FEU Institute of Technology
COLLEGE OF ENGINEERING • COLLEGE OF COMPUTER STUDIES

Credit Card Validation Systems with GUI

Final Project submitted to
Professor Elisa Malasaga

Computer Science Department
FEU Institute of Technology

In partial fulfillment of the requirements in
CS0035 - PROGRAMMING LANGUAGES
Second Trimester, Academic Year 2022-2023

Cajipe, Ralph Henrik I
Español, Kyle G.
Segundo, Sally Jr. P.
Esquivel, Vence T.
Bellosillo, Shiela D.

Section TN32
29/03/2023

FEU Institute of Technology
COLLEGE OF ENGINEERING • COLLEGE OF COMPUTER STUDIES

FINAL PROJECT
CS0035
COMPARISON OF TWO PROGRAMMING LANGUAGES WITH ONE SYSTEM APPLICATION
DOCUMENTATION

I. **NAME OF THE SYSTEM**

Credit Card Validation Systems with GUI

II. **OBJECTIVE OF THE SYSTEM**

- **To validate if the inputted credit card number is valid and to check what kind of card it is, mainly VISA, AMEX and MasterCard.**

III. **COMPARISON OF TWO LANGUAGES (Python & Java)**

COMPARISON	
FEATURES/ ADVANTAGES OF THE (FIRST LANGUAGE)	**FEATURES/ ADVANTAGES OF THE (SECOND LANGUAGE)**
● User-friendly interface: The GUI is designed to be user-friendly, with clear instructions for users to input their credit card number and initiate the validation process. ● Input validation: The program validates the user input to ensure it only contains digits and optional spaces, and prompts the user to enter a valid credit card number if it does not meet the requirement. ● Credit card type identification: The program identifies the type of credit card (e.g., VISA, AMEX, MASTERCARD) based on the first few digits of the credit card number. ● Credit card number validation: The program validates the credit card number using Luhn's	● User-friendly interface: The program has a simple graphical user interface that allows users to input credit card numbers and check their validity. ● Built using Java Swing: The program is built using Java Swing, which is a powerful GUI toolkit that enables developers to create user interfaces for Java applications. ● Error handling: The program includes error handling code that checks for invalid input

algorithm, which is a widely accepted method for validating credit card numbers. ● Clear validation result: The program displays a clear validation result to the user, indicating whether the credit card number is valid and its type (if valid), or that the credit card number is invalid. ● Reset button: The program provides a reset button for users to clear the input field and validation result label. Overall, this Python code provides a useful and user-friendly way to validate credit card numbers using a GUI.	and displays appropriate error messages to the user. ● Card validation: The program includes a validation algorithm that determines the type of card based on its number and checks if it is a valid card number for that type. ● Efficient: The program uses efficient algorithms, such as Luhn's algorithm, to validate card numbers and determine their type. ● Portable: The program can be compiled and run on any platform that supports Java, making it highly portable.
VARIABLE DECLARATION ● tk - A variable name used to import the tkinter module for creating GUIs. ● CreditCardValidatorGUI - A class name that serves as the blueprint for the credit card validator GUI. ● master - A variable used to store the tkinter root window object. ● self - A variable that refers to the instance of the CreditCardValidatorGUI class. ● label - A variable that refers to a Label widget used to display a prompt for the user. ● entry - A variable that refers to an Entry widget used for the user to input the credit card number.	**VARIABLE DECLARATION (SECOND LANGUAGE)** ● In the main method, there are several local variable declarations, such as: ● JFrame frame: a reference variable of type JFrame used to hold the reference to the main GUI window. ● JPanel mainPanel: a reference variable of type JPanel used to hold the reference to the main panel of the GUI. ● JPanel inputPanel: a reference variable of type JPanel used to

● button_frame - A variable that refers to a Frame widget used to hold the Validate and Reset buttons. ● validate_button - A variable that refers to a Button widget used to trigger the validation process. ● reset_button - A variable that refers to a Button widget used to reset the input field. ● result_label - A variable that refers to a Label widget used to display the result of the validation. ● n - A variable used to store the credit card number as an integer in the is_valid and get_type functions. ● type - A variable used to store the credit card type as a string in the is_valid and get_type functions. ● i - A variable used to loop through the digits of the credit card number in the check_sum function. ● sum - A variable used to store the sum of the digits in the credit card number during the validation process in the check_sum function.	hold the reference to the panel containing the card number input field. ● JLabel inputLabel: a reference variable of type JLabel used to hold the reference to the label for the card number input field. ● JTextField inputField: a reference variable of type JTextField used to hold the reference to the text field for the card number input. ● JTextArea resultArea: a reference variable of type JTextArea used to hold the reference to the text area for displaying the validation result. ● JScrollPane resultScrollPane: a reference variable of type JScrollPane used to hold the reference to the scroll pane that contains the result text area. ● JButton checkButton: a reference variable of type JButton used to hold the reference to the "Check" button. ● JButton resetButton: a reference variable of type JButton used to hold the

