Project Assignment 2
50 Points
References: Referenced Textbook and Week 2, 3, 4, 5, & 6 handouts

Skills Required:

1. Object concepts: instantiation, multiple objects, constructors, private/public, static/instance methods, and object methods/constructors
2. Object data field of another object
3. Message dialogs GUIs
4. Arrays of objects
5. Pass parameters to methods and method returns
6. Arithmetic operations, such as mode (%) function

Description:

1. Create a new java project and name it PP2.
2. This project performs a payment for a number of customers, the payment is successful if and only if the customer credit card is valid
3. All the program interactions are performed using Message Dialog Boxes of the JOptionPane class
4. Data entries validations are required for the empty entries and/or invalid entries
5. The program must implement the Hans Luhn check algorithm to validate the customer card based on Mod 10 checks, as described below:

   Credit card numbers follow certain patterns. A credit card number must have between 13 and 16 digits. It must start with:

   4 for Visa cards
   5 for Master cards
   37 for American Express cards
   6 for Discover cards

   In 1954, Hans Luhn of IBM proposed an algorithm to check credit card numbers. The algorithm determines if a card number is entered correctly by a user. Almost all credit card numbers are following the Luhn check or named as the Mod 10 check. For illustration, consider the card number 4388576018402626, and the algorithm follows these steps:

   1. Double every second digit from right to left. If doubling of a digit results in a two-digit number, add up the two digits to get a single-digit number.
   2 * 2 = 4
   2 * 2 = 4

4 * 2 = 8
1 * 2 = 2
6 * 2 = 12 (1 + 2 = 3)
5 * 2 = 10 (1 + 0 = 1)
8 * 2 = 16 (1 + 6 = 7)
4 * 2 = 8

2. Now add all single-digit numbers from Step 1.
4 + 4 + 8 + 2 + 3 + 1 + 7 + 8 = 37

3. Add all digits in the odd places from right to left in the card number.
6 + 6 + 0 + 8 + 0 + 7 + 8 + 3 = 38

4. Sum the results from Step 2 and Step 3.
37 + 38 = 75

5. If the result from Step 4 is divisible by 10, the card number is valid; otherwise, it is invalid. For example, the number 4388576018402626 is invalid, but the number 4388576018410707 is valid.

Write a program that prompts the user to enter a credit card number as a long integer. Display whether the number is valid or invalid.

