

ASSIGNMENT-8.3

2303a51595

B-10

TASK-1 : Email Validation using TDD

Prompt :

Write a Java program to check whether an email is valid or not. The email should contain one @, at least one ., should not start or end with special characters, and should not contain multiple @ symbols. Display whether the email is valid or invalid.

Code :

```
import java.util.Scanner;
public class EmailValidation {
    public static boolean isValidEmail(String email) {
        // must contain exactly one @
        if (email.indexOf("@") == -1 ||
            email.indexOf("@") != email.lastIndexOf("@")) {
            return false;
        }
        // must contain .
        if (!email.contains(".")) {
            return false;
        }
        // must not start or end with special characters
        if (email.startsWith("@") || email.endsWith("@") ||
            email.startsWith(".") || email.endsWith(".")) {
            return false;
        }

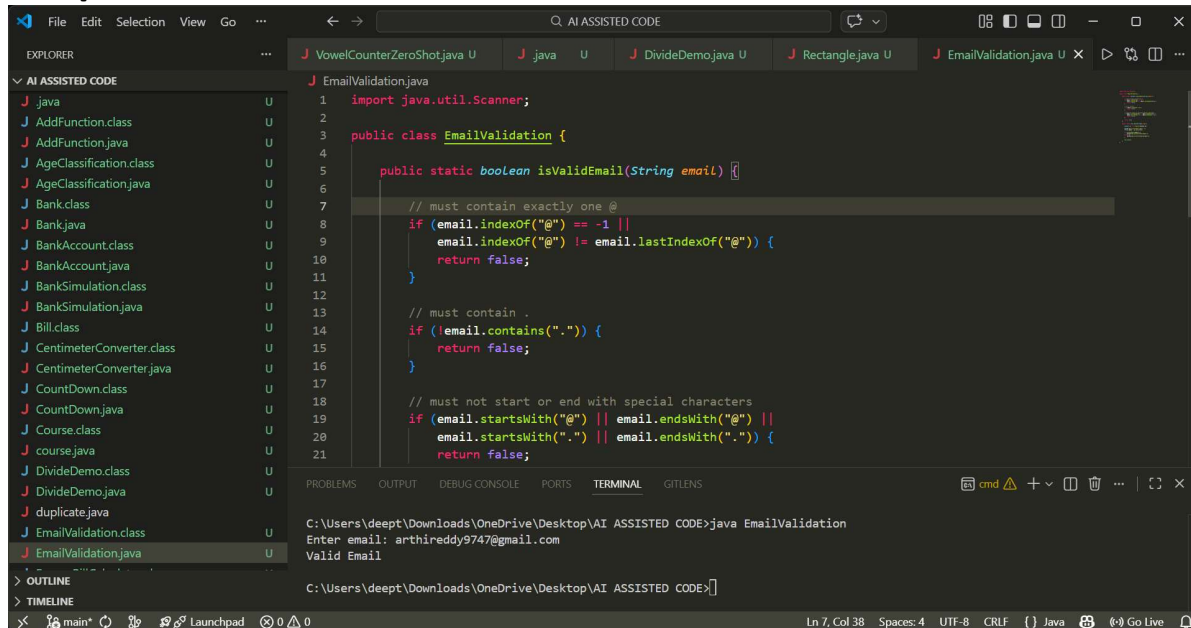
        return true;
    }
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter email: ");
        String email = sc.nextLine();

        if (isValidEmail(email)) {
            System.out.println("Valid Email");
        } else {
            System.out.println("Invalid Email");
        }
    }
}
```

Output :



The screenshot shows an IDE with the Explorer view on the left listing various Java files. The main editor displays the code for `EmailValidation.java`. The code imports `java.util.Scanner` and defines a `public class EmailValidation` with a `public static boolean isValidEmail(String email)` method. The method contains three validation rules: 1. Must contain exactly one '@' symbol. 2. Must contain a period '.'. 3. Must not start or end with special characters. The terminal at the bottom shows the command `java EmailValidation` being executed, with the input `arthireddy9747@gmail.com` and the output `Valid Email`.

```
1 import java.util.Scanner;
2
3 public class EmailValidation {
4
5     public static boolean isValidEmail(String email) {
6
7         // must contain exactly one @
8         if (email.indexOf("@") == -1 ||
9             email.indexOf("@") != email.lastIndexOf("@")) {
10             return false;
11         }
12
13         // must contain .
14         if (!email.contains(".")) {
15             return false;
16         }
17
18         // must not start or end with special characters
19         if (email.startsWith("@") || email.endsWith("@") ||
20             email.startsWith(".") || email.endsWith(".")) {
21             return false;
22         }
23     }
24 }
```

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL GIT LENS

C:\Users\deep\Downloads\OneDrive\Desktop\AI ASSISTED CODE>java EmailValidation
Enter email: arthireddy9747@gmail.com
Valid Email

C:\Users\deep\Downloads\OneDrive\Desktop\AI ASSISTED CODE>

Analysis :

The program takes email input from the user and checks validation rules. If all conditions are satisfied, it prints "Valid Email", otherwise "Invalid Email".