	reference to the "Reset" button. ● In the getType method, there is one local variable declaration: ● int sum: an integer variable used to hold the sum of the digits in the card number, as part of the Luhn's algorithm validation process. ● In the checksum method, there are several local variable declarations: ● int sum: an integer variable used to hold the sum of the digits in the card number, as part of the Luhn's algorithm validation process. ● boolean alternate: a boolean variable used to keep track of whether the current digit is an alternate digit (i.e., every other digit), as part of the Luhn's algorithm validation process.
CONTROL STRUCTURE (FIRST LANGUAGE) ● Conditional statements (if-else statements) are used in the following places:	**CONTROL STRUCTURE (SECOND LANGUAGE)** ● Conditional statements (if-else if-else) in the isValid() and getType() methods to

• In the validate() method to check if the input is a valid credit card number and display the appropriate message • In the is_valid() function to check if a credit card number is valid for a given type and return True or False • In the get_type() function to determine the type of a credit card number and return the type as a string • Loops are used in the following places: • In the check_sum() function to iterate through the digits of the credit card number and compute the sum • In the get_type() function to iterate through the digits of the credit card number and check the starting digits • In the reset() method to delete the contents of the input field • Event-driven programming using tkinter library: • This code uses the tkinter library to create a GUI, with widgets such as Label, Entry, Button, and Frame. • Each widget can trigger an event when the user interacts with it, such as clicking a button or entering text in an input field. • The CreditCardValidatorGUI class defines methods to handle these events, such as the validate() method that is called when the "Validate" button is clicked. • The mainloop() function of tkinter runs an event loop to listen for events and execute the corresponding methods.	determine if a card number is valid for a given card type and to determine the type of a card based on its number. • A loop (for) in the checksum() method to iterate over every other digit of a card number and perform a calculation based on the Luhn's algorithm. • Event listeners (ActionListener) for the checkButton and resetButton buttons that respond to user input and update the user interface accordingly.

LOOPING STATEMENT

- The Credit Card Validator GUI program has one looping statement in the check_sum function which is used to loop through the digits of the credit card number:

```
for i in range(len(str(n)) - 2, -1, -2):
sum += int(str(n)[i]) * 2
if int(str(n)[i]) * 2 > 9:
sum += 1
for i in range(len(str(n)) - 1, -1, -2):
sum += int(str(n)[i])
```

- The first loop starts at the second-to-last digit and iterates backwards by twos until the first digit is reached. Within the loop, every other digit is multiplied by 2, and if the product is greater than 9, 1 is added to the sum of digits of the product.

- The second loop starts at the last digit and iterates backwards by twos until the first digit is reached. Within the loop, each digit is added to the sum.

- These two loops are used to calculate the sum of digits of the credit card number required for Luhn's algorithm.

LOOPING STATEMENT (SECOND LANGUAGE)

- A for-loop in the checksum method that iterates over every other digit of the card number.
- A while-loop in the main method that listens for events and handles them until the program is closed.

DATA TYPES USED

- Integer - used to represent credit card numbers, and in some instances, indexes of strings.
- String - used to store the credit card number entered by the user and for performing string operations like slicing and replacing.

DATA TYPES USED (SECOND LANGUAGE)

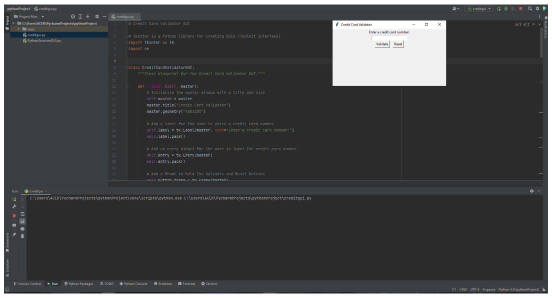
- int: used in the sum variable in the checksum method for storing the sum of digits

