ASSIGNMENT – COURIER MANAGEMENT SYSTEM

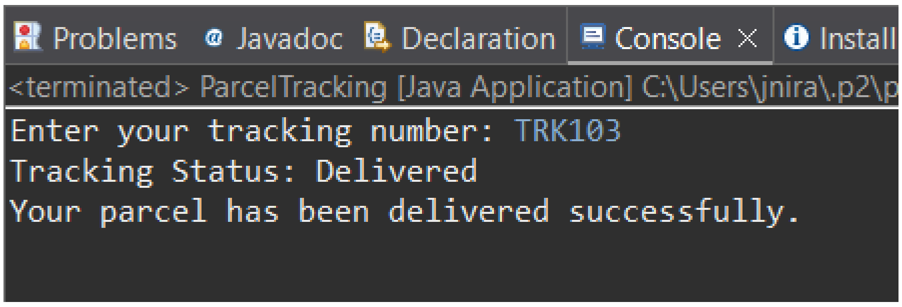
Coding

Task 4: Strings,2d Arrays, user defined functions,Hashmap

9. Parcel Tracking: Create a program that allows users to input a parcel tracking number.Store the tracking number and Status in 2d String Array. Initialize the array with values. Then, simulate the tracking process by displaying messages like "Parcel in transit," "Parcel out for delivery," or "Parcel delivered" based on the tracking number's status.

```
import java.util.*;

public class ParcelTracking {


    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);


    String[][] trackingData = {
        {"TRK101", "In Transit"},
        {"TRK102", "Out for Delivery"},
        {"TRK103", "Delivered"},
        {"TRK104", "In Transit"},
        {"TRK105", "Delivered"}
    };



    System.out.print("Enter your tracking number: ");
    String inputTracking = sc.nextLine();
    boolean found = false;

    for (int i = 0; i < trackingData.length; i++) {
        if (trackingData[i][0].equalsIgnoreCase(inputTracking)) {
            System.out.println("Tracking Status: " + trackingData[i][1]);


                switch (trackingData[i][1]) {
                case "In Transit":
                    System.out.println("Parcel is currently moving between
facilities.");
                    break;
                case "Out for Delivery":
                    System.out.println("Your parcel is on the way to your address.");
                    break;
                case "Delivered":
                    System.out.println("Your parcel has been delivered
successfully.");
                    break;
                default:
                    System.out.println("Status unknown.");
            }


            found = true;
            break;
        }
    }


    if (!found) {
        System.out.println("Tracking number not found.");
    }
```

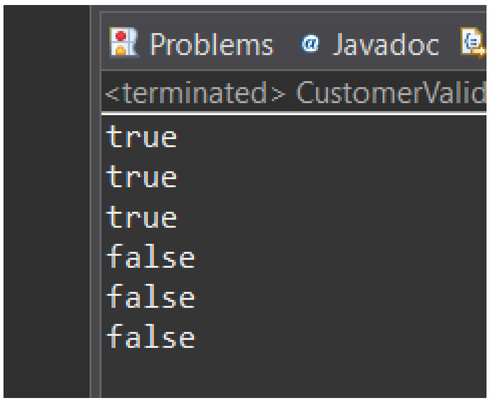
```
        sc.close();
          }
}
```



```
<terminated> ParcelTracking [Java Application] C:\Users\jnira\.p2\p
Enter your tracking number: TRK103
Tracking Status: Delivered
Your parcel has been delivered successfully.
```