TASK-2 : Grade Assignment using Loops

Prompt :

Write a Java program to assign grades based on score using conditions. Handle boundary values (60, 70, 80, 90) correctly and show an error for invalid inputs like negative numbers, values above 100.

Code :

```
import java.util.Scanner;
public class GradeAssignment {
    public static String assignGrade(int score) {

        if (score < 0 || score > 100) {
            return "Invalid Input";
        }

        if (score >= 90)
            return "A";
        else if (score >= 80)
            return "B";
        else if (score >= 70)
```

```

        return "C";
    else if (score >= 60)
        return "D";
    else
        return "F";
}

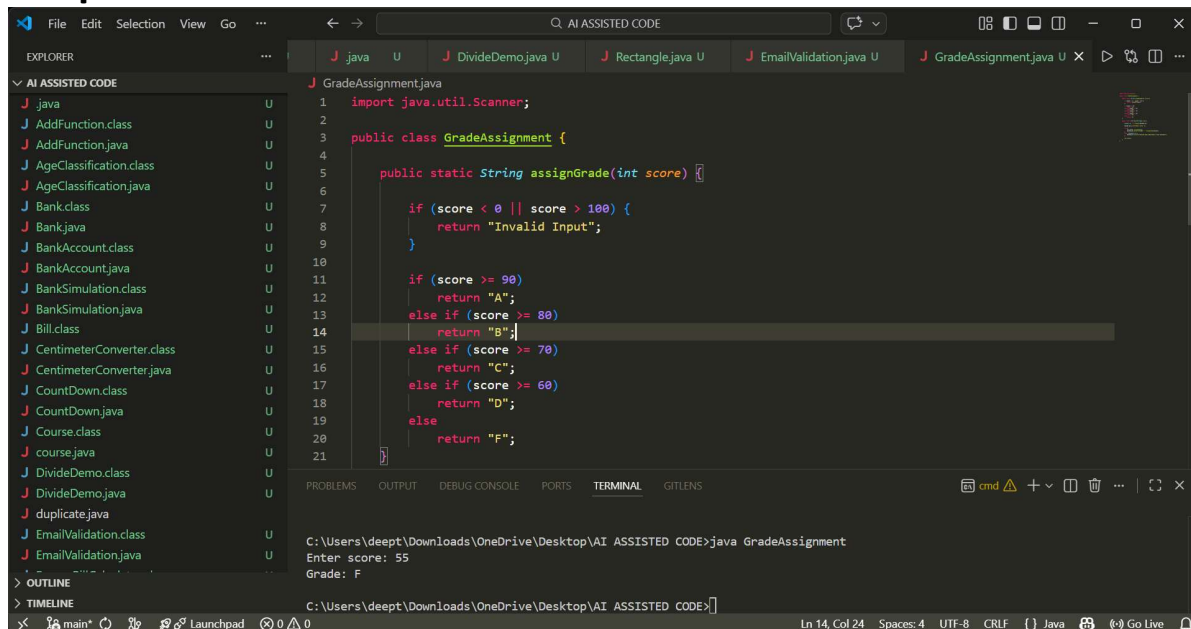
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter score: ");
    try {

        int score = sc.nextInt();
        System.out.println("Grade: " + assignGrade(score));
    }
    catch (Exception e) {
        System.out.println("Invalid Input (Non-numeric value entered)");
    }
    \

    sc.close();
}
}

```

Output :



The screenshot shows an IDE with the following components:

- EXPLORER:** A list of files on the left, including 'AI ASSISTED CODE' and various Java files like 'AddFunction.class', 'AgeClassification.class', etc.
- GradeAssignment.java:** The main file being edited, showing the following code:


```

1  import java.util.Scanner;
2
3  public class GradeAssignment {
4
5      public static String assignGrade(int score) {
6
7          if (score < 0 || score > 100) {
8              return "Invalid Input";
9          }
10
11          if (score >= 90)
12              return "A";
13          else if (score >= 80)
14              return "B";
15          else if (score >= 70)
16              return "C";
17          else if (score >= 60)
18              return "D";
19          else
20              return "F";
21      }
22  }

```
- TERMINAL:** The bottom panel showing the command prompt output:


```

C:\Users\deept\Downloads\OneDrive\Desktop\AI ASSISTED CODE>java GradeAssignment
Enter score: 55
Grade: F

```

Analysis :

The program checks if the score is within 0–100 and assigns grades using if-else conditions. Invalid numeric or non-numeric inputs are handled safely using range check and try-catch.

