Java lab DA-3

Done by Arshdeep Singh Bhatia 19BCB0086

Contents

Q1 1

CODE 1

Output 3

Q2 5

Code 5

Output after 10 sec each: 6

Q1

Question - 1

Write the java program to this Loan class (refer the course page) to throw IllegalArgumentException if the loan amount, interest rate, or number of years is less than or equal to zero

CODE

```
package da3;
import java.util.*;
public class Loan {
    private double annualInterestRate;
    private int numberOfYears;
    private double loanAmount;
    private java.util.Date loanDate;
    /** Default constructor */
    public Loan() {
    this(2.5, 1, 1000);
    }
    /** Construct a loan with specified annual interest rate,
     number of years, and loan amount
    */
    public Loan(double annualInterestRate, int numberOfYears,
    double loanAmount) {
    setAnnualInterestRate(annualInterestRate);
    setNumberOfYears(numberOfYears);
    setLoanAmount(loanAmount);
    loanDate = new java.util.Date();
    }
    /** Return annualInterestRate */
    public double getAnnualInterestRate() {
    return annualInterestRate;
```

```
}
/** Set an new annualInterestRate */
public void setAnnualInterestRate(double annualInterestRate)
throws IllegalArgumentException {
if (annualInterestRate <= 0) {
throw new IllegalArgumentException(
"Annual interest rate must be greater than 0");
}
this.annualInterestRate = annualInterestRate;
}
/** Return numberOfYears */
public int getNumberOfYears() {
return numberOfYears;
}
/** Set a new numberOfYears */
public void setNumberOfYears(int numberOfYears)
throws IllegalArgumentException {
if (numberOfYears <= 0) {
throw new IllegalArgumentException(
"Number of years must be greater than 0");
}
this.numberOfYears = numberOfYears;
}
/** Return loanAmount */
public double getLoanAmount() {
return loanAmount;
}
/** Set a new loanAmount */
public void setLoanAmount(double loanAmount)
throws IllegalArgumentException {
if (loanAmount <= 0) {
throw new IllegalArgumentException(
"Loan amount must be greater than 0");
}
this.loanAmount = loanAmount;
}
/** Find monthly payment */
public double getMonthlyPayment() {
double monthlyInterestRate = annualInterestRate / 1200;
double monthlyPayment = loanAmount * monthlyInterestRate / (1 -
(1 / Math.pow(1 + monthlyInterestRate, numberOfYears * 12)));
return monthlyPayment;
}
/** Find total payment */
public double getTotalPayment() {
double totalPayment = getMonthlyPayment() * numberOfYears * 12;
return totalPayment;
}
```

```
    /** Return loan date */
    public java.util.Date getLoanDate() {
    return loanDate;
    }
public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.println("ENTER ANNUAL INTEREST RATE");
    double a=sc.nextDouble();
    System.out.println("ENTER NUMBER OF YRS");
    int n=sc.nextInt();
    System.out.println("ENTER LOAN AMOUNT");
    double l=sc.nextDouble();
    sc.close();
    try {
        Loan loan = new Loan(a,n,l);
        System.out.println("ANNUAL INT RATE: " + loan.getAnnualInterestRate())
;
        System.out.println("MONTHLY PAYMENT "+ loan.getMonthlyPayment());
        System.out.println("NUMBER OF YEARS : "+loan.getNumberOfYears());
        System.out.println("LOAN DATE : "+loan.getLoanDate());
        System.out.println("LOAN AMOUNT : " +loan.getLoanAmount());
        System.out.println("LOAN TOTAL PAYMENT : " + loan.getTotalPayment());

    } catch (IllegalArgumentException ex) {
        System.out.println("IllegalArgumentException: " + ex.getMessage());
    }
}
}
```

Output

Case 1 no errors

```
ENTER ANNUAL INTEREST RATE
15
ENTER NUMBER OF YRS
5
ENTER LOAN AMOUNT
10000
ANNUAL INT RATE: 15.0
MONTHLY PAYMENT 237.89930086358785
LOAN DATE : Thu Mar 25 17:14:13 IST 2021
LOAN AMOUNT10000.0
LOAN TOTAL PAYMENT14273.958051815272
```

