# Purpose

The purpose of this assignment is to gain some experience using the ***Eclipse*** ***IDE*** to create, edit, and debug a Java console application with multiple classes. The context of the assignment is the management and verification of credit card information. In doing so, you will use ***JOptionPane*** dialogs, enumerated data types, Strings, date information, and other capabilities of Java.

# Overview

Using ***Eclipse***, create at least two Java classes (you may create more than two) including a driver class to handle all I/O and a ***CreditCard*** class that encapsulates and manages the information for a credit card. The solution will input data from the user, create a credit card object that manages and verifies that data, and produces a brief report. The program should continue as long as the user has additional credit card information to process.

# Specifications

* All I/O should be done in the ***driver*** and it should use ***JOptionPane*** dialogs.
* The driver should be menu driven and give the user an opportunity to enter credit card information from which a report on that card will be generated until the user has no more information to process.
* The information for each credit card will include the card holder’s name (***String***), the credit card number (***String***), and the expiration date in the form ***MM/YYYY*** (***String***). The credit card number should be entered as a string of digits without spaces or other separator characters (for example, the user will type 1234567890120987 rather than 1234 5678 9012 0987).
* The CreditCard class will maintain the information input from the credit card in separate fields (attributes). It may have additional fields for things such as whether the credit card has expired and whether the credit card number is valid. It should also have a field for the credit card type (types used for our purposes are ***INVALID***, ***VISA***, ***MASTERCARD***, ***AMERICAN***\_***EXPRESS***, ***DISCOVER***, and ***OTHER***). The expiration date field can be two separate fields for month and year if you wish.
* The type of card should be managed as an ***enumerated data type*** in Java, and it can be determined from the credit card number. See the attachment at the end of this assignment for details.
* A credit card has expired if the expiration date is this month or before – e.g., a card that expires 2/2019 expires on 2/1/2019. You will have to get the current date from the computer and compare the current month and current year to the month and year of expiration to determine whether or not the card has expired.
* The attachment at the end of this assignment also describes how to check whether a credit card number is a valid number. Note that a number may be valid without belonging to the customer whose name is on the card, but your task does not involve matching the name and the number – only verifying that the number is a valid number.

# Documentation

Program documention is **NOT** optional. See the **Documentation Policies** on the course website. If your version of Eclipse is set up and used as we described in the first lab session, it will provide much of your required documentation. However, it is up to you to fill in the details (such as method and parameter purpose) that Eclipse cannot supply.

# Deliverables

* Next week, submit your initial design for this solution. The website describes a design document and gives an example. No code is required in the initial design.
* By the specified due date, complete, test, and debug your assignment. Turn in only the .java files in their packages according to the instructions in the course fact sheet (on the website) along with the updated final design document.
* Due dates are posted on the website

# Hints and Suggestions

* Data in this assignment is ***String*** information. You will need to use some of the methods in the ***String*** class and methods in the ***Integer*** class to manipulate this data for this assignment. Complete documentation for these classes can be found at <https://docs.oracle.com/javase/10/docs/api/index.html?overview-tree.html>. The textbook also contains helpful information. Of particular interest are methods such as ***length*** ( ), ***split* ( )**, and ***substring*** ( ) in the ***String*** class and the ***parseInt*** ( ) method in the ***Integer*** class.
* You can use ***substring*** to access one character at a time in the credit card number, convert it to an integer using ***parseInt*** from the ***Integer*** class, and then do the manipulation required in the following instructions to validate the number. Do not allow the validation to be more complicated than necessary – more than about 15 lines of code is excessive.
* To get the ***current*** ***date*** from the computer to determine whether the expiration date has passed, use the following code (must ***import java.util.Calendar***):



* The code above leaves the ***current*** ***date*** in the ***Calendar*** object named ***today***. Individual components of the date such as the ***month number*** and the ***year*** can be retrieved using code such as the following.

***int currentMonth = ; // currentMonth is now in range 1 - 12***

***int currentYear = today.get (*Calendar.YEAR*);*** ***// currentYear is a 4-digit number such as 2019***

**Introduction (edited from an article posted on** [www.CodeProject.com](http://www.CodeProject.com)**)**

The following article provides brief details about what information is contained in a credit card number and demonstrates to how to validate a credit card number using the mod 10 (Luhn) algorithm.

**Background**

**Card Length**

Typically, credit card numbers are all numeric and the length of the credit card number is between 12 digits and 19 digits. The spaces on a credit card between groups of numbers are only for human readers and are not part of the card number.

* 14, 15, 16 digits – Diners Club
* 15 digits – American Express
* 13, 16 digits – Visa
* 16 digits - MasterCard

For more information please refer <http://en.wikipedia.org/wiki/Bank_card_number>.

**Embedded information**



1. ***Major Industry Identifier (MII)***

The first digit of the credit card number is the Major Industry Identifier (MII). It designates the category of the entity which issued the card.