10. Customer Data Validation: Write a function which takes 2 parameters, data-denotes the data and detail-denotes if it is name addtress or phone number.Validate customer information based on following critirea. Ensure that names contain only letters and are properly capitalized, addresses do not contain special characters, and phone numbers follow a specific format (e.g., ###-###-####).

```
public class CustomerValidator {


            public static boolean validateData(String data, String detail) {
                switch (detail.toLowerCase()) {
                    case "name":


                        return data.matches("[A-Z][a-zA-Z]*");


                    case "address":


                        return data.matches("[a-zA-Z0-9 ,.-]+");


                    case "phone":


                        return data.matches("[6-9][0-9]{9}");
                    default:
                        return false;
                }
            }

            public static void main(String[] args) {
                System.out.println(validateData("Ram", "name"));
                System.out.println(validateData("123 Main Street", "address"));
                System.out.println(validateData("9876543210", "phone"));
                System.out.println(validateData("raM@", "name"));
                System.out.println(validateData("Street#45", "address"));
                System.out.println(validateData("123456", "phone"));
            }

}
```



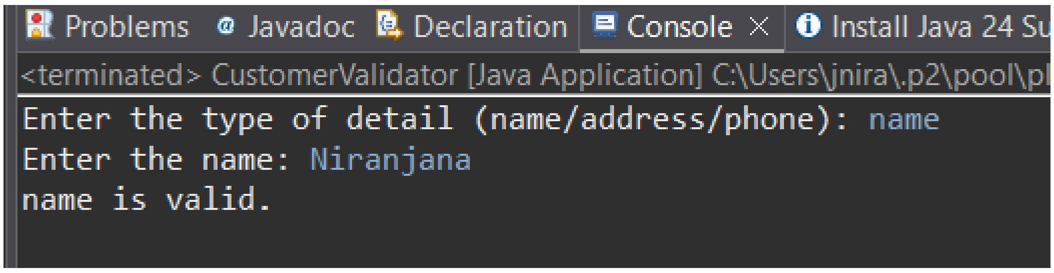
```
<terminated> CustomerValid
true
true
true
false
false
false
```

By getting custom input from user:

```
import java.util.*;

public class CustomerValidator {
```

```
public static boolean validateData(String data, String detail) {
      switch (detail.toLowerCase()) {
         case "name":

            return data.matches("[A-Z][a-zA-Z]*");

         case "address":

            return data.matches("[a-zA-Z0-9 ,.-]+");

         case "phone":

            return data.matches("[6-9][0-9]{9}");

         default:
            return false;
      }
  }

  public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);


      System.out.print("Enter the type of detail (name/address/phone): ");
      String detail = sc.nextLine();

      System.out.print("Enter the " + detail + ": ");
```

```
        String data = sc.nextLine();

        if (validateData(data, detail)) {
            System.out.println(detail + " is valid.");
        } else {
            System.out.println(detail + " is invalid.");
        }

        sc.close();

    }
}
```

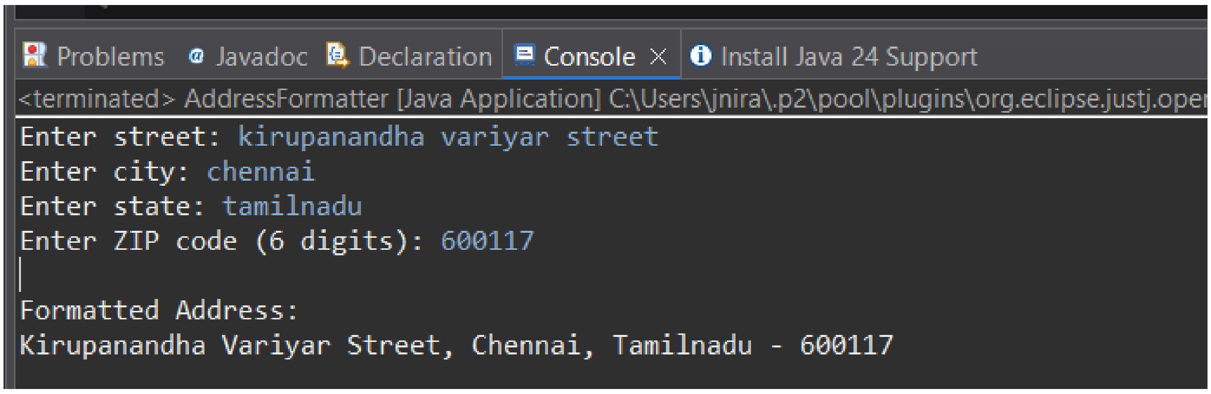


```
<terminated> CustomerValidator [Java Application] C:\Users\jnira\.p2\pool\pl
Enter the type of detail (name/address/phone): name
Enter the name: Niranjana
name is valid.
```

