ASSIGNMENT-8.3

2303a51595
B-10

TASK-1 : Email Validation using TDD

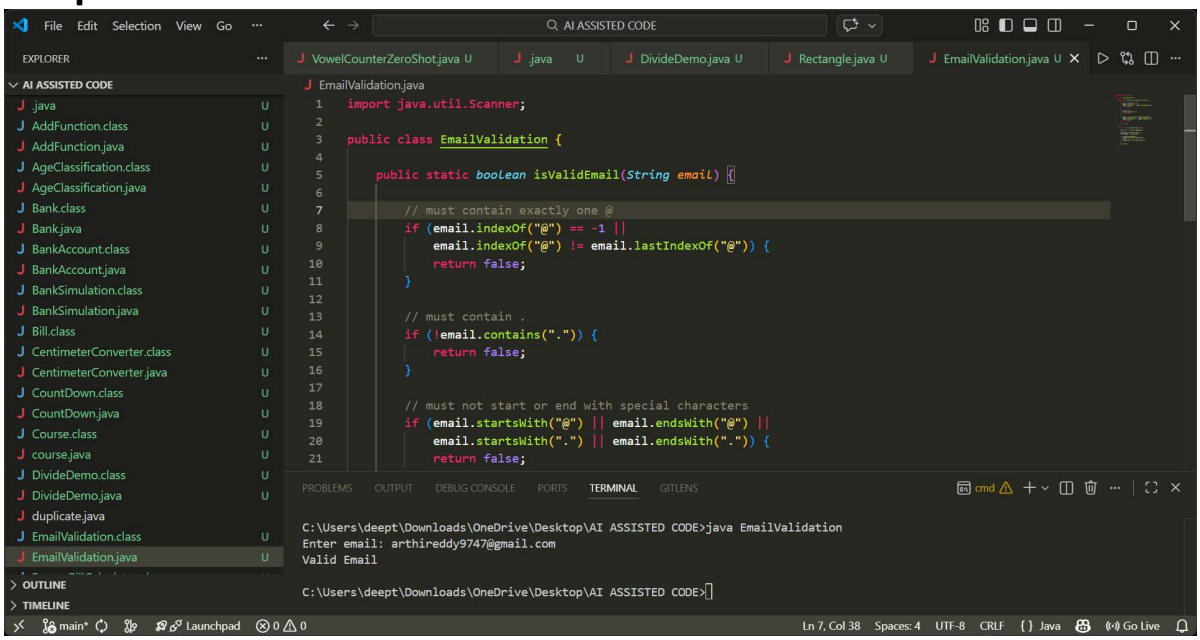
Prompt :

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 :



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 :

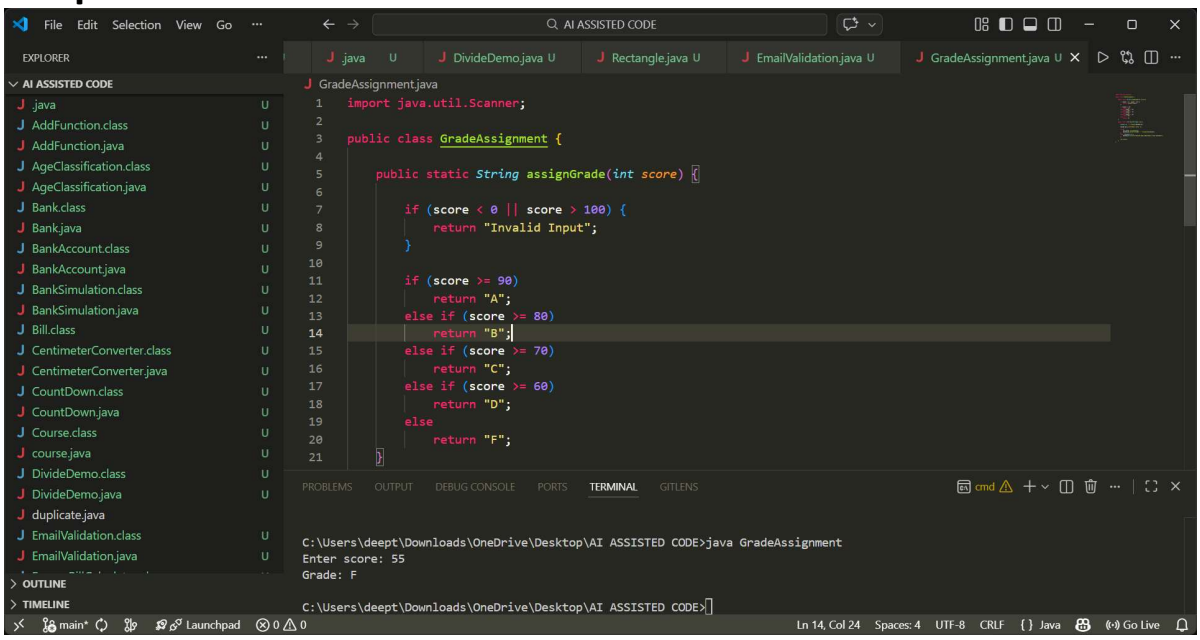
Write a Java program to assign grades based on score using conditions. Handle boundary values (60, 70, 80, 90) correctly and show an error for invalid inputs like negative numbers, values above 100.