● Boolean - used to represent the result of the validation process (True or False). ● List - used to store the starting digits of the credit card numbers for different types of credit cards. ● NoneType - used to represent the absence of a value. It is the default return value for functions and methods that do not explicitly return a value. ● tkinter objects - used to create the graphical user interface. These objects are provided by the tkinter module and include classes like Frame, Label, Entry, Button, and Tk.	● long: used in the n parameter of the isValid and getType methods for storing the credit card number as a long integer value ● String: used in several places for storing text data, including the type variable in the isValid and getType methods, the input field text in the inputField variable, and the result area text in the resultArea variable ● boolean: used for boolean expressions and return values, including the return values of the isValid, getType, and checksum methods ● Color: used in the resultArea variable to set the color of the text in the result area ● Font: used in several places to specify the font for different GUI components, including inputLabel, inputField, resultArea, checkButton, and resetButton.
OTHER SYNTAX/STATEMENT USED ● Conditional statements: These statements are used to execute different code depending on whether a certain condition is true or false. Examples include if, else, elif (short for else-if), switch, and case statements.	**OTHER SYNTAX/STATEMENT USED (SECOND LANGUAGE)** ● Conditional Statements: These are used to make decisions in the program, such as if-else

● Functions: Functions are reusable blocks of code that can be called from other parts of the program. They can accept input arguments, perform a specific task, and return a value. Examples include def (short for define), return, and lambda (used for creating anonymous functions) statements. ● Exception handling: This is a way to handle errors or unexpected events that may occur during program execution. Examples include try, except, finally, and raise statements. ● Input and output: These statements are used for interacting with the user or external files. Examples include print, input, open, read, write, close, and with statements. ● Object-oriented programming: This is a programming paradigm that uses objects as the building blocks of programs. Examples include class, object, attribute, method, and self statements.	statements, switch statements, and ternary operators. ● Loops: These are used to repeat a block of code until a certain condition is met. Common loop types include for loops, while loops, and do-while loops. ● Functions/Methods: These are blocks of code that perform a specific task and can be called from other parts of the program. They can take inputs and return outputs. ● Classes/Objects: These are used in object-oriented programming and allow you to create custom data types with their own properties and methods. ● Exception Handling: This is used to handle errors or unexpected situations that may occur during program execution. ● File I/O: This allows programs to read and write data to files on the computer's storage. ● Comments: These are used to add notes or explanations to the code for other programmers or for yourself in the future.

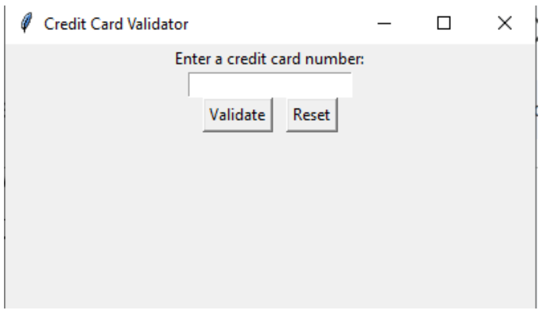
FEU Institute of Technology
COLLEGE OF ENGINEERING • COLLEGE OF COMPUTER STUDIES

