# 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<br><br>long contactNumber<br><br>String emailId<br><br>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

```java
public class Customer {
	private String customerName;

	private long contactNumber;

	private String emailId;

	private int age;

	public String getCustomerName() {
		return customerName;
	}

	public void setCustomerName(String customerName) {
		this.customerName = customerName;
	}

	public long getContactNumber() {
		return contactNumber;
	}

	public void setContactNumber(long contactNumber) {
		this.contactNumber = contactNumber;
	}

	public String getEmailId() {
		return emailId;
	}

	public void setEmailId(String emailId) {
		this.emailId = emailId;
	}

	public int getAge() {
		return age;
	}

	public void setAge(int age) {
		this.age = age;
	}



}

```

### Main.java

```java
import java.util.Scanner;

public class Main {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
      Scanner sc=new Scanner(System.in);
      Customer c=new Customer();
      System.out.println("Enter the Name:");
       String name=(sc.nextLine());
       System.out.println("Enter the ContactNumber:");
       long no=sc.nextLong();
```

```
        sc.nextLine();
        System.out.println("Enter the EmailId:");
        String mail=sc.nextLine();

        System.out.println("Enter the Age:");
        int age=sc.nextInt();
        c.setCustomerName(name);
        c.setContactNumber(no);
        c.setEmailId(mail);
        c.setAge(age);
        System.out.println("Name:"+c.getCustomerName());
        System.out.println("ContactNumber:"+c.getContactNumber());
        System.out.println("EmailId:"+c.getEmailId());
        System.out.println("Age:"+c.getAge());



            }

}
```

## 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)

| Component Name | Type(Class) | Attributes | Methods | Responsibilities |
|---|---|---|---|---|
| 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)

| Component Name | Type(Class) | Attributes | Methods | Responsibilities |
|---|---|---|---|---|
| | Cheque | String chequeNo | Include a public getter and setter | |

|  |  |  |  |  |
|---|---|---|---|---|
|  |  | int chequeAmount<br><br>Date dateOfIssue | 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<br><br>String cardType | Include a public getter and setter method for all the attributes. |  |

|  |  |  |  |  |
|---|---|---|---|---|
|  |  | int creditCardAmo unt |  |  |
| Make payment for EMI amount | Credi t |  | 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

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;
public class Main {

        public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the due amount:");
        int dueAmount=sc.nextInt();

        System.out.println("Enter the mode of payment(cheque/cash/credit):");
        String mode=sc.next();
        Bill b = new Bill();
        if(mode.equals("cheque"))
        {
                System.out.println("enter the cheque number:");
                String chequeNumber=sc.next();
                System.out.println("enter the cheque amount:");
                int chequeAmount=sc.nextInt();
                System.out.println("enter the date of issue:");
                String date=sc.next();
                SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");
                Date dateOfIssue=null;
                try
                {
                    dateOfIssue = dateFormat.parse(date);
                }
                catch (ParseException e)
                {

                }
                Cheque cheque= new Cheque();
                cheque.setChequeNo(chequeNumber);
                cheque.setChequeAmount(chequeAmount);
                cheque.setDateOfIssue(dateOfIssue);
                cheque.setDueAmount(dueAmount);
                System.out.println(b.processPayment(cheque));
        }
        else if(mode.equals("cash"))
        {
                System.out.println("enter the cash amount:");
                int CashAmount=sc.nextInt();
                Cash cash=new Cash();
                cash.setCashAmount(CashAmount);
                cash.setDueAmount(dueAmount);
                System.out.println(b.processPayment(cash));
        }
        else if(mode.equals("credit"))
        {
                System.out.println("enter the credit card number:");
                int creditCardNumber=sc.nextInt();
                System.out.println("enter the card type:");
                String cardType=sc.next();

                Credit credit=new Credit();
                credit.setCreditCardNo(creditCardNumber);
                credit.setCardType(cardType);
                credit.setDueAmount(dueAmount);
                System.out.println(b.processPayment(credit));
        }
}
}
```

## Payment.java

```
public class Payment {
    private int dueAmount;

        public boolean payAmount()
        {
                if(dueAmount == 0)
                        return true;
                else
                        return false;
        }

        public int getDueAmount() {
                return dueAmount;
        }

        public void setDueAmount(int dueAmount) {
                this.dueAmount = dueAmount;
        }
}
```

## Cheque.java

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
public class Cheque extends Payment {
    String chequeNo;
        int chequeAmount;
        Date dateOfIssue;
        public String getChequeNo() {
                return chequeNo;
        }
        public void setChequeNo(String chequeNo) {
                this.chequeNo = chequeNo;
        }
        public int getChequeAmount() {
                return chequeAmount;
        }
        public void setChequeAmount(int chequeAmount) {
                this.chequeAmount = chequeAmount;
        }
        public Date getDateOfIssue() {
                return dateOfIssue;
        }
        public void setDateOfIssue(Date dateOfIssue) {
```

```java
            this.dateOfIssue = dateOfIssue;
    }

    @Override
    public boolean payAmount()
    {
            SimpleDateFormat format = new SimpleDateFormat("dd-MM-yyyy");
            Date today = new Date();
            try
            {
                today = format.parse("01-01-2020");
            }
            catch (ParseException e)
            {
                return false;
            }
            long diff = today.getTime()-dateOfIssue.getTime();
            int day = (int) Math.abs(diff/(1000*60*60*24));
            int month = day/30;
            if(month <=6)
            {

                if(chequeAmount>=getDueAmount())
                {
                    return true;
                }
                else
                    return false;

            }
            else
                return false;
    }


}
```

## Cash.java

```java
public class Cash extends Payment {
  int cashAmount;

    public int getCashAmount() {
        return cashAmount;
    }

    public void setCashAmount(int cashAmount) {
        this.cashAmount = cashAmount;
    }

    @Override
    public boolean payAmount()
    {
        if(cashAmount>=getDueAmount())
            return true;
        else
            return false;
    }


}
```

## Credit.java

```java
public class Credit extends Payment {
  int creditCardNo;
    String cardType;
```

```java
	int creditCardAmount;
	public int getCreditCardNo() {
		return creditCardNo;
	}
	public void setCreditCardNo(int creditCardNo) {
		this.creditCardNo = creditCardNo;
	}
	public String getCardType() {
		return cardType;
	}
	public void setCardType(String cardType) {
		this.cardType = cardType;
	}
	public int getCreditCardAmount() {
		return creditCardAmount;
	}
	public void setCreditCardAmount(int creditCardAmount) {
		this.creditCardAmount = creditCardAmount;
	}


	@Override
	public boolean payAmount()
	{
		int netAmount = 0;
		if(cardType.equals("silver"))
		{
			netAmount = (int) (getDueAmount()*1.02);
			creditCardAmount = 10000;
		}
		else if(cardType.equals("gold"))
		{
			netAmount = (int) (getDueAmount()*1.05);
			creditCardAmount = 50000;
		}
		else if(cardType.equals("platinum"))
		{
			netAmount = (int) (int) (getDueAmount()*1.1);
			creditCardAmount = 100000;
		}

		if(creditCardAmount>=netAmount)
		{
			creditCardAmount = creditCardAmount - netAmount;
			return true;
		}
		else
			return false;
	}


}
```

## Bill.java

```java
public class Bill {
  public String processPayment(Payment obj)
{
		String res="";
		if(obj instanceof Cheque)
		{
			if(obj.payAmount())
				res = "Payment done successfully via cheque";
			else
				res = "Payment not done and your due amount is "+obj.getDueAmount();
		}
```

```
			else if(obj instanceof Cash)
			{
				if(obj.payAmount())
					res = "Payment done successfully via cash";
				else
					res = "Payment not done and your due amount is "+obj.getDueAmount();
			}
			else if(obj instanceof Credit)
			{
				Credit c = (Credit) obj;
				if(obj.payAmount())
					res = "Payment done successfully via credit card. Remaining amount in your "+c.getCardType()+" card is "+c.getCreditCardAmount();
				else
					res = "Payment not done and your due amount is "+obj.getDueAmount();
			}
			return res;
		}
}
```