TASK-3 : Sentence Palindrome Checker

Prompt :

Write a Java program to check whether a given sentence is a palindrome or not. The program should ignore case, spaces, and punctuation marks and return true if the sentence is a palindrome, otherwise false.

Code :

```
import java.util.Scanner;
public class SentencePalindrome {
    public static boolean isSentencePalindrome(String sentence) {
        // remove spaces and punctuation, convert to lowercase
        String cleaned = sentence.replaceAll("[^a-zA-Z0-9]",
        "").toLowerCase();
        // reverse the cleaned string
        String reversed = new StringBuilder(cleaned).reverse().toString();

        return cleaned.equals(reversed);
    }

    public static void main(String[] args) {

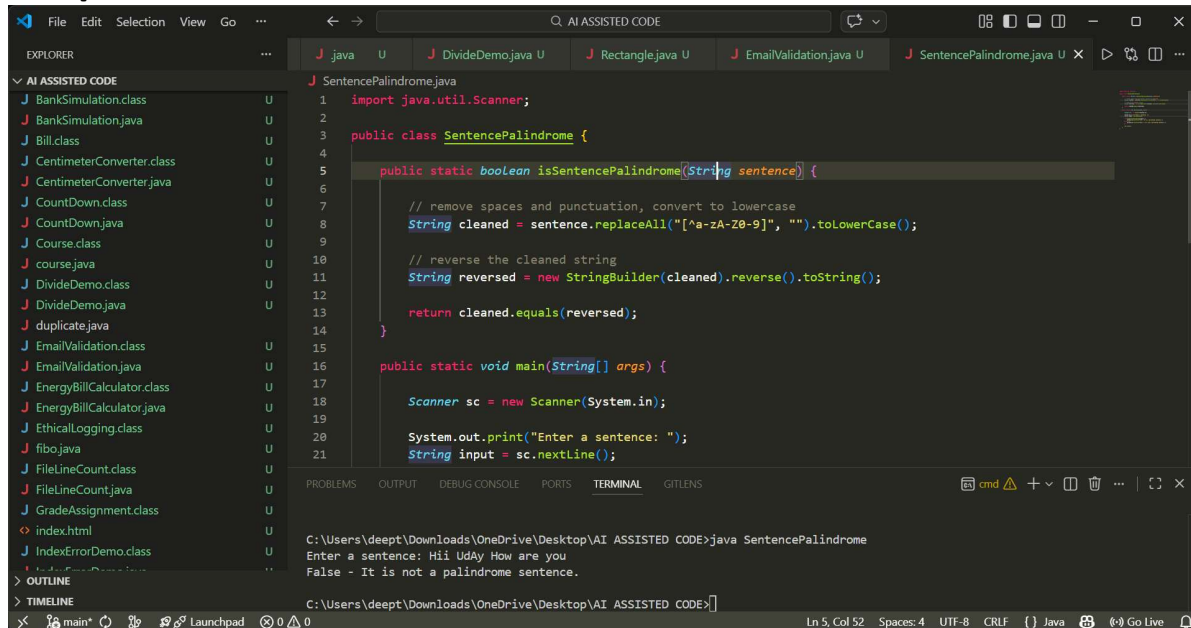
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a sentence: ");
        String input = sc.nextLine();

        if (isSentencePalindrome(input)) {
            System.out.println("True - It is a palindrome sentence.");
        } else {
            System.out.println("False - It is not a palindrome sentence.");
        }

        sc.close();
    }
}
```

