1.Electricity Bill

Sam seems to be a busy man with very little time to spare for personal work like paying his electricity bill. The due date for payment of the electricity bill is nearing and as he is busy, decides to pay it online this time.

Sam calculates the bill according to the Unit Slab rates listed in the following table:

|  |  |
| --- | --- |
| Unit Slabs | Tariff Rates |
| First 50 Units | 2.60 |
| Next100 Units | 3.25 |
| Next 200 Units | 5.26 |
| Next 1000 Units | 7.75 |

Write JUNIT test class named “EBBillTest” to test the method calculateBillAmount() in the given EBBill class.

You need to write only JUNIT code.

**Note:** Do not alter the code present inside the EBBill class.

//Solution

#### **ElectricityBill/src/EBBill.java**

1

2 *import* java.util.Scanner;

3

4 *public* *class* EBBill {

5 *private* *int* units;

6

7 *public* EBBill(*int* units)

8 {

9 *this*.units = units;

10 }

11 *public* *double* calculateBillAmount()

12 {

13 *double* amount=0.0;

14 *if*(units > 0){

15 *if*(units <= 50)

16 {

17 amount = units \* 2.60;

18 }

19 *else* *if* (units <= 100)

20 {

21 amount = 130 + ((units - 50) \* 3.25);

22 }

23 *else* *if*(units <= 200)

24 {

25 amount = 130 + 162.50 + ((units - 100 ) \* 5.26);

26 }

27 *else* *if*(units <= 1000)

28 {

29 amount = 130 + 162.50 + 526 + ((units - 200 ) \* 7.75);

30 }

31 *else* *if*(units <= 5000) {

32 amount = 130 + 162.50 + 526 + 6200 + ((units - 1000) \* 10);

33 }

34 *else* {

35 amount = 0.0;

36 }

37

38 }

39

40 *return* amount;

41 }

42 }

43

#### **ElectricityBill/src/EBBillTest.java**

1 *import* *static* org.junit.Assert.assertTrue;

2 *import* *static* org.junit.Assert.fail;

3 *import* org.junit.Test;

4

5

6 *public* *class* EBBillTest

7 {

8

9 //Write JUNIT Test Code

10 @Test

11 *public* *void* testCalculateBillAmountq()

12 {

13 EBBill e=*new* EBBill(0);

14 assertTrue(e.calculateBillAmount()==0);

15 }

16 @Test

17 *public* *void* testCalculateBillAmount()

18 {

19 EBBill e=*new* EBBill(20);

20 assertTrue(e.calculateBillAmount()==52);

21 }

22 @Test

23 *public* *void* testCalculateBillAmount1()

24 {

25 EBBill e=*new* EBBill(90);

26 assertTrue(e.calculateBillAmount()==260);

27 }

28 @Test

29 *public* *void* testCalculateBillAmount2()

30 {

31 EBBill e=*new* EBBill(150);

32 assertTrue(e.calculateBillAmount()==555.5);

33 }

34 @Test

35 *public* *void* testCalculateBillAmount3()

36 {

37 EBBill e=*new* EBBill(900);

38 assertTrue(e.calculateBillAmount()==6243.5);

39 }

40 @Test

41 *public* *void* testCalculateBillAmount4()

42 {

43 EBBill e=*new* EBBill(4000);

44 assertTrue(e.calculateBillAmount()==37018.5);

45 }

46

47 }

#### **ElectricityBill/src/UserInterface.java**

1 *import* org.junit.runner.JUnitCore;

2 *import* org.junit.runner.Result;

3 *import* org.junit.runner.notification.Failure;

4 *import* *static* org.junit.Assert.\*;

5 *public* *class* UserInterface

6 {

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

8 {

9 Result result=JUnitCore.runClasses(EBBillTest.*class*);

10

11 *if*(result.getFailureCount()==0)

12 {

13 System.out.println("There are No Failures...\n Test Passed...");

14 }

15 *else*

16 {

17 *for*(Failure failure: result.getFailures())

18

19 {

20 System.out.println("The Test execution failed...\n"+failure.getMessage());

21 }

22 }

23 System.out.println("Result"+result.wasSuccessful());

24 }

25 }

## 2. Loan EMI Calculator

Centric  Bank allots various loans like Housing Loan, Vehicle Loan and Personal Loan. Antony wants to know the rate of interest on these loan types so that he can avail any one of these loans.

