**Group -1**

## AirVoice - Registration

Grade settings: Maximum grade: 100

Run: Yes Evaluate: Yes

Automatic grade: Yes Maximum execution time: 16 s

SmartBuy is a leading mobile shop in the town. After buying a product, the customer needs to provide a few personal details for the invoice to be generated.

You being their software consultant have been approached to develop software to retrieve the personal details of the customers, which will help them to generate the invoice faster.

Component Specification: Customer

| Type(Class) | Attributes | Methods | Responsibilities |
| --- | --- | --- | --- |
| Customer | String customerName  long contactNumber  String emailId  int age | Include the getters and setters method for all the attributes. |  |

In the Main class, create an object for the Customer class.

Get the details as shown in the sample input and assign the value for its attributes using the setters.

Display the details as shown in the sample output using the getters method.

All classes and methods should be public, Attributes should be private.

Note:

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

Ensure to follow the object oriented specifications provided in the question.

Ensure to provide the names for classes, attributes and methods as specified in the question.

Adhere to the code template, if provided.

Sample Input 1:

Enter the Name:

john

Enter the ContactNumber:

9874561230

Enter the EmailId:

john@gmail.com

Enter the Age:

32

Sample Output 1:

Name:john

ContactNumber:9874561230

EmailId:john@gmail.com

Age:32

### Automatic evaluation[[+]](about:blank)

#### Customer.java

1 *public* *class* Customer {

2 *private* String customerName;

3

4 *private* *long* contactNumber;

5

6 *private* String emailId;

7

8 *private* *int* age;

9

10 *public* String getCustomerName() {

11 *return* customerName;

12 }

13

14 *public* *void* setCustomerName(String customerName) {

15 *this*.customerName = customerName;

16 }

17

18 *public* *long* getContactNumber() {

19 *return* contactNumber;

20 }

21

22 *public* *void* setContactNumber(*long* contactNumber) {

23 *this*.contactNumber = contactNumber;

24 }

25

26 *public* String getEmailId() {

27 *return* emailId;

28 }

29

30 *public* *void* setEmailId(String emailId) {

31 *this*.emailId = emailId;

32 }

33

34 *public* *int* getAge() {

35 *return* age;

36 }

37

38 *public* *void* setAge(*int* age) {

39 *this*.age = age;

40 }

41

42

43

44 }

45

#### Main.java

1 *import* java.util.Scanner;

2

3 *public* *class* Main {

4

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

6 // TODO Auto-generated method stub

7 Scanner sc=*new* Scanner(System.in);

8 Customer c=*new* Customer();

9 System.out.println("Enter the Name:");

10 String name=(sc.nextLine());

11 System.out.println("Enter the ContactNumber:");

12 *long* no=sc.nextLong();

13 sc.nextLine();

14 System.out.println("Enter the EmailId:");

15 String mail=sc.nextLine();

16

17 System.out.println("Enter the Age:");

18 *int* age=sc.nextInt();

19 c.setCustomerName(name);

20 c.setContactNumber(no);

21 c.setEmailId(mail);

22 c.setAge(age);

23 System.out.println("Name:"+c.getCustomerName());

24 System.out.println("ContactNumber:"+c.getContactNumber());

25 System.out.println("EmailId:"+c.getEmailId());

26 System.out.println("Age:"+c.getAge());

27

28

29

30 }

31

32 }

## Grade

Reviewed on Monday, 7 February 2022, 4:45 PM by Automatic grade  
**Grade** 100 / 100  
**Assessment report**  
[[+]](about:blank)**Grading and Feedback**

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

1. 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  int chequeAmount  Date dateOfIssue | Include a public getter and setter method for all the attributes. |  |
| Make payment for EMI amount | Cheque |  | public boolean payAmount() | This is an overridden method of the parent class. It should return true if the cheque is valid and the amount is valid. Else return false. |

**Note:**

·        The cheque is valid for 6 months from the date of issue.

·        Assume the current date is 01-01-2020 in dd-MM-yyyy format.

