:Rose:Platinum**

Enter the Member Details:

**105:Tint:Silver**

Enter the number of times Membership category needs to be searched:

**4**

Enter the Category

**Gold**

Enter the Category

**Silver**

Enter the Category

**Platinum**

Enter the Category

**Gold**

Gold:2

Silver:2

Platinum:1

Gold:2

---

Automatic evaluation [[+](#)]

*Member.java*

```
1  
2 public class Member {  
3
```

```

4   private String memberId;
5   private String memberName;
6   private String category;
7
8   public String getMemberId() {
9       return memberId;
10  }
11  public void setMemberId(String memberId) {
12      this.memberId = memberId;
13  }
14  public String getMemberName() {
15      return memberName;
16  }
17  public void setMemberName(String memberName) {
18      this.memberName = memberName;
19  }
20  public String getCategory() {
21      return category;
22  }
23  public void setCategory(String category) {
24      this.category = category;
25  }
26
27  public Member(String memberId, String memberName, String category) {
28      super();
29      this.memberId = memberId;
30      this.memberName = memberName;
31      this.category = category;
32  }
33
34
35 }
```

## Main.java

```

1 import java.util.*;
2 public class Main {
3
4     public static void main(String args[]){
5         // Fill the code here
6         List<Member> memberList = new ArrayList<Member>();
7         Scanner scan = new Scanner(System.in);
8         System.out.println("Enter the no of Members");
9         int memberCount = scan.nextInt();
10        String templp;
11        while(memberCount>0){
12            System.out.println("Enter the member details");
13            templp = scan.next();
14            String tempArr[] = templp.split(":");
15            memberList.add(new Member(tempArr[0],tempArr[1],tempArr[2]));
16            memberCount--;
17        }
18        System.out.println("Enter the number of times Membership category needs to be searched");
19        int noOfTimes = scan.nextInt();
20        String[] tempArr = new String[noOfTimes];
21        for(int index=0;index<noOfTimes;index++){
22            System.out.println("Enter the category");
23            tempArr[index] = scan.next();
24        }
25        int countArr[] = new int [noOfTimes];
26        for(int i=0; i<noOfTimes;i++){
27            ZEEShop thread = new ZEEShop(tempArr[i],memberList);
28            thread.run();
29            /*try{
30                thread.join();
31            }catch(InterruptedException e){
32            }*/
33    }
```

```

34     countArr[i] = thread.getCount();
35 }
36 for(int i=0;i<noOfTimes;i++){
37     System.out.println(tempArr[i]+ ":"+countArr[i]);
38 }
39 scan.close();
40 /*List<ZEEShop> zList = new ArrayList<ZEEShop>()
41 for(int i = 0;i<count;i++){
42     ZEEShop zs = new ZEEShop(category , memList);
43     zList.add(zs);
44 }
45 for(ZEEShop z: zeelist){
46     z.start();
47     try{
48         z.join();
49     }catch(Exception e){
50         e.printStackTrace();
51     }
52 }*/
53 }
54 }
55

```

## ZEEShop.java

```

1 import java.util.*;
2 public class ZEEShop extends Thread {
3     // Fill the code here
4     private String memberCategory;
5     private int count;
6     private List<Member> memberList;
7     public ZEEShop(String memberCategory, List memberList){
8         super();
9         this.memberCategory = memberCategory;
10        this.memberList = memberList;
11    }
12    public int getCount(){
13        return count;
14    }
15    public String getMemberCategory(){
16        return memberCategory;
17    }
18    public List<Member> getMemberList(){
19        return memberList;
20    }
21    public void setMemberCategory(String memberCategory){
22        this.memberCategory = memberCategory;
23    }
24    public void setMemberList(List<Member> memberList){
25        this.memberList = memberList;
26    }
27    public void setCount(int count){
28        this.count = count;
29    }
30    public void run(){
31
32        synchronized(this)
33        {
34            for(Member m : memberList){
35                if(m.getCategory().equals(memberCategory))
36                    count++;
37            }
38
39        }
40    }
41 }
42

```

# Grade

Reviewed on Friday, 7 January 2022, 7:25 PM by Automatic grade

**Grade** 100 / 100

**Assessment report**

[\[+\]Grading and Feedback](#)

---

---

## 2. Grade Calculation

**Grade settings:** Maximum grade: 100

**Disable external file upload, paste and drop external content:** Yes

**Run:** Yes **Evaluate:** Yes

**Automatic grade:** Yes **Maximum execution time:** 32 s

### Grade Calculation

Rita is working as a science teacher in an International school. She is the Class Teacher of class V and was busy in calculating the grade for each student in her class, based on his/her total marks obtained in SA1 assessment.

Since she found it very difficult to calculate the grade, she approached you to develop an application which can be used for completing her task faster. You need to implement a java program using thread to calculate the grade for each student. Student details should be obtained from the user in the console.

**Requirement 1:** Calculate the grade for each student.

Calculate the grade based on total marks (sum of all marks) as shown below obtained by each student and set the same in result attribute for respective student.

Total Marks	Grade
400 to 500	A
300 to 399	B
200 to 299	C
Less than 200	E

**Assumption:** Each student will have only five subjects and marks of each subject will be greater than or equal to 0 and lesser than or equal to 100. Hence the maximum Total marks obtained by each student will be 500. And the minimum Total marks obtained by each student will be 0.

**Component Specification: GradeCalculator (Thread Class)**

Component Name	Type(Class)	Attributes	Methods	Responsibilities
	GradeCalculator	String studName	Include getters and setter method for all	Set the values for all the attributes via constructor.

		char result int[] marks	the attributes. Include a two argument constructor in the given order – studName and marks.	
calculate the grade for each student	GradeCalculator		public void run()	Calculate the grade based on total marks and set the same to result attribute.

**Note:** The class and methods should be declared as public and all the attributes should be declared as private.

Create a class called **Main** with the main method and get the inputs like **number of threads** and **Student details** from the user.

The student details will be in the form of String in the following format **studName:mark1:mark2:mark3:mark4:mark5**.

Parse the student details and set the values of studName and marks attributes in **GradeCalculator** thread class using **constructor**.

Invoke the **GradeCalculator** thread class to calculate the grade based on total marks and set the same to result attribute.

Display the Student name and Grade obtained by each student as shown in the sample input and output.

**Note:**

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the remaining text represents the output.

Ensure to follow object oriented specifications provided in the question description.  
Ensure to provide the names for classes, attributes and methods as specified in the question description.

Adhere to the code template, if provided.

**Sample Input / Output 1:**

Enter the number of Threads:

**4**

Enter the String:

**Jeba:100:80:90:40:55**

Enter the String

**David:10:8:9:40:5**

Enter the String

**Adam:90:80:90:50:75**

Enter the String

**Rohit:99:99:99:99:99**

Jeba:B

David:E

Adam:B

Rohit:A

## Automatic evaluation[+]

### Main.java

```
1 import java.util.Scanner;
2 import java.io.BufferedReader;
3 import java.io.InputStreamReader;
4 public class Main {
5     public static void main(String[] args) throws Exception {
6         BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
7         System.out.println("Enter the number of Threads");
8         int th=Integer.parseInt(br.readLine());
9         GradeCalculator obj=null;
10        String str="";
11        String[] details=new String[th];
12        for(int i=0;i<th;i++)
13        {
14            System.out.println("Enter the String");
15            str=br.readLine();
16            details[i]=str;
17        }
18        for(int i=0;i<th;i++)
19        {
20            String sp[]=details[i].split(":");
21            int k=0;
22            int arr[]=new int[sp.length];
23            for(int j=1;j<sp.length;j++)
24                arr[k++]=Integer.parseInt(sp[j]);
25            obj=new GradeCalculator(sp[0],arr);
26            obj.start();
27            try{
28                Thread.sleep(1000);
29            }
30            catch(Exception e)
31            {
32                System.out.println(e);
33            }
34        }
35        //Fill your code here
36
37    }
38}
```

```

39 }
GradeCalculator.java
1
2 public class GradeCalculator extends Thread{
3   private String studName;
4   private char result;
5   private int[] marks;
6   public String getStudName()
7 {
8   return studName;
9 }
10 public void setStudName()
11 {
12   this.studName=studName;
13 }
14 public char getResult()
15 {
16   return result;
17 }
18 public void setResult(char result)
19 {
20   this.result=result;
21 }
22 public int[] getMarks()
23 {
24   return marks;
25 }
26 public void setMarks(int[] marks)
27 {
28   this.marks=marks;
29 }
30 public GradeCalculator(String studName,int[] marks)
31 {
32   this.studName=studName;
33   this.marks=marks;
34 }
35 public void run()
36 {
37   int sum=0;
38   int[] score=getMarks();
39   for(int i=0;i<score.length;i++)
40     sum=sum+score[i];
41   if(400<=sum)&&(sum<=500)
42     System.out.println(getStudName()+":"+'A');
43   if(300<=sum)&&(sum<=399)
44     System.out.println(getStudName()+":"+'B');
45   if(200<=sum)&&(sum<=299)
46     System.out.println(getStudName()+":"+'C');
47   if(sum<200)
48     System.out.println(getStudName()+":"+'E');
49 }
50 }