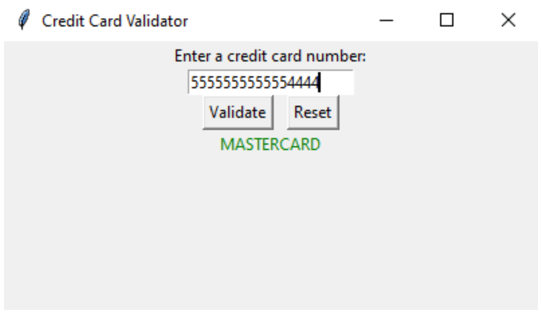
SCREEN SHOTS/LAYOUT



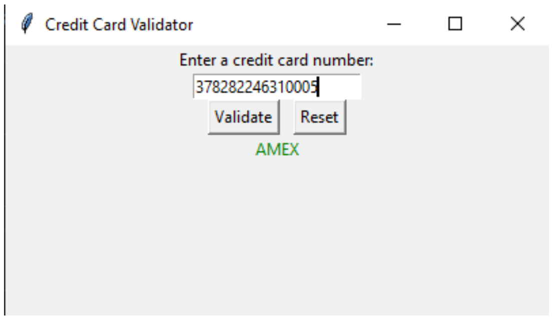
```
# Credit Card Validator GUI

# tkinter is a Python library for creating GUIs (Toolkit Interface)
import tkinter as tk
import re


class CreditCardValidatorGUI:
    """Class blueprint for the Credit Card Validator GUI."""

    def __init__(self, master):
        # Initialize the master window with a title and size
        self.master = master
        master.title("Credit Card Validator")
        master.geometry("400x200")

        # Add a label for the user to enter a credit card number
        self.label = tk.Label(master, text="Enter a credit card number:")
        self.label.pack()

        # Add an entry widget for the user to input the credit card number
        self.entry = tk.Entry(master)
        self.entry.pack()

        # Add a frame to hold the Validate and Reset buttons
```

```
C:\Users\ACER\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ACER\PycharmProjects\pythonProject\creditgui.py
```



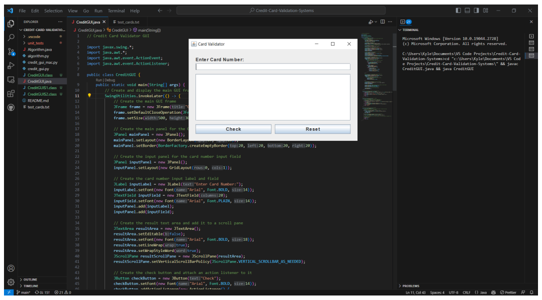


```
5555555555554444
MASTERCARD
```

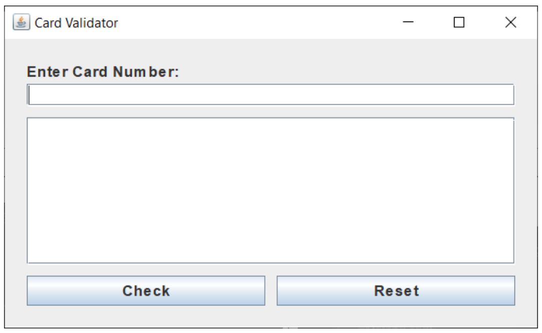


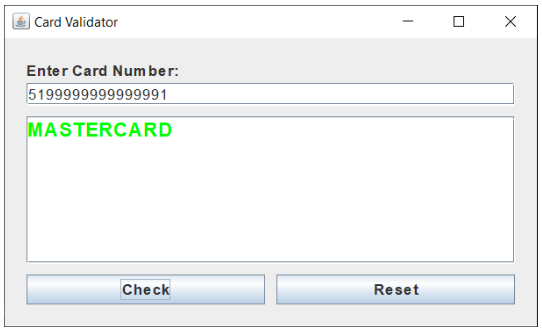
```
378282246310005
AMEX
```

SCREEN SHOTS/LAYOUT (SECOND LANGUAGE)

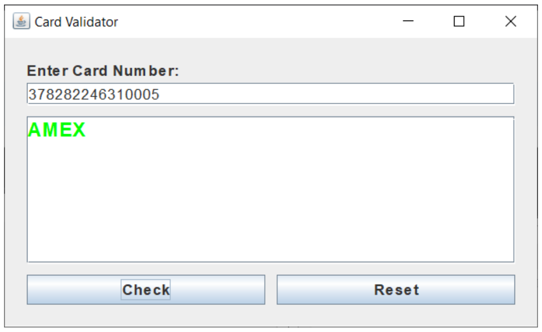


```
// Credit Card Validator GUI

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class CreditGUI {
    public static void main(String[] args) {
        // Create and display the main GUI fra
        SwingUtilities.invokeLater(() -> {
            // Create the main GUI frame
            JFrame frame = new JFrame(title:"
            frame.setDefaultCloseOperation(JFr
            frame.setSize(width:500, height:30

            // Create the main panel for the
            JPanel mainPanel = new JPanel();
            mainPanel.setLayout(new BorderLayo
            mainPanel.setBorder(BorderFactory.createEmptyBorder(top:20, left:20, bottom:20, right:20));

            // Create the input panel for the card number input field
            JPanel inputPanel = new JPanel();
            inputPanel.setLayout(new GridLayout(rows:0, cols:1));

            // Create the card number input label and field
            JLabel inputLabel = new JLabel(text:"Enter Card Number:");
            inputLabel.setFont(new Font(name:"Arial", Font.BOLD, size:14));
            JTextField inputField = new JTextField(columns:20);
            inputField.setFont(new Font(name:"Arial", Font.PLAIN, size:14));
            inputPanel.add(inputLabel);
            inputPanel.add(inputField);

            // Create the result text area and add it to a scroll pane
            JTextArea resultArea = new JTextArea();
            resultArea.setEditable(b:false);
            resultArea.setFont(new Font(name:"Arial", Font.BOLD, size:18));
            resultArea.setLineWrap(wrap:true);
            resultArea.setWrapStyleWord(word:true);
            JScrollPane resultScrollPane = new JScrollPane(resultArea);
            resultScrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

            // Create the check button and attach an action listener to it
            JButton checkButton = new JButton(text:"Check");
            checkButton.setFont(new Font(name:"Arial", Font.BOLD, size:14));
```





```
5199999999999991
MASTERCARD
```



```
378282246310005
AMEX
```