Case 2 (interest rate negative)

```
ENTER ANNUAL INTEREST RATE
-12
ENTER NUMBER OF YRS
ENTER LOAN AMOUNT
200
IllegalArgumentException: Annual interest rate must be greater than 0
```

Case 3 (no of years negative)

```
ENTER ANNUAL INTEREST RATE
12
ENTER NUMBER OF YRS
-20
ENTER LOAN AMOUNT
400
IllegalArgumentException: Number of years must be greater than 0
```

Case 4 (loan amount negative)

```
PS C:\Users\arshd\Desktop\java>  c:; cd 'c:\Users\arshd\Desktop\java'
 'C:\Program Files\Java\jdk-15.0.2\bin\java.exe' '--enable-preview' '
a\Roaming\Code\User\workspaceStorage\d288d8b2eede9c9dc7c034c0343ce4af
ENTER ANNUAL INTEREST RATE
12
ENTER NUMBER OF YRS
20
ENTER LOAN AMOUNT
-100
IllegalArgumentException: Loan amount must be greater than 0
PS C:\Users\arshd\Desktop\java>
```

Q2

Question - 2

Simulate a Bike race using the Multithread in java. Created a class by your own that extends Thread. Assume that 5 bikes are scheduled for the race. Generate a random number for bike name/no, time and add 3 stage for the race. Print the results after each stage as given below. Use thread sleep between 2 to 5 seconds for the same.

Code

```
package da3;
import java.util.*;
class bike extends Thread{
    int number;
    bike(int n){
        this.number=n;
    }
    public int getn(){
        return this.number;
    }

    public void run()
    {
        try {
            System.out.println("Bike #"+getn()+" reached");
        } catch (Exception e) {
            System.out.println("Exception is caught");
        }
    }
}
public class bikerace {
    public static void stage(int stages,int bikes){
        System.out.println("Stage number: "+stages);
        for(int i=1;i<=bikes;i++){
            bike newbike= new bike(i);
            newbike.start();
        }
       try {
        Thread.sleep(10000);
       }
       catch (InterruptedException e) {
        e.printStackTrace();
    }
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter number of bikes");
        int b=sc.nextInt();
        System.out.println("Enter number of stages");
        int s=sc.nextInt();
```

```
        sc.close();
        for(int i=1;i<=s;i++){
            stage(i,b);
        }
    }
}
```

Output after 10 sec each:

```
Enter number of bikes
5
Enter number of stages
3
Stage number: 1
Bike #1 reached
Bike #2 reached
Bike #3 reached
Bike #4 reached
Bike #5 reached
```

```
Enter number of bikes
5
Enter number of stages
3
Stage number: 1
Bike #1 reached
Bike #2 reached
Bike #3 reached
Bike #4 reached
Bike #5 reached
Stage number: 2
Bike #1 reached
Bike #3 reached
Bike #2 reached
Bike #4 reached
Bike #5 reached
```

```
Enter number of bikes
5
Enter number of stages
3
Stage number: 1
Bike #1 reached
Bike #2 reached
Bike #3 reached
Bike #4 reached
Bike #5 reached
Stage number: 2
Bike #1 reached
Bike #3 reached
Bike #2 reached
Bike #4 reached
Bike #5 reached
Stage number: 3
Bike #1 reached
Bike #3 reached
Bike #2 reached
Bike #5 reached
Bike #4 reached
```

Case 2

Different input

1

```
Enter number of bikes
3
Enter number of stages
2
Stage number: 1
Bike #1 reached
Bike #3 reached
Bike #2 reached
Stage number: 2
Bike #1 reached
Bike #3 reached
Bike #2 reached
PS C:\Users\arshd\Desktop\java>
```

2

```
Enter number of bikes
12
Enter number of stages
2
Stage number: 1
Bike #1 reached
Bike #3 reached
Bike #2 reached
Bike #7 reached
Bike #6 reached
Bike #5 reached
Bike #4 reached
Bike #12 reached
Bike #11 reached
Bike #10 reached
Bike #9 reached
Bike #8 reached
Stage number: 2
Bike #1 reached
Bike #3 reached
Bike #2 reached
Bike #8 reached
Bike #5 reached
Bike #4 reached
Bike #12 reached
Bike #11 reached
Bike #9 reached
Bike #6 reached
Bike #7 reached
Bike #10 reached
PS C:\Users\arshd\Desktop\java>
```