## 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

```
import java.util.*;
public class Main
{
        public static void main(String[] args)
        {
            Scanner sc=new Scanner(System.in);
            //Fill the code
            int m=sc.nextInt();
            if(m<=0){
                System.out.println(""+m+" is an invalid");
                return;
             }
             int n=sc.nextInt();
             if(n<=0){
                System.out.println(""+n+" is an invalid");
                return;
             }
             if(m>=n){
                System.out.println(""+m+" is not less than "+n);
                return;
             }
             for(int i=1;i<=n;i++){
                System.out.print((int)Math.pow(m,i)+"");
             }
         }
}
```

## 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 |
|---|---|---|---|
| **Account** | long accountNumber<br><br>double balanceAmount | Include the getters and setters method for all the attributes.<br><br>Include a parametrized constructor of two arguments in the order – accountNumber,balanceAmount to intialize 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<br><br>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.<br><br>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

```java
import java.text.DecimalFormat;
import java.util.Scanner;
import java.util.Scanner;


public class Main{
    static Account ac=new Account(0, 0);
    public static void main (String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the account number:");
        ac.setAccountNumber(sc.nextLong());
        System.out.println("Enter the available amount in the account:");
        ac.setBalanceAmount(sc.nextDouble());
        System.out.println("Enter the amount to be deposited:");
        ac.deposit(sc.nextDouble());
        System.out.printf("Available balance is:%.2f",ac.getBalanceAmount());
        System.out.println();
        System.out.println("Enter the amount to be withdrawn:");
        ac.withdraw(sc.nextDouble());
        System.out.printf("Available balance is:%.2f",ac.getBalanceAmount());
         //Fill the code
    }
}



```

### Account.java

```java

public class Account {
    long accountNumber;
    double balanceAmount;


    public Account(long accno, double bal){
        super();
        this.accountNumber=accno;
        this.balanceAmount=bal;
    }
    public long getAccountNumber(){
        return accountNumber;
    }
    public void setAccountNumber(long accno){
        this.accountNumber=accno;
    }
    public double getBalanceAmount(){
        return balanceAmount;
    }
    public void setBalanceAmount(double bal) {
        this.balanceAmount=bal;
    }
    public void deposit(double depositAmt){
        float total=(float)(balanceAmount+depositAmt);
        balanceAmount=total;
    }
    public boolean withdraw(double withdrawAmt){
        float total;
        if(withdrawAmt>balanceAmount){
            System.out.println("Insufficient balance");

            return false;
        }else{
            total=(float)(balanceAmount-withdrawAmt);
            setBalanceAmount(total);
            return true;
        }
    }
}
```

## 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

```java
import java.util.Scanner;
import java.lang.String.*;
import java.util.*;
public class Main{
    public static void main(String[] args){
        String[] words;
            Scanner read =new Scanner(System.in);
            String sentence=read.nextLine();
            words=sentence.split(" ");
            if(words.length<3)
            System.out.println("Invalid Sentence");
            else{
                String a=words[0].substring(0,1);
                String b=words[1].substring(0,1);
                String c=words[2].substring(0,1);
                if(a.equalsIgnoreCase(b)&&b.equalsIgnoreCase(c))
                {
                    StringBuilder k= new StringBuilder();
                    k.append(words[words.length-1]);
                    k=k.reverse();
                    k.append(words[0]);
                    System.out.println(k);
                }
                else{
                    StringBuilder k = new StringBuilder();
                    k.append(words[0]);
                    k=k.reverse();
                    k.append(words[words.length-1]);
                    System.out.println(k);
                }
            }
    }
}
```

## 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<br><br>String customerName<br><br>long mobileNumber<br><br>String emailId<br><br>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 numberOfTickets, String ACFacility) | This method should calculate the ticket cost according to the seat tier and return the same. |

**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 ACFacility) | 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

```java

public class BookAMovieTicket {

	protected String ticketId;
	protected String customerName;
	protected long mobileNumber;
	protected String emailId;
	protected String movieName;

	public String getTicketId() {
		return ticketId;
	}
	public void setTicketId(String ticketId) {
		this.ticketId = ticketId;
	}
	public String getCustomerName() {
		return customerName;
	}
	public void setCustomerName(String customerName) {
		this.customerName = customerName;
	}
	public long getMobileNumber() {
		return mobileNumber;
	}
	public void setMobileNumber(long mobileNumber) {
		this.mobileNumber = mobileNumber;
	}
	public String getEmailId() {
		return emailId;
	}
	public void setEmailId(String emailId) {
		this.emailId = emailId;
	}
	public String getMovieName() {
		return movieName;
	}
	public void setMovieName(String movieName) {
		this.movieName = movieName;
	}

	public BookAMovieTicket(String ticketId, String customerName, long mobileNumber, String emailId, String movieName) {
		this.ticketId = ticketId;
		this.customerName = customerName;
		this.mobileNumber = mobileNumber;
		this.emailId = emailId;
		this.movieName = movieName;

	}



}

```

## GoldTicket.java

```java

public class GoldTicket extends BookAMovieTicket{
  public GoldTicket(String ticketId,String customerName, long mobileNumber,
  String emailId, String movieName){
     super(ticketId, customerName, mobileNumber, emailId, movieName);
  }

	public boolean validateTicketId(){
		int count=0;
		if(ticketId.contains("GOLD"));
		count++;
		char[] cha=ticketId.toCharArray();
		for(int i=4;i<7;i++){
		   if(cha[i]>='1'&& cha[i]<='9')
		   count++;
		}
		if(count==4)
		return true;
		else
		return false;
	}


	// Include Constructor

	public double calculateTicketCost(int numberOfTickets, String ACFacility){
		double amount;
		if(ACFacility.equals("yes")){
		   amount=500*numberOfTickets;
		}
		else{
		   amount=350*numberOfTickets;
		}

		return amount;
	}

}
```

## PlatinumTicket.java

```java
public class PlatinumTicket extends BookAMovieTicket{
  public PlatinumTicket(String ticketId, String customerName, long mobileNumber,
  String emailId, String movieName){
    super(ticketId, customerName, mobileNumber, emailId, movieName);
  }

        public boolean validateTicketId(){
                        int count=0;
                        if(ticketId.contains("PLATINUM"));
                        count++;
                        char[] cha=ticketId.toCharArray();
                        for(int i=8;i<11;i++){
                          if(cha[i]>='1'&& cha[i]<='9')
                          count++;
                        }
                        if(count==4)
                        return true;
                        else
                        return false;
        }

        // Include Constructor

        public double calculateTicketCost(int numberOfTickets, String ACFacility){
                        double amount;
                        if(ACFacility.equalsIgnoreCase("yes")){
                          amount=750*numberOfTickets;
                        }
                        else{
                          amount=600*numberOfTickets;
                        }

                        return amount;
        }

}

```

## SilverTicket.java

```java

public class SilverTicket extends BookAMovieTicket{
  public SilverTicket(String ticketId, String customerName, long mobileNumber,
  String emailId, String movieName){
    super(ticketId, customerName, mobileNumber, emailId, movieName);
  }

        public boolean validateTicketId(){
                        int count=0;
                        if(ticketId.contains("SILVER"));
                        count++;
                        char[] cha=ticketId.toCharArray();
                        for(int i=6;i<9;i++){
                          if(cha[i]>='1'&& cha[i]<='9')
                          count++;
                        }
                        if(count==4)
                        return true;
                        else
                        return false;
        }

        // Include Constructor

        public double calculateTicketCost(int numberOfTickets, String ACFacility){
                        double amount;
                        if(ACFacility.equals("yes")){
                          amount=250*numberOfTickets;
                        }
```

```java
                else{
                    amount=100*numberOfTickets;
                }

                return amount;
        }

}