SOURCE CODE

```
# Credit Card Validator GUI

# tkinter is a Python library for creating GUIs
(Toolkit Interface)
import tkinter as tk
import re


class CreditCardValidatorGUI:
       """Class blueprint for the Credit Card
Validator GUI."""

   def __init__(self, master):
       # Initialize the master window with a title
and size
       self.master = master
       master.title("Credit Card Validator")
       master.geometry("400x200")

          # Add a label for the user to enter a
credit card number
       self.label = tk.Label(master, text="Enter a
credit card number:")
       self.label.pack()

       # Add an entry widget for the user to input
the credit card number
       self.entry = tk.Entry(master)
       self.entry.pack()

          # Add a frame to hold the Validate and
Reset buttons
       self.button_frame = tk.Frame(master)
       self.button_frame.pack()

       # Add a button that triggers the validation
process
```

SOURCE CODE (SECOND LANGUAGE)

```
// Credit Card Validator GUI

import javax.swing.*;
import java.awt.*;
import
java.awt.event.ActionEvent;
import
java.awt.event.ActionListener;

public class CreditGUI {
   public static void
main(String[] args) {
       // Create and display the
main GUI frame on the event
dispatch thread

SwingUtilities.invokeLater(() →
{
           // Create the main GUI
frame
           JFrame frame = new
JFrame("Card Validator");

frame.setDefaultCloseOperation(JF
rame.EXIT_ON_CLOSE);
           frame.setSize(500,
300);

           // Create the main
panel for the GUI
           JPanel mainPanel = new
JPanel();

mainPanel.setLayout(new
BorderLayout(10, 10));

mainPanel.setBorder(BorderFactory
                              self.validate_button    =
tk.Button(self.button_frame,       text="Validate",
command=self.validate)
             self.validate_button.pack(side=tk.LEFT,
padx=(0, 10))

        # Add a button to reset the input field
                                 self.reset_button    =
tk.Button(self.button_frame,          text="Reset",
command=self.reset)
        self.reset_button.pack(side=tk.LEFT)

         # Add a label to display the result of the
validation
               self.result_label  =  tk.Label(master,
text="")
        self.result_label.pack()

    def validate(self):
            """Method  to  validate  the  credit  card
number."""
        # Get the credit card number from the entry
widget and remove any trailing spaces
        card_number = self.entry.get().rstrip()

        # Check if the input is a valid credit card
number (contains only digits and optional spaces)
            if  not  re.match("^[0-9]+(\s*[0-9]+)*$",
card_number):
               self.result_label.config(text="Please
enter a valid credit card number")
        else:
                #  Remove  any  spaces  from  the  credit
card number
              card_number = card_number.replace(" ",
"")

            # Determine the type of credit card
            card_type = get_type(int(card_number))
```

```
.createEmptyBorder(20, 20, 20,
20));

            // Create the input
panel for the card number input
field
            JPanel inputPanel =
new JPanel();

inputPanel.setLayout(new
GridLayout(0, 1));

            // Create the card
number input label and field
            JLabel inputLabel =
new JLabel("Enter Card Number:");
            inputLabel.setFont(new
Font("Arial", Font.BOLD, 14));
            JTextField inputField
= new JTextField(20);
            inputField.setFont(new