·        The chequeAmount is valid if it is greater than or equal to the dueAmount.

**Component Specification: Cash** **class** (Needs to be a child of Payment class)

| **Component Name** | **Type(Class)** | **Attributes** | **Methods** | **Responsibilities** |
| --- | --- | --- | --- | --- |
| Make payment for EMI amount | Cash | int cashAmount | Include a public getter and setter method for the attribute. |  |
| Make payment for EMI amount | Cash |  | public boolean payAmount() | This is an overridden method of the parent class. It should return true if the cashAmount is greater than or equal to the dueAmount. Else return false. |

**Component Specification: Credit** **class** (Needs to be a child of Payment class)

| **Component Name** | **Type (Class)** | **Attributes** | **Methods** | **Responsibilities** |
| --- | --- | --- | --- | --- |
| Make payment for EMI amount | Credit | int creditCardNo    String cardType    int creditCardAmount | Include a public getter and setter method for all the attributes. |  |
| Make payment for EMI amount | Credit |  | public boolean payAmount() | This is an overridden method of the parent class. It should deduct the   dueAmount and service tax from the creditCardAmount and return true if the credit card payment was done successfully. Else return false. |

**Note:**

·        The payment can be done if the credit card amount is greater than or equal to the sum of due amount and service tax. Else payment cannot be made.

·        The cardType can be “silver” or “gold” or “platinum”. Set the creditCardAmount based on the cardType.

·        Also service tax is calculated on dueAmount based on cardType.

| **Credit Card Type** | **Credit Card Amount** | **Service Tax** |
| --- | --- | --- |
| silver | 10000 | 2% of the due amount |
| gold | 50000 | 5% of the due amount |
| platinum | 100000 | 10% of the due amount |

·        The boolean  payAmount() method should deduct the due amount and the service tax amount from a credit card. If the creditCardAmount is less than the dueAmount+serviceTax, then the payment cannot be made.

·        The balance in credit card amount after a successful payment should be updated in the creditCardAmount by deducting the sum of dueAmount and serviceTax from creditCardAmount itself.

**Component Specification: Bill** **class**

| **Component Name** | **Type(Class)** | **Attributes** | **Methods** | **Responsibilities** |
| --- | --- | --- | --- | --- |
| Payment Status Report | Bill |  | public String processPayment (Payment obj) | This method should return a message based on the status of the payment made. |

**Note:**

·        If the payment is successful, processPayment method should return a message “Payment done successfully via cash” or “Payment done successfully via cheque” or “Payment done successfully via creditcard. Remaining amount in your <<cardType>> card is <<balance in CreditCardAmount>>”

·        If the payment is a failure, then return a message “Payment not done and your due amount is <<dueAmount>>”

Create a **public class Main** with the main method to test the application.

**Note:**

·        Assume the current date as 01-01-2020 in dd-MM-yyyy format.

·        In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

·        Ensure to follow the object oriented specifications provided in the question.

·        Ensure to provide the names for classes, attributes and methods as specified in the question.

·        Adhere to the code template, if provided.

·        Adhere to the sample input and output.

**Sample Input 1:**

Enter the due amount:

**3000**  
Enter the mode of payment(cheque/cash/credit):

**cash**  
Enter the cash amount:

**2000**

**Sample Output 1:**

Payment not done and your due amount is 3000

**Sample Input 2:**

Enter the due amount:

**3000**  
Enter the mode of payment(cheque/cash/credit):

**cash**  
Enter the cash amount:

**3000**

**Sample Output 2:**

Payment done successfully via cash

**Sample Input 3:**

Enter the due amount:

**3000**  
Enter the mode of payment(cheque/cash/credit):

**cheque**  
Enter the cheque number:

**123**  
Enter the cheque amount:

**3000**  
Enter the date of issue:

**21-08-2019**

**Sample Output 3:**

Payment done successfully via cheque

**Sample Input 4:**

Enter the due amount:

**3000**  
Enter the mode of payment(cheque/cash/credit):

**credit**  
Enter the credit card number:

**234**  
Enter the card type(silver,gold,platinum):

**silver**

**Sample Output 4:**

Payment done successfully via credit card. Remaining amount in your silver card is 6940

Automatic evaluation[[+]](about:blank)

Main.java

1 *import* java.text.ParseException;

2 *import* java.text.SimpleDateFormat;

3 *import* java.util.Date;

4 *import* java.util.Scanner;

5 *public* *class* Main {

6

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

8

9 Scanner sc=*new* Scanner(System.in);

10 System.out.println("Enter the due amount:");

11 *int* dueAmount=sc.nextInt();

12

13 System.out.println("Enter the mode of payment(cheque/cash/credit):");

14 String mode=sc.next();

15 Bill b = *new* Bill();

16 *if*(mode.equals("cheque"))

17 {

18 System.out.println("enter the cheque number:");

19 String chequeNumber=sc.next();

20 System.out.println("enter the cheque amount:");

21 *int* chequeAmount=sc.nextInt();

22 System.out.println("enter the date of issue:");

23 String date=sc.next();

24 SimpleDateFormat dateFormat = *new* SimpleDateFormat("dd-MM-yyyy");

25 Date dateOfIssue=*null*;

26 *try*

27 {

28 dateOfIssue = dateFormat.parse(date);

29 }

30 *catch* (ParseException e)

31 {

32

33 }

34 Cheque cheque= *new* Cheque();

35 cheque.setChequeNo(chequeNumber);

36 cheque.setChequeAmount(chequeAmount);

37 cheque.setDateOfIssue(dateOfIssue);

38 cheque.setDueAmount(dueAmount);

39 System.out.println(b.processPayment(cheque));

40 }

41 *else* *if*(mode.equals("cash"))

42 {

43 System.out.println("enter the cash amount:");

44 *int* CashAmount=sc.nextInt();

45 Cash cash=*new* Cash();

46 cash.setCashAmount(CashAmount);

47 cash.setDueAmount(dueAmount);

48 System.out.println(b.processPayment(cash));

49 }

50 *else* *if*(mode.equals("credit"))

51 {

52 System.out.println("enter the credit card number:");

53 *int* creditCardNumber=sc.nextInt();

54 System.out.println("enter the card type:");

55 String cardType=sc.next();

56

57 Credit credit=*new* Credit();

58 credit.setCreditCardNo(creditCardNumber);

59 credit.setCardType(cardType);

60 credit.setDueAmount(dueAmount);

61 System.out.println(b.processPayment(credit));

62 }

63 }

64 }

Payment.java

1 *public* *class* Payment {

2 *private* *int* dueAmount;

3

4 *public* *boolean* payAmount()

5 {

6 *if*(dueAmount == 0)

7 *return* *true*;

8 *else*

9 *return* *false*;

10 }

11

12 *public* *int* getDueAmount() {

13 *return* dueAmount;

14 }

15

16 *public* *void* setDueAmount(*int* dueAmount) {

17 *this*.dueAmount = dueAmount;

18 }

19 }

Cheque.java

1 *import* java.text.ParseException;

2 *import* java.text.SimpleDateFormat;

3 *import* java.util.Date;

4 *public* *class* Cheque *extends* Payment {

5 String chequeNo;

6 *int* chequeAmount;

7 Date dateOfIssue;

8 *public* String getChequeNo() {

9 *return* chequeNo;

10 }

11 *public* *void* setChequeNo(String chequeNo) {

12 *this*.chequeNo = chequeNo;

13 }

14 *public* *int* getChequeAmount() {

15 *return* chequeAmount;

16 }

17 *public* *void* setChequeAmount(*int* chequeAmount) {

18 *this*.chequeAmount = chequeAmount;

19 }

20 *public* Date getDateOfIssue() {

21 *return* dateOfIssue;

