Lesson – 3

Reading work: Do the reading of concepts discussed in this chapter.

Problem: 1 – Class & Objects

Create your own Customer class. A Customer has a firstName, lastName, socSecurityNum (which you can represent as a String), also it has billingAddress and a shippingAddress(which you can represent as a type of Address. Initialize billingAddress and shippingAddress through its setter from Customer class.

Create a constructor for your Customer class to initialize firstName, lastName and socSecurityNum. Create getter, and setter methods for all five attributes.

Create an Address class with the attributes such as street, city, state and zip(which you can represent as a String). Create a constructor to initialize fields of Address class.

Your Customer class should have a toString() method that provides a string representation of the customer. A typical toString() output would be "[Joe, Smith, ssn: 332-221-4444]". Just copy this code in your Customer class.

```
public String toString() {
        return "[" + firstName + ", " + lastName + ", " + "ssn: " + socSecurityNum + "]";
        }
```

In the main method of a Main class, create three instances of Customer (be sure to create instances of Address to populate their billingAddress and shippingAddress fields using setters). Add these instances to an array. Then loop through the array and print to the console all those Customers whose billingAddress is located in the city of Chicago (when you create instances of Customer initially, be sure to create at least one Customer whose billing address is in Chicago).

Problem 2:

Develop a Java application that helps a global team schedule meetings across different time zones. The application should allow users to create an event by specifying a date and time, view the event in a different time zone, and calculate the days until the event. Nicely show the date and time display.

Requirements:

1. Create an Event to the input the details and print the requested details:
    a. Allow users to input the event name, date and time for a new event.
    b. Display the day of the week (Eg: Sunday, Monday) for the event date and check if it is in a leap year.
    c. Calculate and display the number of days from the current date to the event date.
2. Write a function to Format Event Details.
    a. A Function, nicely format and display the event date and time, including the default system time zone.
    Eg: Sunday, September 20, 2025 @ 10:30 [America/Chicago]
3. Time Zone Conversion function:
    a. Convert the event's date and time from the system's default time zone to any other, specified time zone entered by the user.

