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.

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., ###-###-####).

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.

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.

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.

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.

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.

In this task, I will work on a series of functionalities for a courier system. The goal is to use various concepts in Java such as strings, 2D arrays, user-defined functions, and HashMap to implement useful features like parcel tracking, customer data validation, address formatting, order confirmation emails, shipping cost calculation, password generation, and finding similar addresses. We will create a user-friendly console-based system where users can interact with different options, input data, and receive processed results.

The following tasks are covered:

Task 4.1: Parcel Tracking

Task 4.2: Customer Data Validation

Task 4.3 Address Formatting

Task 4.4: Order Confirmation Email

Task 4.5: Calculate Shipping costs

Task 4.6: Password Generator

Task 4.7: Find Similar Addresses

Main Method (psvm)

In Task 4, the public static void main(String[] args) (PSVM) method acts as the entry point of the program. It initializes the program, providing a menu-driven interface for the user to select different tasks such as parcel tracking, customer data validation, address formatting, and more. The main method uses a while(true) loop to repeatedly display the menu options until the user chooses to exit. Based on the user's input, it calls the respective methods to handle specific functionalities. The scanner object is used to capture user inputs throughout the program, allowing interaction with the system. Each case within the switch statement corresponds to one of the functionalities defined in the program, invoking the relevant method when a specific option is selected. This structure ensures the user can interact with the various tasks smoothly and choose what they want to do.

The PSVM method essentially acts as the control center for the task-based program.

```
package main;

import java.util.Random;
import java.util.Scanner;

public class Task4 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\nTask 4 - Choose an option:");
            System.out.println("1. Parcel Tracking");
            System.out.println("2. Customer Data Validation");
            System.out.println("3. Address Formatting");
            System.out.println("4. Order Confirmation Email");
            System.out.println("5. Calculate Shipping Costs");
            System.out.println("6. Generate Password");
            System.out.println("7. Find Similar Addresses");
            System.out.println("8. Exit");
            System.out.print("Enter your choice: ");
```

```
        int choice = scanner.nextInt();
        scanner.nextLine();

        switch (choice) {
            case 1:
                parcelTracking(scanner);
                break;
            case 2:
                System.out.print("Enter name: ");
                String name = scanner.nextLine();
                customerDataValidation(name, type: "name");

                System.out.print("Enter phone number: ");
                String phone = scanner.nextLine();
                customerDataValidation(phone, type: "phone");

                System.out.print("Enter address: ");
                String address = scanner.nextLine();
                customerDataValidation(address, type: "address");
                break;
            case 3:
                addressFormatting(scanner);
                break;
            case 4:
                orderConfirmationEmail(scanner);
                break;
            case 5:
                calculateShippingCost(scanner);
                break;
            case 6:
                System.out.print("Enter the desired password length: ");
                int length = scanner.nextInt();

                String password = generateRandomPassword(length);
                System.out.println("Generated password: " + password);
                break;
            case 7:
                findSimilarAddresses(scanner);
                break;
            case 8:
                System.out.println("Exiting Task 4...");
                scanner.close();
                return;
            default:
                System.out.println("Invalid choice! Please select a valid option.");
        }
    }
}
```

Task 4.1: Parcel Tracking

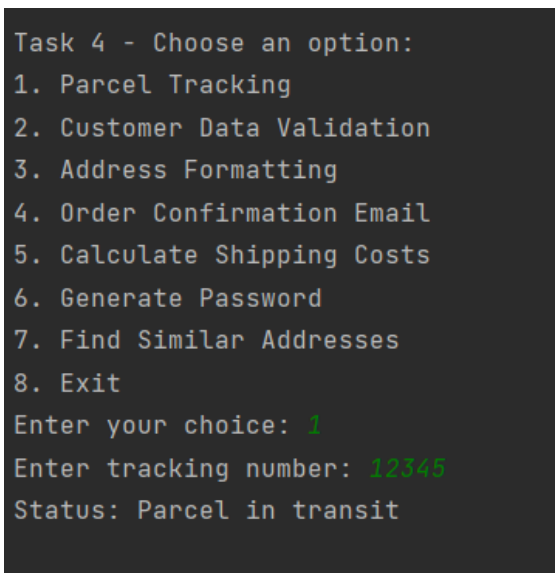
Create a program that allows users to input a parcel tracking number.Store the tracking number and Status in 2d String Array. Initialize the array with values. Then, simulate the tracking process by displaying messages like "Parcel in transit," "Parcel out for delivery," or "Parcel delivered" based on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

