# Day 6

package day6;;

@FunctionalInterface

interface Instrument {

    void play();

}

public class InstrumentMain {

    public static void main(String[] args) {

        // 1. Create an array of 10 Instrument references

        Instrument[] instruments = new Instrument[10];

        // 2. Assign different types of instruments using lambda expressions

        instruments[0] = () -> System.out.println("Piano is playing: Tan tan tan tan");

        instruments[1] = () -> System.out.println("Flute is playing: Tooo tooo tooo");

        instruments[2] = () -> System.out.println("Guitar is playing: Tin tin tin");

        instruments[3] = () -> System.out.println("Piano is playing: Tan tan tan tan");

        instruments[4] = () -> System.out.println("Flute is playing: Tooo tooo tooo");

        instruments[5] = () -> System.out.println("Guitar is playing: Tin tin tin");

        instruments[6] = () -> System.out.println("Piano is playing: Tan tan tan tan");

        instruments[7] = () -> System.out.println("Flute is playing: Tooo tooo tooo");

        instruments[8] = () -> System.out.println("Guitar is playing: Tin tin tin");

        instruments[9] = () -> System.out.println("Piano is playing: Tan tan tan tan");

        // 3. Demonstrate polymorphic behavior by calling the play method

        System.out.println("Demonstrating polymorphic behavior:");

        for (int i = 0; i < instruments.length; i++) {

            System.out.print("Instrument at index " + i + ": ");

            instruments[i].play();

        }

        // 4. Use instanceof to print which object is stored at which index

        System.out.println("\nUsing instanceof to check object type:");

        for (int i = 0; i < instruments.length; i++) {

            if (instruments[i] instanceof Instrument) {

                System.out.println("Index " + i + " contains an Instrument instance.");

            }

        }

    }

}

Instrument at index 0: Piano is playing: Tan tan tan tan

Instrument at index 1: Flute is playing: Tooo tooo tooo

Instrument at index 2: Guitar is playing: Tin tin tin

Instrument at index 3: Piano is playing: Tan tan tan tan

Instrument at index 4: Flute is playing: Tooo tooo tooo

Instrument at index 5: Guitar is playing: Tin tin tin

Instrument at index 6: Piano is playing: Tan tan tan tan

Instrument at index 7: Flute is playing: Tooo tooo tooo

Instrument at index 8: Guitar is playing: Tin tin tin

Instrument at index 9: Piano is playing: Tan tan tan tan

Using instanceof to check object type:

Index 0 contains an Instrument instance.

Index 1 contains an Instrument instance.

Index 2 contains an Instrument instance.

Index 3 contains an Instrument instance.

Index 4 contains an Instrument instance.

Index 5 contains an Instrument instance.

Index 6 contains an Instrument instance.

Index 7 contains an Instrument instance.

Index 8 contains an Instrument instance.

Index 9 contains an Instrument instance.

2)

THIS CODE HAS TO BE MODIFIED

package day6;

import java.time.\*;

import java.time.format.DateTimeFormatter;

import java.util.Scanner;

public class HealthBoxApp {

    private static LocalDateTime appointmentDateTime;

    private static ZoneId customerZone;

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        int choice;

        do {

            System.out.println("\nHealthBox Appointment System:");

            System.out.println("1. Schedule an appointment");

            System.out.println("2. Print appointment details");

            System.out.println("3. Reschedule an appointment");

            System.out.println("4. Get reminder");

            System.out.println("5. Cancel appointment");

            System.out.println("6. Exit");

            System.out.print("Enter your choice: ");

            choice = scanner.nextInt();

            scanner.nextLine(); // Consume newline

            switch (choice) {

                case 1:

                    scheduleAppointment(scanner);

                    break;

                case 2:

                    printAppointmentDetails();

                    break;

                case 3:

                    rescheduleAppointment(scanner);

                    break;

                case 4:

                    getReminder();

                    break;

                case 5:

                    cancelAppointment();

                    break;

                case 6:

                    System.out.println("Exiting the application. Goodbye!");

                    break;

                default:

                    System.out.println("Invalid choice. Please try again.");

            }

        } while (choice != 6);

        scanner.close();

    }

    private static void scheduleAppointment(Scanner scanner) {

        try {

            System.out.print("Enter appointment date and time (yyyy-MM-dd HH:mm): ");

            String dateTimeInput = scanner.nextLine();

            System.out.print("Enter your time zone (e.g., Asia/Kolkata): ");

            String zoneInput = scanner.nextLine();

            DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");

            appointmentDateTime = LocalDateTime.parse(dateTimeInput, formatter);

            customerZone = ZoneId.of(zoneInput);

            System.out.println("Appointment scheduled successfully!");

        } catch (Exception e) {

            System.out.println("Invalid input. Please try again.");

