Regis University CC&IS
CS210 Introduction to Programming
Java Programming Assignment 6: Objects and Loops

Your previous Alice programs implemented the **count (for)** and **while** loops.

This assignment will apply the same concepts to Java, along with a third type of loop, the **do-while** loop. You will again use an **object** to store data. Then looping statements will be used to run some of the statements repeatedly.

WARNING: From this point in the course forward, programs that **do not compile** without errors **will NOT be accepted**. So be sure your code compiles and runs before submitting it.

Problem Summary and Equations

Write a program to compare how long it will take to pay off a credit card balance by paying only the minimum required payment each month, or by paying a larger amount each month. The program will work for any credit card balance of $500 or more.

- The credit card will have an annual interest rate between 3% and 25%.
- Interest will be applied monthly. This means that $1/12^{th}$ of the annual interest will be added to the current balance each month.
- The user can choose to pay down anywhere from 6% to 33% of the balance every month. The payment made each month will always be at least the minimum required payment, but will never be more than the remaining balance.

NOTE: For an example of a Java program implementing loops, see Online Content section 12.13.

Program Requirements

This program will implement a general-use input reading method called **readPercent** that can be used to read and validate a percentage is within a specified range. The method will include 3 parameters:

- the lowest percentage allowed
- the highest percentage allowed
- a String description of what percentage is being read from the user

By passing in these values as parameters, you can re-use the method to read any range of percentages needed for any data value.

For example, if you were to call the method to read a mortgage rate, which is a percentage between 3.5% and 20%, the call would be:

```
double mortgageRate = readPercent(3.5, 20, "annual mortgage rate");
```

Or to read a down payment percent, which is a percentage between 10% and 30%, the call would be:

```
double downRate = readPercent(10, 30, "down payment percent");
```

Required Classes and Methods

Two separate classes will be required for this program.

1. Create a new Java NetBeans project, named as follows:

   `LastnameJavaAssn6`
   For example: **`SmithJavaAssn6`**

2. Define a Java **class** with properties and methods for a Credit Card Account, named:

   `CreditCardAccount`

   The class will have the following **private** properties:

   - ❖ current balance
   - ❖ annual interest rate (percentage – e.g. 12.5%)
   - ❖ percent of current balance to pay each month (e.g. 10%)

Within the **CreditCardAccount** class you will define methods, as follows:

- Define a **constructor**, with parameters that pass in the user entered values for ***only two*** of the properties, the initial balance and annual interest rate.
  - Use the parameters to initialize:
    - the **current balance** property to the initial balance parameter value
    - the **annual interest rate property** to the annual interest rate parameter value
  - Initialize the **percent to pay off each month** to 0.
- Define a second **constructor**, with parameters that pass in the user entered values for ***all three*** of the properties, and use them to initialize the property values.
- Define a **getter** for the percent of current balance to pay each month.
- Define an **instance** method to determine ***and return*** the minimum required payment for a month:
  - Define and use three local ***constants:***
    - A ***low balance minimum payment*** ($50)
    - A ***high balance percentage*** (5%)
    - A ***low balance limit*** ($1000)
  - If the current balance is below the ***low balance limit***, then the minimum required payment will be the ***low balance minimum payment***.
  - Otherwise, the minimum required payment will be the ***high balance percentage*** of the current balance.
  - Return the correct minimum required payment
- Define a **make payment** instance method to record and display a payment for one month (note that there will be ***no loops*** in this method). This method will:
  - Calculate values and display one line of output, containing the data about one month's payment, as follows:
    - Display the current (starting) balance.
    - Use the current balance to calculate the interest to be applied for the month, and add that interest to the current balance.
    - Display the interest charge and current balance (with interest).
    - Using the current balance (with interest), calculate the payment for the month, based on the ***percent of current balance to pay each month*** data field.
    - Call the instance method to determine the minimum required payment.

        - Check to see if the calculated payment is less than the minimum required payment.
            - If so, set the calculated payment to the minimum required payment.
        - The calculated payment may be higher than the remaining balance. So check to see if the calculated payment is higher than the remaining balance.
            - If so, set the calculated payment to the remaining balance.
        - Display the calculated payment that will be paid for the month.
        - Subtract the calculated payment from the current balance (with interest).
        - Display the new current balance (i.e. the ending balance, after the payment has been made).
- Define a **payoff** instance method to calculate and display information each month until the credit card is paid off. This method will:
    - Display a line describing the percent of current balance that will be paid each month
        - If the percent is 0, display "minimum payment" instead of a percent.
    - Display headers for the results (see sample output below).
    - Use a **while** loop to update and display information about the credit card account ***every month***, as follows:
        - Display the month number (starting with month 1, and incrementing by 1 each time the code loops).
        - Call the above defined **make payment** method to update and display the data values for the month.
    - Stop looping when the current balance reaches 0.
    - ***Return*** the number of months needed to pay off the card.