Output :



```
File Edit Selection View Go ... Q AI ASSISTED CODE
EXPLORER
  AI ASSISTED CODE
    BankSimulation.class
    BankSimulation.java
    Bill.class
    CentimeterConverter.class
    CentimeterConverter.java
    Countdown.class
    Countdown.java
    Course.class
    course.java
    DivideDemo.class
    DivideDemo.java
    duplicate.java
    EmailValidation.class
    EmailValidation.java
    EnergyBillCalculator.class
    EnergyBillCalculator.java
    EthicalLogging.class
    fibo.java
    FileLineCount.class
    FileLineCount.java
    GradeAssignment.class
    index.html
    IndexErrorDemo.class
  OUTLINE
  TIMELINE
  main
  Launchpad
  0 0 0

SentencePalindrome.java
1 import java.util.Scanner;
2
3 public class SentencePalindrome {
4
5     public static boolean isSentencePalindrome(String sentence) {
6
7         // remove spaces and punctuation, convert to lowercase
8         String cleaned = sentence.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();
9
10        // reverse the cleaned string
11        String reversed = new StringBuilder(cleaned).reverse().toString();
12
13        return cleaned.equals(reversed);
14    }
15
16    public static void main(String[] args) {
17
18        Scanner sc = new Scanner(System.in);
19
20        System.out.print("Enter a sentence: ");
21        String input = sc.nextLine();
22    }
23 }

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL GITLENS
C:\Users\deep\Downloads\OneDrive\Desktop\AI ASSISTED CODE>java SentencePalindrome
Enter a sentence: Hii Uday How are you
False - It is not a palindrome sentence.

C:\Users\deep\Downloads\OneDrive\Desktop\AI ASSISTED CODE>
```

Analysis :

The program removes spaces and punctuation using `replaceAll()` and converts the sentence to lowercase. Then it compares the cleaned string with its reversed version to check if it is a palindrome.

TASK-4 : ShoppingCart Class

Prompt :

Write a Java program to create a `ShoppingCart` class with methods to add items, remove items, and calculate the total cost. The program should correctly update the cart, handle empty cart situations, and display the total bill accurately.

CODE :

```
import java.util.*;

class ShoppingCart {
    private HashMap<String, Double> items = new HashMap<>();
    // add item
    public void addItem(String name, double price) {
        items.put(name, price);
        System.out.println(name + " added to cart.");
    }
}
```

```

    }

    // remove item
    public void removeItem(String name) {
        if (items.containsKey(name)) {
            items.remove(name);
            System.out.println(name + " removed from cart.");
        } else {
            System.out.println("Item not found in cart.");
        }
    }

    // calculate total cost
    public double totalCost() {
        double total = 0;
        for (double price : items.values()) {
            total += price;
        }
        return total;
    }

    // display items
    public void displayCart() {
        if (items.isEmpty()) {
            System.out.println("Cart is empty.");
        } else {
            System.out.println("Items in cart:");
            for (String name : items.keySet()) {
                System.out.println(name + " - " + items.get(name));
            }
        }
    }
}

public class ShoppingCartDemo {
    public static void main(String[] args) {
        ShoppingCart cart = new ShoppingCart();
    }
}

```

```

// AI-generated test cases
cart.displayCart(); // empty cart
cart.addItem("Book", 500);
cart.addItem("Pen", 50);
cart.displayCart();
System.out.println("Total Cost: " + cart.totalCost());
cart.removeItem("Pen");
System.out.println("Total Cost after removal: " + cart.totalCost());
}
}

```

Output :

The screenshot shows an IDE with the following components:

- EXPLORER:** A list of files including `SentimentAnalysis.class`, `Server.class`, `ShoppingCart.class`, `ShoppingCartDemo.class`, `SimpleMLModel.class`, `Student.class`, `StudentTest.class`, `Sum.class`, `SumNumbers.class`, `UserInfoCollector.class`, and `VowelCounterZeroShot.class`.
- EDITOR:** The `ShoppingCartDemo.java` file is open, showing a `class ShoppingCart {` definition.
- TERMINAL:** The output of the program is displayed, showing a menu with 5 options: 1. Add Item, 2. Remove Item, 3. Display Cart, 4. Total Cost, and 5. Exit. The user enters choice 1, then item name 'pens' and price 30. The output shows 'pens added to cart.' and the total cost as 30.0.

Analysis :

The program uses Scanner to take user input and manage cart operations through a menu. Items are added, removed, and total cost is calculated using loops and HashMap.

TASK-5 : Date Format Conversion

Prompt :

Write a Java program to convert a date from YYYY-MM-DD format to DD-MM-YYYY format. If the input format is wrong, print "Invalid format". Test the program with some valid and invalid dates.