Font("Arial", Font.PLAIN, 14));

inputPanel.add(inputLabel);

inputPanel.add(inputField);

            // Create the result
text area and add it to a scroll
pane
            JTextArea resultArea =
new JTextArea();

resultArea.setEditable(false);
            resultArea.setFont(new
Font("Arial", Font.BOLD, 18));

resultArea.setLineWrap(true);
```

```
               # If the card is invalid, display
"INVALID" in red text
           if card_type == "INVALID":

self.result_label.config(text="INVALID", fg="red")
           else:
               # If the card is valid, display its
type in green text
                     if is_valid(int(card_number),
card_type):

self.result_label.config(text=card_type,
fg="green")
               # If the card is not valid, display
"INVALID" in red text
               else:

self.result_label.config(text="INVALID", fg="red")

   def reset(self):
          """Method to reset the input field and
result label."""
       self.entry.delete(0, tk.END)
       self.result_label.config(text="")


def is_valid(n, type):
    """Function to check if a credit card number is
valid for a given type."""
    # Check the length and starting digits of the
credit card number for the given type
    if type == "VISA" and (len(str(n)) == 13 or
len(str(n)) == 16):
       return True
   elif type == "AMEX" and len(str(n)) == 15:
       return True
    elif type == "MASTERCARD" and len(str(n)) ==
16:
       return True
   else:
```

```

resultArea.setWrapStyleWord(true)
;
           JScrollPane
resultScrollPane = new
JScrollPane(resultArea);

resultScrollPane.setVerticalScrol
lBarPolicy(JScrollPane.VERTICAL_S
CROLLBAR_AS_NEEDED);

           // Create the check
button and attach an action
listener to it
           JButton checkButton =
new JButton("Check");

checkButton.setFont(new
Font("Arial", Font.BOLD, 14));

checkButton.addActionListener(new
ActionListener() {
               @Override
               public void
actionPerformed(ActionEvent e) {
                   // Validate
the card number input and display
the result
                   long n;
                   try {
                       n =
Long.parseLong(inputField.getText
().trim());
                   } catch
(NumberFormatException ex) {

resultArea.setText("INVALID");

resultArea.setForeground(Color.RE
D);
```

```python
        return False


def get_type(n):
    """Function to determine the type of a credit
card number."""
    # Check the starting digits of the credit card
number to determine its type
    # and call the check_sum function to validate
the number
    if int(str(n)[0]) == 4 and check_sum(n):
        return "VISA"
    elif (int(str(n)[:2]) == 34 or int(str(n)[:2])
== 37) and check_sum(n):
        return "AMEX"
    elif (int(str(n)[:2]) >= 51 and int(str(n)[:2])
<= 55) and check_sum(n):
        return "MASTERCARD"
    else:
        return "INVALID"


def check_sum(n):
    """Function to validate a credit card number
using Luhn's algorithm."""
    # Initialize a sum variable and loop through
the digits of the credit card number
    sum = 0
    for i in range(len(str(n)) - 2, -1, -2):
        # Multiply every other digit by 2, starting
        # with the second-to-last digit, and add
the digits of the products together
        sum += int(str(n)[i]) * 2
        if int(str(n)[i]) * 2 > 9:
            sum += 1

    # Add the sum to the sum of the digits that
weren't multiplied by 2
    for i in range(len(str(n)) - 1, -1, -2):
        sum += int(str(n)[i])
```