22 }

23 *public* *void* setDateOfIssue(Date dateOfIssue) {

24 *this*.dateOfIssue = dateOfIssue;

25 }

26

27 @Override

28 *public* *boolean* payAmount()

29 {

30 SimpleDateFormat format = *new* SimpleDateFormat("dd-MM-yyyy");

31 Date today = *new* Date();

32 *try*

33 {

34 today = format.parse("01-01-2020");

35 }

36 *catch* (ParseException e)

37 {

38 *return* *false*;

39 }

40 *long* diff = today.getTime()-dateOfIssue.getTime();

41 *int* day = (*int*) Math.abs(diff/(1000\*60\*60\*24));

42 *int* month = day/30;

43 *if*(month <=6)

44 {

45

46 *if*(chequeAmount>=getDueAmount())

47 {

48 *return* *true*;

49 }

50 *else*

51 *return* *false*;

52

53 }

54 *else*

55 *return* *false*;

56 }

57

58

59 }

Cash.java

1 *public* *class* Cash *extends* Payment {

2 *int* cashAmount;

3

4 *public* *int* getCashAmount() {

5 *return* cashAmount;

6 }

7

8 *public* *void* setCashAmount(*int* cashAmount) {

9 *this*.cashAmount = cashAmount;

10 }

11

12 @Override

13 *public* *boolean* payAmount()

14 {

15 *if*(cashAmount>=getDueAmount())

16 *return* *true*;

17 *else*

18 *return* *false*;

19 }

20

21

22 }

Credit.java

1 *public* *class* Credit *extends* Payment {

2 *int* creditCardNo;

3 String cardType;

4 *int* creditCardAmount;

5 *public* *int* getCreditCardNo() {

6 *return* creditCardNo;

7 }

8 *public* *void* setCreditCardNo(*int* creditCardNo) {

9 *this*.creditCardNo = creditCardNo;

10 }

11 *public* String getCardType() {

12 *return* cardType;

13 }

14 *public* *void* setCardType(String cardType) {

15 *this*.cardType = cardType;

16 }

17 *public* *int* getCreditCardAmount() {

18 *return* creditCardAmount;

19 }

20 *public* *void* setCreditCardAmount(*int* creditCardAmount) {

21 *this*.creditCardAmount = creditCardAmount;

22 }

23

24

25 @Override

26 *public* *boolean* payAmount()

27 {

28 *int* netAmount = 0;

29 *if*(cardType.equals("silver"))

30 {

31 netAmount = (*int*) (getDueAmount()\*1.02);

32 creditCardAmount = 10000;

33 }

34 *else* *if*(cardType.equals("gold"))

35 {

36 netAmount = (*int*) (getDueAmount()\*1.05);

37 creditCardAmount = 50000;

38 }

39 *else* *if*(cardType.equals("platinum"))

40 {

41 netAmount = (*int*) (*int*) (getDueAmount()\*1.1);

42 creditCardAmount = 100000;

43 }

44

45 *if*(creditCardAmount>=netAmount)

46 {

47 creditCardAmount = creditCardAmount - netAmount;

48 *return* *true*;

49 }

50 *else*

51 *return* *false*;

52 }

53

54

55 }

Bill.java

1 *public* *class* Bill {

2 *public* String processPayment(Payment obj)

3 {

4 String res="";

5 *if*(obj *instanceof* Cheque)

6 {

7 *if*(obj.payAmount())

8 res = "Payment done successfully via cheque";

9 *else*

10 res = "Payment not done and your due amount is "+obj.getDueAmount();

11 }

12 *else* *if*(obj *instanceof* Cash)

13 {

14 *if*(obj.payAmount())

15 res = "Payment done successfully via cash";

16 *else*

17 res = "Payment not done and your due amount is "+obj.getDueAmount();

18 }

19 *else* *if*(obj *instanceof* Credit)

20 {

21 Credit c = (Credit) obj;

22 *if*(obj.payAmount())

23 res = "Payment done successfully via credit card. Remaining amount in your "+c.getCardType()+" card is "+c.getCreditCardAmount();