```

## UserInterface.java

```java
import java.util.*;

public class UserInterface {

        public static void main(String[] args){
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter Ticket Id");
                String tid=sc.next();
                System.out.println("Enter Customer name");
                String cnm=sc.next();
                System.out.println("Enter Mobile number");
                long mno=sc.nextLong();
                System.out.println("Enter Email id");
                String email=sc.next();
                System.out.println("Enter Movie name");
                String mnm=sc.next();
                System.out.println("Enter number of tickets");
                int tno=sc.nextInt();
                System.out.println("Do you want AC or not");
                String choice =sc.next();
                if(tid.contains("PLATINUM")){
                    PlatinumTicket PT= new PlatinumTicket(tid,cnm,mno,email,mnm);
                    boolean b1=PT.validateTicketId();
                    if(b1==true){
                        double cost=PT.calculateTicketCost(tno, choice);
                        System.out.println("Ticket cost is "+String.format("%.2f",cost));
                    }
                    else if(b1==false){
                        System.out.println("Provide valid Ticket Id");
                        System.exit(0);
                    }
                }
                else if(tid.contains("GOLD")){
                    GoldTicket GT= new GoldTicket(tid,cnm,mno,email,mnm);
                    boolean b2=GT.validateTicketId();
                    if(b2==true){
                        double cost=GT.calculateTicketCost(tno,choice);
                        System.out.println("Ticket cost is "+String.format("%.2f",cost));
                    }
                    else if (b2==false){
                        System.out.println("Provide valid Ticket Id");
                        System.exit(0);
                    }
                }
                else if(tid.contains("SILVER")){
                    SilverTicket ST= new SilverTicket(tid,cnm,mno,email,mnm);
                    boolean b3=ST.validateTicketId();
                    if(b3==true){
                        double cost=ST.calculateTicketCost(tno,choice);
                        System.out.println("Ticket cost is "+String.format("%.2f",cost));
                    }
                    else if (b3==false){
                        System.out.println("Provide valid Ticket Id");
                        System.exit(0);
                    }
        }
        }
}


```

## Grade

Reviewed on Monday, 7 February 2022, 4:18 PM by Automatic grade
**Grade** 100 / 100
**Assessment report**
[+]**Grading and Feedback**

==================================================================================

# Group-2

## 1. Flight record retrieval

**Grade settings**: Maximum grade: 100
**Based on**: JAVA CC JDBC - MetaData V1 - ORACLE (w/o Proj Struc)
**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) | Attributes | Methods | Responsibilities |
|---|---|---|---|
| **Flight** | int flightId<br><br>String source<br><br>String destination<br><br>int noOfSeats<br><br>double flightFare | Include getters and setter method for all the attributes.<br><br>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:

| Column Name | Datatype |
|---|---|
| flightId | integer |
| source | varchar2(30) |
| destination | varchar2(30) |
| noofseats | integer |
| flightfare | double |

Sample records available in **flight** table are:

| Flightid | Source | Destination | Noofseats | Flightfare |
|---|---|---|---|---|
| 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**.**

**D**isplay 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

```java

public class Flight {

	private int flightId;
	private String source;
	private String destination;
	private int noOfSeats;
	private double flightFare;
	public int getFlightId() {
		return flightId;
	}
	public void setFlightId(int flightId) {
		this.flightId = flightId;
	}
	public String getSource() {
		return source;
	}
	public void setSource(String source) {
		this.source = source;
	}
	public String getDestination() {
		return destination;
	}
	public void setDestination(String destination) {
		this.destination = destination;
	}
	public int getNoOfSeats() {
		return noOfSeats;
	}
	public void setNoOfSeats(int noOfSeats) {
		this.noOfSeats = noOfSeats;
	}
	public double getFlightFare() {
		return flightFare;
	}
	public void setFlightFare(double flightFare) {
		this.flightFare = flightFare;
	}
	public Flight(int flightId, String source, String destination,
			int noOfSeats, double flightFare) {
		super();
		this.flightId = flightId;
		this.source = source;
		this.destination = destination;
		this.noOfSeats = noOfSeats;
		this.flightFare = flightFare;
	}



}

```

### FlightManagementSystem.java

```java
import java.util.ArrayList;
import java.sql.*;


public class FlightManagementSystem {

    public ArrayList<Flight> viewFlightBySourceDestination(String source, String destination){
        ArrayList<Flight> flightList = new ArrayList<Flight>();
        try{
            Connection con = DB.getConnection();

            String query="SELECT * FROM flight WHERE source= '" + source + "' AND destination= '" + destination + "' ";

            Statement st=con.createStatement();

            ResultSet rst= st.executeQuery(query);

            while(rst.next()){
                int flightId= rst.getInt(1);
                String src=rst.getString(2);
                String dst=rst.getString(3);
                int noofseats=rst.getInt(4);
                double flightfare=rst.getDouble(5);

                flightList.add(new Flight(flightId, src, dst, noofseats, flightfare));
            }
        }catch(ClassNotFoundException | SQLException e){
            e.printStackTrace();
        }
        return flightList;
    }

}
```

## Main.java

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

public class Main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the source");
        String source=sc.next();
        System.out.println("Enter the destination");
         String destination=sc.next();

         FlightManagementSystem fms= new FlightManagementSystem();
         ArrayList<Flight> flightList=fms.viewFlightBySourceDestination(source,destination);
         if(flightList.isEmpty()){
             System.out.println("No flights available for the given source and destination");
             return;
         }
         System.out.println("Flightid Noofseats Flightfare");
         for(Flight flight : flightList){
             System.out.println(flight.getFlightId()+" "+flight.getNoOfSeats()+" "+flight.getFlightFare());
         }

    }
}
```

## DB.java

```java
import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;
```

```java

public class DB {

        private static Connection con = null;
        private static Properties props = new Properties();


    //ENSURE YOU DON'T CHANGE THE BELOW CODE WHEN YOU SUBMIT
        public static Connection getConnection() throws ClassNotFoundException, SQLException {
            try{

                                    FileInputStream fis = null;
                                    fis = new FileInputStream("database.properties");
                                    props.load(fis);

                                    // load the Driver Class
                                    Class.forName(props.getProperty("DB_DRIVER_CLASS"));

                                    // create the connection now
            con =
DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),props.getProperty("DB_PASSWORD"));
            }
            catch(IOException e){
                e.printStackTrace();
            }
                        return con;
        }
}

```

## database.properties

```
#IF NEEDED, YOU CAN MODIFY THIS PROPERTY FILE
#ENSURE YOU ARE NOT CHANGING THE NAME OF THE PROPERTY
#YOU CAN CHANGE THE VALUE OF THE PROPERTY
#LOAD THE DETAILS OF DRIVER CLASS, URL, USERNAME AND PASSWORD IN DB.java using this properties file only.
#Do not hard code the values in DB.java.

DB_DRIVER_CLASS=oracle.jdbc.driver.OracleDriver
DB_URL=jdbc:oracle:thin:@127.0.0.1:1521:XE
DB_USERNAME=${sys:db_username}
DB_PASSWORD=${sys:db_password}

```

# Grade

Reviewed on Monday, 7 February 2022, 6:33 PM by Automatic grade

**Grade** 100 / 100

**Assessment report**

**Assessment Completed Successfully**

[+]**Grading and Feedback**

==================================================

## 2. Get Text and Display Welcome Message

**Grade settings**: Maximum grade: 100
**Run**: Yes **Evaluate**: Yes
**Automatic grade**: Yes **Maximum execution time**: 16 s

Amir owns “Bouncing Babies” an exclusive online store for baby toys.

He desires to display a welcome message whenever a customer visits his online store and makes a purchase.

Help him do this by incorporating the customer name using the Lambda expression.

**Requirement 1:** Display Welcome message

Amir wants to display a welcome message for his customers. The method displayText is used to display the name of the customer who made an online purchase from his store.

**Component Specification: DisplayText Interface – This is a Functional Interface.**

| Type(Interface) | Methods | Responsibilities |
|---|---|---|
| DisplayText | public void displayText(String text) | The purpose of this method is to display the welcome message by including the text provided as an argument by using Lambda expression. |
| DisplayText | public default String getInput() | This method should get a String (name of the customer) as input from the user and return the same. This method should be a default method. |

**Annotate the interface with the appropriate annotation**

**Component Specification: Main class**

| Component Name | Type(Class) | Methods | Responsibilities |
|---|---|---|---|
| Display welcome message | Main | public static DisplayText welcomeMessage() | This method should return a DisplayText object. To do this, implement the lambda expression to print the text received as a parameter in the displayText method as “Welcome <text>”. |

**In the Main class write the main method and perform the given steps :**

- Invoke the static method welcomeMessage(). It returns a DisplayText object.
- Capture the DisplayText object in a reference variable.
- Using that reference, invoke the default method getInput.

- It will return a String. Capture that String in a variable.
- Using the reference of DisplayText, invoke the displayText method by passing the String as a parameter.
- The output should be as shown in the sample data mentioned below.

**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 name for classes, interfaces and methods as specified in the question.

Adhere to the code template, if provided.

**Sample Input 1 :**

**Watson**

**Sample Output 1 :**

Welcome Watson

## Automatic evaluation[+]

### DisplayText.java