11. Address Formatting: Develop a function that takes an address as input (street, city, state, zip code) and formats it correctly, including capitalizing the first letter of each word and properly formatting the zip code.

```
import java.util.*;
public class AddressFormatter {

    public static String capitalizeWords(String input) {
        String[] words = input.trim().toLowerCase().split(" ");
        String formatted = "";
        for (String word : words) {
            if (!word.isEmpty()) {
                formatted += Character.toUpperCase(word.charAt(0)) +
word.substring(1) + " ";
            }
        }

        return formatted.trim();
    }
    public static String formatAddress(String street, String city, String
state, String zipCode) {
        if (!zipCode.matches("\\d{6}")) {
            return "Invalid ZIP Code. It must be 6 digits.";
        }
        street = capitalizeWords(street);
        city = capitalizeWords(city);
        state = capitalizeWords(state);
        return street + ", " + city + ", " + state + " - " + zipCode;
    }
  public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
  System.out.print("Enter street: ");
  String street = sc.nextLine();
  System.out.print("Enter city: ");
  String city = sc.nextLine();
  System.out.print("Enter state: ");
  String state = sc.nextLine();
  System.out.print("Enter ZIP code (6 digits): ");
  String zipCode = sc.nextLine();
  String result = formatAddress(street, city, state, zipCode);
```

```
        System.out.println("\nFormatted Address:");
        System.out.println(result);
        sc.close();

    }
}
```



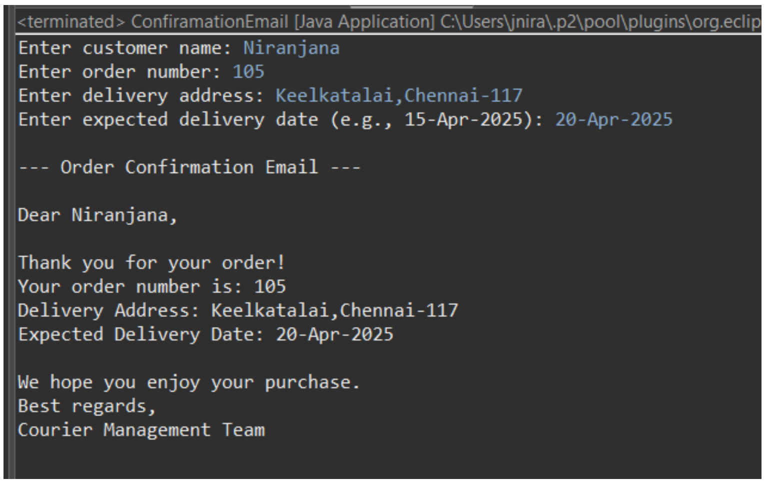
Problems Javadoc Declaration Console Install Java 24 Support

```
<terminated> AddressFormatter [Java Application] C:\Users\jnira\.p2\pool\plugins\org.eclipse.justj.oper
Enter street: kirupanandha variyar street
Enter city: chennai
Enter state: tamilnadu
Enter ZIP code (6 digits): 600117

Formatted Address:
Kirupanandha Variyar Street, Chennai, Tamilnadu - 600117
```