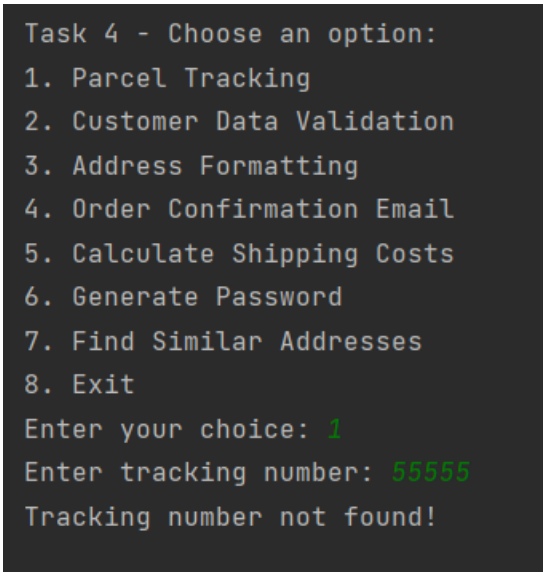
In this task, we will implement a parcel tracking system where users can input a tracking number, and the system will return the status of the parcel. We will use a 2D array to store tracking numbers and their corresponding statuses. The user will be prompted to input a tracking number, and the system will check if it exists in the array. Based on the tracking number, the parcel status will be displayed.

Code :

```
// 1. Parcel Tracking
public static void parcelTracking(Scanner scanner) {
    String[][] trackingData = {
            {"12345", "Parcel in transit"},
            {"67890", "Parcel out for delivery"},
            {"54321", "Parcel delivered"},
            {"98765", "Parcel in transit"},
            {"11111", "Parcel shipped"},
            {"22222", "Parcel processing"},
            {"33333", "Parcel at sorting facility"},
            {"44444", "Parcel pending pickup"}
    };

    System.out.print("Enter tracking number: ");
    String trackingNumber = scanner.nextLine();

    boolean found = false;
    for (String[] parcel : trackingData) {
        if (parcel[0].equals(trackingNumber)) {
            System.out.println("Status: " + parcel[1]);
            found = true;
            break;
        }
    }

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

Output :





Task 4.2: 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., ###-###-####).

In this task, we will implement a function to validate customer data like name, phone number, and address. The system will check if the data meets specific formats: names should contain only letters, addresses should not contain special characters, and phone numbers should match a specific format (###-###-####). If the data is valid, a confirmation message will be displayed.

Code :

```java
// 2. Customer Data Validation
public static void customerDataValidation(String data, String type) {
    switch (type.toLowerCase()) {
        case "name":
            if (data.matches( regex: "[A-Za-z ]+")) {
                System.out.println("Valid name format.");
            } else {
                System.out.println("Invalid name! Name should contain only letters.");
            }
            break;
        case "address":
            if (data.matches( regex: "[A-Za-z0-9 ,.-]+")) {
                System.out.println("Valid address format.");
            } else {
                System.out.println("Invalid address! Address should not contain special characters.");
            }
            break;
```

```
            break;
        case "phone":
            if (data.matches( regex: "\\d{3}-\\d{3}-\\d{4}")) {
                System.out.println("Valid phone number format.");
            } else {
                System.out.println("Invalid phone number! Format should be ###-###-####.");
            }
            break;
        default:
            System.out.println("Invalid detail type!");
    }
}
```

Output :

```
Task 4 - Choose an option:
1. Parcel Tracking
2. Customer Data Validation
3. Address Formatting
4. Order Confirmation Email
5. Calculate Shipping Costs
6. Generate Password
7. Find Similar Addresses
8. Exit
Enter your choice: 2
Enter name: Reshmika
Valid name format.
Enter phone number: 123-456-7890
Valid phone number format.
Enter address: 29 , Prasanna Colony , 9th Street , Madurai
Valid address format.
```

Task 4.3 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.

In this task, we will develop a function that takes an address and formats it correctly. The function will capitalize the first letter of each word in the address and ensure the zip code is correctly formatted. It will handle the input for street, city, state, and zip code.

Code :