 The Interest rates for the various loan types provided by the bank is listed in the following table.

|  |  |
| --- | --- |
| Loan Type | Interest per Annum |
| Housing Loan | 12% |
| Vehicle Loan | 11% |
| Personal Loan | 10% |

Antony calculates the loan amount to be paid every month for the loan types based on the above listed interest rates.

Write JUNIT test class named “EMICalculatorTest” to test the method calculateEMI() in the given EMICalculator class.

You need to write only JUNIT code.

**Note:** Do not alter the code present inside the EMCalculatorl class.

//Solution

#### **LoanEMICalculator/src/EMICalculator.java**

1 *import* java.util.\*;

2 *public* *class* EMICalculator {

3

4 *public* *double* calculateEMI(*double* principal,String loanType,*double* durationInYears)

5 {

6 *double* rate=0.0;

7 *double* loanAmount=0.0;

8 *if*((principal>=10000) && (durationInYears>0 && durationInYears<=20)) {

9 *if*(loanType.equals("Housing Loan"))

10 {

11 rate = 0.12;

12 rate=rate/(12\*100);

13 durationInYears = durationInYears\*12;

14 loanAmount = (principal\*rate\*Math.pow(1+rate,durationInYears))/(Math.pow(1+rate,durationInYears)-1);

15

16 }

17 *else* *if*(loanType.equals("Vehicle Loan"))

18 {

19 rate = 0.11;

20 rate=rate/(12\*100);

21 durationInYears = durationInYears\*12;

22 loanAmount = (principal\*rate\*Math.pow(1+rate,durationInYears))/(Math.pow(1+rate,durationInYears)-1);

23

24 }

25 *if*(loanType.equals("Personal Loan"))

26 {

27 rate = 0.10;

28 rate=rate/(12\*100);

29 durationInYears = durationInYears\*12;

30 loanAmount = (principal\*rate\*Math.pow(1+rate,durationInYears))/(Math.pow(1+rate,durationInYears)-1);

31

32 }

33 }

34 *return* loanAmount;

35 }

36 }

#### **LoanEMICalculator/src/EMICalculatorTest.java**

1 *import* *static* org.junit.Assert.assertEquals;

2 *import* *static* org.junit.Assert.fail;

3 *import* *static* org.junit.Assert.assertFalse;

4 *import* java.util.\*;

5 *import* org.junit.Before;

6 *import* org.junit.Test;

7 *import* org.junit.runner.RunWith;

8 *import* org.junit.runners.Parameterized;

9 *import* org.junit.runners.Parameterized.Parameters;

10 @RunWith(Parameterized.*class*)

11 *public* *class* EMICalculatorTest {

12

13 //Write JUNIT Test Code

14 *private* String input;

15 *private* String output;

16 *public* EMICalculatorTest(String input,String output)

17 {

18 *super*();

19 *this*.input=input;

20 *this*.output=output;

21 }

22 @Parameters

23 *public* *static* Collection<String[]> testConditions()

24 {

25 String expectedOutputs[][]={{"Housing Loan","1667.7501986013428"},{"Vehicle Loan","1667.66"},{"Personal Loan","1667.57"}};

26 *return* Arrays.asList(expectedOutputs);

27 }

28 EMICalculator ec;

29 @Before

30 *public* *void* setup()

31 {

32 ec=*new* EMICalculator();

33 }

34 @Test

35 *public* *void* testCalculateEMIn()

36 {

37 *try* {

38 assertEquals(0.0,ec.calculateEMI(5000,"Housing Loan",1),0.001);

39 } *catch*(*final* RuntimeException e) {

40 fail();

41 } *finally* {

42 }

43 }

44 @Test

45 *public* *void* testCalculateEMI()