24 *else*

25 res = "Payment not done and your due amount is "+obj.getDueAmount();

26 }

27 *return* res;

28 }

29 }

Grade

Reviewed on Wednesday, 1 December 2021, 10:08 PM by Automatic grade  
**Grade** 100 / 100  
**Assessment report**  
TEST CASE PASSED  
[[+]](about:blank)**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[[+]](about:blank)

#### Main.java

1 *import* java.util.\*;

2 *public* *class* Main

3 {

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

5 {

6 Scanner sc=*new* Scanner(System.in);

7 //Fill the code

8 *int* m=sc.nextInt();

9 *if*(m<=0){

10 System.out.println(""+m+" is an invalid");

11 *return*;

12 }

13 *int* n=sc.nextInt();

14 *if*(n<=0){

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

16 *return*;

17 }

18 *if*(m>=n){

19 System.out.println(""+m+" is not less than "+n);

20 *return*;

21 }

22 *for*(*int* i=1;i<=n;i++){

23 System.out.print((*int*)Math.pow(m,i)+"");

24 }

25 }

26 }

## Grade

Reviewed on Monday, 7 February 2022, 4:46 PM by Automatic grade  
**Grade** 100 / 100  
**Assessment report**  
TEST CASE PASSED  
[[+]](about:blank)**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  double balanceAmount | Include the getters and setters method for all the attributes.  Include a    parametrized constructor of two arguments in the order – accountNumber,balanceAmount to 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  This method should perform the deposit,by adding the deposited amount to the balanceAmount |

**Requirement 2: Being able to withdraw money from the account anytime**

As per this requirement, the customer should be able to withdraw money from his account anytime he wants. The amount to be withdrawn should be less than or equal to the balance in the account. After the withdrawal, the account should reflect the balance amount

**Component Specification:** **Account**

| **Component Name** | **Type(Class)** | **Methods** | **Responsibilities** |
| --- | --- | --- | --- |
| Withdraw amount from an account | Account | public boolean withdraw(double withdrawAmt) | This method should take the amount to be withdrawn as an argument.  This method should check the balanceAmount and deduct the withdraw amount from the balanceAmount and return true. If there is insufficient balance then return false. |

In the **Main**class, Get the details as shown in the sample input.

Create an object for the Account class and invoke the deposit method to deposit the amount and withdraw method to withdraw the amount from the account.

All classes and methods should be public, Attributes should be private.

**Note:**

Balance amount should be displayed corrected to 2 decimal places.

Order of the transactions to be performed (Display,Deposit,Withdraw).

If the balance amount is insufficient then display the message as shown in the Sample Input / Output.

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.  
  
Ensure to follow the object-oriented specifications provided in the question.

Ensure to provide the names for classes, attributes, and methods as specified in the question.

Adhere to the code template, if provided.

**Sample Input/Output 1:**

Enter the account number:

**1234567890**

Enter the available amount in the account:

**15000**

Enter the amount to be deposited:

**1500**

Available balance is:16500.00

Enter the amount to be withdrawn:

**500**

Available balance is:16000.00

**Sample Input/Output 2:**

Enter the account number:

**1234567890**

Enter the available amount in the account:

**15000**

Enter the amount to be deposited:

**1500**

Available balance is:16500.00

Enter the amount to be withdrawn:

**18500**

Insufficient balance

Available balance is:16500.00

### Automatic evaluation[[+]](about:blank)

#### Main.java

1 *import* java.text.DecimalFormat;

2 *import* java.util.Scanner;

3 *import* java.util.Scanner;

4

5

6 *public* *class* Main{

7 *static* Account ac=*new* Account(0, 0);

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

9 Scanner sc=*new* Scanner(System.in);

10 System.out.println("Enter the account number:");

11 ac.setAccountNumber(sc.nextLong());

12 System.out.println("Enter the available amount in the account:");

13 ac.setBalanceAmount(sc.nextDouble());