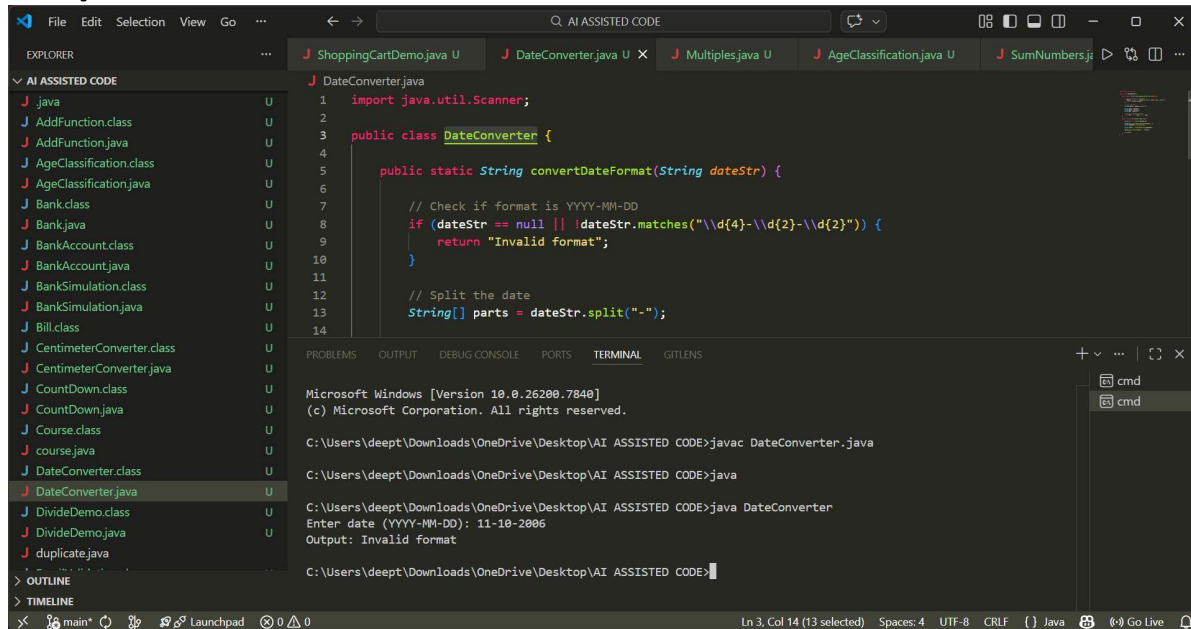
Code :

```
import java.util.Scanner;
public class DateConverter {
    public static String convertDateFormat(String dateStr) {
        // Check if format is YYYY-MM-DD
        if (dateStr == null || !dateStr.matches("\\d{4}-\\d{2}-\\d{2}")) {
            return "Invalid format";
        }
        // Split the date
        String[] parts = dateStr.split("-");
        String year = parts[0];
        String month = parts[1];
        String day = parts[2];
        // Return in DD-MM-YYYY format
        return day + "-" + month + "-" + year;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter date (YYYY-MM-DD): ");
        String inputDate = sc.nextLine();
        String result = convertDateFormat(inputDate);
        System.out.println("Output: " + result);

        String[] parts = dateStr.split("-");
        String year = parts[0];
        String month = parts[1];
        String day = parts[2];

        sc.close();
    }
}
```


Output :



The screenshot shows an IDE with the following components:

- EXPLORER:** A list of files on the left, including `DateConverter.java`, which is currently selected.
- EDITOR:** The main workspace showing the code for `DateConverter.java`. The code is as follows:

```
1 import java.util.Scanner;  
2  
3 public class DateConverter {  
4  
5     public static String convertDateFormat(String dateStr) {  
6  
7         // Check if format is YYYY-MM-DD  
8         if (dateStr == null || !dateStr.matches("\\d{4}-\\d{2}-\\d{2}")) {  
9             return "Invalid format";  
10        }  
11  
12        // Split the date  
13        String[] parts = dateStr.split("-");  
14    }  
}
```
- TERMINAL:** The bottom panel showing the execution of the program. The commands and output are:

```
C:\Users\deep\Downloads\OneDrive\Desktop\AI ASSISTED CODE> javac DateConverter.java  
C:\Users\deep\Downloads\OneDrive\Desktop\AI ASSISTED CODE> java  
C:\Users\deep\Downloads\OneDrive\Desktop\AI ASSISTED CODE> java DateConverter  
Enter date (YYYY-MM-DD): 11-10-2006  
Output: Invalid format  
C:\Users\deep\Downloads\OneDrive\Desktop\AI ASSISTED CODE>
```

Analysis :

It checks the format using regex and converts it to DD-MM-YYYY, otherwise prints "Invalid format".