```

## Grade

Reviewed on Friday, 7 January 2022, 7:24 PM by Automatic grade

**Grade** 100 / 100

**Assessment report**

[\[+\]Grading and Feedback](#)

---

## 3. Query Data Set

**Grade settings:** Maximum grade: 100  
**Disable external file upload, paste and drop external content:** Yes  
**Run:** Yes **Evaluate:** Yes  
**Automatic grade:** Yes **Maximum execution time:** 32 s

## Query Data Set

Jackson is pursuing his Bachelor's degree in TekSpec University, Alaska. His professor has given him a weekend assignment to develop an application in java using inner class concept.

You being his best friend, help him in completing his weekend assignment. You need to implement a java program using inner class concept to display the details available in primaryDataSet, secondaryDataSet and Query.

**Requirement 1:** Display the details in primaryDataSet, secondaryDataSet and Query

### Component Specification: Query (Model Class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
	Query	String queryId String queryCategory DataSet primaryDataSet DataSet secondaryDataSet	Include getters and setter method for all the attributes.	
	DataSet	String theatreId String theatreName String location int noOfScreen double ticketCost	Include getters and setter method for all the attributes.	The DataSet class must be available only to Query class. So the DataSet class should come as a Inner class in Query class.
Display the details	Query	toString()		Override the toString method to produce the output as specified in the sample output

**Note:** The class and methods should be declared as public and all the attributes should be declared as private.

Create a class called **TestApplication** with the main method and get the inputs for primary data set and secondary data set like **theatreId**, **theatreName**, **location**, **noOfScreen** and **ticketCost**, and details of Query like **queryId** and **queryCategory** from the user.

Display the details of primary data set, secondary data set and Query as shown in the sample input and output.

**Note:**

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the remaining text represents the output.

Ensure to follow object oriented specifications provided in the question description.

Ensure to provide the names for classes, attributes and methods as specified in the question description.

Adhere to the code template, if provided.

**Sample Input / Output 1:**

Enter the Details for primary data set

Enter the theatre id

**PNR6001**

Enter the theatre name

**KV cinemas**

Enter the location

**Chennai**

Enter the no of screens

**8**

Enter the ticket cost

**120**

Enter the Details for secondary data set

Enter the theatre id

**RNV5001**

Enter the theatre name

**Inoxe**

Enter the location

**Bangalore**

Enter the no of screens

**5**

Enter the ticket cost

**150**

Enter the query id

**Q510**

Enter the query category

**DML**

Primary data set

Theatre id : PNR6001

Theatre name : KV cinemas

Location : Chennai

No of Screen : 8

Ticket Cost : 120

Secondary data set

Theatre id : RNV5001

Theatre name : Inoxe

Location : Bangalore

No of Screen : 5

Ticket Cost : 150

Query id : Q510

Query category : DML

---

## Automatic evaluation[\[+\]](#)

### Query.java

```
1  
2 //Write the required business logic as expected in the question description  
3  
4 public class Query {  
5     private String queryId;  
6     private String queryCategory;  
7     private DataSet primaryDataSet;  
8     private DataSet secondaryDataSet;  
9 }
```

```

10  @Override
11  public String toString()
12  {
13      String g="";
14      g+="Primary data set"+"\n";
15      g+="Theatre id :" +primaryDataSet.getTheatreId()+"\n";
16      g+="Theatre name :" +primaryDataSet.getTheatreName()+"\n";
17      g+="Location :" +primaryDataSet.getLocation()+"\n";
18      g+="No of Screen :" +primaryDataSet.getNoOfScreen()+"\n";
19      g+="Ticket Cost :" +primaryDataSet.getTicketCost()+"\n";
20
21      g+="Secondary data set"+"\n";
22      g+="Theatre id :" +secondaryDataSet.getTheatreId()+"\n";
23      g+="Theatre name :" +secondaryDataSet.getTheatreName()+"\n";
24      g+="Location :" +secondaryDataSet.getLocation()+"\n";
25      g+="No of Screen :" +secondaryDataSet.getNoOfScreen()+"\n";
26      g+="Ticket Cost :" +secondaryDataSet.getTicketCost()+"\n";
27      g+="Query id : "+queryId+ "\n";
28      g+="Query category : "+queryCategory+ "\n";
29
30      return g;
31  }
32  public class DataSet{
33      private String theatreId;
34      private String theatreName;
35      private String location;
36      private int noOfScreen;
37      private double ticketCost;
38
39      public double getTicketCost()
40      {
41          return ticketCost;
42      }
43      public void setTicketCost(double a)
44      {
45          ticketCost=a;
46      }
47
48      public int getNoOfScreen()
49      {
50          return noOfScreen;
51      }
52      public void setNoOfScreen(int a)
53      {
54          noOfScreen=a;
55      }
56      public String getLocation()
57      {
58          return location;
59      }
60      public void setLocation(String a)
61      {
62          location=a;
63      }
64      public String getTheatreName ()
65      {
66          return theatreName;
67      }
68      public void setTheatreName(String a)
69      {
70          theatreName=a;
71      }
72
73      public String getTheatreId()
74      {
75          return theatreId;
76      }

```

```

77     public void setTheatreId(String a)
78     {
79         theatreId=a;
80     }
81 }
82 public void setSecondaryDataSet(DataSet pD)
83 {
84     this.secondaryDataSet=pD;
85 }
86 public DataSet getSecondaryDataSet()
87 {
88     return this.secondaryDataSet;
89 }
90 public void setPrimaryDataSet(DataSet pD)
91 {
92     this.primaryDataSet=pD;
93 }
94 public DataSet getPrimaryDataSet()
95 {
96     return this.primaryDataSet;
97 }
98 public void setQueryId (String queryId)
99 {
100    this.queryId=queryId;
101 }
102 public void setQueryCategory(String queryCategory)
103 {
104    this.queryCategory=queryCategory;
105 }
106 public String getQueryId()
107 {
108     return this.queryId;
109 }
110 public String getQueryCategory()
111 {
112     return this.queryCategory;
113 }
114
115 }

```

## *TestApplication.java*

```

1 import java.util.*;
2 public class TestApplication {
3     //Write the required business logic as expected in the question description
4     public static void main (String[] args) {
5         Scanner sc= new Scanner (System.in);
6         Query q= new Query();
7         Query.DataSet pd= q.new DataSet();
8         Query.DataSet sd= q.new DataSet();
9         System.out.println("Enter the Details for primary data set");
10        System.out.println("Enter the theatre id");
11        pd.setTheatreId(sc.nextLine());
12        System.out.println("Enter the theatre name");
13        pd.setTheatreName(sc.nextLine());
14        System.out.println("Enter the location");
15        pd.setLocation(sc.nextLine());
16        System.out.println("Enter the no of screens");
17        pd.setNoOfScreen(sc.nextInt());
18        System.out.println("Enter the ticket cost");
19        pd.setTicketCost(sc.nextDouble());
20        System.out.println("Enter the Details for secondary data set");
21        System.out.println("Enter the theatre id");
22
23        String id2=sc.next();
24        //System.out.println(id2);
25        sd.setTheatreId(id2);
26        System.out.println("Enter the theatre name");