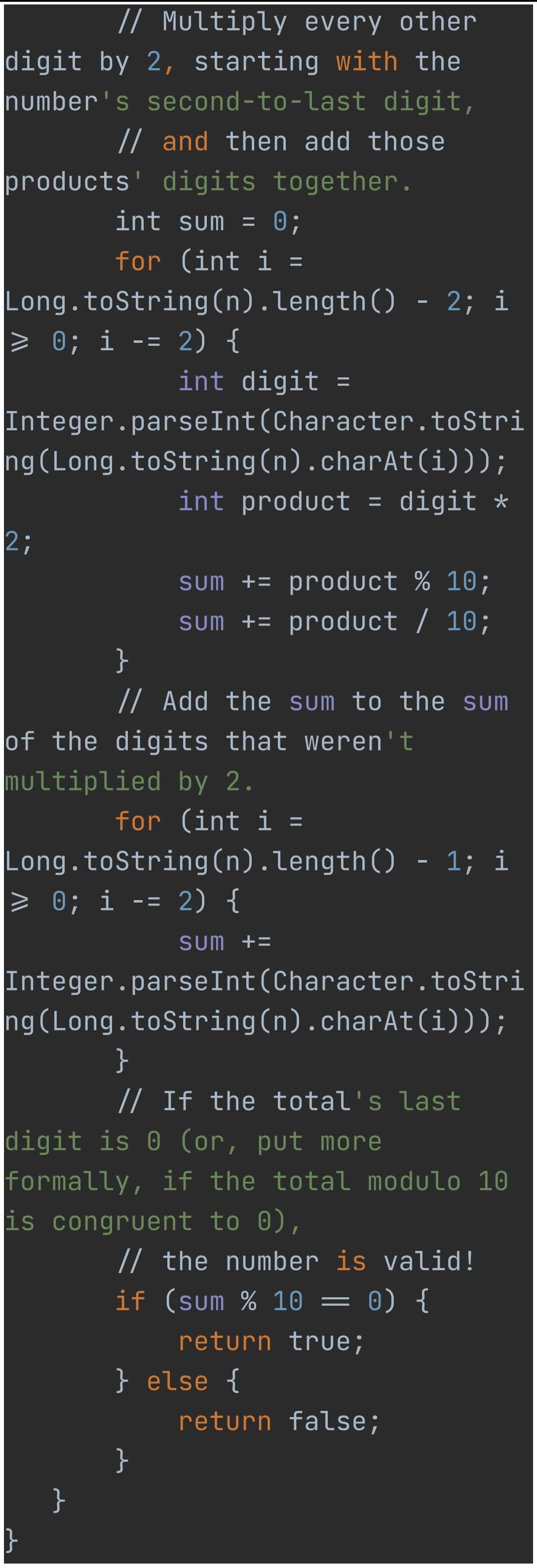
```java
                                        return;
                                }
                                String type =
getType(n);
                                if
(type.equals("INVALID")) {

resultArea.setText("INVALID");

resultArea.setForeground(Color.RE
D);
                                } else {
                                        if
(isValid(n, type)) {

resultArea.setText(type);

resultArea.setForeground(Color.GR
EEN);
                                        } else {

resultArea.setText("INVALID");

resultArea.setForeground(Color.RE
D);
                                        }
                                }
                        }
                });

                // Create the reset
button and attach an action
listener to it
                JButton resetButton =
new JButton("Reset");

resetButton.setFont(new
Font("Arial", Font.BOLD, 14));
```

```
   # If the last digit of the sum is 0, the credit
card number is valid
   if sum % 10 == 0:
       return True
   else:
       return False


if __name__ == "__main__":
   root = tk.Tk()
   my_gui = CreditCardValidatorGUI(root)
   root.mainloop()
```

```
resetButton.addActionListener(new
ActionListener() {
               @Override
               public void
actionPerformed(ActionEvent e) {
                   // Reset the
card number input and result text
area

inputField.setText("");

resultArea.setText("");

resultArea.setForeground(Color.BL
ACK);
               }
           });

           // Create the button
panel for the check and reset
buttons
           JPanel buttonPanel =
new JPanel();

buttonPanel.setLayout(new
GridLayout(1, 2, 10, 10));

buttonPanel.add(checkButton);

buttonPanel.add(resetButton);

           // Add the input
panel, result scroll pane, and
button panel to the main panel

mainPanel.add(inputPanel,
BorderLayout.NORTH);
```

```
mainPanel.add(resultScrollPane,
BorderLayout.CENTER);

mainPanel.add(buttonPanel,
BorderLayout.SOUTH);

           // Add the main panel
to the main frame and display
           frame.add(mainPanel);

frame.setVisible(true);
       });
   }

   // Check if a card number is
valid for a given type of card
   public static boolean
isValid(long n, String type) {
       if (type.equals("VISA") &&
(Long.toString(n).length() == 13
|| Long.toString(n).length() ==
16)) {
           return true;
       } else if
(type.equals("AMEX") &&
Long.toString(n).length() == 15)
{
           return true;
       } else if
(type.equals("MASTERCARD") &&
Long.toString(n).length() == 16)
{
           return true;
       } else {
           return false;
       }
   }
```

```java
   // Determine the type of a card based on its number
   public static String getType(long n) {
       // Check if the card number is a Visa, Amex, or Mastercard
       // using the first few digits of the number and Luhn's algorithm
       if (Long.toString(n).charAt(0) == '4' && checksum(n)) {
           return "VISA";
       } else if ((Long.toString(n).substring(0, 2).equals("34") || Long.toString(n).substring(0, 2).equals("37"))
               && checksum(n)) {
           return "AMEX";
       } else if (Long.parseLong(Long.toString(n).substring(0, 2)) >= 51
               && Long.parseLong(Long.toString(n).substring(0, 2)) <= 55 && checksum(n)) {
           return "MASTERCARD";
       } else {
           return "INVALID";
       }
   }

   // Calculate the checksum using Luhn's algorithm
   public static boolean checksum(long n) {
```