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 :



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

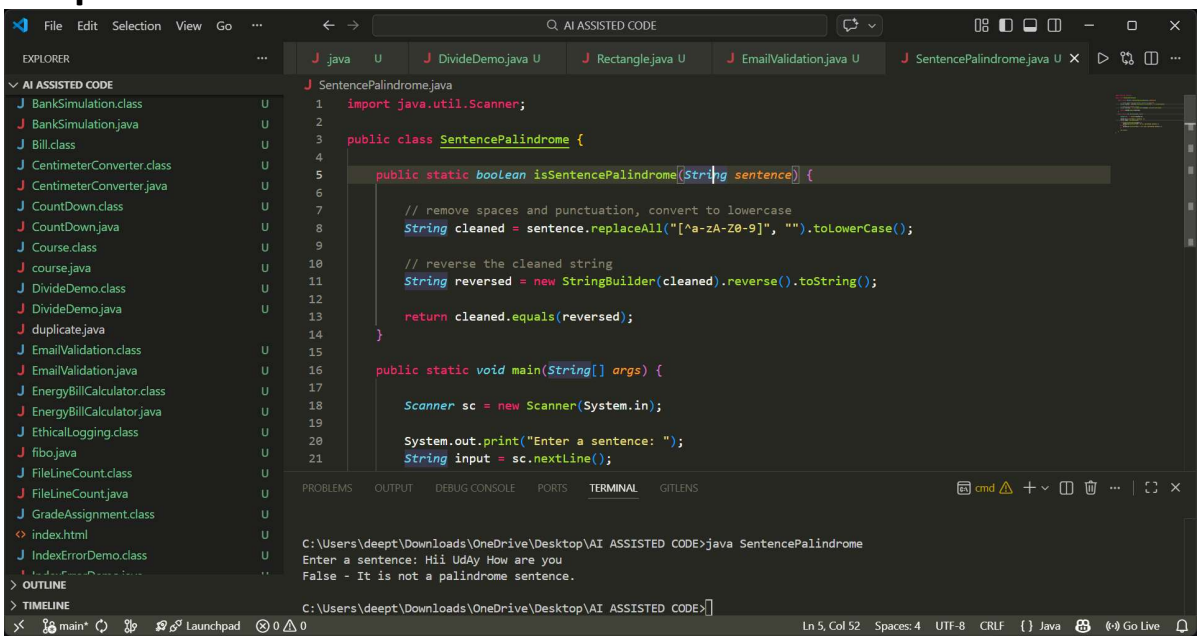
Prompt :

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 :



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.