12. Order Confirmation Email: Create a program that generates an order confirmation email. The email should include details such as the customer's name, order number, delivery address, and expected delivery date.

```
 import java.util.*;
public class ConfiramationEmail {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Taking inputs
        System.out.print("Enter customer name: ");
        String customerName = sc.nextLine();

        System.out.print("Enter order number: ");
        String orderNumber = sc.nextLine();
```

```
                System.out.print("Enter delivery address: ");
                String deliveryAddress = sc.nextLine();


                System.out.print("Enter expected delivery date (e.g., 15-Apr-2025):
");
                String deliveryDate = sc.nextLine();


                String emailMessage = "\nDear " + customerName + ",\n\n"
                        + "Thank you for your order!\n"
                        + "Your order number is: " + orderNumber + "\n"
                        + "Delivery Address: " + deliveryAddress + "\n"
                        + "Expected Delivery Date: " + deliveryDate + "\n\n"
                        + "We hope you enjoy your purchase.\n"
                        + "Best regards,\n"
                        + "Courier Management Team";
                System.out.println("\n--- Order Confirmation Email ---");
                System.out.println(emailMessage);
                sc.close();
        }
}
```

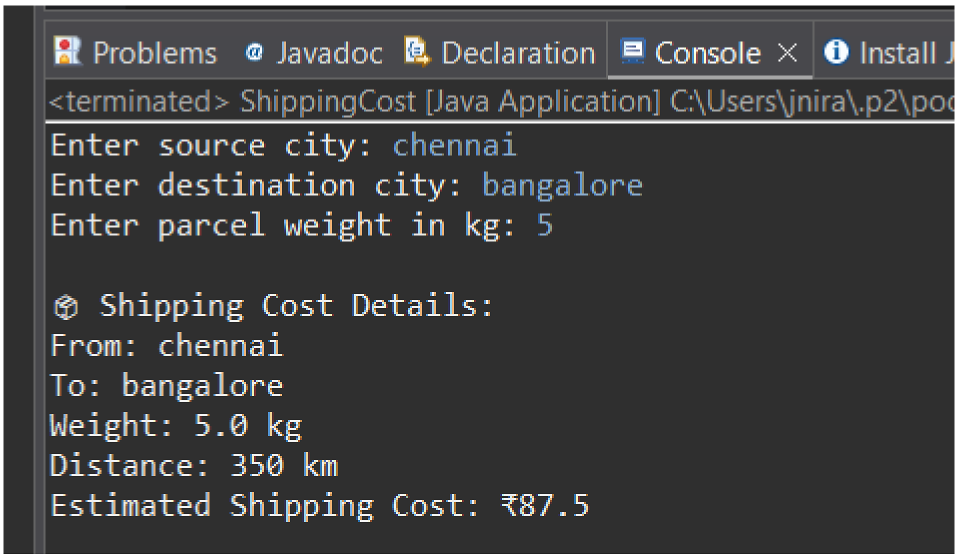


```
<terminated> ConfiramationEmail [Java Application] C:\Users\jnira\.p2\pool\plugins\org.eclip
Enter customer name: Niranjana
Enter order number: 105
Enter delivery address: Keelkatalai,Chennai-117
Enter expected delivery date (e.g., 15-Apr-2025): 20-Apr-2025

--- Order Confirmation Email ---

Dear Niranjana,

Thank you for your order!
Your order number is: 105
Delivery Address: Keelkatalai,Chennai-117
Expected Delivery Date: 20-Apr-2025

We hope you enjoy your purchase.
Best regards,
Courier Management Team
```