NOTE: When one instance method needs to call another instance method, the second instance method should be called using the **`this`** reference to reference the current object.

3. Define a second **class** containing a **main** method, and additional methods, to compare paying only the minimum required payment each month with paying a larger amount each month. Name the class:

    `PayoffComparison`

    Within the **PayoffComparison** class:

    - Define a **static** method to prompt for, read, and validate an initial credit card balance. The method will:
        - Define and use a constant to hold the lowest initial balance allowed ($500). This constant should be used to specify the lowest value allowed in the prompts, and to test for it within conditions.
        - Prompt for and read the initial balance.
        - If the user enters an initial balance that is below the lowest initial balance allowed:
            - Issue an error message, and loop to re-prompt the user and read another value
        - Loop until a valid value is entered.
        - Return a valid initial balance.

- Define the general-use static **readPercent** method described at the top of this assignment, to prompt for, read, and validate a percentage. This method will:
  - Include the parameters, from the description at the top of this assignment.
  - Implement a loop that uses a ***boolean variable*** as the condition.
  - Prompt for and read a percentage.
    - The prompt should use the parameters to specify the percentage being read (description) and the lowest and highest percentage rates allowed.
  - If the percent entered is not valid (i.e. not between the lowest and highest rate, inclusive):
    - Issue an error message, and loop to re-prompt the user and read another value
  - Loop until the percent entered is valid.
  - Return a valid percentage.

- Define a **static** method to prompt for, read, and validate the percentage of the balance to be paid off each month. This method will:
  - Define constants to hold the lowest and highest paydown percentages allowed (6 and 33).
  - Within a loop, until the user enter a valid choice:
    - Let the user choose from a menu of five choices on what minimum payment to make:

      1 - 10% of remaining balance each month

      2 - 20% of remaining balance each month

      3 - 30% of remaining balance each month

      4 – Some other percent of the balance each month between 6 and 33
    - If the user enters 4, call the generic ***readPercent*** method to read the percentage of the balance to be paid off each month, between 6 and 33.
    - If the user enters an invalid value, loop to display the menu and read another choice.
  - Return a valid percentage of the balance to be paid off each month.

- Define a **main** method that will:
  - Define constants to hold the lowest and highest annual interest rates allowed (3 and 25).
  - Display a description of what the program will do to the user.
  - Read input from the user as follows:
    - Call the static methods to read the initial balance.
    - Call the static **readPercent** method to read the annual interest rate.
    - Call the static **readPercent** method to read percentage of balance to pay off each month.
  - After reading all the input, create two new objects of the **CreditCardAccount** class
    - Create the first object using the first constructor and the first two user input values.
    - Create the second object using the second constructor and all the user input values.

    NOTE: After creating the objects, the variables used to store the values read from the user should ***not*** be used again.

Sample Description and Input

```
This program calculates how long it will take to pay off your credit
card balance with minimum payments vs. larger payments.

Enter the beginning balance (at least $500): 1111.11
the credit card's annual interest rate (between 3% and 25%): 7

What will the monthly payment will be?
    1 - 10% of balance with interest
    2 - 20% of balance with interest
    3 - 30% of balance with interest
    4 - Some other percent
Enter choice from Menu above: 2
```

- Display a couple of blank lines after reading all of the user input.
- Call the **payoff** method with the **first** object, and store the returned number of months.
- Call the **payoff** method with the **second** object, and store the returned number of months.
- Display a statement comparing the number of months required to pay off the credit card with only the minimum monthly payments, vs paying off the credit card with a larger monthly payment (use a **getter** to get the payoff percentage per month).

  (The figures in each column should line up with each other on the right – see **Sample Output** on next page for example)

4. The program must implement both a **while** and **do-while** loop somewhere in the code.

5. The program must follow the **CS210 Coding Standards** from Content section 6.10.
   Be sure to ***include*** the following comments:
   - Comments at the ***top of each code file*** describing what the class does
     - Include **tags** with the author's name (i.e. your full name) and the version of the code (e.g. version 1.0, Java Assn 5)
   - Comments at the ***top of each method***, describing what the method does
     - Include **tags** with names and descriptions of ***each*** parameter and return value.