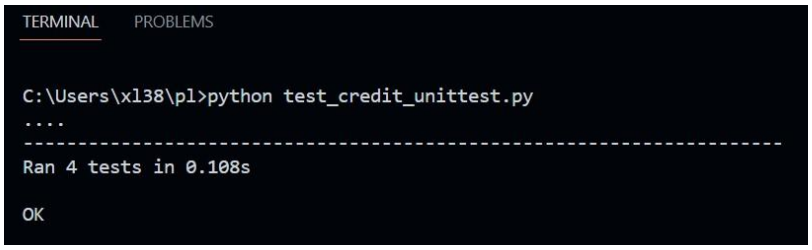


```
        // Multiply every other
digit by 2, starting with the
number's second-to-last digit,
        // and then add those
products' digits together.
        int sum = 0;
        for (int i =
Long.toString(n).length() - 2; i
>= 0; i -= 2) {
            int digit =
Integer.parseInt(Character.toStri
ng(Long.toString(n).charAt(i)));
            int product = digit *
2;
            sum += product % 10;
            sum += product / 10;
        }
        // Add the sum to the sum
of the digits that weren't
multiplied by 2.
        for (int i =
Long.toString(n).length() - 1; i
>= 0; i -= 2) {
            sum +=
Integer.parseInt(Character.toStri
ng(Long.toString(n).charAt(i)));
        }
        // If the total's last
digit is 0 (or, put more
formally, if the total modulo 10
is congruent to 0),
        // the number is valid!
        if (sum % 10 == 0) {
            return true;
        } else {
            return false;
        }
    }
}
```

UNIT TESTS

Using Python’s built in unittest testing framework

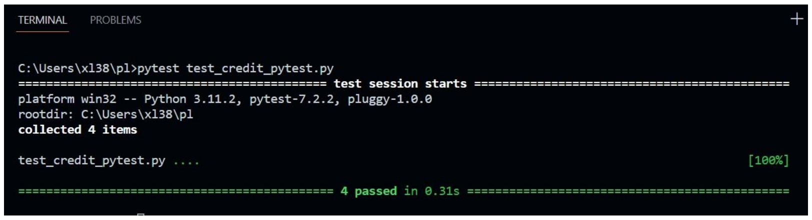
Status: 4 tests passed (all) ✅



```
TERMINAL    PROBLEMS

C:\Users\xl38\pl>python test_credit_unittest.py
....
----------------------------------------------------------------------
Ran 4 tests in 0.108s

OK
```

Using a Python library called pytest

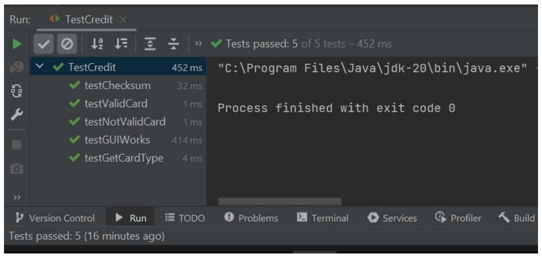
Status: 4 tests passed (all) ✅



```
TERMINAL    PROBLEMS

C:\Users\xl38\pl>pytest test_credit_pytest.py
========================================== test session starts ===========================================
platform win32 -- Python 3.11.2, pytest-7.2.2, pluggy-1.0.0
rootdir: C:\Users\xl38\pl
collected 4 items

test_credit_pytest.py ....                                                                          [100%]

=========================================== 4 passed in 0.31s ============================================
```

Using Java’s built in Junit testing framework

Status: 5 tests passed (all) ✅



```
Run:  TestCredit
Tests passed: 5 of 5 tests – 452 ms

TestCredit           452 ms    "C:\Program Files\Java\jdk-20\bin\java.exe"
  testChecksum        32 ms
  testValidCard        1 ms    Process finished with exit code 0
  testNotValidCard     1 ms
  testGUIWorks       414 ms
  testGetCardType      4 ms

Tests passed: 5 (16 minutes ago)
```

IV. NOTE:

To submit your documentation kindly use the filename:

GROUP NAME-FINAL PROJECT

(NOTE: The Leader of each group will be in charge in submitting your Final Project documentation)