CHRIST(Deemed to be University)
Department of Computer Science
MAI271-JAVA Programming Date:23-10-2024
Lab-Exe:1

Program 1:

You are a **security expert** at the **JJJ (JavaJuggernautsJourney)** bank, which is facing numerous cases of invalid credit card numbers being used. They are encountering numerous cases of invalid credit cards of their bank in use. You are a JAVA geek of 2 MScAI&ML and the Chief Technical Officer (CTO) wants you to take up the job of "security expert" in the bank. Though they are convinced of your skillset, the technical team lead wants to implement the final test using the JAVA program and conditions are described below and You are required to implement the solution using **constructors**.

Task:
Write a Java program that reads a credit card number (variable name `ccNumber`) and validates it according to the following rules.

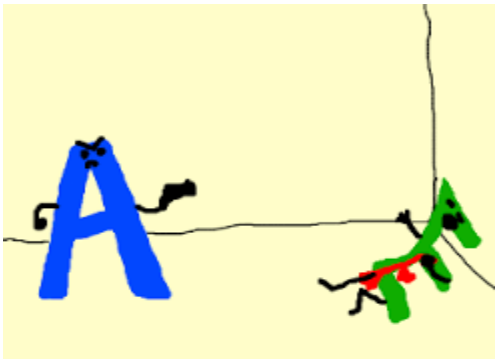
Rules for a valid credit card number:
1. The credit card number should have a minimum of 8 digits and a maximum of 9 digits. If the number is outside this range, display: **"Invalid credit card number"**.
2. If the number is within the valid range, perform the following steps using a switch-case to check if the credit card number is valid:
   - **Step a:** Remove the last digit of the `ccNumber`.
   - **Step b:** Reverse the remaining digits.

- **Step c:**Double the digits that are in the odd-numbered positions (1st, 3rd, 5th, etc.). If the result of the doubling is a double-digit number, add the digits of that result (for example, if `5` is doubled to `10`, then add `1 + 0 = 1`).
- **Step d:**Add up all the digits.
- **Step e:**Subtract the last digit obtained in step a from 10.
- **Step f:**Compare the result of step e with the last digit obtained in step a. If they match, the card number is valid; otherwise, it is invalid.

Example:
For `ccNumber = 123467575
- Step a:Last digit = `5`, remaining number = `12346757`
- Step b:Reverse the number: `75764321`
- Step c:After doubling the odd-positioned digits: `55568341`
- Step d: Sum of all digits = `37`
- Step e: `10 - 7 = 3`
- Step f: `3` (result of step e) does not match `5` (last digit from step a), so the card is invalid.

Instructions:
- Implement this using Java and use a constructor for reading the credit card number input and validating the card based on the rules provided.
- Use a switch-case to manage the steps from Step a to Step f.
- Display appropriate messages for whether the credit card is valid or invalid.

Program 2:



2)Imagine you really enjoy playing games, especially "**CALL OF DUTY**," and you're pretty good at using Java. Now, someone gives you a cool task: make a game called "**Alphabet War**." This game is all about a playful fight between letters, split into two teams—the left-side letters and the right-side letters.

The left-side letters act like superheroes, each having their own strengths:
- 'w' is super strong (strength 4)
- 'p' is quite strong (strength 3)
- 'b' is okay strong (strength 2)
- 's' is kinda strong (strength 1)

On the other side, the right-side letters also have their own strengths:
- 'm' is super strong (strength 4)
- 'q' is quite strong (strength 3)
- 'd' is okay strong (strength 2)
- 'z' is kinda strong (strength 1)

Your job is to set up some rules for this friendly letter battle. If the left-side letters win, you say "Left side wins!" If the right-side letters win, you say "Right side wins!" If it's a tie or something else, you say "Let's fight again!"

To make things fun in Java, you create a class called `AlphabetWarGame`. This class helps you figure out who wins using different rules for different scenarios. You can even change the strengths of the letters if you want.

For example, you might create this class in different ways using **constructor overloading:**
- One version where the strengths are set by default.
- Another version where you can customize the strengths.

And, you can figure out who wins in different ways using **method overloading:**
- One method where you pass in just one word.
- Another method where you pass in separate left and right words.

Following are few sample testcases for you to implement:

```
AlphabetWar("z") => Right side wins!
AlphabetWar("zdqmwpbs")=> Let's fight again!
AlphabetWar("wwwwwwz" )=> Left side wins!
```

It's like making your own world of letter superheroes in Java, having a blast experimenting with different strengths and battle scenarios!