Kingdom of Saudi Arabia
Ministry of Education
Imam Abdulrahman bin Faisal University
Computer Science department
College of Science and Humanities

CSC231 Project Milestone (1): Project Report of OOP1 Application

SAYAH

By

Leader	ID	Member Name
☆	2240003018	Zainab Hamed Hamada
	2240000820	ALhawraa Fahmi Ali
	2240003242	Lama Bakhet Alzahrani
	2240005883	Jana Ayed Almarri
	2240002063	Noor Mohammed Alhamadi
	2240000754	Munira Salmeen Alsiari
	2240007041	Lubna Sami Alamri

Introduction

Saudi Arabia, a country rich in cultural heritage, offers countless opportunities for tourism, ranging from its historical landmarks to modern attractions. In line with Saudi Arabia's Vision 2030, the tourism sector has seen significant development to attract local and international tourists. This project, the SAYAH Application, aims to provide tourists with an efficient and interactive way to explore the country's regions and tourist attractions. By integrating language preferences and providing detailed information about services, the app seeks to enhance the overall user experience.

Tourism plays a vital role in showcasing the beauty and diversity of Saudi Arabia's landscapes, from the ancient ruins of Al-Ula to the bustling streets of Riyadh and the serene beaches of the Red Sea. The increasing number of visitors demands a modern, accessible platform that can cater to their needs effectively. This project aspires to bridge the gap between tourists and the wealth of experiences Saudi Arabia has to offer, leveraging technology to create a seamless journey for all.

Objectives

The main objective of the SAYAH Application is to assist tourists in:

Choosing a language: Offering support for both Arabic and English, ensuring inclusivity and ease of use for local and international users.

Exploring regions: Providing comprehensive information about various tourist regions, including their historical significance, modern attractions, and natural wonders.

Accessing services: Enabling tourists to learn about available services such as hotels, restaurants, and historical landmarks, making trip planning more efficient.

Providing guidance: Ensuring ease of navigation and offering help options for tourists unfamiliar with certain locations, thus enhancing their confidence to explore new destinations.

Code Explanation

This project, SAYAH, is a Java application designed to provide information about tourist regions, weather, emergency services, and general help in Saudi Arabia. It helps users access organized information and interact with the app intuitively."

The application utilizes object-oriented programming (OOP) principles such as encapsulation, inheritance, polymorphism, and abstraction to create a modular and maintainable codebase. Below are the key classes used in the project:

SaudiTourist Class

The main class of the application that integrates all features.

This class acts as the central hub of the application. It creates objects of other classes like TouristRegion, Weather, Emergency, and Help and calls their methods to display the relevant information to the user."

```java
package com.tour.sauditourist;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.List;
import java.util.Scanner;

// Main Application
public class SaudiTourist {
    private static String language = "English"; // Default language
    private static Scanner scanner = new Scanner(source: System.in);

    public static void main(String[] args) {
        selectLanguage();
         if (!performLogin()) {
            System.out.println(x: getLocalizedMessage(english:"Login failed. Exiting application.", arabic: "فشل تسجيل الدخول. إنهاء التطبيق
            return;
        }
//        displayWelcomeMessage();

        List<TouristRegion> regions = createRegions();

        boolean exit = false;
        while (!exit) {
            System.out.println("\n" + getLocalizedMessage(english:"Choose a region or view help:", arabic: "اختر منطقة أو صفحة المساعدة:"));
            displayRegions(regions);
            System.out.println((regions.size() + 1) + ". " + getLocalizedMessage(english:"Help", arabic: "مساعدة"));
            System.out.println("0. " + getLocalizedMessage(english:"Exit", arabic: "خروج"));
            System.out.print(s: "> ");

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

            if (choice == 0) {
                exit = true;
```

TouristRegion Class

This class represents a region in Saudi Arabia and encapsulates attributes such as the region's name, a list of services, and a list of available guides.

Represents a tourist region with a name and list of attractions.

- It stores data about regions and attractions.
- Uses encapsulation to keep the fields private and provides getter methods to access them.

Code Snippet:

```java
// Classes and Relationships
class TouristRegion {
    private String name;
    private List<String> attractions;

    public TouristRegion(String name, List<String> attractions) {
        this.name = name;
        this.attractions = attractions;
    }

    public String getName() {
        return name;
    }

    public List<String> getAttractions() {
        return attractions;
    }
}
```

Encapsulation: The class uses private fields with public getters to control access to its attributes.

Abstraction: The class hides implementation details, exposing only the necessary methods for interaction.

C. Weather Class

Provides weather information.

- Stores weather details like currentWeather.
- Displays weather information using the displayWeather() method.

```java
public void updateWeather(String region) {
    // Simulated weather data based on the region
    switch (region) {
        case "Riyadh":
            currentWeather = "Sunny";
            forecast = "Clear skies for the next few days.";
            temperature = 40;
            break;
        case "Jeddah":
            currentWeather = "Humid";
            forecast = "Occasional clouds with a chance of rain.";
            temperature = 35;
            break;
        case "Eastern Province":
            currentWeather = "Windy";
            forecast = "Mild winds expected to continue.";
            temperature = 38;
            break;
        case "Abha":
            currentWeather = "Cool";
            forecast = "Clear skies with mild temperatures.";
            temperature = 25;
            break;
        case "Tabuk":
            currentWeather = "Mild";
            forecast = "Clear skies with cool evenings.";
            temperature = 28;
            break;
        case "Al-Jouf":
            currentWeather = "Dry";
            forecast = "Sunny with no significant changes expected.";
            temperature = 30;
            break;
        case "Hail":
```

D. Information Class

A base class for other information-related classes (Emergency and Help).