```java
// 3. Address Formatting
public static void addressFormatting(Scanner scanner) {
    System.out.print("Enter Street: ");
    String street = scanner.nextLine();
    System.out.print("Enter City: ");
    String city = scanner.nextLine();
    System.out.print("Enter State: ");
    String state = scanner.nextLine();
    System.out.print("Enter Zip Code: ");
    String zip = scanner.nextLine();

    if (!zip.matches( regex: "\\d{6}(-\\d{4})?")) {
        System.out.println("Invalid Zip Code! Format should be ##### or #####-####.");
        return;
    }

    street = capitalizeWords(street);
    city = capitalizeWords(city);
    state = capitalizeWords(state);

    System.out.println("Formatted Address: " + street + ", " + city + ", " + state + " - " + zip);
}
```

```java
private static String capitalizeWords(String str) {
    String[] words = str.split( regex: " ");
    StringBuilder capitalizedStr = new StringBuilder();
    for (String word : words) {
        if (!word.isEmpty()) {
            capitalizedStr.append(Character.toUpperCase(word.charAt(0)))
                    .append(word.substring( beginIndex: 1).toLowerCase())
                    .append(" ");
        }
    }
    return capitalizedStr.toString().trim();
}
```

Output :

```
Task 4 - Choose an option:
1. Parcel Tracking
2. Customer Data Validation
3. Address Formatting
4. Order Confirmation Email
5. Calculate Shipping Costs
6. Generate Password
7. Find Similar Addresses
8. Exit
Enter your choice: 3
Enter Street: 29,Prasanna Colony , 9th Street
Enter City: Madurai
Enter State: Tamil Ndu
Enter Zip Code: 625012
Formatted Address: 29,prasanna Colony , 9th Street, Madurai, Tamil Ndu - 625012
```

Task 4.4: 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.

In this task, we will generate an order confirmation email for a customer. The system will take input such as customer name, order number, delivery address, and expected delivery date, and generate a personalized email containing the order details.

Code :

```java
// 4. Order Confirmation Email
public static void orderConfirmationEmail(Scanner scanner) {

    System.out.print("Enter customer name: ");
    String name = scanner.nextLine();

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

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

    System.out.print("Enter expected delivery date (YYYY-MM-DD): ");
    String deliveryDate = scanner.nextLine();

    String emailContent = generateOrderConfirmationEmail(name, orderNumber, deliveryAddress, deliveryDate);

    System.out.println("\n--- Order Confirmation Email ---");
    System.out.println(emailContent);
}
```

```java
public static String generateOrderConfirmationEmail(String name, String orderNumber, String deliveryAddress, String deliveryDate) {
    return "Dear " + name + ",\n\n" +
            "Thank you for your order! Your order details are as follows:\n" +
            "Order Number: " + orderNumber + "\n" +
            "Delivery Address: " + deliveryAddress + "\n" +
            "Expected Delivery Date: " + deliveryDate + "\n\n" +
            "We appreciate your business and will notify you once your parcel is shipped.\n" +
            "\nBest regards,\n" +
            "The Courier Team";
}
```

Output :

```
Task 4 - Choose an option:
1. Parcel Tracking
2. Customer Data Validation
3. Address Formatting
4. Order Confirmation Email
5. Calculate Shipping Costs
6. Generate Password
7. Find Similar Addresses
8. Exit
Enter your choice: 4
Enter customer name: Reshmika
Enter order number: 123
Enter delivery address: KK nagar , Chennai
Enter expected delivery date (YYYY-MM-DD): 2025-04-05
```

```
--- Order Confirmation Email ---
Dear Reshmika,

Thank you for your order! Your order details are as follows:
Order Number: 123
Delivery Address: KK nagar , Chennai
Expected Delivery Date: 2025-04-05

We appreciate your business and will notify you once your parcel is shipped.

Best regards,
The Courier Team
```

Task 4.5: 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.

In this task, we will calculate the shipping cost based on the distance between two locations and the weight of the parcel. The user will input the source and destination addresses, as well as the weight of the parcel. The cost will be calculated using the distance (calculated by the difference in address length) and the weight.

Code :

```java
// 5. Calculate Shipping Costs
public static void calculateShippingCost(Scanner scanner) {

    System.out.print("Enter source location: ");
    String source = scanner.nextLine();
    System.out.print("Enter destination location: ");
    String destination = scanner.nextLine();
    System.out.print("Enter the weight of the parcel (in kg): ");

    double weight = scanner.nextDouble();
    double costPerKm = 2.0;
    double costPerKg = 5.0;

    double distance = calculateDistance(source, destination);

    double shippingCost = (costPerKm * distance) + (costPerKg * weight);

