Problem Statement 1: Java 8- Lambda Expressions

```
interface Instrument {
    void play();
}

public class LambdaExpressionExample {
    public static void main(String[] args) {
        Instrument piano = () -> System.out.println("Piano is playing tan
tan tan tan");
        Instrument flute = () -> System.out.println("Flute is playing
toot toot toot toot");
        Instrument guitar = () -> System.out.println("Guitar is playing
tin tin tin");

        Instrument[] instruments = new Instrument[10];
        instruments[0] = piano;
        instruments[1] = flute;
        instruments[2] = guitar;
        instruments[3] = piano;
        instruments[4] = flute;
        instruments[5] = guitar;
        instruments[6] = piano;
        instruments[7] = flute;
        instruments[8] = guitar;
        instruments[9] = piano;

        for (int i = 0; i < instruments.length; i++) {
            instruments[i].play();
            if (instruments[i] instanceof Instrument) {
                System.out.println("Instrument at index " + i + " is " +
instruments[i].getClass().getSimpleName());
            }
        }
    }
}
```

Problem Statement 2: New Date-Time API in Java 8

```
import java.time.*;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;

class Appointment {
    private LocalDateTime dateTime;
    private ZoneId zoneId;

    public void schedule(LocalDate date, LocalTime time, ZoneId zone) {
        this.dateTime = LocalDateTime.of(date, time);
        this.zoneId = zone;
        System.out.println("Successfully Booked");
    }

    public void print() {
        if (dateTime == null) {
            System.out.println("No appointment booked.");
        } else {
            ZonedDateTime zonedDateTime = dateTime.atZone(zoneId);
```

```
System.out.println(zonedDateTime.format(DateTimeFormatter.ofPattern("E,
MMM dd yyyy hh:mm a z")));
        }
    }

    public void reschedule(int days, LocalTime newTime) {
        if (dateTime == null) {
            System.out.println("No appointment booked.");
        } else {
            this.dateTime =
this.dateTime.plusDays(days).withHour(newTime.getHour()).withMinute(newTi
me.getMinute());
            print();
        }
    }

    public void getReminder() {
        if (dateTime == null) {
            System.out.println("No appointment booked.");
        } else {
            LocalDateTime reminderTime = dateTime.minusDays(1);
            ZonedDateTime reminderZoned = reminderTime.atZone(zoneId);

System.out.println(reminderZoned.format(DateTimeFormatter.ofPattern("E,
MMM dd yyyy hh:mm a z")));
        }
    }

    public void cancel() {
        if (dateTime == null) {
            System.out.println("No appointment booked.");
        } else {
            this.dateTime = null;
            this.zoneId = null;
            System.out.println("Appointment has been cancelled!!");
        }
    }
}

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

        while (true) {
            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 the Appointment");
            System.out.println("6. Exit");
            System.out.print("Enter an option: ");
            int option = sc.nextInt();

            switch (option) {
                case 1:
                    System.out.print("Enter Date (dd/MM/yyyy): ");
                    String date = sc.next();
                    System.out.print("Enter Time (HH:mm): ");
                    String time = sc.next();
                    System.out.println("Available Zones are:");
                    System.out.println("A: America/Anchorage");
                    System.out.println("B: Europe/Paris");
                    System.out.println("C: Asia/Tokyo");
                    System.out.println("D: America/Phoenix");
                    System.out.print("Select the Zone: ");
                    String zone = sc.next();
                    ZoneId zoneId = ZoneId.of(zone);

                    LocalDate localDate = LocalDate.parse(date,
DateTimeFormatter.ofPattern("dd/MM/yyyy"));
                    LocalTime localTime = LocalTime.parse(time,
DateTimeFormatter.ofPattern("HH:mm"));
                    appointment.schedule(localDate, localTime, zoneId);
                    break;
                case 2:
                    appointment.print();
                    break;
                case 3:
                    System.out.println("Current Appointment Date is:");