46 {

47 assertEquals(Double.parseDouble(output),ec.calculateEMI(20000,input,1),0.001);

48 }

49 @Test

50 *public* *void* testCalculateEMI1()

51 {

52 assertEquals(Double.parseDouble(output),ec.calculateEMI(20000,input,1),0.001);

53 }

54 @Test

55 *public* *void* testCalculateEMI2()

56 {

57 assertEquals(Double.parseDouble(output),ec.calculateEMI(20000,input,1),0.001);

58 }

59

60 }

61

#### **LoanEMICalculator/src/UserInterface.java**

1 *import* org.junit.runner.JUnitCore;

2 *import* org.junit.runner.Result;

3 *import* org.junit.runner.notification.Failure;

4 *import* *static* org.junit.Assert.\*;

5 *public* *class* UserInterface

6 {

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

8 {

9 Result result=JUnitCore.runClasses(EMICalculatorTest.*class*);

10

11 *if*(result.getFailureCount()==0)

12 {

13 System.out.println("There are No Failures...\n Test Passed...");

14 }

15 *else*

16 {

17 *for*(Failure failure: result.getFailures())

18

19 {

20 System.out.println("The Test execution failed...\n"+failure.getMessage());

21 }

22 }

23 System.out.println("Result..."+result.wasSuccessful());

24 }

25 }

## 3. Product Login Test suite

Varunika is a software tester in an organization. She tests multiple test cases. She wants to make testing very simple so she decides to create a test suite which can run multiple test cases under a single unit.

 Write JUNIT test class named LoginTest to test the methoddeleteLogin() in the given LoginDAO class.

 Write JUNIT test class named ProductTest to test the method deleteproduct() in the given ProductDAO class.

 Write a JUNIT Test Suite named TestSuiteRunner to run the test cases in both LoginTest and ProductTest classes.

//Solution

#### **ProductLogin/src/Login.java**

1 *public* *class* Login {

2 *private* String userName;

3 *private* String Password;

4 *public* Login(String userName, String password) {

5 *super*();

6 *this*.userName = userName;

7 Password = password;

8 }

9 *public* String getUserName() {

10 *return* userName;

11 }

12 *public* *void* setUserName(String userName) {

13 *this*.userName = userName;

14 }

15 *public* String getPassword() {

16 *return* Password;

17 }

18 *public* *void* setPassword(String password) {

19 Password = password;

20 }

21

22

23 }

24

#### **ProductLogin/src/LoginDAO.java**

1 *import* java.util.\*;

2 *public* *class* LoginDAO {

3 ArrayList<Login> loginlist = *new* ArrayList<Login>();

4 *public* *boolean* addLogin(Login obj)

5 {

6 *boolean* flag=*false*;

7 *if* (obj != *null*){

8 flag = loginlist.add(obj);

9

10 }

11

12 *return* flag;

13

14 }

15

16 *public* *boolean* deleteLogin(Login obj)

17 {

18 *boolean* flag = *false*;

19 *if*(obj!=*null*){

20 flag = loginlist.remove(obj);

21

22 }

23

24 *return* flag;

25 }

26

27

28 }

29

#### **ProductLogin/src/LoginTest.java**

1 *import* org.junit.Test;

2

3 *import* *static* org.junit.Assert.fail;

4

5 *import* *static* org.junit.Assert.assertTrue;

6

7 *import* *static* org.junit.Assert.assertFalse;

8

9

10

11

12

13 *public* *class* LoginTest {

14

15

16

17 //Write the code for adding and deleting Login data

18

19 LoginDAO ld=*new* LoginDAO();

20

21 Login l=*new* Login("Niveditha","sai@123");

22

23 @Test

24

25 *public* *void* testDeleteLogin()

26

27 {

28

29 assertTrue(ld.deleteLogin(l));

30

31 }

32

33 @Test

34

35 *public* *void* testDeleteLogin1()

36

37 {

38

39 Login l=*null*;

40

41 assertFalse(ld.deleteLogin(l));

42

43 }

44

45 @Test

46

47 *public* *void* testAddLogin()