WARNING: The objects, classes, and methods must be implemented exactly as specified above. If your program produces correct output, but you did not create and use the object as specified, and implement the required classes and methods, you will lose a significant number of points.

Testing

- Run, debug, and test your Java program with different inputs, until you are sure that all control structures within your program work correctly.
- The sample inputs and output can be used as the initial test to test your program. But be sure you thoroughly test it using other values as well.

Sample Output

```
Results when paying minimum required payment per month
-------------------------------------------------------------------
         Initial    Month's     Balance                    End
Month    Balance    Interest    w/Interest    Payment    Balance
-------------------------------------------------------------------
    1    1111.11        6.48       1117.59      55.88    1061.71
    2    1061.71        6.19       1067.91      53.40    1014.51
    3    1014.51        5.92       1020.43      51.02     969.41
    4     969.41        5.65        975.06      50.00     925.06
    5     925.06        5.40        930.46      50.00     880.46
    6     880.46        5.14        885.59      50.00     835.59
    7     835.59        4.87        840.47      50.00     790.47
    8     790.47        4.61        795.08      50.00     745.08
    9     745.08        4.35        749.43      50.00     699.43
   10     699.43        4.08        703.51      50.00     653.51
   11     653.51        3.81        657.32      50.00     607.32
   12     607.32        3.54        610.86      50.00     560.86
   13     560.86        3.27        564.13      50.00     514.13
   14     514.13        3.00        517.13      50.00     467.13
   15     467.13        2.72        469.86      50.00     419.86
   16     419.86        2.45        422.30      50.00     372.30
   17     372.30        2.17        374.48      50.00     324.48
   18     324.48        1.89        326.37      50.00     276.37
   19     276.37        1.61        277.98      50.00     227.98
   20     227.98        1.33        229.31      50.00     179.31
   21     179.31        1.05        180.36      50.00     130.36
   22     130.36        0.76        131.12      50.00      81.12
   23      81.12        0.47         81.59      50.00      31.59
   24      31.59        0.18         31.78      31.78       0.00

Results when paying 20% of the balance per month
-------------------------------------------------------------------
         Initial    Month's     Balance                    End
Month    Balance    Interest    w/Interest    Payment    Balance
-------------------------------------------------------------------
    1    1111.11        6.48       1117.59     223.52     894.07
    2     894.07        5.22        899.29     179.86     719.43
    3     719.43        4.20        723.63     144.73     578.90
    4     578.90        3.38        582.28     116.46     465.82
    5     465.82        2.72        468.54      93.71     374.83
    6     374.83        2.19        377.02      75.40     301.62
    7     301.62        1.76        303.37      60.67     242.70
    8     242.70        1.42        244.12      50.00     194.12
    9     194.12        1.13        195.25      50.00     145.25
   10     145.25        0.85        146.10      50.00      96.10
   11      96.10        0.56         96.66      50.00      46.66
   12      46.66        0.27         46.93      46.93       0.00

Paying minimum required payment per month,
    it will take 24 months to pay off the credit card.

Paying 20% of the balance per month,
    it will take 12 months to pay off the credit card.
```

Program Submission

This programming assignment is due by midnight of the date listed in the **Course Assignments by Week**.

Programs that **do not compile** without errors **will NOT be accepted**.

Again, you will submit a single **zip** file containing all of the files in your project.

- First export your project from NetBeans:
  - Highlight the project name.
  - Click on **File** from the top menu, and select **Export Project**.
  - Select **To ZIP**
  - Name your export file in the following format:
    `<lastname>Assn<x>.zip`

    For example:
    `SmithAssn6.zip`

  NOTE: Save this zip file to some other directory, not your project directory.

- Then submit your **.zip** file to the **Java Prog Assn 6** assignment submission folder (located under the **Assignments/Dropbox** tab in the online course).

  - Warning: Only NetBeans export files will be accepted.
    Do not use any other kind of archive or zip utility.

Grading

This program will be graded using the **rubric** that is linked on the same assignment page from which this program requirements file was downloaded.

WARNING:

*Programs submitted more than 5 days past the due date will **not** be accepted, and will receive a grade of 0.*