        }

    }

    private static void printAppointmentDetails() {

        if (appointmentDateTime != null) {

            ZonedDateTime zonedDateTime = appointmentDateTime.atZone(customerZone);

            System.out.println("\nAppointment Details:");

            System.out.println("Date and Time: " + zonedDateTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm z")));

        } else {

            System.out.println("No appointment has been booked.");

        }

    }

    private static void rescheduleAppointment(Scanner scanner) {

        if (appointmentDateTime != null) {

            scheduleAppointment(scanner);

        } else {

            System.out.println("No appointment to reschedule.");

        }

    }

    private static void getReminder() {

        if (appointmentDateTime != null) {

            LocalDateTime reminderDateTime = appointmentDateTime.minusDays(1);

            ZonedDateTime reminderZoned = reminderDateTime.atZone(customerZone);

            System.out.println("\nReminder: Your appointment is scheduled for " +

                appointmentDateTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm")) + ".");

            System.out.println("Reminder will trigger on: " +

                reminderZoned.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm z")));

        } else {

            System.out.println("No appointment has been booked.");

        }

    }

    private static void cancelAppointment() {

        if (appointmentDateTime != null) {

            LocalDateTime now = LocalDateTime.now();

            if (appointmentDateTime.isAfter(now)) {

                appointmentDateTime = null;

                customerZone = null;

                System.out.println("Appointment cancelled successfully.");

            } else {

                System.out.println("Cannot cancel the appointment as it is already past the scheduled date.");

            }

        } else {

            System.out.println("No appointment to cancel.");

        }

    }

}

3)

package day6;

import java.util.Random;

// Employee class

class Employee {

    String name;

    long[] phoneNo;

    String passportNo;

    Integer licenseNo;

    String panCardNo;

    Integer voterId;

    Integer employeeId;

    // Constructor with passportNo

    public Employee(String name, long[] phoneNo, Integer employeeId, String passportNo) {

        this.name = name;

        this.phoneNo = phoneNo;

        this.employeeId = employeeId;

        this.passportNo = passportNo;

    }

    // Constructor with licenseNo and panCardNo

    public Employee(String name, long[] phoneNo, Integer employeeId, Integer licenseNo, String panCardNo) {

        this.name = name;

        this.phoneNo = phoneNo;

        this.employeeId = employeeId;

        this.licenseNo = licenseNo;

        this.panCardNo = panCardNo;

    }

    // Constructor with voterId and licenseNo

    public Employee(String name, long[] phoneNo, Integer employeeId, Integer voterId, Integer licenseNo) {

        this.name = name;

        this.phoneNo = phoneNo;

        this.employeeId = employeeId;

        this.voterId = voterId;

        this.licenseNo = licenseNo;

    }

}

// Student class

class Student {

    String name;

    long[] phoneNo;

    String passportNo;

    Integer licenseNo;

    String panCardNo;

    Integer voterId;

    // Constructor with passportNo

    public Student(String name, long[] phoneNo, String passportNo) {

        this.name = name;

        this.phoneNo = phoneNo;

        this.passportNo = passportNo;

    }

    // Constructor with licenseNo and panCardNo

    public Student(String name, long[] phoneNo, Integer licenseNo, String panCardNo) {

        this.name = name;

        this.phoneNo = phoneNo;

        this.licenseNo = licenseNo;

        this.panCardNo = panCardNo;

    }

    // Constructor with voterId and licenseNo

    public Student(String name, long[] phoneNo, Integer voterId, Integer licenseNo) {

        this.name = name;

        this.phoneNo = phoneNo;

        this.voterId = voterId;

        this.licenseNo = licenseNo;

    }

}

// Generic Register class

class Register<T> {

    private String registerId;

    // Method to generate registration ID

    public String generateRegisterId(int n) {

        String characters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

        Random random = new Random();

        StringBuilder idBuilder = new StringBuilder();

        for (int i = 0; i < n; i++) {

            idBuilder.append(characters.charAt(random.nextInt(characters.length())));

        }

        this.registerId = idBuilder.toString();

        return this.registerId;

    }

    // Method to display user details

    public void display(T obj) {

        System.out.println("\nUser Details:");

        System.out.println(obj.toString());

        System.out.println("Registration ID: " + this.registerId);

    }

}

// Tester class

public class Generic {

    public static void main(String[] args) {

        // Employee Registration

        long[] empPhone = {9876543210L, 9123456780L};

        Employee emp = new Employee("John Doe", empPhone, 12345, "A1234567");

        Register<Employee> empRegister = new Register<>();

        String empRegId = empRegister.generateRegisterId(8);

        empRegister.display(emp);

        // Student Registration

        long[] stuPhone = {9876543210L, 9876501234L};

        Student stu = new Student("Jane Smith", stuPhone, "B9876543");

        Register<Student> stuRegister = new Register<>();

        String stuRegId = stuRegister.generateRegisterId(10);

        stuRegister.display(stu);

    }

}