```
import java.util.*;
@FunctionalInterface
public interface DisplayText
{
  public void displayText(String text);
  public default String getInput()
  {
      Scanner read = new Scanner(System.in);
      String str = read.next();
       return str;
       //return null;
   }
}
```

### Main.java

```
public class Main
{
    public static DisplayText welcomeMessage()
    {

        DisplayText dis = (str)->{

             System.out.println("Welcome "+str);
        };
         return dis;
     }
    public static void main(String args[])
    {
      DisplayText dis=welcomeMessage();
      String text = dis.getInput();
      dis.displayText(text);

    }
}
```

## Grade

Reviewed on Wednesday, 1 December 2021, 10:14 PM by Automatic grade

**Grade** 100 / 100

**Assessment report**

[+]**Grading and Feedback**

========================================

## 3. Generate Password

**Grade settings**: Maximum grade: 100
**Run**: Yes **Evaluate**: Yes
**Automatic grade**: Yes
**Important Instructions:**

· **Please read the document thoroughly before you code.**

· **Import the given skeleton code into your Eclipse.(if provided)**

· **Do not change the Skeleton code or the package structure, method names, variable names, return types, exception clauses, access specifiers etc.**

· **You can create any number of private methods inside the given class.**

· **You can test your code from main() method of the program**

The system administrator of an organization wants to set password for all the computers for security purpose. To generate a strong password, he wants to combine the username of each user of the system with the reverse of their respective usernames. Help them by using Lambda expressions that caters to their requirement.

**Requirement 1:** PasswordInfo

The Administrator wants to generate password for each system by making use of the passwordGeneration method based on the username which is passed as a string.

**Component Specification:** Password Info Interface – This is a Functional Interface.

| Type(Interface) | Methods | Responsibilities |
|---|---|---|
| **PasswordInfo** | public String passwordGeneration(String username) | This method is used to generate the password based on the username and hence returns the generated password |

**Component Specification: Computer Class**

| Type(Class) | Methods | Responsibilities |
|---|---|---|
| Computer | public static PasswordInfo passwordPropagation() | This method should return a PasswordInfo object. To do this, implement the lambda expression to get the password. |
| | public static void displayUserDetails(String systemNo,String username,PasswordInfo passwordInfoObj) | This method is used to print the Password Info such as the systemNo, password along with the message, “Your password is generated successfully!!!” based |

| | | |
|---|---|---|
| | | on the systemNo, username, passwordInfoObj which is passed as an argument. |

In the Computer class write the main method and perform the given steps:

- Get the systemNo and username from the user.
- Invoke the static method passwordPropagation(). It returns a passwordInfo object with the definition of the passwordGeneration method.
- Capture the PasswordInfo object in a reference variable.
- Invoke the displayUserDetails method by passing systemNo, username and passwordInfoObj as parameters.
- Inside the userDetails method, you should invoke the passwordGeneration method using the passwordInfo object and the output should be displayed as shown in the sample input/output.
- The output should be as shown in the sample data mentioned below.

**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 use the lambda expression.
- Ensure to follow the object oriented specifications provided in the question.
- Ensure to provide the name for classes, interfaces and methods as specified in the question.
- Adhere to the code template, if provided.

**Sample Input 1:**

Enter system no

**Tek/1234**

Enter username

**Manoj Kumar**

**Sample Output 1:**

Password Info

System no: Tek/1234

Password: Manoj KumarramuK jonaM

Your password is generated successfully!!!

==================================================================

## 4. Watican Museum Manipulation

**Grade settings**: Maximum grade: 100
**Run**: Yes **Evaluate**: Yes
**Automatic grade**: Yes **Maximum execution time**: 60 s **Maximum memory used**: 64 MiB **Maximum execution file size**: 320 KiB
**Important Instructions:**

**· Please read the document thoroughly before you code.**

**· Import the given skeleton code into your Eclipse.(if provided)**

**· Do not change the Skeleton code or the package structure, method names, variable names, return types, exception clauses, access specifiers etc.**

**· You can create any number of private methods inside the given class.**

**· You can test your code from the main() method of the program.**

Watican Museum is one of the famous museums, they have collections of houses paintings, and sculptures from artists. The Museum management stores their visitor's details in a text file. Now, they need an application to analyze and manipulate the visitor details based on the visitor visit date and the visitor address.

**You are provided with a text file – VisitorDetails.txt, which contains all the visitor details like the visitor Id, visitor name, mobile number, date of visiting and address. Your application should satisfy the following requirements.**

1. View visitor details within two given dates.

2. View visitor details which are above a particular mentioned visitor address.

You are provided with a code template which includes the following:

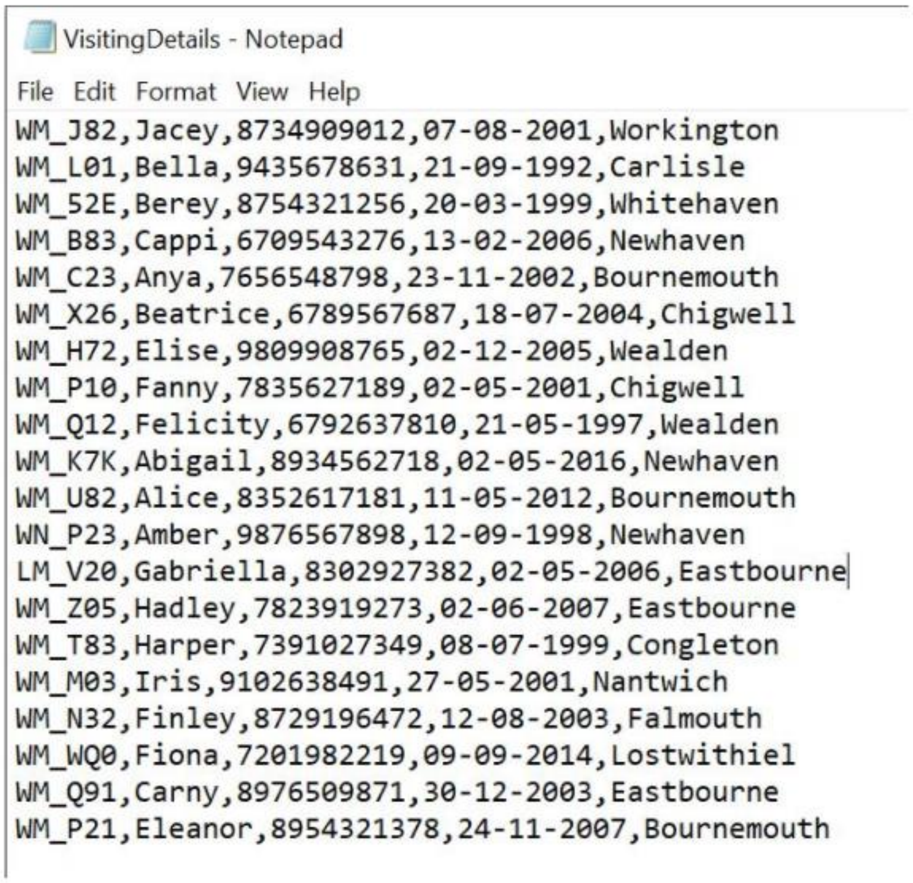
- Visitor class which includes the attributes visitorId, visitorName, mobileNumber, dateOfVisiting and address with all the getters and setters.
- VisitorUtility class which includes the following method declarations.
    - public List<Visitor> generateVisitor(String filePath)
    - public boolean isValidVisitorId(String visitorId)
    - public List<Visitor> viewVisitorDetailsByDateOfVisiting(Stream<Visitor> visitorStream, String fromDate, String toDate)
    - public Stream<Visitor> viewVisitorDetailsByAddress (Stream<Visitor> visitorStream, double address)
- InvalidVisitorIdException class which inherits the Exception class.
- Main class with a main method which creates the required user interface for the application.
- VisitorDetails.txt which contains all the visitor details like visitor id, visitor name, mobile number, date of visiting and address.

**Note:**

- The Visitor class and the Main class will be provided with all the necessary codes. Please do not edit or delete any line of the code in these two classes.

- Fill your code in the InvalidVisitorIdException class to create a constructor as described in the functional requirements below.
- Fill your code in the respective methods of VisitorUtility class to fulfil all the functional requirements.
- In the VisitorDetails.txt file, each visitor detail has information separated by a comma, and it is given as one customer detail per line.

**Sample data in VisitorDetails.txt file**



