**CS1180**

**Project 3**

**Purpose:** This program determines if a credit card number is valid. It will give you experience manipulating strings, writing loops, and using methods.

For full credit on this assignment, **you must use separate methods for each task** and each check. Be sure to declare all variables as “locally” as possible. **Global variables are not allowed**!

**Upload two separate zip files to Pilot – one for Part I and one for Part II.**

**Part I:** Prompt the user to enter the card number, store it into a string. Use input validation to ensure that it contains only spaces and digits. Then remove any embedded spaces within the string and check for validity as indicated below. Hint: StringBuilder can be very useful! Output the card number and card type (unknown if the type cannot be determined). If the card type could be determined, your output should indicate whether the number was valid or invalid. **A sample program run is provided on the next page.**

**Part II:** Re-write your program so that it reads card numbers from a file rather than asking for user input. A file is provided on Pilot for your use. Be sure to use a try..catch in case the file does not open properly. Do not assume the test file used when grading your project will be the same length as the provided file. Be sure to read and process card numbers until the end of file is reached. You may assume that each card number is on a separate line within the input file. If a card number does not contain all digits, simply output “*[cardNumber]* card cannot be processed.” Do not perform any additional checks. Modify your code further so that the output is displayed to a separate file. Display “Program finished” to the console at the end of your program.

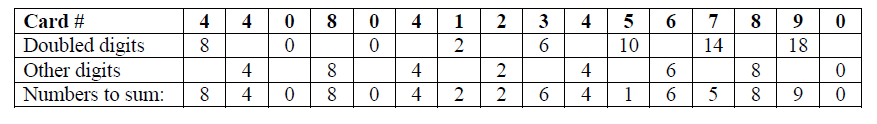
**Testing validity:** Perform the following checks/determinations on the input string in the order given. The string must be all digits and embedded spaces must be removed before these checks. If any check shows the card number is not valid, do not perform subsequent checks.

* Determine card type based on the following. It must be one of these or it is not valid.
* Visa card numbers always begin with a 4.
* MasterCard numbers begin with a 51, 52, 53, 54, or 55.
* American Express numbers begin with 34 or 37.
* Discover card numbers begin with 6011.
* Determine if the length of the card number is valid.
  + Visa card numbers are either 13 or 16 digits in length
  + MasterCard numbers are 16 digits in length
  + American Express numbers are 15 digits in length
  + Discover card numbers are 16 digits in length.
* Determine if the checkSum value is valid. The checkSum algorithm works as follows:

Beginning with the **2nd digit from the right** and **working right to left** (and remembering that not all card numbers are the same length), double every-other digit; if the resulting number is greater than or equal to 10, subtract 9. Add this result to a running total. Then, beginning with the last digit on the right and working right to left, add every-other digit to the running total used in the previous step.

If the resulting total is evenly divisible by 10, the credit card number is valid.

The following table illustrates this algorithm using card number 4408 0412 3456 7890



The resulting sum is 67. Since this is not evenly divisible by 10, this is not a valid card number.

**Sample program runs for Part I:**

**Run #1:**

Please enter the credit card number:

5123 4567 ab01 2345

The credit card number must contain only the digits 0-9; please re-enter:

5123 4567 8901 2\*45

The credit card number must contain only the digits 0-9; please re-enter:

5123 4567 8901 2345

The card number is 5123456789012345

The card type is MasterCard

The card number is invalid.

**Run #2:**

Please enter the credit card number:

5123 4567 8901 2346

The card number is 5123456789012346

The card type is MasterCard

The card number is valid.

**Run #3:**

Please enter the credit card number:

601155126708921

The card number is 601155126708921

The card type is Discover

The card number is invalid.

**Run #4:**

Please enter the credit card number:

1234 5678 9012 3456

The card number is 1234567890123456

The card type is unknown

You may run additional tests by using your own credit card numbers or using the numbers in the provided file.

**Grading:**

Part 1. (65 pts)

The program…

(10 pts) correctly gets a card number from the user (all digits) and removes embedded spaces

(10 pts) correctly determines card type (or indicates type is unknown)

(8 pts) correctly checks that length is valid

(12 pts) correctly computes and uses checkSum to determine if card number is valid

(10 pts) displays output in a user-friendly format

(15 pts) uses a separate method for each task with appropriate return types and parameters

Part 2. (20 pts)

The program…

(10) correctly opens an input file and reads/processes all card numbers in the file

(10) correctly writes the output to an output file and displays a “Program finished” line to the

console

Parts 1 and 2 – Style (15 pts)

(3 pts) meaningful identifiers (variable and method names)

(3 pts) proper formatting – indenting, alignment of braces, etc.

(4 pts) meaningful comments to make algorithm intent clear

(5) javadoc comments before each method as described in the Style Requirements document

**Turn in information:**

Upload to Pilot separate zip files for part 1 and for part 2. Also, upload to Pilot separate .txt files with your code for part 1 and for part 2.