```

```

27     sc.nextLine();
28     sd.setTheatreName(sc.nextLine());
29     System.out.println("Enter the location");
30     String gll=sc.nextLine();
31     sd.setLocation(gll);
32     System.out.println("Enter the no of screens");
33
34 //System.out.println(gll);
35 //String pp=sc.nextLine();
36 //System.out.println(pp);
37
38     sd.setNoOfScreen(sc.nextInt());
39     System.out.println("Enter the ticket cost");
40     sd.setTicketCost(sc.nextDouble());
41     sc.nextLine();
42     System.out.println("Enter the query id");
43     q.setQueryId(sc.nextLine());
44     System.out.println("Enter the query category");
45     q.setQueryCategory(sc.nextLine());
46
47     q.setSecondaryDataSet(sd);
48     q.setPrimaryDataSet(pd);
49     System.out.println(q.toString());
50 }
51 }
```

## Grade

Reviewed on Friday, 17 December 2021, 6:59 PM by Automatic grade

**Grade** 100 / 100

**Assessment report**

[\[+\]Grading and Feedback](#)

---



---

## 4. Retrieve Flights Based on Source and Destination

**Grade settings:** Maximum grade: 100

**Disable external file upload, paste and drop external content:** Yes

**Run:** Yes **Evaluate:** Yes

**Automatic grade:** Yes **Maximum execution time:** 32 s

### Retrieve Flights Based on Source and Destination

Zaro Flight System wants to automate the process in their organization. The flight details are available in the database, the customer should have the facility to view flights which are from a particular source to destination.

You being their software consultant have been approached by them to develop an application which can be used for managing their business. You need to implement a java program to view all the flight based on source and destination.

**Component Specification: Flight (Model Class)**

Type	Class	Attribute	Methods	Responsibilitie
------	-------	-----------	---------	-----------------

	s		s
<b>Flight</b>	int flightId String source String destination int noOfSeats double flightFare	Include getters and setter method for all the attributes.  Include a five argument constructor in the given order – flightId, source, destination, noOfSeats and flightFare.	

**Note:** The class and methods should be declared as public and all the attributes should be declared as private.

**Requirement 1:** Retrieve all the flights with the given source and destination

The customer should have the facility to view flights which are from a particular source to destination. Hence the system should fetch all the flight details for the given source and destination from the database. Those flight details should be added to a ArrayList and return the same.

#### Component Specification: FlightManagementSystem

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Retrieve all the flights with the given source and destination	FlightManagement System		public ArrayList<Flight> viewFlightBySourceDestination(String source, String destination)	This method should accept a Source and a destination as parameter and retrieve all the flights with the given source and destination from the database. Return these details as ArrayList<Flight>.

**Note:** The class and methods should be declared as public and all the attributes should be declared as private.

The **flight** table is already created at the backend. The structure of flight table is:

<b>Column Name</b>	<b>Datatype</b>
flightId	int
source	varchar2(30)
destination	varchar2(30)
noofseats	int
flightfare	number(8,2)

Sample records available in **flight** table are:

<b>Flightid</b>	<b>Source</b>	<b>Destination</b>	<b>Noofseats</b>	<b>Flightfare</b>
18221	Malaysia	Singapore	50	5000
18222	Dubai	Kochi	25	50000
18223	Malaysia	Singapore	150	6000
18224	Malaysia	Singapore	100	7000

To connect to the database you are provided with **database.properties** file and **DB.java** file. (**Do not change any values in database.properties file**)

Create a class called **Main** with the main method and get the inputs like **source** and **destination** from the user.

Display the details of flight such as flightId, noofseats and flightfare for all the flights returned as `ArrayList<Flight>` from the method **viewFlightBySourceDestination** in **FlightManagementSystem** class.

If no flight is available in the list, the output should be “**No flights available for the given source and destination**”.

**Note:**

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the remaining text represents the output.

Ensure to follow object oriented specifications provided in the question description.  
Ensure to provide the names for classes, attributes and methods as specified in the question description.

Adhere to the code template, if provided.

**Sample Input / Output 1:**

Enter the source

**Malaysia**

Enter the destination

**Singapore**

Flightid Noofseats Flightfare

18221 50 5000.0

18223 150 6000.0

18224 100 7000.0

### Sample Input / Output 2:

Enter the source

**Malaysia**

Enter the destination

**Dubai**

No flights available for the given source and destination

---

### Automatic evaluation [+]

#### *Flight.java*

```
1  public class Flight {  
2      private int flightId;  
3      private String source;  
4      private String destination;  
5      private int noOfSeats;  
6      private double flightFare;  
7      public int getFlightId() {  
8          return flightId;  
9      }  
10     public void setFlightId(int flightId) {  
11         this.flightId = flightId;  
12     }  
13     public String getSource() {  
14         return source;  
15     }  
16     public void setSource(String source) {  
17         this.source = source;  
18     }  
19     public String getDestination() {  
20         return destination;  
21     }  
22     public void setDestination(String destination) {  
23         this.destination = destination;  
24     }  
25     public int getNoOfSeats() {  
26         return noOfSeats;  
27     }  
28     public void setNoOfSeats(int noOfSeats) {  
29         this.noOfSeats = noOfSeats;  
30     }  
31     public double getFlightFare() {  
32         return flightFare;  
33     }  
34     public void setFlightFare(double flightFare) {  
35         this.flightFare = flightFare;  
36     }  
37     public Flight(int flightId, String source, String destination,  
38                 int noOfSeats, double flightFare) {  
39 }
```

```

41         super();
42         this.flightId = flightId;
43         this.source = source;
44         this.destination = destination;
45         this.noOfSeats = noOfSeats;
46         this.flightFare = flightFare;
47     }
48
49
50
51 }
52

```

### *FlightManagementSystem.java*

```

1 import java.sql.Connection;
2 import java.sql.ResultSet;
3 import java.sql.SQLException;
4 import java.util.ArrayList;
5 import java.util.List;
6 import java.sql.PreparedStatement;
7 public class FlightManagementSystem {
8     public ArrayList <Flight> viewFlightBySourceDestination(String source, String destination){
9         Connection conn = null;
10        ResultSet Rs = null;
11        String sql = "select * from flight where source=? and destination=? order by flightid";
12        ArrayList<Flight> flight = new ArrayList<>();
13        try{
14            conn = DB.getConnection();
15            PreparedStatement ps = conn.prepareStatement(sql);
16
17            ps.setString(1, source);
18            ps.setString(2, destination);
19
20            Rs=ps.executeQuery();
21            while(Rs.next()){
22                Flight F = new Flight(Rs.getInt(1),source,destination,Rs.getInt(4),Rs.getInt(5));
23                flight.add(F);
24            }
25        }catch(ClassNotFoundException e){
26            e.printStackTrace();
27        }catch(SQLException e){
28            e.printStackTrace();
29        }
30
31        return flight;
32    }
33 }

```

### *Main.java*

```

1 import java.util.Comparator;
2 import java.util.Scanner;
3 import java.util.ArrayList;
4
5
6 public class Main{
7     public static void main(String[] args){
8         Scanner sc=new Scanner(System.in);
9         // fill your code here
10        System.out.println("Enter the source");
11        String source=sc.nextLine();
12        System.out.println("Enter the destination");
13        String destination=sc.nextLine();
14        ArrayList<Flight> flight = new
FlightManagementSystem().viewFlightBySourceDestination(source,destination);
15        if(flight.isEmpty())
16        {
17            System.out.println("No flights available for the given source and destination");

```

```

18
19     }
20     else
21     {
22         System.out.println("Flightid Noofseats Flightfare");
23         for(Flight f : flight)
24         {
25             System.out.println(f.getFlightId()+" "+f.getNoOfSeats()+" "+f.getFlightFare());
26         }
27     }
28
29
30 }
31}