    System.out.println("Shipping Cost from " + source + " to " + destination + " is: Rs. " + shippingCost);
}
```

```java
public static double calculateDistance(String source, String destination) {
    return Math.abs(source.length() - destination.length()) * 10;
}
```

Output :

```
Task 4 - Choose an option:
1. Parcel Tracking
2. Customer Data Validation
3. Address Formatting
4. Order Confirmation Email
5. Calculate Shipping Costs
6. Generate Password
7. Find Similar Addresses
8. Exit
Enter your choice: 5
Enter source location: KK Nagar
Enter destination location: Nehru Street
Enter the weight of the parcel (in kg): 8
Shipping Cost from KK Nagar to Nehru Street is: Rs. 120.0
```

Task 4.6: 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.

In this task, we will create a secure password generator for courier system accounts. The generated passwords will include a mix of uppercase letters, lowercase letters, digits, and special characters. The user will be prompted to specify the desired password length.

Code :

```java
//6. Generate Password
public static String generateRandomPassword(int length) {
    String upperCaseLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    String lowerCaseLetters = "abcdefghijklmnopqrstuvwxyz";
    String digits = "0123456789";
    String specialCharacters = "!@#$%^&*()-_+=<>?/[]{}|";

    String allCharacters = upperCaseLetters + lowerCaseLetters + digits + specialCharacters;
    Random random = new Random();

    StringBuilder password = new StringBuilder();

    for (int i = 0; i < length; i++) {
        int index = random.nextInt(allCharacters.length());
        password.append(allCharacters.charAt(index));
    }

    return password.toString();
}
```

Output :

```
Task 4 - Choose an option:
1. Parcel Tracking
2. Customer Data Validation
3. Address Formatting
4. Order Confirmation Email
5. Calculate Shipping Costs
6. Generate Password
7. Find Similar Addresses
8. Exit
Enter your choice: 6
Enter the desired password length: 8
Generated password: w/8b|4uC
```

Task 4.7: 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.

In this task, we will implement a function that finds similar addresses in the system. It will compare the user's input address with existing addresses stored in the system. The function will return similar addresses by matching the lowercase formatted string without special characters.

Code :

```java
// 7. Find Similar Addresses
public static void findSimilarAddresses(Scanner scanner) {
    String[] existingAddresses = {
            "No 10, Anna Nagar, Chennai, Tamil Nadu 600040",
            "No 45, Tidel Park, Chennai, Tamil Nadu 600100",
            "No 22, Kotturpuram, Chennai, Tamil Nadu 600085",
            "No 89, Egmore, Chennai, Tamil Nadu 600008",
            "No 12, Mount Road, Chennai, Tamil Nadu 600002",
            "No 56, Adyar, Chennai, Tamil Nadu 600020",
            "No 33, Mylapore, Chennai, Tamil Nadu 600004",
            "No 78, T Nagar, Chennai, Tamil Nadu 600017"
    };

    System.out.print("Enter an address to find similar addresses: ");
    String inputAddress = scanner.nextLine().toLowerCase().replaceAll( regex: "[^a-z0-9 ]", replacement: "").trim();

    System.out.println("Searching for similar addresses...");
    boolean foundSimilar = false;

    for (String address : existingAddresses) {
        String formattedAddress = address.toLowerCase().replaceAll( regex: "[^a-z0-9 ]", replacement: "").trim();
        if (formattedAddress.contains(inputAddress)) {
            System.out.println("Similar address found: " + address);
            foundSimilar = true;
        }
    }
```

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


}
```

Output :

```
Task 4 - Choose an option:
1. Parcel Tracking
2. Customer Data Validation
3. Address Formatting
4. Order Confirmation Email
5. Calculate Shipping Costs
6. Generate Password
7. Find Similar Addresses
8. Exit
Enter your choice: 7
Enter an address to find similar addresses: No 12, Mount Road, Chennai, Tamil Nadu 600002
Searching for similar addresses...
Similar address found: No 12, Mount Road, Chennai, Tamil Nadu 600002
```

Conclusion

In this task, we developed several functions to enhance the courier system using various Java programming concepts. We implemented parcel tracking, customer data validation, address formatting, order confirmation email generation, shipping cost calculation, password generation, and similar address finding. Through this exercise, we learned how to handle strings, arrays, and user inputs effectively and created a system that simulates real-world functionalities.