48

49 {

50

51 assertTrue(ld.addLogin(l));

52

53 }

54

55 @Test

56

57 *public* *void* testAddLogin1()

58

59 {

60

61 Login l=*null*;

62

63 assertFalse(ld.addLogin(l));

64

65 }

66

67

68

69

70

71 }

72

73

#### **ProductLogin/src/Product.java**

1 *public* *class* Product {

2 *private* String productId;

3 *private* String productName;

4 *private* *double* price;

5 *public* Product(){

6 String productId="";

7 String productName="";

8 *double* price=0.0;

9 }

10 *public* String getProductId() {

11 *return* productId;

12 }

13 *public* *void* setProductId(String productId) {

14 *this*.productId = productId;

15 }

16 *public* String getProductName() {

17 *return* productName;

18 }

19 *public* *void* setProductName(String productName) {

20 *this*.productName = productName;

21 }

22 *public* *double* getPrice() {

23 *return* price;

24 }

25 *public* *void* setPrice(*double* price) {

26 *this*.price = price;

27 }

28 *public* Product(String productId, String productName, *double* price) {

29 *super*();

30 *this*.productId = productId;

31 *this*.productName = productName;

32 *this*.price = price;

33 }

34

35

36 }

37

#### **ProductLogin/src/ProductDAO.java**

1 *import* java.util.ArrayList;

2

3 *public* *class* ProductDAO {

4 ArrayList<Product> productlist = *new* ArrayList<Product>();

5 *public* *boolean* addProduct(Product obj)

6 {

7 *boolean* flag=*false*;

8 *if*(obj != *null*){

9 flag =productlist.add(obj);

10 }

11 *return* flag;

12 }

13

14 *public* *boolean* deleteProduct(Product obj)

15 {

16 *boolean* flag=*false*;

17 *if*(obj!=*null*)

18 {

19 flag = productlist.remove(obj);

20

21 }

22 *return* flag;

23 }

24 }

25

#### **ProductLogin/src/ProductTest.java**

1 *import* org.junit.Test;

2

3 *import* *static* org.junit.Assert.fail;

4

5 *import* *static* org.junit.Assert.assertTrue;

6

7 *import* *static* org.junit.Assert.assertFalse;

8

9 *public* *class* ProductTest {

10

11

12

13 //Write the code for test methods

14

15 ProductDAO pd=*new* ProductDAO();

16

17 Product p=*new* Product("1","lux",50.0);

18

19 @Test

20

21 *public* *void* testDeleteProduct()

22

23 {

24

25 assertTrue(pd.deleteProduct(p));

26

27 }

28

29 @Test

30

31 *public* *void* testDeleteProduct2()

32

33 {

34

35 Product p=*null*;

36

37 assertFalse(pd.deleteProduct(p));

38

39 }

40

41 @Test

42

43 *public* *void* testAddProduct()

44

45 {

46

47 assertTrue(pd.addProduct(p));

48

49 }

50

51 @Test

52

53 *public* *void* testAddProduct1()

54

55 {

56

57 Product p=*null*;

58

59 assertFalse(pd.addProduct(p));

60

61 }

62

63 }

#### **ProductLogin/src/TestSuiteRunner.java**

1 *import* org.junit.runner.RunWith;

2 *import* org.junit.runners.Suite;

3 *import* org.junit.runners.Suite.SuiteClasses;

4

5 //write code here

6 @RunWith(Suite.*class*)

7 @SuiteClasses({LoginTest.*class*,ProductTest.*class*})

8 *public* *class* TestSuiteRunner {

9

10 }

11

#### **ProductLogin/src/UserInterface.java**

1 *import* org.junit.runner.JUnitCore;

2 *import* org.junit.runner.Result;

3 *import* org.junit.runner.notification.Failure;

4 *import* *static* org.junit.Assert.\*;

5 *public* *class* UserInterface

6 {

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

8 {

9 Result result=JUnitCore.runClasses(TestSuiteRunner.*class*);

10

11 *if*(result.getFailureCount()==0)

12 {

13 System.out.println("There are No Failures...\n Test Passed...");

14 }

15 *else*

16 {

17 *for*(Failure failure: result.getFailures())

18

19 {

20 System.out.println("The Test execution failed...\n"+failure.getMessage());

21 }

22 }

23 System.out.println("Result"+result.wasSuccessful());

24 }

25 }

## 4. Testing Using Assertion

Write JUNIT test class named CustomerTest to test the attributes given in the Customer class.