14 System.out.println("Enter the amount to be deposited:");

15 ac.deposit(sc.nextDouble());

16 System.out.printf("Available balance is:%.2f",ac.getBalanceAmount());

17 System.out.println();

18 System.out.println("Enter the amount to be withdrawn:");

19 ac.withdraw(sc.nextDouble());

20 System.out.printf("Available balance is:%.2f",ac.getBalanceAmount());

21 //Fill the code

22 }

23 }

24

25

26

#### Account.java

1

2 *public* *class* Account {

3 *long* accountNumber;

4 *double* balanceAmount;

5

6

7 *public* Account(*long* accno, *double* bal){

8 *super*();

9 *this*.accountNumber=accno;

10 *this*.balanceAmount=bal;

11 }

12 *public* *long* getAccountNumber(){

13 *return* accountNumber;

14 }

15 *public* *void* setAccountNumber(*long* accno){

16 *this*.accountNumber=accno;

17 }

18 *public* *double* getBalanceAmount(){

19 *return* balanceAmount;

20 }

21 *public* *void* setBalanceAmount(*double* bal) {

22 *this*.balanceAmount=bal;

23 }

24 *public* *void* deposit(*double* depositAmt){

25 *float* total=(*float*)(balanceAmount+depositAmt);

26 balanceAmount=total;

27 }

28 *public* *boolean* withdraw(*double* withdrawAmt){

29 *float* total;

30 *if*(withdrawAmt>balanceAmount){

31 System.out.println("Insufficient balance");

32

33 *return* *false*;

34 }*else*{

35 total=(*float*)(balanceAmount-withdrawAmt);

36 setBalanceAmount(total);

37 *return* *true*;

38 }

39 }

40 }

## Grade

Reviewed on Monday, 7 February 2022, 4:47 PM by Automatic grade  
**Grade** 100 / 100  
**Assessment report**  
[[+]](about:blank)**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[[+]](about:blank)

#### Main.java

1 *import* java.util.Scanner;

2 *import* java.lang.String.\*;

3 *import* java.util.\*;

4 *public* *class* Main{

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

6 String[] words;

7 Scanner read =*new* Scanner(System.in);

8 String sentence=read.nextLine();

9 words=sentence.split(" ");

10 *if*(words.length<3)

11 System.out.println("Invalid Sentence");

12 *else*{

13 String a=words[0].substring(0,1);

14 String b=words[1].substring(0,1);

15 String c=words[2].substring(0,1);

16 *if*(a.equalsIgnoreCase(b)&&b.equalsIgnoreCase(c))

17 {

18 StringBuilder k= *new* StringBuilder();

19 k.append(words[words.length-1]);

20 k=k.reverse();

21 k.append(words[0]);

22 System.out.println(k);

23 }

24 *else*{

25 StringBuilder k = *new* StringBuilder();

26 k.append(words[0]);

27 k=k.reverse();

28 k.append(words[words.length-1]);

29 System.out.println(k);

30 }

31 }

32 }

33 }

## Grade

Reviewed on Monday, 7 February 2022, 5:12 PM by Automatic grade  
**Grade** 90 / 100  
**Assessment report**

*Fail 1 -- test5\_CheckForTheSentenceContainsOtherThanAlphabets::*

*$Expected output:"[Invalid Word]" Actual output:"[tahWme]"$*

[[+]](about:blank)**Grading and Feedback**

6. Dominion cinemas

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

Dominion cinema is a famous theatre in the city. It has different types of seat tiers – Platinum, Gold and Silver. So far the management was manually calculating the ticket cost for all their customers which proved very hectic and time consuming. Going forward they want to calculate ticket cost using their main computer. Assist them in calculating and retrieving the amount to be paid by the Customer.

**Requirements 1: Calculation of Ticket Cost**

The application needs to calculate the ticket cost to be paid by the Customer according to the seat tier.