Sample Time Zones

America/Panama

America/Chicago

America/Indiana/Indianapolis

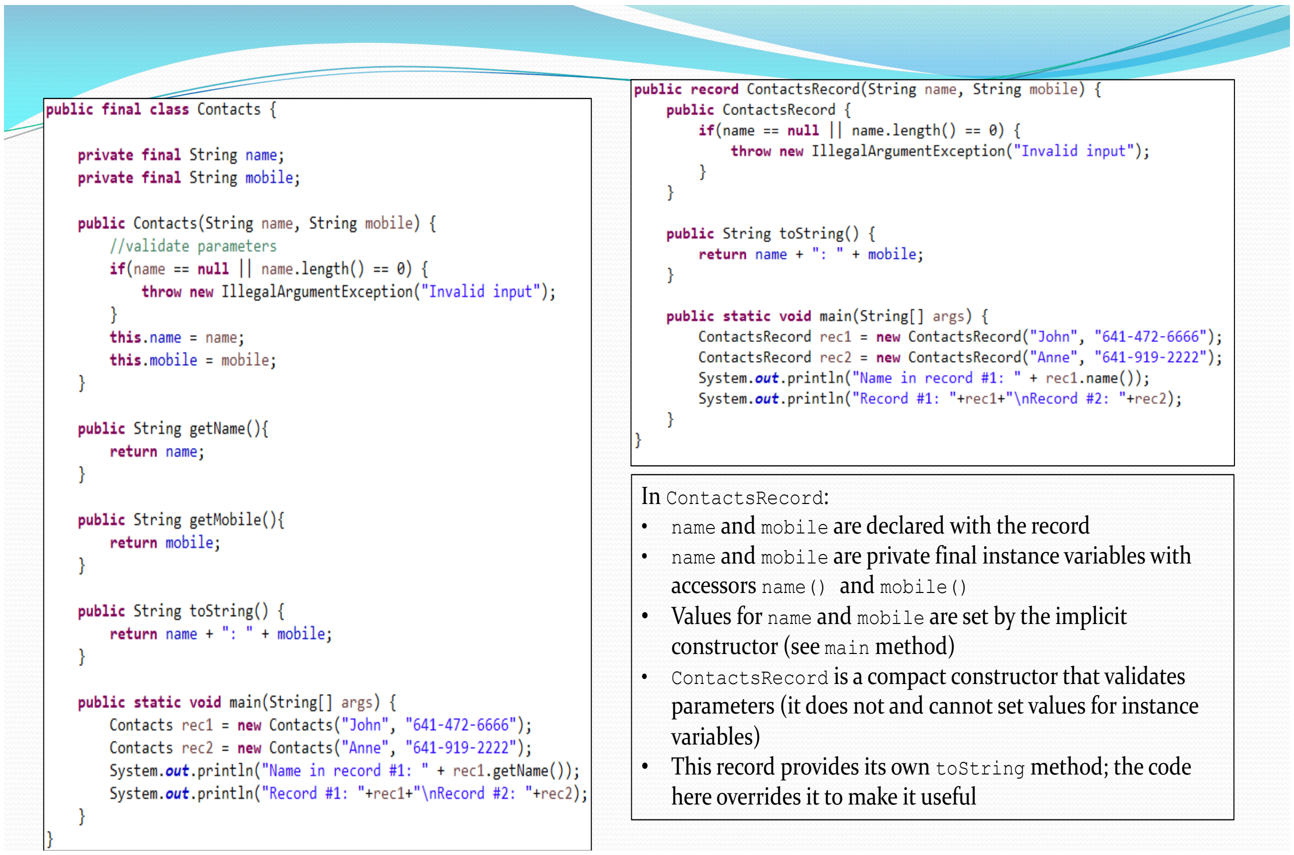
America/Santiago

America/Phoenix

Problem 3: Immutable & Record class

We want to create **three types of travel bookings**. Each booking type should be **immutable** (once created, values cannot be changed). Must follow the Immutable rules discussed in class. **Introduce compact constructor to validates parameters.**

Reference:



```java
public final class Contacts {

    private final String name;
    private final String mobile;

    public Contacts(String name, String mobile) {
        //validate parameters
        if(name == null || name.length() == 0) {
            throw new IllegalArgumentException("Invalid input");
        }
        this.name = name;
        this.mobile = mobile;
    }

    public String getName(){
        return name;
    }

    public String getMobile(){
        return mobile;
    }

    public String toString() {
        return name + ": " + mobile;
    }

    public static void main(String[] args) {
        Contacts rec1 = new Contacts("John", "641-472-6666");
        Contacts rec2 = new Contacts("Anne", "641-919-2222");
        System.out.println("Name in record #1: " + rec1.getName());
        System.out.println("Record #1: "+rec1+"\nRecord #2: "+rec2);
    }
}
```

```java
public record ContactsRecord(String name, String mobile) {
    public ContactsRecord {
        if(name == null || name.length() == 0) {
            throw new IllegalArgumentException("Invalid input");
        }
    }

    public String toString() {
        return name + ": " + mobile;
    }

    public static void main(String[] args) {
        ContactsRecord rec1 = new ContactsRecord("John", "641-472-6666");
        ContactsRecord rec2 = new ContactsRecord("Anne", "641-919-2222");
        System.out.println("Name in record #1: " + rec1.name());
        System.out.println("Record #1: "+rec1+"\nRecord #2: "+rec2);
    }
}
```

In `ContactsRecord`:
- `name` and `mobile` are declared with the record
- `name` and `mobile` are private final instance variables with accessors `name()` and `mobile()`
- Values for `name` and `mobile` are set by the implicit constructor (see `main` method)
- `ContactsRecord` is a compact constructor that validates parameters (it does not and cannot set values for instance variables)
- This record provides its own `toString` method; the code here overrides it to make it useful

1. FlightBooking (use record class)

- Instance Fields: origin, destination, distanceKm
- Method: compute and return distanceKm / avgSpeed

public double computeFlightTime(double avgSpeed)

2. HotelBooking (use record class)

- Instance Fields: hotelName, nights, pricePerNight
- Method: compute and return nights * pricePerNight

public double totalCost()

3. CarRental (manual immutable class)

- Fields: carModel, days, ratePerDay, milesPerDay
- Methods:

```
public double totalRentalCost()      // return days * ratePerDay
public double totalMilesAllowed()    // return days * milesPerDay
```

4. Main Class (with loop)

- Show menu:

```
Enter F for Flight Booking
Enter H for Hotel Booking
Enter C for Car Rental
```

- Based on user choice:
    - Ask for input fields from the console using Scanner.
    - Create the object based on the Input request.
    - Call the associated computation method and print result on the console.
- Ask: Do you want to continue (y/n)?
- Stop if user enters n.