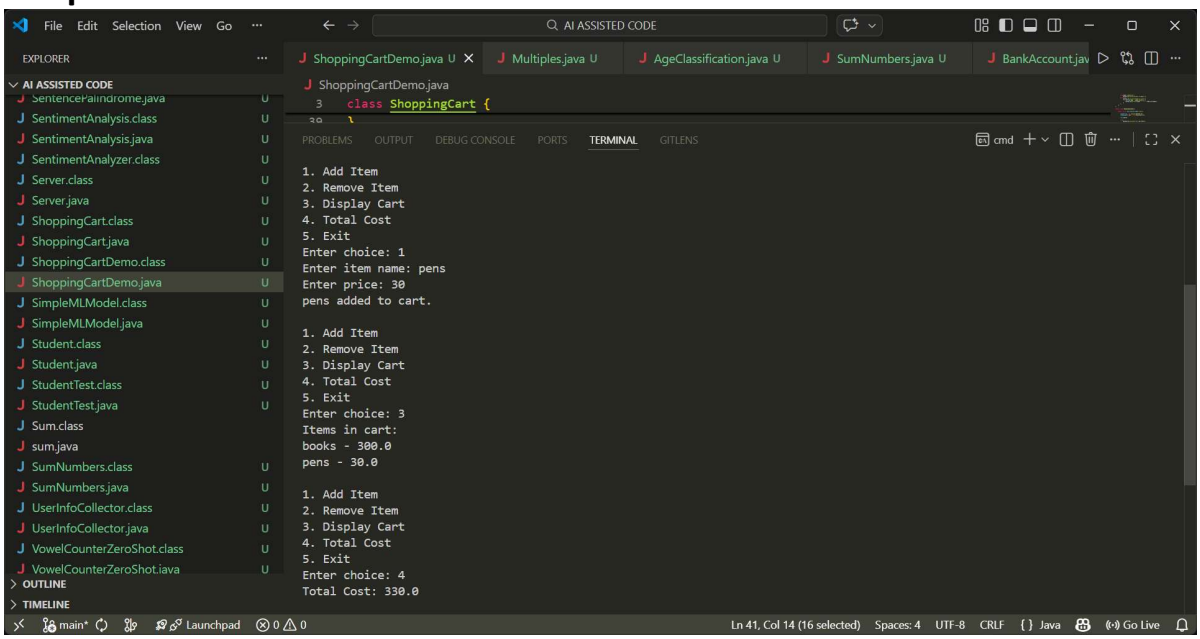
CODE :

```
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 :



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

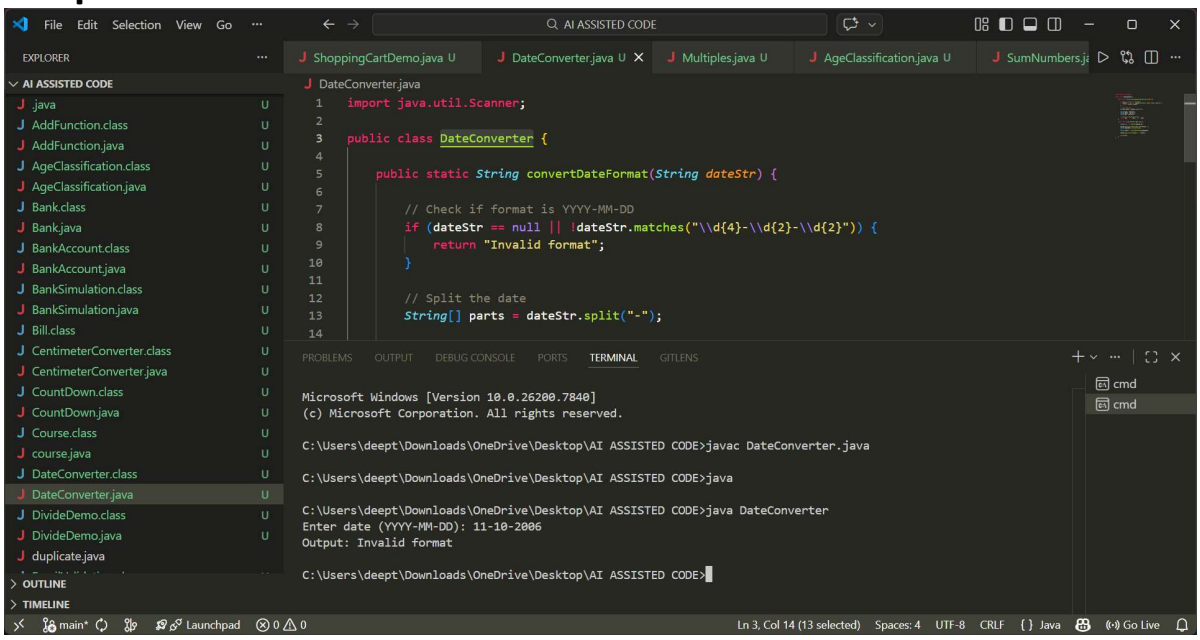
Prompt :

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 :



Analysis :

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