```

## DB.java

```

1 import java.io.FileInputStream;
2 import java.io.IOException;
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6 import java.util.Properties;
7
8 public class DB {
9
10    private static Connection con = null;
11    private static Properties props = new Properties();
12
13
14 //ENSURE YOU DON'T CHANGE THE BELOW CODE WHEN YOU SUBMIT
15    public static Connection getConnection() throws ClassNotFoundException, SQLException {
16        try{
17
18            FileInputStream fis = null;
19            fis = new FileInputStream("database.properties");
20            props.load(fis);
21
22            // load the Driver Class
23            Class.forName(props.getProperty("DB_DRIVER_CLASS"));
24
25            // create the connection now
26            con =
27        DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),props.getProperty("DB_PASSWORD"));
28        }
29        catch(IOException e){
30            e.printStackTrace();
31        }
32        return con;
33    }
34}

```

## database.properties

```

1 #IF NEEDED, YOU CAN MODIFY THIS PROPERTY FILE
2 #ENSURE YOU ARE NOT CHANGING THE NAME OF THE PROPERTY
3 #YOU CAN CHANGE THE VALUE OF THE PROPERTY
4 #LOAD THE DETAILS OF DRIVER CLASS, URL, USERNAME AND PASSWORD IN DB.java using this
properties file only.
5 #Do not hard code the values in DB.java.
6
7 DB_DRIVER_CLASS=com.mysql.jdbc.Driver
8 DB_URL=jdbc:mysql://localhost:3306/${sys:DB_USERNAME}
9 DB_USERNAME=${sys:DB_USERNAME}
10 DB_PASSWORD=${sys:DB_USERNAME}
11

```

## Grade

Reviewed on Wednesday, 12 May 2021, 6:31 AM by Automatic grade

**Grade** 100 / 100

**Assessment report**

[\[+\]Grading and Feedback](#)

## CLUB MEMBER DETAILS

### ClubMember.java\*

```
public class ClubMember{  
    private int memberId;  
    private String memberName;  
    private String memberType;  
    private double membershipFees;  
  
    public void setMemberId(int memberId){  
        this.memberId = memberId;  
    }  
  
    public int getMemberId(){  
        return memberId;  
    }  
  
    public void setMemberName(String memberName){  
        this.memberName = memberName;  
    }  
  
    public String getMemberName(){  
        return memberName;  
    }  
  
    public void setMemberType(String memberType){  
        this.memberType = memberType;  
    }  
  
    public String getMemberType(){  
        return memberType;  
    }  
  
    public void setMembershipFees(double membershipFees){
```

```
this.membershipFees = membershipFees;  
}  
  
public double getMembershipFees(){  
    return membershipFees;  
}  
  
public ClubMember(int memberId, String memberName, String memberType){  
    this.memberId = memberId;  
    this.memberName = memberName;  
    this.memberType = memberType;  
}  
  
public void calculateMembershipFees (){  
    if (memberType.equals("Gold")) membershipFees = 50000.0;  
    else if (memberType.equals("Premium")) membershipFees = 75000.0;  
}  
}
```

### Main.java\*

```
import java.util.Scanner;  
  
public class Main{  
  
    public static void main (String[] args) {  
        Scanner sc=new Scanner(System.in);  
        System.out.println("Enter Member Id");  
        int memberId = sc.nextInt();  
        sc.nextLine();  
        System.out.println("Enter Name");  
        String memberName = sc.nextLine();
```

```
System.out.println("Enter Member Type");
String memberType = sc.next();

ClubMember clubMemberObj = new ClubMember(memberId,memberName,memberType);
clubMemberObj.calculateMembershipFees();

System.out.println("Member Id is " + clubMemberObj.getMemberId());
System.out.println("Member Name is " + clubMemberObj.getMemberName());
System.out.println("Member Type is " + clubMemberObj.getMemberType());
System.out.println("Membership Fees is " + clubMemberObj.getMembershipFees());
}

}
```

## CreditCardValidator

### CreditCard.java\*

```
package com.cts.entity;

public class CreditCard {

    private String number;

    public CreditCard() {

    }

    public CreditCard(String number) {
        super();
        this.number = number;
    }

    public String getNumber() {
        return number;
    }

    public void setNumber(String number) {
        this.number = number;
    }
}
```

### CreditCardService.java\*

```
package com.cts.services;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
```

```
import java.util.List;
import java.nio.file.*;

import com.cts.entity.CreditCard;

public class CreditCardService {

    //check whether the card is blocklisted and card contains only 16 digits

    public String validate(CreditCard card,String fileName) throws IOException
    {

        String msg=null;
        if(validateAgainstBlocklist(card, fileName))
        {
            msg="Card is blocked";
        }
        else if(validateNumber(card.getNumber()))
        {
            msg="card is not having 16 digits";
        }
        else
        {
            msg="valid card";
        }
        return msg;
    }

    // Validate a credit card against a blocklist.

    public boolean validateAgainstBlocklist(CreditCard card, String fileName) throws IOException {

        //write your code here
        boolean bol = true;
        String str = "";
        str = new String(Files.readAllBytes(Paths.get(fileName)));
        String dig[] = str.split(",");
        String str2 = dig[0];
        String str3 = dig[1];
        if(card.getNumber().equalsIgnoreCase(str2) || card.getNumber().equalsIgnoreCase(str3))

```

```
{  
    bol=true;  
}  
  
else{  
    bol=false;  
}  
  
}  
  
return bol;  
}  
  
// Validate the card number length  
public boolean validateNumber(String number) {  
    int len = number.length();  
    boolean bol=true;  
    if(len!=16)  
    {  
        bol=true;  
    }  
    else{  
        bol=false;  
    }  
  
    return bol;  
}  
  
// Get the blocklisted no's from the file and return list of numbers  
public List<String> getBlockListNumbers(String fileName) throws IOException {  
  
    List<String> li = new ArrayList<String>();  
    String data = "";  
    data = new String(Files.readAllBytes(Paths.get(fileName)));  
    String dig1[] = data.split(",");  
    for(int i=0;i<dig1.length;i++)  
    {  
        li.add(dig1[i]);  
    }  
}
```

```
    }

    return li;
}

}
```

### SkeletonValidator.java\*

```
package com.cts.skeletonvalidator;

import java.lang.reflect.Method;
import java.util.logging.Level;
import java.util.logging.Logger;

public class SkeletonValidator {

    public SkeletonValidator() {
        validateClassName("com.cts.entity.CreditCard");
        validateClassName("com.cts.services.CreditCardService");
        validateMethodSignature(
            "validate:String,validateAgainstBlocklist:boolean,validateNumber:boolean,getBlockListNumbers>List","com.cts.services.CreditCardService");
    }

    private static final Logger LOG = Logger.getLogger("SkeletonValidator");

    protected final boolean validateClassName(String className) {

        boolean iscorrect = false;
        try {
            Class.forName(className);
            iscorrect = true;
            LOG.info("Class Name " + className + " is correct");
        } catch (ClassNotFoundException e) {
```

```

LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use the correct package "
        + "and class name as provided in the skeleton");

} catch (Exception e) {

LOG.log(Level.SEVERE, "There is an error in validating the " + "Class Name. Please manually verify that the "
        + "Class name is same as skeleton before uploading");

}

return iscorrect;

}

protected final void validateMethodSignature(String methodWithExcptn, String className) {

Class cls = null;

try {

    String[] actualmethods = methodWithExcptn.split(",");
    boolean errorFlag = false;
    String[] methodSignature;
    String methodName = null;
    String returnType = null;

    for (String singleMethod : actualmethods) {
        boolean foundMethod = false;
        methodSignature = singleMethod.split(":");
        methodName = methodSignature[0];
        returnType = methodSignature[1];
        cls = Class.forName(className);
        Method[] methods = cls.getMethods();
        for (Method findMethod : methods) {
            if (methodName.equals(findMethod.getName())) {
                foundMethod = true;
                if (!(findMethod.getReturnType().getSimpleName().equals(returnType))) {
                    errorFlag = true;
                    LOG.log(Level.SEVERE, " You have changed the " + "return type in " + methodName

```

```

+ "" method. Please stick to the " + "skeleton provided");

} else {
    LOG.info("Method signature of " + methodName + " is valid");
}

}

if (!foundMethod) {
    errorFlag = true;
    LOG.log(Level.SEVERE, " Unable to find the given public method " + methodName
        + ". Do not change the " + "given public method name. " +
        "Verify it with the skeleton");
}

}

if (!errorFlag) {
    LOG.info("Method signature is valid");
}

}

} catch (Exception e) {
    LOG.log(Level.SEVERE," There is an error in validating the " + "method structure. Please
    manually verify that the "+ "Method signature is same as the skeleton before uploading");
}

}

}