* 1 and 2 – Airlines
* 3 – Travel
* 4 and 5 – Banking and Financial
* 6 – Merchandising and Banking/Financial
* 7 – Petroleum
* 8 – Healthcare, Telecommunications
* 9 – National Assignment

1. ***Issuer Identification Number***

The first 6 digits are the Issuer Identification Number (IIN). It will identify the institution that issued the card. Following are some (but by no means all) of the familiar IIN’s.

* **American Express** – 34xxxx, 37xxxx
* **Visa** – 4xxxxxx
* **MasterCard** – 51xxxx – 55xxxx
* **Discover** – 6011xx, 644xxx, 65xxxx

The remaining digits among these first 6 may identify a bank or other entity that issued the card. For example, if the first six digits are 416724, the card is a Wells Fargo Bank Debit Visa (USA) card.

### **Issuer Identification Numbers Samples**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Issuer Identification Number | **420767** | | Card Brand | **VISA** | | Issuing Bank Name | **CAPITAL ONE BANK N.A** | | Card Type | **CREDIT** | | Card Level | **SIGNATURE** | | Iso Country Name | **UNITED STATES** | | |  |  | | --- | --- | | Issuer Identification Number | **414709** | | Card Brand | **VISA** | | Issuing Bank Name | **CITIBANK N.A.** | | Card Type | **CREDIT** | | Card Level | **PLATINUM** | | Iso Country Name | **UNITED STATES** | |
| |  |  | | --- | --- | | Issuer Identification Number | **540168** | | Card Brand | **MASTERCARD** | | Issuing Bank Name | **JPMORGAN CHASE BANK N.A.** | | Card Type | **CREDIT** | | Card Level | **STANDARD** | | Iso Country Name | **UNITED STATES** | | |  |  | | --- | --- | | Issuer Identification Number | **374322** | | Card Brand | **AMERICAN EXPRESS** | | Issuing Bank Name | **BANK OF AMERICA N.A.** | | Card Type | **CREDIT** | | Card Level | **CLASSIC** | | Iso Country Name | **UNITED STATES** | |
| |  |  | | --- | --- | | Issuer Identification Number | **374328** | | Card Brand | **AMERICAN EXPRESS** | | Issuing Bank Name | **WWW.AEPREPAID.COM** | | Card Type | **DEBIT** | | Card Level | **PREPAID** | | Iso Country Name | **UNITED STATES** | | |  |  | | --- | --- | | Issuer Identification Number | **512107** | | Card Brand | **MASTERCARD** | | Issuing Bank Name | **CITIBANK N.A.** | | Card Type | **CREDIT** | | Card Level | **GOLD** | | Iso Country Name | **UNITED STATES** | |

1. ***Account Number***

Taking away the 6 identifier digits and the last digit, the remaining digits are the person’s unique account number (the 7th and following digits up to but excluding the last digit)

1. ***Check Digit***

The last digit is known as a check digit or checksum. It is used to validate the credit card number using the Luhn algorithm (Mod 10 algorithm) that is described below.

**Luhn algorithm (Mod 10)**

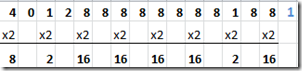
The Luhn algorithm, also known as the “modulus 10″ or “mod 10″ algorithm, is a simple checksum algorithm used to validate a variety of identification numbers, such as credit card numbers, IMEI numbers, National Provider Identifier numbers in the US and Canadian Social Insurance Numbers. It was created by IBM scientist Hans Peter Luhn (<http://en.wikipedia.org/wiki/Luhn_algorithm>).

When you implement ecommerce applications, it is a best practice to validate the credit card number before sending it to the bank for validation. Note that just because a credit card number is verified by Luhn’s algorithm, that does not mean the number has been assigned to the person whose name is on the card. It only means that the number meets all requirements for being a valid number. A number that fails Luhn’s algorithm cannot be a valid credit card number.

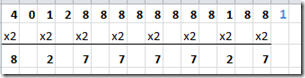
Here are the Luhn steps which can used to validate the credit card number.

**4 0 1 2 8 8 8 8 8 8 8 8 1 8 8 1**

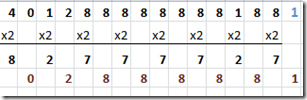
**Step 1** - Starting with the check digit double the value of every other digit (right to left every 2nd digit)



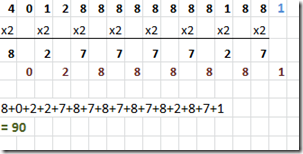
**Step 2** - If doubling a number results in a two digit number, add the digits to get a single digit number. This will result in eight single digit numbers.



**Step 3** - Now add the un-doubled digits to the odd places



**Step 4** - Add up all the digits in this number



If the final sum is divisible by 10, then the credit card number is **valid**. If it is not divisible by 10, the number is **invalid**.

**Some Test Data**

Some data you may use to test is in the following table. Use other (both valid and invalid) data as well, possibly including your own credit card number(s).

|  |  |
| --- | --- |
| MasterCard | 5499990123456781 |
| Visa | 4003000123456781 |
| AMEX | 373953244361001 |
| Discover | 6011000997235373 |

Bottom of Form