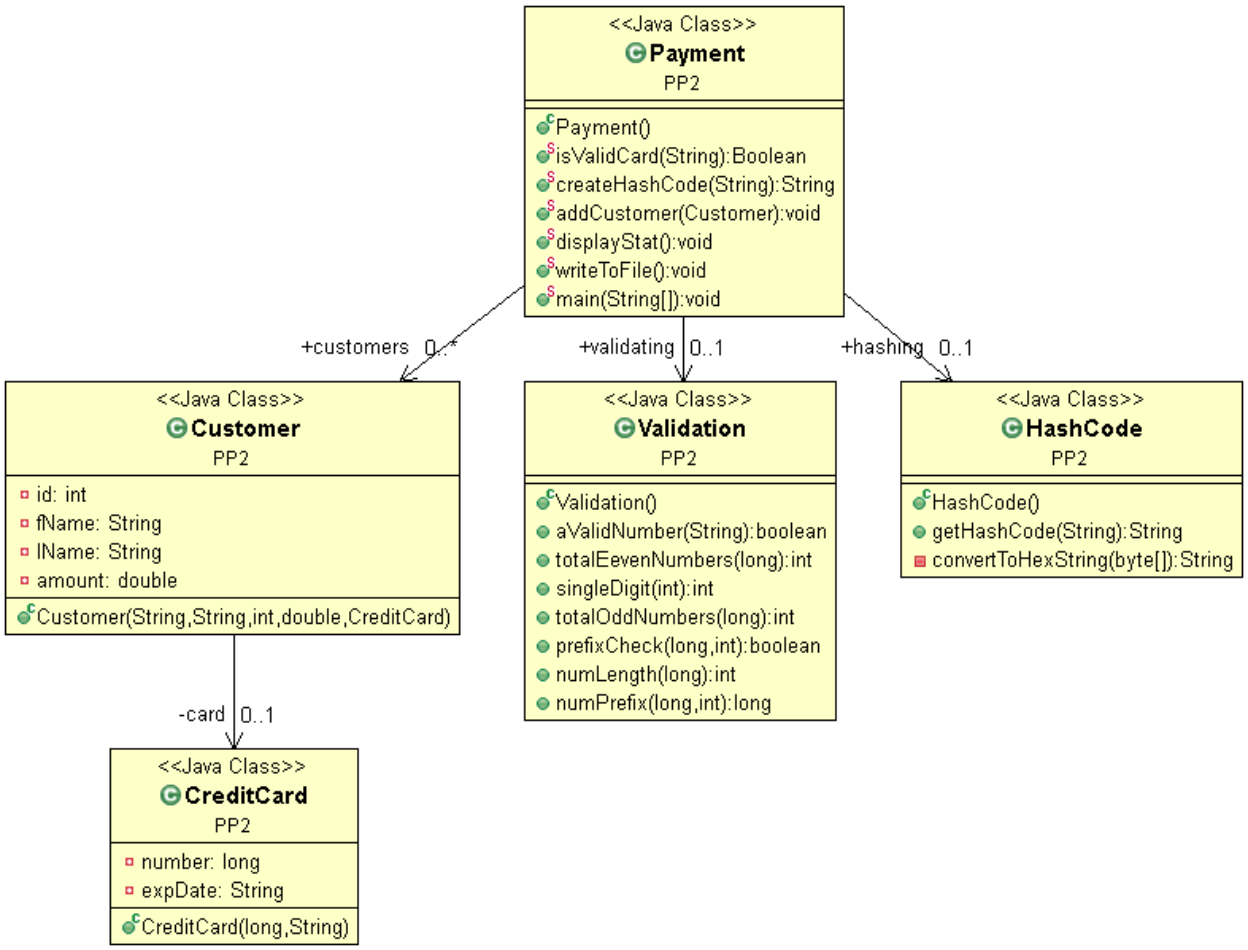
Here are sample runs of the program, using Message Dialog Boxes for user interactions:

Sample 1:
Enter a credit card number as a long integer: 4246345689049834
4246345689049834 is invalid

Sample 2:
Enter a credit card number as a long integer: 4388576018410707
4388576018410707 is valid

You should test the program for some valid and invalid card numbers. See the associated credit cards text file (CreditCards.txt) that has valid card numbers in order to test your program

6. The implementation of the Luhn check algorithm must NOT use any custom arrays, such as arrays of Strings, Int, Long, etc, it should only be implemented based on the Mode 10 arithmetic operation
7. Once the entered card number by a user is valid, the program can perform the payment, and it adds the customer object data to an array of customers, such as Customer[] customers
8. The size of the customers array is equal to the user input of variable n, so it should hold at most n Customer objects, that means you should check the array size before adding a new a Customer object to the array, and a user should see a notification message if the user input exceeds the array size
9. The program should allow users to enter customers payments information, until a user enters a customer id equals to 0, or until it acceptably adds at most n customers payments information
10. The program displays some simple statistics at the end, when a user completes entering the payment information; then the program should traverse the array of customers objects to find out the payments average, the maximum customer payment, and the minimum customer payment
11. After showing the statistics to the user, the customer information with only valid card numbers should be stored in a text file (Customer.txt), all customer information is stored as a plain text, except for the card numbers that should be encrypted using the one way hash method in the `HashCode.java` class
12. Each customer object information must be stored in an exactly one line, so that approximately, there will be about n text lines in the text file



Task Specifications:

1. Design your project PP2 as specified in the UML class diagram model shown above
2. You should copy the Java source code files associated with this document into your eclipse under project PP2, copy all java files in the source SourceFiles folder to your PP2 project folder
3. You should not add any extra methods in these classes, unless it is stated in the Java source files comments to be added
4. Implement the class methods as described in the Java files comments
5. `Payment.java` is the main class that has the static main (entry method) method that should interact with users, and it implements the business logic of the project using the associated classes as shown in the UML class diagram
6. All the `Payment.java` attributes/data fields should be class variables, and also all methods are class methods
7. Since all the `Payment.java` methods are class methods, that means, no need to create an object to call these methods
8. `Payment.java` should have a class array attribute of the Customer objects with size n, `Customer[] customers`

9. `Payment.java` methods should perform the following business logic requirements:
   a) `isValidCard()` // this will check whether a card is valid using the `Validation.java` class
   b) `createHashCode()` // creates a hash code for the credit card number to be stored in file using the `HashCode.java` class
   c) `addCustomer()` // it adds a new customer to the array of customers once the payment was made successful
   d) `displayStat()` // it displays the customers payments AVG, MAX, and MIN payments, only for accepted payments with valid cards numbers
   e) `writeToFile()` // writes customer data to file, the credit card number should be encrypted using one-way hash method in the `HashCode.java` class before storing it in the output file
   f) `main (String[] args)` // Runs once the program starts, it should use Message Dialog Boxes to receive data from users and displays data to users
10. All the other class attributes/methods are object/instance attributes/methods; attributes are only accessed within the same class only, and object methods are accessible from outside
11. A user must enter all the customer payment information as shown in the `Customer.java` data fields, and also the credit card information as shown in the `CreditCard.java` data fields
12. You should complete the setter/getter and toString methods in the `Customer.java` and `CreditCard.java` classes
13. Use the toString methods in the `Customer.java` and `CreditCard.java` classes to display the customer information, once the payment is performed successfully with a valid card number
14. No methods are needed to be implemented in the `HashCode.java` class, you only need to create an object of this class, and use its `getHashCode()` method to encrypt the customer card number
15. All instance methods in the `Validation.java` class are required to be implemented following the aforementioned Luhn check algorithm, except for `aValidNumber()` method that only need to be called from an outside class, which in turn, it calls other methods in the `Validation.java` class, this method should not be changed
16. The unimplemented instance methods in the `Validation.java` class should be implemented as described in the method comments, NO custom arrays should be used to implement these methods

Evaluation Criteria:

1. All tasks must be completed to receive complete credit for this project
2. The application should perform all the requirements correctly, including read user inputs and write data to file

3. The application should not crash from improper input or other user's interactions
4. The application should notify the user for improper inputs or empty text fields for validation
5. The application should stop when a user enters id 0, or successfully enter n customers payments

Submission:

Copy the .java source files from the *src* folder in your *work space* to another folder that should be named following the provided naming format in this course, then zip and upload the file under this assignment answer in Canvas. You should read the project demo document in Canvas before the review time.

File Name: *FLLLLPP2.zip (F = first letter in your first name and LLLL = your last name)*

Grading Rubric – Project 2 (PP02)

Student Names: _____

Requirements	**Comments**	**Max Points Allocated**	**Points Earned**
General Code Structure: Coding of the Java classes, Payment, Customer etc. classes with the expected methods (3) Use of Java comments in the source code (2) Indentations, good variable names/class members (2) Successful compilation (no compile error) and running of code (able to execute the program) (5)		12	
Input, Output, User Interface: Exception handling of the input values (e.g., program will not crash is no value is entered, empty space is entered, invalid value is entered) (4) Use of dialog boxes, as appropriate (3)		10	

Formatting of the output values (3)			
Functionality: Validation of credit card number and Mod10 check (10) Generate a discount between 5% and 20% (3) Hold at most n Customer objects (3) Program must allow users to enter customer payments information, until a user enters a customer id equals to 0 (3) Finding payments average, the maximum customer payment, and the minimum customer payment value (n) Each customer object information must be stored in an exactly one line (3) Use of array (3) Use of Math.random (3)		38	
Total		**50**	

Total: _____/50