```
WM_J82,Jacey,8734909012,07-08-2001,Workington
WM_L01,Bella,9435678631,21-09-1992,Carlisle
WM_52E,Berey,8754321256,20-03-1999,Whitehaven
WM_B83,Cappi,6709543276,13-02-2006,Newhaven
WM_C23,Anya,7656548798,23-11-2002,Bournemouth
WM_X26,Beatrice,6789567687,18-07-2004,Chigwell
WM_H72,Elise,9809908765,02-12-2005,Wealden
WM_P10,Fanny,7835627189,02-05-2001,Chigwell
WM_Q12,Felicity,6792637810,21-05-1997,Wealden
WM_K7K,Abigail,8934562718,02-05-2016,Newhaven
WM_U82,Alice,8352617181,11-05-2012,Bournemouth
WN_P23,Amber,9876567898,12-09-1998,Newhaven
LM_V20,Gabriella,8302927382,02-05-2006,Eastbourne
WM_Z05,Hadley,7823919273,02-06-2007,Eastbourne
WM_T83,Harper,7391027349,08-07-1999,Congleton
WM_M03,Iris,9102638491,27-05-2001,Nantwich
WM_N32,Finley,8729196472,12-08-2003,Falmouth
WM_WQ0,Fiona,7201982219,09-09-2014,Lostwithiel
WM_Q91,Carny,8976509871,30-12-2003,Eastbourne
WM_P21,Eleanor,8954321378,24-11-2007,Bournemouth
```

**Functional Requirements:**

Fill your code in the respective class and method declarations based on the required functionalities as given below.

| Class | Attributes/ Methods | Rules/ Responsibility |
|---|---|---|

|  |  |  |
|---|---|---|
| VisitorUtility | public List < Visitor> generateVisitor(String filePath) | Read the text file and convert each line in the text file as String and store it in a List. Each String from the List should be converted into a visitor object and each visitor object should be stored in a List. Return the List of visitors.<br><br>**Note:**<br><br>Before converting the separated string into a visitor object, the identified visitorId should be validated using the isValidVisitorId method. |
| VisitorUtility | public boolean isValidVisitorId (String visitorId) | Should check whether the provided visitorId is valid or not.<br><br>If valid, this method should return true.<br><br>If invalid, this method should handle an InvalidVisitorIdException with a message “<visitorId> is Invalid Visitor Id”.<br><br>**Validation Rules:**<br><br>· Length of the visitorId should be exactly 6.<br><br>· The visitorId should start with “WM_” and the next letter should be an alphabet (A-Z) in upper case and the last two letters should be positive integers(0-9).<br><br>Example.<br><br>WM_A23 |
| InvalidVisitorIdException | Create a constructor with a single String argument and pass it to the parent class constructor. | This class Should inherit the Exception class. The constructor should pass the String message which is thrown to it by calling the parent class constructor. |

**Requirement 1: View visitor details between the dates of visiting**

| Class | Attributes/ Methods | Rules/ Responsibility |
|---|---|---|

| | | |
|---|---|---|
| VisitorUtility | public List<Visitor> viewVisitorDetailsByDateOfVisiting(Stream<Visitor> visitorStream, String fromDate, String toDate) | From the provided Stream of Visitor, separate the visitor details which has the date of visiting between fromDate and toDate (both inclusive). Return the separated visitor details as a list. |

**Requirement 2: View visitor details which are above a particular mentioned address**

| **Class** | **Attributes/ Methods** | **Rules/ Responsibility** |
|---|---|---|
| VisitorUtility | public Stream<Visitor> viewVisitorDetailsByAddress(Stream<Visitor> visitorStream, String address) | From the given Stream of Visitor, separate the visitor details based on address, which has **a particular mentioned** address as provided. Return the separated Stream of visitor. |

**Note:**

1. All inputs/ outputs for processing the functional requirements should be case sensitive.
2. Adhere to the Sample Inputs/ Outputs
3. In the Sample Inputs/ Outputs provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.
4. All the Date values used in this application must be in “dd-MM-yyyy” format.
5. Adhere to the code template.
6. Fill all your required codes in the respective blocks. Do not edit or delete the codes provided in the code template.
7. The Sample Inputs/ Outputs given below are generated based on the Sample data given in the VisitorDetails.txt file.
8. Please do not hard code the output.

**Sample Input/ Output 1:**