13. Calculate Shipping Costs: Develop a function that calculates the shipping cost based on the distance between two locations and the weight of the parcel. You can use string inputs for the source and destination addresses

```
import java.util.*;

public class ShippingCost {


     public static int getDistance(String source, String destination) {
    source = source.toLowerCase();
    destination = destination.toLowerCase();


    if (source.equals(destination)) {
      return 0;
    }


    if ((source.equals("chennai") && destination.equals("bangalore")) ||
      (source.equals("bangalore") && destination.equals("chennai"))) {
      return 350;
    } else if ((source.equals("chennai") && destination.equals("mumbai")) ||
          (source.equals("mumbai") && destination.equals("chennai"))) {
      return 1330;
    } else if ((source.equals("bangalore") && destination.equals("mumbai")) ||
          (source.equals("mumbai") && destination.equals("bangalore"))) {
      return 980;
    } else {
      return -1; // indicates unknown route
    }
  }
```

```
    // Function to calculate cost
    public static double calculateShippingCost(int distance, double weight) {
        double costPer100KmPerKg = 5.0;
        double cost = (distance / 100.0) * costPer100KmPerKg * weight;
        return Math.round(cost * 100.0) / 100.0;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

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

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

        System.out.print("Enter parcel weight in kg: ");
        double weight = sc.nextDouble();

        int distance = getDistance(source, destination);

        if (distance == -1) {
            System.out.println("\n Service not available between " + source + " and
" + destination + ".");
        } else {
            double cost = calculateShippingCost(distance, weight);
            System.out.println("\n Shipping Cost Details:");
            System.out.println("From: " + source);
            System.out.println("To: " + destination);
            System.out.println("Weight: " + weight + " kg");
            System.out.println("Distance: " + distance + " km");
            System.out.println("Estimated Shipping Cost: ₹" + cost);
        }
        sc.close();
         }
}
```



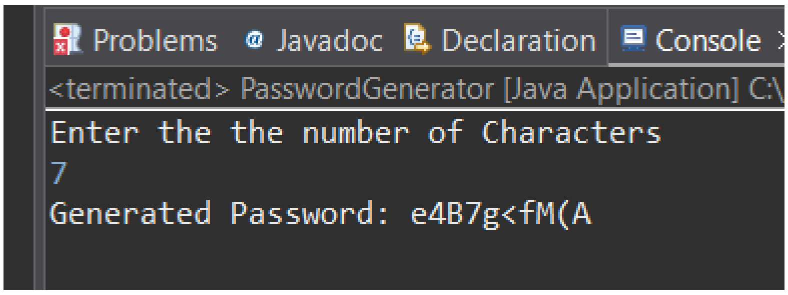
```
<terminated> ShippingCost [Java Application] C:\Users\jnira\.p2\poo
Enter source city: chennai
Enter destination city: bangalore
Enter parcel weight in kg: 5

📦 Shipping Cost Details:
From: chennai
To: bangalore
Weight: 5.0 kg
Distance: 350 km
Estimated Shipping Cost: ₹87.5
```

14. Password Generator: Create a function that generates secure passwords for courier system accounts. Ensure the passwords contain a mix of uppercase letters, lowercase letters, numbers, and special characters.

```
import java.util.*;


public class PasswordGenerator {


          public static String generatePassword(int length) {
         if (length < 4) {
```

```
    return "Password length should be at least 4.";
}

String upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
String lower = "abcdefghijklmnopqrstuvwxyz";
String digits = "0123456789";
String special = "!@#$%^&*()-_+=<>?";

String all = upper + lower + digits + special;
Random rand = new Random();
char[] password = new char[length];

// Ensure 1 character from each required type
password[0] = upper.charAt(rand.nextInt(upper.length()));
password[1] = lower.charAt(rand.nextInt(lower.length()));
password[2] = digits.charAt(rand.nextInt(digits.length()));
password[3] = special.charAt(rand.nextInt(special.length()));

// Fill the rest randomly
for (int i = 4; i < length; i++) {
    password[i] = all.charAt(rand.nextInt(all.length()));
}

// Shuffle the password so guaranteed characters are not in fixed positions
for (int i = 0; i < password.length; i++) {
    int randomIndex = rand.nextInt(password.length);
    char temp = password[i];
```

```
                password[i] = password[randomIndex];
                password[randomIndex] = temp;
            }
            return new String(password);
        }
        public static void main(String[] args) {
            // Generate and print a 10-character secure password
              Scanner sc=new Scanner(System.in);
              System.out.println("Enter the the number of Characters");
            String password = sc.nextLine();
            String password1 = generatePassword(10);
            System.out.println("Generated Password: " + password1);
    }
}
```