```

### **CreditCardValidatorMain.java\***

```

package com.cts;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

```

```
import com.cts.entity.CreditCard;
import com.cts.services.CreditCardService;
import com.cts.skeletonvalidator.SkeletonValidator;

public class CreditCardValidatorMain {

    public static void main(String[] args) throws IOException {
        // CODE SKELETON - VALIDATION STARTS
        // DO NOT CHANGE THIS CODE
        BufferedReader b = new BufferedReader(new InputStreamReader(System.in));

        new SkeletonValidator();

        // CODE SKELETON - VALIDATION ENDS

        // Please start your code from here

        String cardNumber = b.readLine();
        CreditCard creditCard = new CreditCard();
        creditCard.setNumber(cardNumber);
        //Write your code here read card numnber and create CreditCard object based on cardnumber
        CreditCardService creditCardService = new CreditCardService();

        String validationMessage=creditCardService.validate(creditCard, "resources/blacklist.csv");
        System.out.println(validationMessage);
    }
}
```

## ESHOPPING

### Main.java\*

```
package com.cts.eshopping.main;

import com.cts.eshopping.orderservice.CartService;
import com.cts.eshopping.skeletonvalidator.SkeletonValidator;
import com.cts.eshopping.vo.OrderLineItem;

public class Main {

    public static void main(String ag[]) {

        // CODE SKELETON - VALIDATION STARTS
        // DO NOT CHANGE THIS CODE
        SkeletonValidator validator = new SkeletonValidator();
        // CODE SKELETON - VALIDATION ENDS

        // Please start your code from here

        OrderLineItem it1 = new OrderLineItem("AM33","Book",200,3);
        OrderLineItem it2 = new OrderLineItem("AM345","Watch",1000,2);
        CartService cs  = new CartService();

        OrderLineItem[] arr = {it1, it2};
        double amt = cs.calculateOrderTotalAmount(arr);
        System.out.println(cs.calculateDiscount(amt));

    }
}
```

**CartService.java\*/orderService**

```
package com.cts.eshopping.orderservice;

import com.cts.eshopping.vo.OrderLineItem;

/**
 *
 */
public class CartService {

    /**
     * Method to calculate total purchase amount for all the order line items
     *
     * @param orderLineItems
     * @return totalOrderAmount
     */
    public double calculateOrderTotalAmount(OrderLineItem[] orderLineItems) {

        double totalOrderAmount = 0;
        int qt = 0;
        double cost = 0.0;

        for(int i=0;i<orderLineItems.length;i++){
            qt = orderLineItems[i].quantity;
            cost = orderLineItems[i].itemCostPerQuantity;
            totalOrderAmount += (qt*cost);
        }

        return totalOrderAmount; // TODO change this return value
    }

    /**
     * Method to calculate discount based on order total amount
     *
     * @param totalOrderAmount
     */
}
```

```

* @return discount
*/
public double calculateDiscount(double totalOrderAmount) {
    double discount = 0.0;

    if(totalOrderAmount<1000){
        discount = (totalOrderAmount*10)/100;
    }
    else if(totalOrderAmount>=1000 && totalOrderAmount<10000){
        discount = (totalOrderAmount*20)/100;
    }
    else if(totalOrderAmount>=10000){
        discount = (totalOrderAmount*30)/100;
    }

    return discount; // TODO change this return value
}

/***
 * Method to verify if the order line item is flagged as Bulk Order or not
 *
 * @param linelitem
 * @return boolean
 */
public boolean isBulkOrder(OrderLineItem linelitem) {
    boolean result=false;

    if(linelitem.quantity>5){
        result = true;
    }
    else if(linelitem.quantity<=5 && linelitem.quantity>=1){
        result=false;
    }

    return result; // TODO change this return value
}

```

```

/**
 * Count the number of line items which are ordered in bulk
 *
 * @param orderLineItems
 * @return
 */
public int countOfBulkOrderLineItems(OrderLineItem[] orderLineItems) {
    int count = 0;

    for(int i=0;i<orderLineItems.length;i++){
        if(isBulkOrder(orderLineItems[i])){
            count++;
        }
    }

    return count; // TODO change this return value
}

}

```

### **SkeletonValidator.java\***

```

package com.cts.eshopping.skeletonvalidator;

import java.lang.reflect.Method;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author 222805
 *
 * This class is used to verify if the Code Skeleton is intact and not modified by participants thereby ensuring smooth
 * auto evaluation
 *
 */

```

```
public class SkeletonValidator {  
  
    public SkeletonValidator() {  
        validateClassName("com.cts.eshopping.orderservice.CartService");  
        validateClassName("com.cts.eshopping.vo.OrderLineItem");  
        validateMethodSignature(  
            "calculateOrderTotalAmount:double,calculateDiscount:double,isBulkOrder:boolean,countOfBulkOrderLineItems:int",  
            "com.cts.eshopping.orderservice.CartService");  
  
    }  
  
    private static final Logger LOG = Logger.getLogger("SkeletonValidator");  
  
    protected final boolean validateClassName(String className) {  
  
        boolean iscorrect = false;  
        try {  
            Class.forName(className);  
            iscorrect = true;  
            LOG.info("Class Name " + className + " is correct");  
  
        } catch (ClassNotFoundException e) {  
            LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use the correct package "  
                + "and class name as provided in the skeleton");  
  
        } catch (Exception e) {  
            LOG.log(Level.SEVERE,"There is an error in validating the " + "Class Name. Please manually  
            verify that the " + "Class name is same as skeleton before uploading");  
        }  
        return iscorrect;  
    }  
}
```

```

protected final void validateMethodSignature(String methodWithExcptn, String className) {

    Class cls = null;

    try {

        String[] actualmethods = methodWithExcptn.split(",");
        boolean errorFlag = false;
        String[] methodSignature;
        String methodName = null;
        String returnType = null;

        for (String singleMethod : actualmethods) {

            boolean foundMethod = false;
            methodSignature = singleMethod.split(":");

            methodName = methodSignature[0];
            returnType = methodSignature[1];
            cls = Class.forName(className);
            Method[] methods = cls.getMethods();
            for (Method findMethod : methods) {
                if (methodName.equals(findMethod.getName())) {
                    foundMethod = true;
                    if (!(findMethod.getReturnType().getName().equals(returnType))) {
                        errorFlag = true;
                        LOG.log(Level.SEVERE, " You have changed the " + "return type in '" + methodName
                                + "' method. Please stick to the " + "skeleton provided");
                }
            }
            if (!foundMethod) {
                errorFlag = true;
            }
        }
    }
}

```

```

        LOG.log(Level.SEVERE, " Unable to find the given public method " + methodName
+ ". Do not change the " + "given public method name. " + "Verify it with the skeleton");
    }

}

if (!errorFlag) {
    LOG.info("Method signature is valid");
}

}

} catch (Exception e) {
    LOG.log(Level.SEVERE," There is an error in validating the " + "method structure. Please
manually verify that the "+ "Method signature is same as the skeleton before uploading");
}

}

}

```

### **OrderLineItem.java\***

```

package com.cts.eshopping.vo;

/**
 * @author Value Object - OrderLineItem
 *
 */
public class OrderLineItem {

    public String itemId;
    public String itemName;
    public double itemCostPerQuantity;
    public int quantity;

    public String getItemId(){
        return itemId;
    }
}

```

```
public void setItemId(String itemId){  
    this.itemId = itemId;  
}  
  
public String getItemName(){  
    return itemName;  
}  
  
public void setItemName(String itemName){  
    this.itemName = itemName;  
}  
  
public double getItemCostPerQuantity(){  
    return itemCostPerQuantity;  
}  
  
public void setItemCostPerQuantity(double itemCostPerQuantity){  
    this.itemCostPerQuantity = itemCostPerQuantity;  
}  
  
public int getQuantity(){  
    return quantity;  
}  
  
public void setItemId(int quantity){  
    this.quantity = quantity;  
}  
  
public OrderLineItem(String itemId, String itemName, double itemCostPerQuantity, int quantity){  
    this.itemId = itemId;  
    this.itemName = itemName;  
    this.itemCostPerQuantity=itemCostPerQuantity;  
    this.quantity = quantity;  
}  
}
```

