Regis University CC&IS
CS210 Introduction to Programming
Java Programming Assignment 4: Objects and Instance Methods

All of your Alice programs have been using objects already. And the predefined function and procedure methods in Alice are equivalent to instance methods for those objects in Java. You also defined your own methods within an Alice class, which had to be applied to a specific object in order to run.

This assignment will apply the same concepts to Java. You will create a new class, and use the new class to create an object in Java. Methods defined within the new class (instance methods, similar to the Alice procedure and function methods) will be called using the dot (.) operator, to send a message to the object.

> WARNING: This assignment will be ***more challenging*** than the previous assignments, so make sure to ***start early*** and ***allocate enough time*** to complete it, including time you might need to seek any necessary help.

Problem Summary

Suppose you wanted to modify your Mortgage Calculator from Java programming assignment 3.
You could:

- Store all the previous data about one mortgage loan as data fields within an **object**
- Add a mortgage loan identifier data field
- Revise the calculation methods to be instance methods within the object's class
    - NOTE: This will eliminate the need to pass the loan data into the methods via parameters.

Since you will be modifying your Mortgage Calculator program from Java Assn 3, make sure you use your instructor's feedback to fix any mistakes from Java Assn 3 before you submit Java Assn 4.

Overview of Program

This program will contain two classes, in two separate files within your project:

- A new **MortgageLoan** class to define the data fields and methods for a MortgageLoan object, containing:
    - Six data field definitions for a MortgageLoan object
    - A **constructor** method to create a MortgageLoan object
    - Four **setter** methods to set the values for four of the data fields
    - Four **getter** methods to get the values of four of the data fields
    - An instance method to compute the monthly property tax.
    - An instance method to compute the monthly insurance premium.
    - An instance method to compute the monthly principle and interest loan payment.
- The main **MortageCalculator** class, modified to define only 4 methods, containing:
    - A **main** method to display the program description, create an object, read the inputs from the user, and call the other methods.
    - A static method to display a description of what the program will do.
    - A static method to display loan details.
    - A static method to calculate and display the results of the program.

Since there will be multiple files, the program will be submitted via a **.zip** file (see last page).

NOTE: For an example of a Java program using objects, see Online Content section 10.13.

Program Requirements

Modify the program you wrote for Java Assn 3, as follows:

1. Define an additional Java **class** (File | New File | Java Class) named:

   `MortgageLoan`

   Within the MortgageLoan class, define the following **private** data fields to store properties about the mortgage loan:
   - loan identifier (**String**) ← new!
   - home value
   - down payment (**double** value that will hold a whole number percent of home value, e.g. 10.0)
   - loan amount
   - length of loan (in whole years)
   - loan annual interest rate (floating point percentage, e.g. 4.5)

   Remember that all private data fields defined at the class-level can be accessed by any method within the ***same*** class (without parameter passing).

   Also within the new **MortgageLoan** class, you will define instance methods that can be used with an object of the MortgageLoan class, as follows:

   - Define a default **constructor** with ***no*** parameters that will create the object with the following initial data values set:

     Loan identifier set to `""` (an empty String)

     Home value, loan amount, and interest rate all set to 0.0

     Down payment set to 10%

     Loan length set to 30 years

   - Create **setters** for the following data fields:
     - Setter for loan identifier
       - Input parameters will be two Strings: the home buyer's last name and zip code.
       - Use built-in Java **String** methods to uppercase and extract first 4 letters of last name and concatenate them with last 3 digits of zip code to create the loan identifier.
       - Set the loan identifier data field to this value.
     - Setter for home value
       - Input parameter will be home value.
       - Set the home value data field to the parameter value.
     - Setter for loan amount
       - No parameters
       - Calculate the loan amount using the home value and the down payment percent data fields. Be sure to convert the down payment from whole number value to a value that can be used in a math formula.
       - Set the loan amount data field to the calculated value.
     - Setter for loan annual interest rate
       - Input parameter will be the annual interest rate.
       - Set the loan annual interest rate data field to the parameter value

- Create **getters** for the following data fields in the **MortgageLoan** class:
    - loan identifier (String)
    - loan amount
    - length of loan (in years)
    - loan annual interest rate (percentage)

  Each of the getters should return the data field's current value.

- Move the calculation methods (methods 2 – 4) from Java Assn 3 from the **MortgageCalculator** class to the **MortgageLoan** class. Then modify them, so they can be used as **instance** methods in the new **MortgageLoan** class, as follows:
    - Eliminate the parameter lists from each method (parameters will no longer be necessary).
    - Keep all constants previously defined in the methods, except for the number of loan months defined in method 4.
        - Use the loan length data field from the object to calculate the number of months in the loan, instead of the previous constant value.
    - Modify the formulas to use the object data fields, instead of parameters.

2. Within the original **MortgageCalculator** class that contains the **main** method:

- Make sure you deleted the old static calculation methods when you moved them to the **MortgageLoan** class.
- Modify the **main** method to perform the following additional tasks (beyond Java Assn 3):
    - After displaying program description, create an object of the new **MortgageLoan** class type.
    - Prompt for (using descriptive prompts) and read:
        - the home buyer's last name (you can assume there will be no spaces in the last name and it will contain at least 4 letters).
        - the home buyer's zip code
    - Same as before, prompt for (using descriptive prompts) and read the home value and the annual interest rate.