```
WM_52E is Invalid Visitor Id

WM_K7K is Invalid Visitor Id

WN_P23 is Invalid Visitor Id

LM_V20 is Invalid Visitor Id

WM_WQ0 is Invalid Visitor Id

1. ViewVisitorDetailsByDateOfVisiting

2. ViewVisitorDetailsByAddress

Enter your choice
```

**1**

Enter the starting date

**19-05-2004**

Enter the ending date

**07-04-2012**

WM_B83 Cappi 6709543276 13-02-2006 Newhaven

WM_X26 Beatrice 6789567687 18-07-2004 Chigwell

WM_H72 Elise 9809908765 02-12-2005 Wealden

WM_Z05 Hadley 7823919273 02-06-2007 Eastbourne

WM_P21 Eleanor 8954321378 24-11-2007 Bournemouth

**Sample Input/ Output 2:**

WM_52E is Invalid Visitor Id

WM_K7K is Invalid Visitor Id

WN_P23 is Invalid Visitor Id

LM_V20 is Invalid Visitor Id

WM_WQ0 is Invalid Visitor Id

1. viewVisitorDetailsByDateOfVisiting

2. viewVisitorDetailsByAddress

Enter your choice

**2**

Enter the address

**Eastbourne**

WM_Z05 Hadley 7823919273 02-06-2007 Eastbourne

WM_Q91 Carny 8976509871 30-12-2003 Eastbourne

# 5. Hospital Management_Streams

**Grade settings**: Maximum grade: 100
**Run**: Yes **Evaluate**: Yes
**Automatic grade**: Yes
**Laxmi Hospital** is a world-class health care institution providing patient treatment with specialized medical and nursing staff and medical equipment. It typically provides an emergency department to treat urgent health problems ranging from fire and accident victims to sudden illness. The hospital maintains a register to maintain the records of the patients who enter the emergency department. The receptionist at the helpdesk would like to filter the patients based on a criterion. Develop a java application for the same using Streams.

**Requirements:**

1. Read the patient records from the file.

2. Retrieve the patient details for the specified date interval.

3. Retrieve the  patient details which are from a particular area (address).

**Component Specification: Patient (POJO Class)**

| **Type (Class)** | **Attributes** | **Methods** |
|---|---|---|
| Patient | String patientId<br><br>String patientName<br><br>String contactNumber<br><br>String dateOfVisit<br><br>String patientAddress | Getters and Setters are given in the code skeleton. |

**Component Specification: PatientUtility**

| **Type (Class)** | **Methods** | **Responsibilities** |
|---|---|---|
| PatientUtility | public List <Patient> fetchPatient(String filePath) | Read the file using File I/O or Java Streams and return the validated list of patient records. It should filter the valid patient records based on the valid patient Id using |

| | | |
|---|---|---|
| | | the method isValidPatientId ().<br><br>**Note:** Make sure that the user-defined exception is handled in this method itself. |
| PatientUtility | public boolean isValidPatientId (String patientId) | **Validation Guidelines for Valid Patient ID:**<br><br>• The length of the Patient Id should be exactly 6.<br>• The Patient Id should start with "WM_" and the next letter should be an alphabet (A-Z) in upper case and the last two letters should be positive integers(0-9). Example. WM_A10.<br><br>Check whether the patient Id is valid or not. If invalid, this method should handle an InvalidPatientIdException with a message "<patientid> is an Invalid Patient Id". |
| PatientUtility | public List<Patient> retrievePatientRecords_ByDateOfVisit(Stream<Patient> patientStream, String fromDate, String toDate) | From the provided stream of patient, separate the patient details which has the date of visit between fromDate and toDate (both inclusive) and return the resultant patient records as a list. |
| PatientUtility | public Stream<Patient> retrievePatientRecords_ByAddress(Stream<Patient> patientStream, String address) | From the given stream of patient, filter the patient details based on the user input address, and return the separated Stream of patients. |

**Component Specification: InvalidPatientIdException (User defined Exception)**

| **Type (Class)** | **Methods** | **Responsibilities** |
|---|---|---|
| InvalidPatientIdException | public InvalidPatientIdException(String message) | This constructor should set the message to the superclass. |

**Note:** The class and methods should be declared as public and all the attributes should be declared as private.

You are provided with a text file –PatientRegister.txt, which contains all the patient details like the patient Id, patient name, contact number, date of visit, and patient address. You can add any number of records in the text file to test your code.

**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 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:**

Invalid Patient Id are:

WM_52E is an Invalid Patient Id

WM_K7K is an Invalid Patient Id

WN_P23 is an Invalid Patient Id

LM_V20 is an Invalid Patient Id

WM_WQ0 is an Invalid Patient Id

Retrieve Patient Details

1. By Date of Visit

2. By Address

Enter your choice:

**1**

Enter the start date

**02-03-2003**

Enter the end date

**02-12-2005**

WM_X26 Beatrice 6789567687 18-07-2004 Texas

WM_H72 Elise 9809908765 02-12-2005 Washington

WM_N32 Finley 8729196472 12-08-2003 Pennsylvania

WM_Q91 Carny 8976509871 30-12-2003 Virginia

**Sample Input/Output 2:**

Invalid Patient Id are:

WM_52E is an Invalid Patient Id

WM_K7K is an Invalid Patient Id

WN_P23 is an Invalid Patient Id

LM_V20 is an Invalid Patient Id

WM_WQ0 is an Invalid Patient Id

Retrieve Patient Details

1. By Date of Visit

2. By Address

Enter your choice:

**2**

Enter the address

**Carolina**

WM_C23 Anya 7656548798 23-11-2002 Carolina

WM_T83 Harper 7391027349 08-07-1999 Carolina

WM_P21 Eleanor 8954321378 24-11-2007 Carolina

**Sample Input/Output 3:**

Invalid Patient Id are:

WM_52E is an Invalid Patient Id

WM_K7K is an Invalid Patient Id

WN_P23 is an Invalid Patient Id

LM_V20 is an Invalid Patient Id

WM_WQ0 is an Invalid Patient Id

Retrieve Patient Details

1. By Date of Visit

2. By Address

Enter your choice:

**1**

Enter the start date

**03-02-2020**

Enter the end date

**02-02-2021**

No patient records available during this interval

**Sample Input/Output 4:**

Invalid Patient Id are:

WM_52E is an Invalid Patient Id

WM_K7K is an Invalid Patient Id

WN_P23 is an Invalid Patient Id

LM_V20 is an Invalid Patient Id

WM_WQ0 is an Invalid Patient Id

Retrieve Patient Details

1. By Date of Visit

2. By Address

Enter your choice:

**3**

Invalid Option

## Automatic evaluation[+]

### HospitalManagement/PatientRegister.txt

```
WM_J82,Jacey,8734909012,07-08-2001,Colorado
WM_L01,Bella,9435678631,21-09-1992,Connecticut
WM_52E,Berey,8754321256,20-03-1999,Indiana
WM_B83,Cappi,6709543276,13-02-2006,Pennsylvania
WM_C23,Anya,7656548798,23-11-2002,Carolina
WM_X26,Beatrice,6789567687,18-07-2004,Texas
WM_H72,Elise,9809908765,02-12-2005,Washington
WM_P10,Fanny,7835627189,02-05-2001,Virginia
WM_Q12,Felicity,6792637810,21-05-1997,Colorado
WM_K7K,Abigail,8934562718,02-05-2016,Indiana
WM_U82,Alice,8352617181,11-05-2012,Indiana
WN_P23,Amber,9876567898,12-09-1998,Pennsylvania
LM_V20,Gabriella,8302927382,02-05-2006,Connecticut
WM_Z05,Hadley,7823919273,02-06-2007,Connecticut
WM_T83,Harper,7391027349,08-07-1999,Carolina
WM_M03,Iris,9102638491,27-05-2001,Texas
WM_N32,Finley,8729196472,12-08-2003,Pennsylvania
WM_WQ0,Fiona,7201982219,09-09-2014,Washington
WM_Q91,Carny,8976509871,30-12-2003,Virginia
WM_P21,Eleanor,8954321378,24-11-2007,Carolina
```

### HospitalManagement/src/InvalidPatientIdException.java

```java

//public class InvalidPatientIdException{
        //FILL THE CODE HERE
        public class InvalidPatientIdException extends Exception{
            public InvalidPatientIdException(String message){
                super(message);
            }
        }




```

### HospitalManagement/src/Main.java

```java
public class Main {

        public static void main(String[] args){

                                                        // CODE SKELETON - VALIDATION STARTS
                                                        // DO NOT CHANGE THIS CODE

                                                         new SkeletonValidator();
                                                        // CODE SKELETON - VALIDATION ENDS

                                                        // FILL THE CODE HERE

                                }

        }


```

## HospitalManagement/src/Patient.java

```java
//DO NOT ADD/EDIT THE CODE
public class Patient {

	private String patientId;
	private String patientName;
	private String contactNumber;
	private String dateOfVisit;
	private String patientAddress;

	//Setters and Getters

	public String getPatientId() {
		return patientId;
	}
	public void setPatientId(String patientId) {
		this.patientId = patientId;
	}
	public String getPatientName() {
		return patientName;
	}
	public void setPatientName(String patientName) {
		this.patientName = patientName;
	}
	public String getContactNumber() {
		return contactNumber;
	}
	public void setContactNumber(String contactNumber) {
		this.contactNumber = contactNumber;
	}
	public String getDateOfVisit() {
		return dateOfVisit;
	}
	public void setDateOfVisit(String dateOfVisit) {
		this.dateOfVisit = dateOfVisit;
	}
	public String getPatientAddress() {
		return patientAddress;
	}
	public void setPatientAddress(String patientAddress) {
		this.patientAddress = patientAddress;
	}




}

```

## HospitalManagement/src/PatientUtility.java

```java
import java.util.List;
import java.util.stream.Stream;
import java.util.ArrayList;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.regex.*;
import java.util.stream.Collectors;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;


```

```
public class PatientUtility {

        public List <Patient> fetchPatient(String filePath) {


                        //FILL THE CODE HERE
                        List <Patient> patients =new ArrayList<>();
                        try{
                          File register =new File(filePath);
                          Scanner reader=new Scanner(register);
                          while(reader.hasNextLine()){
                          Patient p = new Patient();
                          String[] infos=reader.nextLine().split(",");
                          try{
                            if(isValidPatientId(infos[0])){
                              p.setPatientId(infos[0]);
                              p.setPatientName(infos[1]);
                              p.setContactNumber(infos[2]);
                              p.setDateOfVisit(infos[3]);
                              p.setPatientAddress(infos[4]);
                              patients.add(p);
                            }
                          }
                          catch(InvalidPatientIdException e1){
                            System.out.println(e1.getMessage());
                          }
                        }
                        reader.close();
                        }
                        catch(FileNotFoundException e){}
                        return patients;

                        //return null;
        }


        public boolean isValidPatientId (String patientId)throws InvalidPatientIdException
        {

                        //FILL THE CODE HERE
                        Pattern p =Pattern.compile("WM_[A-Z][0-9]{2}$");
                        Matcher m=p.matcher(patientId);
                        boolean ne =m.matches();
                        if(!ne){
                          throw new InvalidPatientIdException(patientId+"is an Invalid Patient Id.");