## Fixed Deposit Details

### FDScheme.java\*

```
import java.util.*;  
  
class FDScheme{  
  
    private int schemeNo;  
  
    private double depositAmt;  
  
    private int period;  
  
    private float rate;  
  
    public FDScheme(int schemeNo, double depositAmt, int period){  
  
        super();  
  
        this.schemeNo=schemeNo;  
  
        this.depositAmt=depositAmt;  
  
        this.period=period;  
  
        calculateInterestRate();  
  
    }  
  
    public int getSchemeNo(){  
  
        return schemeNo;  
  
    }  
  
    public void setSchemeNo(int schemeNo)  
  
    {  
  
        this.schemeNo=schemeNo;  
  
    }  
  
    public double getDepositAmt(){  
  
        return depositAmt;  
  
    }  
  
    public void setDepositAmt(double depositAmt)  
  
    {  
  
        this.depositAmt=depositAmt;  
  
    }  
  
    public int getPeriod()  
  
    {  
  
        return period;  
  
    }  
  
    public void setPeriod(int period){
```

```

    this.period=period;
}

public float getRate(){
    return rate;
}

public void setRate(float rate){
    this.rate=rate;
}

public void calculateInterestRate()
{
    if(period>=1 && period<=90)
    {
        this.rate=(float)5.5;
    }
    else if(period>=91 && period<=180)
    {
        this.rate=(float)6.25;
    }
    else if(period>=181 && period<=365)
    {
        this.rate=(float)7.5;
    }
    System.out.println("Interest rate for"+period+"days is"+this.rate);
}
}

```

### Main.java\*

```

import java.util.Scanner;

public class Main{

    public static void main (String[] args) {
        Scanner sc=new Scanner(System.in);

```

```
System.out.println("Enter Scheme no");
int no=sc.nextInt();
sc.nextLine();
System.out.println("Enter Deposit amount");
double amt=sc.nextDouble();
System.out.println("enter period of deposit");
int prd=sc.nextInt();
FDScheme obj=new
FDScheme(no,amt,prd);
}
}
```

## GPA CALCULATION

### UserInterface.java\*

```
package com.ui;

import com.utility.*;
import java.util.*;

public class UserInterface {

    public static void main(String []args)
    {

        GPACalculator gpa = new GPACalculator();

        gpa.setGradePointList(new ArrayList<Integer>());

        int option=0;

        double gpa1=0;

        Scanner sc = new Scanner(System.in);

        do

        {

            System.out.println("1. Add Grade\n2. Calculate GPA\n3. Exit");

            System.out.println("Enter your choice");

            option = Integer.valueOf(sc.nextLine());

            switch(option)

            {

                case 1: System.out.println("Enter the obtained grade");

                    char grade = sc.nextLine().charAt(0);

                    gpa.addGradePoint(grade);

                    break;

                case 2 : gpa1 = gpa.calculateGPAScored();

                    if(gpa1 > 0)

                    {

                        System.out.println("GPA Scored");

                        System.out.println(gpa1);

                    }

                    else

                    {

                        System.out.println("No GradePoints available");
                    }
            }
        }
    }
}
```

```

        }

        break;

    case 3 : break;

}

}while(option!=3);

System.out.println("Thank you for using the Application");

}

}

```

### **GPACalculator.java\***

```

package com.utility;

import java.util.*;

public class GPACalculator {

    private List<Integer> gradePointList;

    public List<Integer> getGradePointList() {
        return gradePointList;
    }

    public void setGradePointList(List<Integer> gradePointList) {
        this.gradePointList = gradePointList;
    }

    /*This method should add equivalent grade points based on the grade obtained by the student passed as argument into gradePointList

```

Grade	S	A	B	C	D	E
Grade Point	10	9	8	7	6	5

For example if the grade obtained is A, its equivalent grade points is 9 has to be added into the gradePointList\*/

```

public void addGradePoint(char gradeObtained) {

    if(gradeObtained == 'S')
    {
        gradePointList.add(10);
    }
    else if(gradeObtained == 'A')
    {
        gradePointList.add(9);
    }
    else if(gradeObtained == 'B')
    {
        gradePointList.add(8);
    }
    else if(gradeObtained == 'C')
    {
        gradePointList.add(7);
    }
    else if(gradeObtained == 'D')
    {
        gradePointList.add(6);
    }
    else
    {
        gradePointList.add(5);
    }
}

/* This method should return the GPA of all grades scored in the semester
GPA can be calculated based on the following formula
GPA= (gradePoint1 + gradePoint2 + ... + gradePointN) / (size of List)

```

For Example:

if the list contains the following marks [9,10,8,5]

GPA =  $(9 + 10 + 8 + 5) / (4)$ = 8.0 \*/

```
public double calculateGPAScored() {  
  
    double gpa=-1;  
    double total=0,value=0,size=0;  
  
    size = gradePointList.size();  
    if(size < 1)  
    {  
        return 0;  
    }  
    // fill the code  
    Iterator i = gradePointList.iterator();  
    while(i.hasNext())  
    {  
        value = (Integer)i.next();  
        total += value;  
    }  
    gpa = total/size;  
  
    return gpa;  
}  
}
```

## HUNGER EATS

### FoodProduct.java\*

```
package com.bean;

public class FoodProduct {

    private int foodId;
    private String foodName;
    private double costPerUnit;
    private int quantity;

    public int getFoodId() {
        return foodId;
    }

    public void setFoodId(int foodId) {
        this.foodId = foodId;
    }

    public String getFoodName() {
        return foodName;
    }

    public void setFoodName(String foodName) {
        this.foodName = foodName;
    }

    public double getCostPerUnit() {
        return costPerUnit;
    }

    public void setCostPerUnit(double costPerUnit) {
        this.costPerUnit = costPerUnit;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
}
```

```
}
```

```
}
```

### **UserInterface.java\***

```
package com.ui;

import java.util.Scanner;

import com.utility.Order;

import com.bean.FoodProduct;

public class UserInterface {

    public static void main(String[] args) {

        // fill the code
        Scanner sc=new Scanner(System.in);
        int itemno;
        String bank;
        System.out.println("Enter the number of items");
        itemno=sc.nextInt();
        System.out.println("Enter the item details");
        Order z=new Order();
        for(int i=0;i<itemno;i++){
            FoodProduct fd=new FoodProduct();
            System.out.println("Enter the item id");
            fd.setFoodId(sc.nextInt());
            System.out.println("Enter the item name");
            fd.setFoodName(sc.next());
            System.out.println("Enter the cost per unit");
            fd.setCostPerUnit(sc.nextDouble());
            System.out.println("Enter the quantity");
            fd.setQuantity(sc.nextInt());
            z.addToCart(fd);
        }
    }
}
```

```
        System.out.println("Enter the bank name to avail offer");
        bank=sc.next();
        z.findDiscount(bank);
        System.out.println("Calculated Bill Amount:"+z.calculateTotalBill());
    }

}
```