                    appointment.print();
                    System.out.print("Kindly Enter Number of Days to be
postponed: ");
                    int days = sc.nextInt();
                    System.out.print("Enter the new time in HH:mm: ");
                    String newTime = sc.next();
                    LocalTime newLocalTime = LocalTime.parse(newTime,
DateTimeFormatter.ofPattern("HH:mm"));
                    appointment.reschedule(days, newLocalTime);
                    break;
                case 4:
                    appointment.getReminder();
                    break;
                case 5:
                    appointment.cancel();
                    break;
                case 6:
                    sc.close();
                    System.exit(0);
                default:
                    System.out.println("Invalid option. Please try
again.");
            }
        }
    }
}
```

Problem Statement 3: Design the highly general and reusable code with Generic classes

```
import java.util.Random;

class Register<T> {
    private String registerId;
    public String generateRegisterId(int n) {
        String characters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
        Random rand = new Random();
        StringBuilder sb = new StringBuilder(n);
        for (int i = 0; i < n; i++) {

sb.append(characters.charAt(rand.nextInt(characters.length())));
        }
        return sb.toString();
    }

    public void display(T user) {
        System.out.println(user.toString());
    }
}

class Employee {
    private String name;
    private long[] phoneNo;
    private String passportNo;
    private Integer licenseNo;
    private String panCardNo;
    private Integer voterId;
    private Integer employeeId;

    public Employee(String name, long[] phoneNo, Integer employeeId,
String passportNo) {
        this.name = name;
        this.phoneNo = phoneNo;
        this.employeeId = employeeId;
        this.passportNo = passportNo;
    }

    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;
    }

    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;
    }

    @Override
    public String toString() {
        return "Employee [Name=" + name + ", Phone No's=" + phoneNo + ",
Employee Id=" + employeeId + ", Passport No=" + passportNo + ", License
No=" + licenseNo + ", Pan Card No=" + panCardNo + ", Voter Id=" + voterId
+ "]";
    }
```

```
}

class Student {
    private String name;
    private long[] phoneNo;
    private String passportNo;
    private Integer licenseNo;
    private String panCardNo;
    private Integer voterId;

    public Student(String name, long[] phoneNo, String passportNo) {
        this.name = name;
        this.phoneNo = phoneNo;
        this.passportNo = passportNo;
    }

    public Student(String name, long[] phoneNo, Integer licenseNo, String
panCardNo) {
        this.name = name;
        this.phoneNo = phoneNo;
        this.licenseNo = licenseNo;
        this.panCardNo = panCardNo;
    }

    public Student(String name, long[] phoneNo, Integer voterId, Integer
licenseNo) {
        this.name = name;
        this.phoneNo = phoneNo;
        this.voterId = voterId;
        this.licenseNo = licenseNo;
    }

    @Override
    public String toString() {
        return "Student [Name=" + name + ", Phone No's=" + phoneNo + ",
Passport No=" + passportNo + ", License No=" + licenseNo + ", Pan Card
No=" + panCardNo + ", Voter Id=" + voterId + "]";
    }
}

public class Tester {
    public static void main(String[] args) {
        Register<Employee> employeeRegister = new Register<>();
        Employee emp1 = new Employee("Arun", new long[]{9997389981L,
90949309552L}, 1101, "LA788333DG");
        emp1.toString();
        String empRegId1 = employeeRegister.generateRegisterId(7);
        System.out.println("Details of the Employee:\nHurray!! you
availed a discount of 10%\nRegistered Id: " + empRegId1);
        employeeRegister.display(emp1

);

        Register<Student> studentRegister = new Register<>();
        Student student1 = new Student("Joseph", new long[]{9038474875L,
8359493029L}, 2210, "DUPPS2781J");
        String studentRegId1 = studentRegister.generateRegisterId(7);
        System.out.println("Details of the Student:\nHurray!! you availed
a discount of 22%\nRegistered Id: " + studentRegId1);
        studentRegister.display(student1);
    }
}
```