**Lab Assignment – 11   
Due Date: 10/31 (Wednesday), 11.59 pm  
Total Points: 100**

**Objectives:**

* **Learn to define a user-defined class**
* **Learn to validate the values before assigning them to instance variables.**

1. Design a User-Defined class called **Date.java** with three attributes day, month and year. A date object represents a date according to the Gregorian calendar. The class should provide the following:

* Appropriate constructor
* getMonth( ), getDay( ), getYear( ) methods
* setYear( ) – check if the year is great than or equal to 1582 the year that the Gregorian Calendar was adopted. If not throw new IllegalArgumentException with an appropriate error message as a String argument.
* setMonth( ) – check if the month value is between 1 – 12 before assigning to month if not throw new IllegalArgumentException with an appropriate error message as a String argument.
* setDay( ) – this method will check the month first to validate whether the value that will be assigned to day is legal. Example, a value of 31 for the day for month 11 (November) is invalid and should not be assigned and should throw an IllegalArgumentException as shown below:  
  **throw new IllegalArgumentException(** *appropriate error message should go here* **);**If the year is a leap year than a value of 29 for the day for month – 2 (February) is valid. To determine whether a year is a leap year, use the GregorianCalendar class that is part of the util package. This class has a non-static method called isLeapYear( ).
* Change the access modifier for set methods to private
* Explain in the word document why is it a good idea to declare the set methods to private instead of public.
* Implement toString( ) and equals( ) method (*we will discuss toString( ) and equals( ) method during our next lecture*)
* Modify the constructor you created earlier so, you can call the private set methods to ensure that the values being assigned for the day, month and year instance variables are valid. Please see the below constructor -  
  Date(int d, int m, int y)  
  {

set(y);

set(m);

set(d);

}

* Draw a class diagram for Date class using Visio or other diagramming tool.

Next, create a client program DateTest.java with the main method( ). In the program, prompting the user for day, month, and year to create a Date object and display date values using the get methods. Create appropriate test cases to test the set methods of the Date class. A sample of test cases is provided below. You would have to create additional test cases. Use the below format for Test Cases –

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Inputs** | **Expected Output** | **Actual Output** |
| 1 | Year = 2012 Month = 2  Day = 28 | **2/28/2012** | 2/28/2012 |
| 2 | Year = 2013  Month = 3  Day = 32 | IllegalArgumentException generated with the message – “**Day value cannot be greater than 32 for month 3**” | IllegalArgumentException: Day value must be from 1 to 31 for month 3 |
| 4 | Year = 1325  Month = 3  Day = 5 | IllegalArgumentException generated with the message – “**Year must be greater than or equal to 1582**” | IllegalArgumentException: Year must be greater than or equal to 1582 |
| 5 | Year = 2013  Month = 2  Day = 29 | IllegalArgumentException generated with the message – **"At 2013, the day in February must be from 1 to 28"** | IllegalArgumentException: At2013, the day in February must be from 1 to 28 |
| 6 | Year = 2016  Month = 2  Day = 29 | **2/29/2016** | 2/29/2016 |

**(40 points)**

1. Next, design a User-Defined class called Loan.java, an object of which represents the real-world Loan entity. Each loan object should store the following information:
   1. Annual Interest Rate
   2. Number of Years
   3. Loan Amount
   4. Loan Date (You would be specifying Loan Date of the type Date class you created earlier)

Provide appropriate constructor(s) for the Loan class.

Here are the different operations that could be done on any Loan object. Define a method for each of the below operations.

1. An operation that returns the annual interest rate
2. An operation that returns the number of years
3. An operation that returns the loan amount
4. An operation that returns the Loan Date as a String value providing the date information.
5. An operation that sets a new annual interest rate
6. An operation that sets a new number of years
7. An operation that sets a new loan amount
8. An operation that calculates and returns the monthly payment  
   (Calculate the monthly payment using the following formulas:
   * Monthly payment = (mIR \* M) / (1 – (1 / (1 + mIR)(12\*nOY) ))  
     where:
   * mIR = monthly interest rate (annual interest rate / 12)
   * nOY = number of years
   * M = Loan amount
9. An operation that calculates the total payment amount over the life of the loan
10. An operation that returns the overpayment amount
11. toString( ) method – returns the content of the Loan object as a String value
12. equals( ) method – check the contents of the two Loan object to check if they are same.

**Draw a class diagram for the Loan class**

Next, you will implement a test class called LoanTest.java to test the Loan class

You will do the following in the test class:

* Prompt the user for the annual interest rate.
* Prompt the user for the number of years the loan will be held
* Prompt the user for a number representing the loan amount borrowed from the bank.
* Prompt the user for the start date of the loan.
* Include necessary checks to validate the above input values and also check for datatype safe read.
* Create the Loan object
* Output a summary of the mortgage problem as follows:
  + The annual interest rate in percent notation
  + The mortgage amount in dollars
  + The monthly payment in dollars, with only two significant digits after the decimal point
  + The total payment over the years, with only two significant digits after the decimal point
  + The overpayment with only two significant digits after the decimal point
  + The overpayment as a percentage (in percent notation) of the mortgage amount
* After your program displays the above output, it should prompt the user if the user would be like to try again. If the answer is YES then repeat the above operations.

Your output for both the programs has to be verbose and don’t forget to comment and describe your programs. ***(60 points)***

**Things to Turn in:**

1. In a new Word file and save the file naming convention that we have used for earlier labs. *Use landscape page layout and Courier New font.*
2. Enter your name at the top of the document.
3. Paste the class diagram of the Date class
4. Type your answer to why the set methods in Date class were declared to be private.
5. Copy and paste the source code of Program – I (Data.java and DataTest.java) and copy and paste the contents of output window of DataTest.java
6. Copy and paste the screenshot of the output window of DateTest.java for a run with good input and a bad input.
7. Type in the test cases you identified for Programming exercise – I.
8. Paste the class diagram of the Loan class.
9. Copy and paste the source code of Program – II (Loan.java and LoanTest.java) and copy and paste the contents of output window for LoanTest.java.
10. Copy and paste the screenshot of the output window of LoanTest.java for each run of the program for the inputs provided.
11. Zip your NetBeans Project folder by going to File 🡪Zip. Make sure to select a location where you want to save the zipped file and **do not forget to include .zip at the end of the zipped file name.**
12. Submit your word document along with the zipped NetBeans project folder using the ***Lab Assignment – 11 link*** on Blackboard by the due date.