- Encapsulates fields like title, desc, and lastUpd.
- Includes a method showInfo() for displaying general information.

```java

// Base Class: Information
class Information {
    public String title, desc;
    public Date lastUpd;

    public Information(String title, String desc, Date lastUpd) {
        this.title = title;
        this.desc = desc;
        this.lastUpd = lastUpd;
    }

    public void showInfo() {
        System.out.println("Title: " + title);
        System.out.println("Description: " + desc);
        System.out.println("Last Updated: " + lastUpd);
    }
}

```

E. Emergency Class

Extends Information to add emergency-specific features.

- Adds contactNum for storing emergency contacts.
- Overrides showInfo() to include contact details.

```java

// Subclass: Emergency (Overriding)
class Emergency extends Information {
    public String contactNum;

    // Constructor for Emergency
    public Emergency(String title, String desc, Date lastUpd, String contactNum) {
        super(title, desc, lastUpd);
        this.contactNum = contactNum;
    }

    // Overridden method to display emergency-specific information
    @Override
    public void showInfo() {
        System.out.println(x: "=== Emergency Information ===");
        super.showInfo(); // Call superclass's showInfo method
        System.out.println("Contact Number: " + contactNum);
    }

    // Method to display emergency contact numbers
    public void showContact() {
        System.out.println(x: "=== Emergency Contact Numbers ===");
        System.out.println("Ambulance & Medical Emergencies: 997" + "\n"
                + "Police: 999" + "\n"
                + "Civil Defense (Fire & Rescue): 998" + "\n"
                + "Traffic Accidents: 993" + "\n"
                + "Electricity Emergency: 933" + "\n"
                + "Water & Sewage Issues: 020001744" + "\n"
                + "Public Transport Accidents (Trains & Buses): 8001249999" + "\n"
                + "Tourist Police: 939");
    }
}
```

"**Emergency** extends **Information** to reuse its fields like title and desc."

"It adds a *contactNum* field specific to emergencies."

"The showInfo() method is overridden to display both general information and the contact number."

F. Help Class

Provides help-related functionality.

- Overloads the showHelp() method to provide different levels of help (general, keyword-based, or detailed).

```java

// Subclass: Help (Overloading)
class Help extends Information {
    public String instructions;
    public String language;

    // Constructor for Help
    public Help(String title, String desc, Date lastUpd, String instructions, String language) {
        super(title, desc, lastUpd);
        this.instructions = instructions;
        this.language = language;
    }

    // Overloaded methods to display help information
    public void showHelp() {
        System.out.println("=== Help Information ===");
        System.out.println("Title: " + title);
        System.out.println("Description: " + desc);
        System.out.println("Instructions: " + instructions);
    }

    public void showHelp(String keyword) {
        System.out.println("=== Searching Help for Keyword: " + keyword + " ===");
        if (title.contains(s: keyword) || desc.contains(s: keyword)) {
            showHelp();
        } else {
            System.out.println("No help information found for keyword: " + keyword);
        }
    }

    public void showHelp(String keyword, boolean detailed) {
        System.out.println("=== Help Information with Details ===");
        if (title.contains(s: keyword) || desc.contains(s: keyword)) {
            showHelp();
            if (detailed) {
```

Applied OOP principles to make the code modular and reusable:

- **Encapsulation**: Fields like name and attractions are private, accessed only through getters.
- **Inheritance**: Emergency and Help reuse fields and methods from Information.
- **Polymorphism**: The showInfo() method behaves differently in Emergency and Help.
- **Abstraction**: The application hides the implementation details from the user, providing only high-level functionality.

Overloading:

- Help class methods handle different parameters for help-related functionalities.
- **Encapsulation:** Attributes in classes like TouristRegion and Information are encapsulated with appropriate getters and setters.

Output Screenshot

```
Output - Run (SaudiTourist)
--- exec-maven-plugin:3.1.0:exec (default-cli) @ SaudiTourist ---
Please select your language: 1. English  2. عربي
1
Login
Enter username: user
Enter password: user123
Login successful.
Welcome user to Saudi Tourism App!

Choose a region or view help:
1. Riyadh
2. Jeddah
3. Abha
4. Tabuk
5. Al-Jouf
6. Hail
7. Help
0. Exit
> 3

Tourist places in Abha:
1. Abha Palace Dam
2. Al-Soudah Park
3. Habala Village
4. Shada Palace
5. Green Mountain
> 3
You selected: Habala Village

Available services:
1. List of hotels
2. List of tourist guides
```

Conclusion

This project highlights the diverse tourism potential of Saudi Arabia by showcasing attractions, accommodations, and professional guides for various regions through a structured Java application. By leveraging Object-Oriented Programming (OOP) principles, the system ensures maintainability, scalability, and clarity. Encapsulation secures data within classes like TouristRegion, while abstraction simplifies user interaction through high-level methods like displayHotels and displayTouristGuides. The modular design enables easy additions, ensuring future adaptability as tourism offerings grow. Future work could involve integrating inheritance and polymorphism to manage specialized categories of regions or accommodations, enhancing functionality while aligning with Vision 2030's goals to promote Saudi Arabia as a leading global tourist destination.

References

Saudi Vision 2030 Official Website. (https://www.vision2030.gov.sa/)

Java Programming Documentation. (https://docs.oracle.com/javase/tutorial/)

Saudi Tourism Information. (https://www.sauditourism.sa/en/)