(i)   Test whether the Aadhar card no is valid or not by using assertTrue() and assertFalse() methods. The Aadhaar card number should contain 16 digits. It should not start with 0 or 1.

(ii)  Test whether the firstname and lastname are not equal by using assertNotEquals() method

(iii) Test whether the Email id is not null by using assertNotNull() method

 You need to write only JUNIT code.

**Note:** Do not alter the code present inside the Customer class.

//Solution

#### **AssertionTest/src/Customer.java**

1 *import* java.util.regex.Matcher;

2 *import* java.util.regex.Pattern;

3 *public* *class* Customer {

4 *private* String aadharCardNo;

5 *private* String firstName;

6 *private* String lastName;

7 *private* String address;

8 *private* *long* mobileNo;

9 *private* String emailId;

10

11

12 *public* Customer(String aadharCardNo, String firstName, String lastName, String address, *long* mobileNo,String emailId) {

13 *super*();

14 *this*.aadharCardNo = aadharCardNo;

15 *this*.firstName = firstName;

16 *this*.lastName = lastName;

17 *this*.address = address;

18 *this*.mobileNo = mobileNo;

19 *this*.emailId = emailId;

20 }

21

22 *public* String getFirstName() {

23 *return* firstName;

24 }

25

26 *public* String getEmailId() {

27 *return* emailId;

28 }

29

30 *public* String getLastName() {

31 *return* lastName;

32 }

33

34

35 *private* *static* Pattern aadhaarPattern = Pattern.compile("^[2-9]{1}[0-9]{11}$");

36

37

38 *public* *static* *boolean* isValidAadharNo(String name)

39 {

40 Matcher matcher = aadhaarPattern.matcher(name);

41 *return* matcher.find();

42 }

43

44

45 }

46

47

48

49

#### **AssertionTest/src/CustomerTest.java**

1 *import* *static* org.junit.Assert.\*;

2 *import* org.junit.Test;

3

4 *public* *class* CustomerTest {

5 //Write the code for testing assertion using JUNIT

6 Customer c=*new* Customer("454135086518","Aseema","Shaik","Tirupati",99,"shaik.aseema9@gmail.com");

7 @Test

8 *public* *void* testIsValidAadharNoTrue()

9 {

10 *boolean* b=c.isValidAadharNo("454135086518");

11 assertTrue(b);

12 }

13 @Test

14 *public* *void* testIsValidAadharNoFalse()

15 {

16 assertFalse(c.isValidAadharNo("014541350865"));

17 }

18 @Test

19 *public* *void* testFirstAndLastName()

20 {

21 assertNotEquals(c.getFirstName(),c.getLastName());

22 }

23 @Test

24 *public* *void* testEmailId()

25 {

26 assertNotNull(c.getEmailId());

27 }

28 }

29

#### **AssertionTest/src/UserInterface.java**

1 *import* org.junit.runner.JUnitCore;

2 *import* org.junit.runner.Result;

3 *import* org.junit.runner.notification.Failure;

4 *import* *static* org.junit.Assert.\*;

5 *public* *class* UserInterface

6 {

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

8 {

9 Result result=JUnitCore.runClasses(CustomerTest.*class*);

10

11 *if*(result.getFailureCount()==0)

12 {

13 System.out.println("There are No Failures...\n Test Passed...");

14 }

15 *else*

16 {

17 *for*(Failure failure: result.getFailures())

18

19 {

20 System.out.println("The Test execution failed...\n"+failure.getMessage());

21 }

22 }

23 System.out.println("Result"+result.wasSuccessful());

24 }

25 }