### **Order.java\***

```
package com.utility;

import java.util.*;
import com.bean.FoodProduct;

public class Order {

    private double discountPercentage;
    private List<FoodProduct> foodList=new ArrayList<FoodProduct>();

    public double getDiscountPercentage() {
        return discountPercentage;
    }

    public void setDiscountPercentage(double discountPercentage) {
        this.discountPercentage = discountPercentage;
    }

    public List<FoodProduct> getFoodList() {
        return foodList;
    }

    public void setFoodList(List<FoodProduct> foodList) {
        this.foodList = foodList;
    }
}
```

```
//This method should set the discount percentage based on bank passed as argument
```

```
public void findDiscount(String bankName) {
```

```
    // fill the code
```

```
    if(bankName.equals("HDFC")){
```

```
        discountPercentage=15.0;
```

```
    }
```

```
    else if(bankName.equals("ICICI")){
```

```
        discountPercentage=25.0;
```

```
    }
```

```
    else if(bankName.equals("CUB")){
```

```
        discountPercentage=30.0;
```

```
    }
```

```
    else if(bankName.equals("SBI")){
```

```
        discountPercentage=50.0;
```

```
    }
```

```
    else if(bankName.equals("OTHERS")){
```

```
        discountPercentage=0.0;
```

```
    }
```

```
}
```

```
//This method should add the FoodProduct Object into Food List
```

```
public void addToCart(FoodProduct foodProductObject) {
```

```
    // fill the code
```

```
    List<FoodProduct> f=getFoodList();
```

```
    f.add(foodProductObject);
```

```
    setFoodList(f);
```

```
}
```

```
//method should return the total bill amount after discount
```

```
// based on the bank name  
public double calculateTotalBill() {  
  
    // fill the code  
    double bill=0;  
    List<FoodProduct> f=getFoodList();  
    for(int i=0;i<f.size();i++){  
        //  
        // System.out.println(f.get(i).getCostPerUnit());  
        //  
        // System.out.println(f.get(i).getQuantity());  
        bill+=f.get(i).getQuantity()*f.get(i).getCostPerUnit()*1.0;  
    }  
    bill=bill-((bill*discountPercentage)/100);  
  
    return bill;  
}  
  
}
```

## INSURANCE PREMIUM GENERATOR

### PropertyDetails.java\*

```
package com.cts.insurance.entity;

public class PropertyDetails {

    private Integer builtUpArea;
    private Integer builtYear;
    private Integer reconstructionCost;
    private Integer householdValuation;
    private String burglaryCoverReqd;
    private String politicalUnrestCoverReqd;
    private Integer sumAssured;

    public PropertyDetails() {

    }

    public Integer getBuiltUpArea() {
        return builtUpArea;
    }

    public void setBuiltUpArea(Integer builtUpArea) {
        this.builtUpArea = builtUpArea;
    }

    public Integer getBuiltYear() {
        return builtYear;
    }

    public void setBuiltYear(Integer builtYear) {
        this.builtYear = builtYear;
    }

    public Integer getReconstructionCost() {
```

```
    return reconstructionCost;
}

public void setReconstructionCost(Integer reconstructionCost) {
    this.reconstructionCost = reconstructionCost;
}

public Integer getHouseholdValuation() {
    return householdValuation;
}

public void setHouseholdValuation(Integer householdValuation) {
    this.householdValuation = householdValuation;
}

public String getBurglaryCoverReqd() {
    return burglaryCoverReqd;
}

public void setBurglaryCoverReqd(String burglaryCoverReqd) {
    this.burglaryCoverReqd = burglaryCoverReqd;
}

public String getPoliticalUnrestCoverReqd() {
    return politicalUnrestCoverReqd;
}

public void setPoliticalUnrestCoverReqd(String politicalUnrestCoverReqd) {
    this.politicalUnrestCoverReqd = politicalUnrestCoverReqd;
}

public Integer getSumAssured() {
    return sumAssured;
}
```

```

public void setSumAssured(Integer sumAssured) {
    this.sumAssured = sumAssured;
}

public PropertyDetails(Integer builtUpArea,Integer builtYear, Integer reconstructionCost, Integer
householdValuation,
String burglaryCoverReqd, String politicalUnrestCoverReqd) {
super();
this.builtUpArea = builtUpArea;
this.builtYear=builtYear;
this.reconstructionCost = reconstructionCost;
this.householdValuation = householdValuation;
this.burglaryCoverReqd = burglaryCoverReqd;
this.politicalUnrestCoverReqd = politicalUnrestCoverReqd;
}

}


```

#### **Constants.java\***

```

package com.cts.insurance.misc;

public class Constants {
    public final static String YES = "Yes";
    public final static String NO = "No";
    public final static double MIN_PREMIUM_AMOUNT = 5000;
    public final static int MIN_HOUSEHOLD_VALUATION=0;
}

```

#### **CalculatePremiumService.java\***

```

package com.cts.insurance.services;

```

```

import com.cts.insurance.entity.PropertyDetails;
import com.cts.insurance.misc.Constants;

import java.time.LocalDate;

public class CalculatePremiumService {

    public boolean checkOwnerDetails(String name, String mobile) {
        //name cannot have numbers or special characters; minimum length of name=2
        //mobile number begins with any digit between 6 and 9; length=10
        return name.matches("^([a-zA-Z]{2,})$") && mobile.matches("^[6-9][0-9]{9}$");
    }

    public double getPremiumAmount(PropertyDetails propertyDetails) {
        double amountToBePaid = 0;
        double additionalAmount1 = 0;
        double additionalAmount2 = 0;
        /*invoke validatePropertyParameters(propertyDetails) and check the response
         * if true ,calculate premium amount to be paid by calling
         * the methods calculatePremiumByPropertyAge(propertyDetails),
         * calculatePremiumForBurglaryCoverage(propertyDetails, amountToBePaid) and
         * calculatePremiumForPoliticalUnrestCoverage(propertyDetails, amountToBePaid)
         *
         * return the premium amount rounded off to zero decimal places
         * else return 0;
        */
        if(!validatePropertyParameters(propertyDetails)) {
            return 0;
        }

        amountToBePaid = calculatePremiumByPropertyAge(propertyDetails);
        additionalAmount1 = calculatePremiumForBurglaryCoverage(propertyDetails, amountToBePaid);
        additionalAmount2 = calculatePremiumForPoliticalUnrestCoverage(propertyDetails,
amountToBePaid);

        return Math.round(amountToBePaid + additionalAmount1 + additionalAmount2);
    }
}

```

```

}

public boolean validatePropertyParameters(PropertyDetails propertyDetails) {
    /*
     * conditions to be checked
     * builtUpArea between 400 and 15,000 sq. ft.
     * reconstructionCost between Rs.1,000 and Rs.10,000
     * householdValuation either same as Constants.MIN_HOUSEHOLD_VALUATION
     * between Rs.1,00,000 and Rs.15,00,000
     * builtYear between 2000 and current year
     */
    int builtUpArea = propertyDetails.getBuiltUpArea();
    if(!(builtUpArea>=400 && builtUpArea<=15000)) return false;
    int reconstructionCost = propertyDetails.getReconstructionCost();
    if(!(reconstructionCost>=1000 && reconstructionCost<=10000)) return false;
    int householdValuation = propertyDetails.getHouseholdValuation();
    if(!((householdValuation==Constants.MIN_HOUSEHOLD_VALUATION) || (householdValuation >= 100000 && householdValuation <= 1500000))) {
        return false;
    }
    int builtYear = propertyDetails.getBuiltYear();
    if(!(builtYear>=2000 && builtYear<=LocalDate.now().getYear())) {
        return false;
    }
    return true;
}

public double calculatePremiumByPropertyAge(PropertyDetails propertyDetails) {
    //Write your code here based on business rules
    //Use Constants.MIN_PREMIUM_AMOUNT
    int sumAssured =
propertyDetails.getBuiltUpArea()*propertyDetails.getReconstructionCost()+propertyDetails.getHouseholdValuation());
    int propertyAge = LocalDate.now().getYear()-propertyDetails.getBuiltYear();
    propertyDetails.setSumAssured(sumAssured);
    double premium = 0;
    if(propertyAge>15) {

```

```

        premium = Constants.MIN_PREMIUM_AMOUNT+(propertyDetails.getSumAssured()*0.35);

    }

    else if(propertyAge>=6) {

        premium = Constants.MIN_PREMIUM_AMOUNT+(propertyDetails.getSumAssured()*0.2);

    }

    else {

        premium = Constants.MIN_PREMIUM_AMOUNT+(propertyDetails.getSumAssured()*0.1);

    }

    return premium;

}

public double calculatePremiumForBurglaryCoverage(PropertyDetails propertyDetails, double amount) {

    //write your code here based on business rules

    if(propertyDetails.getBurglaryCoverReqd().equalsIgnoreCase(Constants.YES)) {

        return amount*.01;

    }

    return 0;

}

public double calculatePremiumForPoliticalUnrestCoverage(PropertyDetails propertyDetails, double amount) {

    //Write your code here based on business rules

    //Ex:-propertyDetails.getPoliticalUnrestCoverReqd().equalsIgnoreCase(Constants.YES) to check
condition

    if(propertyDetails.getPoliticalUnrestCoverReqd().equalsIgnoreCase(Constants.YES)) {

        return amount*.01;

    }

    return 0;

}

}