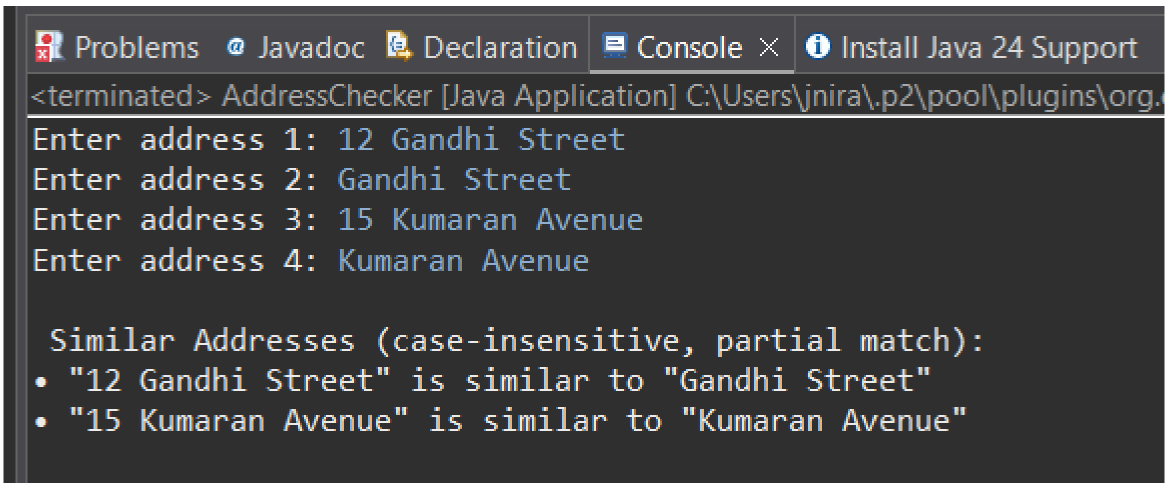

Problems Javadoc Declaration Console

```
<terminated> PasswordGenerator [Java Application] C:\
Enter the the number of Characters
7
Generated Password: e4B7g<fM(A
```

15. Find Similar Addresses: Implement a function that finds similar addresses in the system. This can be useful for identifying duplicate customer entries or optimizing delivery routes. Use string functions to implement this.

```
import java.util.*;


public class AddressChecker {


        public static void findSimilarAddresses(String[] addresses) {
```

```
        System.out.println("\n Similar Addresses (case-insensitive, partial
match):");
        boolean found = false;

        for (int i = 0; i < addresses.length; i++) {
            for (int j = i + 1; j < addresses.length; j++) {
                // Convert both to lowercase and remove spaces for comparison
                String addr1 = addresses[i].toLowerCase().replaceAll("\\s+", "");
                String addr2 = addresses[j].toLowerCase().replaceAll("\\s+", "");

                // Check if one address contains the other (partial match)
                if (addr1.contains(addr2) || addr2.contains(addr1)) {
                    System.out.println("• \"" + addresses[i] + "\" is similar to \"" +
addresses[j] + "\"");
                    found = true;
                }
            }
        }

        if (!found) {
            System.out.println("No similar addresses found.");
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of addresses: ");
        int n = sc.nextInt();
        sc.nextLine(); // consume leftover newline

        String[] addresses = new String[n];

        for (int i = 0; i < n; i++) {
            System.out.print("Enter address " + (i + 1) + ": ");
            addresses[i] = sc.nextLine();
        }

        findSimilarAddresses(addresses);

        sc.close();


    }

}
```



```
<terminated> AddressChecker [Java Application] C:\Users\jnira\.p2\pool\plugins\org.
Enter address 1: 12 Gandhi Street
Enter address 2: Gandhi Street
Enter address 3: 15 Kumaran Avenue
Enter address 4: Kumaran Avenue

 Similar Addresses (case-insensitive, partial match):
• "12 Gandhi Street" is similar to "Gandhi Street"
• "15 Kumaran Avenue" is similar to "Kumaran Avenue"
```