**Component Specification: BookAMovieTicket Class**(Parent Class)

| **Component Name** | **Type(Class)** | **Attributes** | **Methods** | **Responsibilities** |
| --- | --- | --- | --- | --- |
| Calculation of Ticket cost | BookAMovieTicket | String ticketId  String customerName  long mobileNumber  String emailId  String movieName | Public getter and setter method for all the attributes and 5 argument constructor in the given order - ticketId, customerName, mobileNumber, emailId, movieName are provided as a part of the code skeleton. |  |

**Note:**

* The attributes of the BookAMovieTicketclass 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[[+]](about:blank)

#### BookAMovieTicket.java

1

2 *public* *class* BookAMovieTicket {

3

4 *protected* String ticketId;

5 *protected* String customerName;

6 *protected* *long* mobileNumber;

7 *protected* String emailId;

8 *protected* String movieName;

9

10 *public* String getTicketId() {

11 *return* ticketId;

12 }

13 *public* *void* setTicketId(String ticketId) {

14 *this*.ticketId = ticketId;

15 }

16 *public* String getCustomerName() {

17 *return* customerName;

18 }

19 *public* *void* setCustomerName(String customerName) {

20 *this*.customerName = customerName;

21 }

22 *public* *long* getMobileNumber() {

23 *return* mobileNumber;

24 }

25 *public* *void* setMobileNumber(*long* mobileNumber) {

26 *this*.mobileNumber = mobileNumber;

27 }

28 *public* String getEmailId() {

29 *return* emailId;

30 }

31 *public* *void* setEmailId(String emailId) {

32 *this*.emailId = emailId;

33 }

34 *public* String getMovieName() {

35 *return* movieName;

36 }

37 *public* *void* setMovieName(String movieName) {

38 *this*.movieName = movieName;

39 }

40

41 *public* BookAMovieTicket(String ticketId, String customerName, *long* mobileNumber, String emailId, String movieName) {

42 *this*.ticketId = ticketId;

43 *this*.customerName = customerName;

44 *this*.mobileNumber = mobileNumber;

45 *this*.emailId = emailId;

46 *this*.movieName = movieName;

47

48 }

49

50

51

52 }

53

#### GoldTicket.java

1

2 *public* *class* GoldTicket *extends* BookAMovieTicket{

3 *public* GoldTicket(String ticketId,String customerName, *long* mobileNumber,

4 String emailId, String movieName){

5 *super*(ticketId, customerName, mobileNumber, emailId, movieName);

6 }

7

8 *public* *boolean* validateTicketId(){

9 *int* count=0;

10 *if*(ticketId.contains("GOLD"));

11 count++;

12 *char*[] cha=ticketId.toCharArray();

13 *for*(*int* i=4;i<7;i++){

14 *if*(cha[i]>='1'&& cha[i]<='9')

15 count++;

16 }

17 *if*(count==4)

18 *return* *true*;

19 *else*

20 *return* *false*;

21 }

22

23

24 // Include Constructor

25

26 *public* *double* calculateTicketCost(*int* numberOfTickets, String ACFacility){

27 *double* amount;

28 *if*(ACFacility.equals("yes")){

29 amount=500\*numberOfTickets;

30 }

31 *else*{

32 amount=350\*numberOfTickets;

33 }

34

35 *return* amount;

36 }

37

38 }

#### PlatinumTicket.java

1 *public* *class* PlatinumTicket *extends* BookAMovieTicket{

2 *public* PlatinumTicket(String ticketId, String customerName, *long* mobileNumber,

3 String emailId, String movieName){

4 *super*(ticketId, customerName, mobileNumber, emailId, movieName);

5 }

6

7 *public* *boolean* validateTicketId(){

8 *int* count=0;

9 *if*(ticketId.contains("PLATINUM"));

10 count++;

11 *char*[] cha=ticketId.toCharArray();

12 *for*(*int* i=8;i<11;i++){

13 *if*(cha[i]>='1'&& cha[i]<='9')

14 count++;

15 }

16 *if*(count==4)

17 *return* *true*;