Sample Output:

```
Enter F for Flight Booking

Enter H for Hotel Booking

Enter C for Car Rental

F

Enter origin: Chicago

Enter destination: Bangalore

Enter distance in km: 6000

Enter average speed (km/h): 450

Estimated Flight Time: 13.33 hours

Do you want to continue (y/n): y


Enter F for Flight Booking

Enter H for Hotel Booking
Enter C for Car Rental
H
Enter hotel name: Holiday Inn
Enter number of nights: 2
Enter price per night: 250
Total Hotel Cost: 500.00
Do you want to continue (y/n): y

Enter F for Flight Booking
Enter H for Hotel Booking
Enter C for Car Rental
C
Enter car model: 2023
Enter number of days: 2
Enter rate per day: 15
Enter miles per day: 30
Total Rental Cost: 30.00
Total Miles Allowed: 60.00
Do you want to continue (y/n): n
```

Problem 4: Pizza Order Management with Enums and Switch Expressions

A) Define an enum named **PizzaSize** representing different pizza sizes (SMALL, MEDIUM, LARGE).

B) Define an enum named **PizzaType** representing different pizza types (VEGGIE, PEPPERONI, CHEEZE, BBQCHICKEN).

C) Create a Class Pizza includes,

   a. An attribute for the pizza size of type PizzaSize.
   b. An attribute for the pizza type of type PizzaType.
   c. An attribute for the quantity
   d. The attribute price not initialized through constructor, need to calculate the price, based on PizzaSize and PizzaType using a private method calculatePrice().
   e. A constructor to initialize PizzaSize, PizzaType and quantity.

D) Create a private void calculatePrice() method which assign the price instance using Switch expression and enum types. You can assign some default values based on the pizza sizes (SMALL($8), MEDIUM($10), LARGE($12) using Switch expression to return the sizeprice. In the same way another switch return the default price based on the types of Pizza(VEGGIE($1), PEPPERONI($2), CHEEZE($1.5), BBQCHICKEN($2)). Finally assign the price instance by adding the sizeprice and typeprice multiplied by quantity.

E) A method printOrderSummary() that returns a string summarizing the pizza order, including size, type, qty, price, tax amount, and total price. Use String.format() to print the receipt as mentioned below.

```
Pizza Order:
Size: SMALL
Type: VEGGIE
Qty: 2
Price: $18.00
Tax: $0.54
Total Price: $18.54
```

Add 3% of tax from the price calculated from the Task D. Total Price is price + tax.

F) Use the below code

```java
public class PizzaTest {
  public static void main(String[] args) {
    Pizza pizza1 = new Pizza(PizzaSize.SMALL, PizzaType.VEGGIE, 2);
    Pizza pizza2 = new Pizza(PizzaSize.MEDIUM, PizzaType.PEPPERONI, 1);
    Pizza pizza3 = new Pizza(PizzaSize.LARGE, PizzaType.BBQ_CHICKEN, 2);
    System.out.println(pizza1.printOrderSummary());
    System.out.println(pizza2.printOrderSummary());
    System.out.println(pizza3.printOrderSummary());
  }
}
```

Interview Reading Question

Lesson-3-Class and Objects

1. What is Immutable? Tell the Immutable APIs.
2. Speak to difference between a static variable and static method in java.

3. What is the purpose of using OOPs concepts?
4. What are the four main features of OOPs?
5. What is the difference between Heap and Stack Memory in java?
6. What is a singleton class in Java? How do you implement a singleton class?
7. What is an immutable class in Java? How do you implement an immutable class?
8. Describe how Java works as “pass by value” or “pass by reference” phenomenon?