                        }
                        //return inValid;
                        return ne;
        }


        public List<Patient> retrievePatientRecords_ByDateOfVisit(Stream<Patient> patientStream, String
fromDate, String toDate)
        {
                        //FILL THE CODE HERE
                        SimpleDateFormat simpleDateFormat=new SimpleDateFormat("dd-MM-yyyy");
                        return patientStream
                        .filter((p)->{
                          try{
                            Date start=simpleDateFormat.parse(fromDate);
                            Date end= simpleDateFormat.parse(toDate);
                            Date current =simpleDateFormat.parse(p.getDateOfVisit());
                            return start.compareTo(current)*current.compareTo(end)>=0;
                          }
                          catch(ParseException e){}
                          return false;
```

```
		}).collect(Collectors.toList());
	//		return null;
	}



	public Stream<Patient> retrievePatientRecords_ByAddress(Stream<Patient> patientStream, String address)
	{

		//FILL THE CODE HERE
		return patientStream.filter(p->address.equals(p.getPatientAddress()));
		//return null;



	}

}

```

## HospitalManagement/src/SkeletonValidator.java

```
import java.lang.reflect.Method;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.stream.Stream;

/**
 * @author TJ
 *
 * 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("Patient");
		validateClassName("PatientUtility");
		validateClassName("InvalidPatientIdException");
		validateMethodSignature(
	"fetchPatient:java.util.List,isValidPatientId:boolean,retrievePatientRecords_ByDateOfVisit:java.util.List,retrievePatientRecords_ByAddress:java.util.stream.Stream",
						"PatientUtility");

	}

	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");
```

```java

		} 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");

						} 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");
```

```
            }
    }

}
```

## Grade

Reviewed on Monday, 7 February 2022, 6:04 PM by Automatic grade

**Grade** 100 / 100

**Assessment report**

**Assessment Completed Successfully**

[+]**Grading and Feedback**

=================================================================================

## 6. Technology Fest

**Grade settings**: Maximum grade: 100
**Run**: Yes **Evaluate**: Yes
**Automatic grade**: Yes

**Institute of Technology** is organizing an All-India Technology Fest for various engineering colleges across the country. The management would like to automate the registration so that it is easier and more systematic while conducting the fest. Create a java application for the same using Threads.

**Component Specification: Participant (POJO Class)**

| Type (Class) | Attributes | Methods |
|---|---|---|
| Participant | String name<br><br>String yearofstudy<br><br>String department<br><br>String collegeName<br><br>String eventName<br><br>double registrationFee | Getters, Setters, and a five-argument constructor in the given order - name, yearofstudy, department, collegeName, eventName are included in the code Skeleton. |

**Requirements:**

· To calculate the registration fee of the participant based on the event name.

· To calculate the number of participants registered for a particular event.

| Sl No | Event Name | Registration Fee |
|---|---|---|
| 1 | Robocar | 1000 |
| 2 | PaperTalk | 500 |
| 3 | Quiz | 300 |
| 4 | Games | 100 |

***Note that Event name is case in- sensitive**

**Component Specification: EventManagement (Thread Class)**

| Type (Class) | Attributes | Methods | Responsibilities |
|---|---|---|---|
| EventManagement | List<Participant> TechList<br><br>String searchEvent<br><br>int counter | | Include getters and setter methods for all the attributes. |

| | | | |
|---|---|---|---|
| EventManagement | | public void calculateRegistrationFee(List <Participant> list) | Calculate the registration fee of the participant based on the event name. If the event name doesn't exist, throw an InvalidEventException with an error message "Event Name is invalid". |
| EventManagement | | public void run() | Calculate the number of participants registered for a particular event. Increment the counter attribute based on the search. |

**Note:** The class and methods should be declared as public and all the attributes should be declared as private.

**Component Specification: InvalidEventException**

| **Type (Class)** | **Methods** | **Responsibilities** |
|---|---|---|
| InvalidEventException | public InvalidEventException (String message) | To set the message string to the superclass. |

**Create a class called Main with the main method and perform the tasks are given below:**

· Get the inputs as provided in the sample input.

· Call the calculateRegistrationFee () method to calculate the registration fee of the participant based on the event name.

· Print the list of Participant objects with the registration fee.

· Get the event type to search to find the number of the participants registered for that particular event.

· Handle the user-defined exception in the main method.

· Display the output as shown in the sample input/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 represent the output.

· Ensure to follow the 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 entries

**3**

Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName

**rinu/4/EEE/mnm/robocar**

**fina/3/EEE/psg/papertalk**

**rachel/4/civil/kcg/quiz**

Print participant details

ParticipantName=rinu, Yearofstudy=4, Department=EEE, CollegeName=mnm, EventName=robocar, RegistrationFee=1000.0

ParticipantName=fina, Yearofstudy=3, Department=EEE, CollegeName=psg, EventName=papertalk, RegistrationFee=500.0

ParticipantName=rachel, Yearofstudy=4, Department=civil, CollegeName=kcg, EventName=quiz, RegistrationFee=300.0

Enter the event to search

**robocar**

Number of participants for ROBOCAR event is 1

**Sample Input/Output 2:**

Enter the number of entries

**3**

Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName

**rinu/4/EEE/mnm/robocar**

**fina/3/EEE/psg/papertalk**

**rachel/4/civil/kcg/quiz**

Print participant details

ParticipantName=rinu, Yearofstudy=4, Department=EEE, CollegeName=mnm, EventName=robocar, RegistrationFee=1000.0

ParticipantName=fina, Yearofstudy=3, Department=EEE, CollegeName=psg, EventName=papertalk, RegistrationFee=500.0

ParticipantName=rachel, Yearofstudy=4, Department=civil, CollegeName=kcg, EventName=quiz, RegistrationFee=300.0

Enter the event to search

**games**

No participant found

**Sample Input/Output 3:**

Enter the number of entries

**2**

Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName

**vishal/4/mech/vjc/flyingrobo**

**vivek/3/mech/hdl/games**

Event Name is invalid

## Automatic evaluation[+]

### TechnologyFest/src/EventManagement.java

```
import java.util.List;

public class EventManagement implements Runnable {
        private List<Participant> TechList;
        private String searchEvent;
        private int counter=0;
        public List<Participant>getTechList()
        {
            return TechList;

        }
        public void setTechList(List<Participant>techList)
        {
            TechList=techList;
        }
        public String getSearchEvent()
        {
            return searchEvent;
        }
    public void setSearchEvent(String searchEvent)
    {
        this.searchEvent=searchEvent;
    }
    public int getCounter()
    {
        return counter;
    }
    public void setCounter(int counter)
    {
        this.counter=counter;
    }
        //FILL THE CODE HERE
```

```java

        public void calculateRegistrationFee(List<Participant> list) throws InvalidEventException

        {
            for(Participant p:list)
            {
                if(p.getEventName().equalsIgnoreCase("robocar"))
                {
                    p.setRegistrationFee(1000);
                }
                else if(p.getEventName().equalsIgnoreCase("papertalk")){
                    p.setRegistrationFee(500);

                }

                else if(p.getEventName().equalsIgnoreCase("quiz")){
                    p.setRegistrationFee(300);
            }
            else if(p.getEventName().equalsIgnoreCase("games")){
                    p.setRegistrationFee(100);
            }
            else{
                throw new InvalidEventException("Event Name is Invalid");
            }
            }
                        //FILL THE CODE HERE
                        setTechList(list);
        }

        public void run()
        {
            String str="robocarpapertalkquizgames";
            if(str.contains(this.getSearchEvent())){
                for(Participant P:this.getTechList()){
                    if(this.getSearchEvent().equals(P.getEventName())){
                        counter++;
                    }
                }
            }
            setCounter(counter);

                        //FILL THE CODE HERE

        }
}

```

## TechnologyFest/src/InvalidEventException.java

```java
public class InvalidEventException extends Exception{
        //FILL THE CODE HERE
public InvalidEventException(String str){
    super(str);

}

}

```

## TechnologyFest/src/Main.java

```java

import java.util.Scanner;
import java.util.*;
public class Main {
        public static void main(String [] args)
        {
                        // CODE SKELETON - VALIDATION STARTS
            // DO NOT CHANGE THIS CODE