18 *else*

19 *return* *false*;

20 }

21

22 // Include Constructor

23

24 *public* *double* calculateTicketCost(*int* numberOfTickets, String ACFacility){

25 *double* amount;

26 *if*(ACFacility.equalsIgnoreCase("yes")){

27 amount=750\*numberOfTickets;

28 }

29 *else*{

30 amount=600\*numberOfTickets;

31 }

32

33 *return* amount;

34 }

35

36 }

37

#### SilverTicket.java

1

2 *public* *class* SilverTicket *extends* BookAMovieTicket{

3 *public* SilverTicket(String ticketId, String customerName, *long* mobileNumber,

4 String emailId, String movieName){

5 *super*(ticketId, customerName, mobileNumber, emailId, movieName);

6 }

7

8 *public* *boolean* validateTicketId(){

9 *int* count=0;

10 *if*(ticketId.contains("SILVER"));

11 count++;

12 *char*[] cha=ticketId.toCharArray();

13 *for*(*int* i=6;i<9;i++){

14 *if*(cha[i]>='1'&& cha[i]<='9')

15 count++;

16 }

17 *if*(count==4)

18 *return* *true*;

19 *else*

20 *return* *false*;

21 }

22

23 // Include Constructor

24

25 *public* *double* calculateTicketCost(*int* numberOfTickets, String ACFacility){

26 *double* amount;

27 *if*(ACFacility.equals("yes")){

28 amount=250\*numberOfTickets;

29 }

30 *else*{

31 amount=100\*numberOfTickets;

32 }

33

34 *return* amount;

35 }

36

37 }

38

#### UserInterface.java

1 *import* java.util.\*;

2

3 *public* *class* UserInterface {

4

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

6 Scanner sc=*new* Scanner(System.in);

7 System.out.println("Enter Ticket Id");

8 String tid=sc.next();

9 System.out.println("Enter Customer name");

10 String cnm=sc.next();

11 System.out.println("Enter Mobile number");

12 *long* mno=sc.nextLong();

13 System.out.println("Enter Email id");

14 String email=sc.next();

15 System.out.println("Enter Movie name");

16 String mnm=sc.next();

17 System.out.println("Enter number of tickets");

18 *int* tno=sc.nextInt();

19 System.out.println("Do you want AC or not");

20 String choice =sc.next();

21 *if*(tid.contains("PLATINUM")){

22 PlatinumTicket PT= *new* PlatinumTicket(tid,cnm,mno,email,mnm);

23 *boolean* b1=PT.validateTicketId();

24 *if*(b1==*true*){

25 *double* cost=PT.calculateTicketCost(tno, choice);

26 System.out.println("Ticket cost is "+String.format("%.2f",cost));

27 }

28 *else* *if*(b1==*false*){

29 System.out.println("Provide valid Ticket Id");

30 System.exit(0);

31 }

32 }

33 *else* *if*(tid.contains("GOLD")){

34 GoldTicket GT= *new* GoldTicket(tid,cnm,mno,email,mnm);

35 *boolean* b2=GT.validateTicketId();

36 *if*(b2==*true*){

37 *double* cost=GT.calculateTicketCost(tno,choice);

38 System.out.println("Ticket cost is "+String.format("%.2f",cost));

39 }

40 *else* *if* (b2==*false*){

41 System.out.println("Provide valid Ticket Id");

42 System.exit(0);

43 }

44 }

45 *else* *if*(tid.contains("SILVER")){

46 SilverTicket ST= *new* SilverTicket(tid,cnm,mno,email,mnm);

47 *boolean* b3=ST.validateTicketId();

48 *if*(b3==*true*){

49 *double* cost=ST.calculateTicketCost(tno,choice);

50 System.out.println("Ticket cost is "+String.format("%.2f",cost));

51 }

52 *else* *if* (b3==*false*){

53 System.out.println("Provide valid Ticket Id");

54 System.exit(0);

55 }

56 }

57 }

58 }

59

60

## Grade

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

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