```

**SkeletonValidator.java\***

```
package com.cts.insurance.skeleton;

import java.lang.reflect.Method;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author
 *
 * This class is used to verify if the Code Skeleton is intact and not modified by participants thereby ensuring smooth
auto evaluation
*
*/
public class SkeletonValidator {

    public SkeletonValidator() {
        validateClassName("com.cts.insurance.entity.PropertyDetails");
        validateClassName("com.cts.insurance.misc.Constants");
        validateClassName("com.cts.insurance.services.CalculatePremiumService");
        validateClassName("com.cts.insurance.InsurancePremiumGeneratorApp");
        validateMethodSignature(
            "checkOwnerDetails:boolean,getPremiumAmount:double,validatePropertyParameters:boolean,calculatePremiumByPropertyAge:double,calculatePremiumForBurglaryCoverage:double,calculatePremiumForPoliticalUnrestCoverage:double",
            "com.cts.insurance.services.CalculatePremiumService");
    }

    private static final Logger LOG = Logger.getLogger("SkeletonValidator");

    protected final boolean validateClassName(String className) {
        boolean iscorrect = false;
        try {
            Class.forName(className);
            iscorrect = true;
        }
        catch (Exception e) {
            iscorrect = false;
        }
        return iscorrect;
    }
}
```

```

        LOG.info("Class Name " + className + " is correct");

    } catch (ClassNotFoundException e) {
        LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use the
correct package "+ "and class name as provided in the skeleton");

    } catch (Exception e) {
        LOG.log(Level.SEVERE,
                "There is an error in validating the " + "Class Name. Please manually verify that the "
                + "Class name is same as skeleton before uploading");
    }

    return iscorrect;
}

protected final void validateMethodSignature(String methodWithExcptn, String className) {
    Class cls = null;
    try {

        String[] actualmethods = methodWithExcptn.split(",");
        boolean errorFlag = false;
        String[] methodSignature;
        String methodName = null;
        String returnType = null;

        for (String singleMethod : actualmethods) {
            boolean foundMethod = false;
            methodSignature = singleMethod.split(":");
            methodName = methodSignature[0];
            returnType = methodSignature[1];
            cls = Class.forName(className);
            Method[] methods = cls.getMethods();
            for (Method findMethod : methods) {

```

```

        if (methodName.equals(findMethod.getName())) {
            foundMethod = true;
            if (!(findMethod.getReturnType().getSimpleClassName().equals(returnType))) {
                errorFlag = true;
                LOG.log(Level.SEVERE, " You have changed the " + "return type in " + methodName+ "" " method. Please stick to the " + "skeleton provided");
            }
        } else {
            LOG.info("Method signature of " + methodName + " is valid");
        }
    }

    if (!foundMethod) {
        errorFlag = true;
        LOG.log(Level.SEVERE, " Unable to find the given public method " + methodName + ". Do not change the " + "given public method name. " + "Verify it with the skeleton");
    }
}

if (!errorFlag) {
    LOG.info("Method signature is valid");
}
}

} catch (Exception e) {
    LOG.log(Level.SEVERE," There is an error in validating the " + "method structure. Please manually verify that the "+ "Method signature is same as the skeleton before uploading");
}

}
}

```

**InsurancePremiumGeneratorApp.java\***

```
package com.cts.insurance;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

import com.cts.insurance.entity.PropertyDetails;
import com.cts.insurance.misc.Constants;
import com.cts.insurance.services.CalculatePremiumService;
import com.cts.insurance.skeleton.SkeletonValidator;

public class InsurancePremiumGeneratorApp {

    public static void main(String[] args) throws IOException {

        // CODE SKELETON - VALIDATION STARTS
        // DO NOT CHANGE THIS CODE

        SkeletonValidator validator = new SkeletonValidator();

        // CODE SKELETON - VALIDATION ENDS

        // Please start your code from here
        String name = "";
        String mobile = "";
        Integer builtUpArea = 0;
        Integer builtYear=0;
        Integer reconstructionCost = 0;
        Integer householdValuation = Constants.MIN_HOUSEHOLD_VALUATION;
        String burglaryCoverReqd = "";
        String politicalUnrestCoverReqd = "";

        //writer the code for creating BufferedReader object here
```

```

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
CalculatePremiumService premiumService = new CalculatePremiumService();

System.out.println("Enter the name");
//read name
name = br.readLine();

System.out.println("Enter the mobile");
//read mobile
mobile = br.readLine();

//validate name and mobile format; continue if true
if(premiumService.checkOwnerDetails(name, mobile)) {

System.out.println("Enter the Built-Up Area(In sq.ft.)between 400 and 15,000");
//read builtUpArea
builtUpArea = Integer.parseInt(br.readLine());

System.out.println("Enter the year the house was built");
//read builtYear
builtYear = Integer.parseInt(br.readLine());

System.out.println("Enter the Re-construction cost(per sq.ft.)between 1,000 and 10,000");
//read reconstructionCost
reconstructionCost = Integer.parseInt(br.readLine());

System.out.println(
"Do you want to include valuation of HouseHold Articles? Please provide yes/no");

//read response
String response = br.readLine();

//if (response is "yes" case insensitive)
if(response.equalsIgnoreCase("yes")) {

System.out.println("Enter the Household valuation between Rs.1,00,000 and Rs.15,00,000");
//read householdValuation
householdValuation = Integer.parseInt(br.readLine());
}

System.out.println("Do you want to include Burglary cover? Please provide yes/no");

```

```
//read burglaryCoverReqd  
burglaryCoverReqd = br.readLine();  
  
System.out.println("Do you want to include Political unrest cover? Please provide yes/no");  
  
//read politicalUnrestCoverReqd  
politicalUnrestCoverReqd = br.readLine();  
  
//create PropertyDetails Object  
  
PropertyDetails propertyDetails = new PropertyDetails(builtUpArea, builtYear, reconstructionCost,  
householdValuation, burglaryCoverReqd, politicalUnrestCoverReqd);  
  
double premiumAmount = premiumService.getPremiumAmount(propertyDetails);  
  
if(premiumAmount==0.0) {  
  
    System.out.println("Incorrect figures provided");  
  
}else {  
  
    System.out.println("Sum Insured: Rs." +propertyDetails.getSumAssured()+"\nInsurance  
Premium for the property of " + name + ": Rs." + premiumAmount);  
  
}  
  
}  
  
}
```

## NUMEROLOGY NUMBER

### Main.java\*

```
import java.util.Scanner;

public class Main {
    private static int getSum(long num) {
        char[] chars = Long.toString(num).toCharArray();
        int sum = 0;

        for (char ch : chars) {
            sum += Character.digit(ch, 10);
        }

        return sum;
    }

    private static int getNumerology(long num) {
        String string = String.valueOf(num);

        while (string.length() != 1) {
            string = String.valueOf(getSum(Long.parseLong(string)));
        }

        return Integer.parseInt(string);
    }

    private static int getOddCount(long num) {
        int oddCount = 0;

        for (char ch : Long.toString(num).toCharArray()) {
            if (Character.digit(ch, 10) % 2 != 0) {
                ++oddCount;
            }
        }
    }
}
```

```
        return oddCount;
    }

private static int getEvenCount(long num) {
    int evenCount = 0;

    for (char ch : Long.toString(num).toCharArray()) {
        if (Character.digit(ch, 10) % 2 == 0) {
            ++evenCount;
        }
    }

    return evenCount;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.println("Enter the number");
    long num = scanner.nextLong();

    System.out.println("Sum of digits");
    System.out.println(getSum(num));

    System.out.println("Numerology number");
    System.out.println(getNumerology(num));

    System.out.println("Number of odd numbers");
    System.out.println(getOddCount(num));

    System.out.println("Number of even numbers");
    System.out.println(getEvenCount(num));
}
```