                        new SkeletonValidator();

            // CODE SKELETON - VALIDATION ENDS

            Scanner sc=new Scanner(System.in);
            System.out.println("Enter the number of entries");
            int n=sc.nextInt();
            System.out.println("Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName");
            List<Participant> list=new ArrayList<Participant>();
            String strlist[]=new String[n];
            for(int i=0;i<n;i++)
                        {
                            strlist[i]=sc.next();
                            String a[]=strlist[i].split("/");
                            Participant pt=new Participant(a[0],a[1],a[2],a[3],a[4]);
                            list.add(pt);
                        }
                        EventManagement em=new EventManagement();
                        try {
                          em.calculateRegistrationFee(list);
                        }
                        catch(InvalidEventException e)
                        {
                            e.printStackTrace();

                        }
                        System.out.println("Print participant details");
                        for(Participant p:list)
        {
            System.out.println(p);
        }
        System.out.println("Enter the event to search");
        String srch=sc.nextLine();
        em.setSearchEvent(srch);
        em.run();
        int count=em.getCounter();
        if(count<=0){
            System.out.println("No participant found");

        }
        else{
            System.out.println("Number of participants for"+srch+"event is "+count);          }
}
        }
```

## TechnologyFest/src/Participant.java

```java
public class Participant {
        private String name;
        private String yearofstudy;
        private String department;
        private String collegeName;
        private String eventName;
        private double registrationFee;

        //5 argument Constructor
```

```
	public Participant(String name, String yearofstudy, String department, String collegeName, String eventName) {
		super();
		this.name = name;
		this.yearofstudy = yearofstudy;
		this.department = department;
		this.collegeName = collegeName;
		this.eventName = eventName;
	}

	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getYearofstudy() {
		return yearofstudy;
	}
	public void setYearofstudy(String yearofstudy) {
		this.yearofstudy = yearofstudy;
	}
	public String getDepartment() {
		return department;
	}
	public void setDepartment(String department) {
		this.department = department;
	}
	public String getCollegeName() {
		return collegeName;
	}
	public void setCollegeName(String collegeName) {
		this.collegeName = collegeName;
	}
	public String getEventName() {
		return eventName;
	}
	public void setEventName(String eventName) {
		this.eventName = eventName;
	}
	public double getRegistrationFee() {
		return registrationFee;
	}
	public void setRegistrationFee(double registrationFee) {
		this.registrationFee = registrationFee;
	}

	@Override
	public String toString() {
		return "Participant [name=" + name + ", yearofstudy=" + yearofstudy + ", department=" + department
				+ ", collegeName=" + collegeName + ", eventName=" + eventName + ", registrationFee=" + registrationFee
				+ "]";
	}




}

```

## TechnologyFest/src/SkeletonValidator.java

```

import java.lang.reflect.Method;
```

```java
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.stream.Stream;

/**
 * @author TJ
 *
 * 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() {

		//classes
		validateClassName("Main");
		validateClassName("EventManagement");
		validateClassName("Participant");
		validateClassName("InvalidEventException");
		//functional methods
		validateMethodSignature(
				"calculateRegistrationFee:void","EventManagement");
		validateMethodSignature(
				"run:void","EventManagement");

		//setters and getters of HallHandler
		validateMethodSignature(
				"getTechList:List","EventManagement");
		validateMethodSignature(
				"setTechList:void","EventManagement");

		validateMethodSignature(
				"getCounter:int","EventManagement");
		validateMethodSignature(
				"setCounter:void","EventManagement");

		validateMethodSignature(
				"getSearchEvent:String","EventManagement");
		validateMethodSignature(
				"setSearchEvent:void","EventManagement");

		//setters and getters of Hall
		validateMethodSignature(
				"getName:String","Participant");
		validateMethodSignature(
				"setName:void","Participant");

		validateMethodSignature(
				"getYearofstudy:String","Participant");
		validateMethodSignature(
				"setYearofstudy:void","Participant");

		validateMethodSignature(
				"getDepartment:String","Participant");
		validateMethodSignature(
				"setDepartment:void","Participant");

		validateMethodSignature(
				"getCollegeName:String","Participant");
		validateMethodSignature(
				"setCollegeName:void","Participant");

		validateMethodSignature(
				"getEventName:String","Participant");
		validateMethodSignature(
                "setEventName:void","Participant");

        validateMethodSignature(
                "getRegistrationFee:double","Participant");
        validateMethodSignature(
                "setRegistrationFee:void","Participant");

    }

    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().contains(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");
        }
    }

}
```

## Grade

Reviewed on Monday, 7 February 2022, 6:34 PM by Automatic grade

**Grade** 74 / 100

**Assessment report**

```
Fail 1 -- test4CheckTheOutput::
$Expected output:"[Print participant details
 ParticipantName=Weni
 Yearofstudy=3
 Department=civil
 CollegeName=vjc
 EventName=robocar
 RegistrationFee=1000.0
 ParticipantName=gina
 Yearofstudy=2
 Department=mech
 CollegeName=vjc
 EventName=quiz
 RegistrationFee=300.0
 ParticipantName=jos
 Yearofstudy=4
 Department=ece
 CollegeName=vjec
 EventName=games
 RegistrationFee=100.0
 ParticipantName=fida
 Yearofstudy=1
 Department=eee
 CollegeName=vjec
 EventName=papertalk
 RegistrationFee=500.0
 Enter the event to search
 Number of participants for PAPERTALK event is 1]" Actual output:"[Enter the number of
entries
```

```
 Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName
 Print participant details
 Participant [name=Weni
 yearofstudy=3
 department=civil
 collegeName=vjc
 eventName=robocar
 registrationFee=1000.0]
 Participant [name=gina
 yearofstudy=2
 department=mech
 collegeName=vjc
 eventName=quiz
 registrationFee=300.0]
 Participant [name=jos
 yearofstudy=4
 department=ece
 collegeName=vjec
 eventName=games
 registrationFee=100.0]
 Participant [name=fida
 yearofstudy=1
 department=eee
 collegeName=vjec
 eventName=papertalk
 registrationFee=500.0]
 Enter the event to search
 No participant found]"$
 Check your code with the input :Weni/3/civil/vjc/robocar
gina/2/mech/vjc/quiz
jos/4/ece/vjec/games
fida/1/eee/vjec/papertalk



Fail 2 -- test6CheckTheOutputfor_NCount::
$Expected output:"[Print participant details
 ParticipantName=philip
 Yearofstudy=4
 Department=eee
 CollegeName=mvc
 EventName=robocar
 RegistrationFee=1000.0
 ParticipantName=susan
 Yearofstudy=4
 Department=eee
 CollegeName=mvc
 EventName=robocar
 RegistrationFee=1000.0
 ParticipantName=vivek
 Yearofstudy=3
 Department=civil
 CollegeName=mvc
 EventName=quiz
 RegistrationFee=300.0
 ParticipantName=vishal
 Yearofstudy=3
 Department=civil
 CollegeName=mvc
 EventName=papertalk
 RegistrationFee=500.0
 Enter the event to search
 Number of participants for ROBOCAR event is 2]" Actual output:"[Enter the number of
entries
 Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName
 Print participant details
 Participant [name=philip
 yearofstudy=4
 department=eee
 collegeName=mvc
 eventName=robocar
 registrationFee=1000.0]
 Participant [name=susan
 yearofstudy=4
 department=eee
 collegeName=mvc
 eventName=robocar
 registrationFee=1000.0]
 Participant [name=vivek
 yearofstudy=3
 department=civil
 collegeName=mvc
 eventName=quiz
 registrationFee=300.0]
 Participant [name=vishal
 yearofstudy=3
 department=civil
 collegeName=mvc
 eventName=papertalk
 registrationFee=500.0]
 Enter the event to search
 No participant found]"$
 Check your code with the input :philip/4/eee/mvc/robocar
susan/4/eee/mvc/robocar
vivek/3/civil/mvc/quiz
vishal/3/civil/mvc/papertalk
robocar
```

**Obtained Pass Percentage. Still few testcases failed . Kindly revisit the Solution**

**[+]Grading and Feedback**

======================================================================