      Sample Input

      ```
      Please enter the home buyer's last name: Smith
      Please enter the home's zip code:  80221
      Please enter the home value: 222222
      Please enter the annual interest rate: 4.25
      ```

    - Use the **setters** to set the values in the following data fields of the object:
        - Set the values for the loan identifier and home value fields first.
        - Then set the values for the loan amount and loan annual interest rate.

      Be sure to pass any necessary parameters to the setters.
    - After setting all the data fields, display a couple of blank lines to separate input and output.
    - Call the two static display methods:
        - A new method to display the loan details (details below)

- A revised method to calculate and display the loan payment information (details below)

The **main** method code should ***not*** calculate any results itself, nor display any loan details or loan payment information. All this should be done from within the two display methods, after being called from the **main** method code.

- The new static method to **display the loan details** will:
  - Have only one parameter: the mortgage loan object
  - Use the **getters** to get and display the loan's information (loan identifier, loan amount, loan length, and annual percentage rate), formatted as shown in the Sample Output below:
    - The loan information is displayed under the header "Loan Details" and be indented.
    - The loan amount should be displayed to 2 decimal places
    - The loan length should be displayed in years
    - The annual percentage rate should be displayed to 3 decimal places
    - All values should line up with each other on the right
- The static method to **calculate and display results** from Assn 3 will need to be modified to use the new mortgage loan object. It will:
  - Have only one parameter: a mortgage loan object
  - Send messages to the loan object to compute the three parts of the monthly mortgage payment, by calling each of the calculation instance methods.
    - **Save** the **result returned** from each calculation method in separate local variables.
  - As in Java Assn 3 display the results:
    - The loan payment figures should be displayed under the header "Monthly Mortgage Payment", and be indented.
    - Display each of the calculated values (the three parts of the mortgage payment) to 2 decimal places, lined up on the decimal points.
    - Calculate and display the total monthly mortgage payment.

  Line all figures up on the right, to the same place as the figures that were output from the loan details method, as shown below.

```
Loan Details
  Loan identifier                  SMIT221
  Loan amount                   199999.80
  Loan length                    30 years
  Annual interest rate             4.250%

Monthly Mortgage Payment
  Monthly Taxes                    102.08
  Monthly Insurance                 91.00
  Monthly Principle & Interest     983.88
                                  --------
  Total Monthly Mortgage Payment  1176.96
```

Output from new "loan details" method

Output from revised "calculate and display results" method

WARNING: The methods must be implemented exactly as specified in these requirements. If your program produces correct output, but you did not implement the methods as specified (with correct parameter passing and return values), you will lose a significant number of points.

Coding Standards

The program must also follow the **CS210 Coding Standards** from Content section 6.10.

You must ***include*** the following comments:

- Comments at the ***top of the file, above*** the **main** class, 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 3)
- Comments at the ***top of each method***, ***above*** the method header, describing what the method does (***only*** this method – do not refer to actions of any other methods)
  - Include **tags** with names and descriptions of ***each*** parameter and return value.

Delete any default comments supplied by the IDE that you did not use.

Debugging and Testing

Run, test, and debug your Java program, until it works.

Then test your program with different inputs to make sure it provides correct results.

Outline of what your MortgageCalculator.java file code should look like:

```
/*
 * The mortgage calculator program will do the following:
 *   Put program description here (expand to as many lines as needed)
 */

import java.util.Scanner;

/**
 * @author Mary Jones          // your full name
 * @version 1.0, Java Assn 4
 */
public class MortgageCalculator {

    public static void main(String[] args) {
        // Call to method 1 goes here
        // Statement to create object goes here
        // Statements to read user inputs go here
        // Statements to call setters go here
        // Statements to call display methods go here
    }


    // Existing method to display program description goes here


   /**
     * Description of method to display loan details
     *
     * @param name - description of parameter
     */
    public static datatype methodName (parameterList) {
        // Method body display statements go here
    }
```

```
    /**
     * Description of method to calculate and display mortgage payment
     *
     * @param name - description of parameter
     */
    public static datatype methodName (parameterList) {
        // Method body calculation statements go here
        // Method body display statements go here
    }


    /**
     * Description of method to calculate and display mortagage payment
     *
     * @param name - description of parameter
     * @return name - description of return value
     */
    public static datatype method4name (parameterList) {
        // Constant definition goes here

        // Method body statements go here here
    }
}
```

Outline of what your MortgageLoan.java file code should look like:

```
public class MortgageLoan {
    // Data field definitions go here

    // Constructor method documentation and definition goes here

    // Setter method documentation and definitions go here

    // Getter method documentation and definitions go here

    // Calculation documentation and method definitions go here
}
```

Submission

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

Now that you have multiple class files within your project, 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:
    `SmithAssn4.zip`

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

- Then submit your **.zip** file to the **Java Prog Assn 4** 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

Your 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.