CECS 575 Object Oriented Analysis and Design

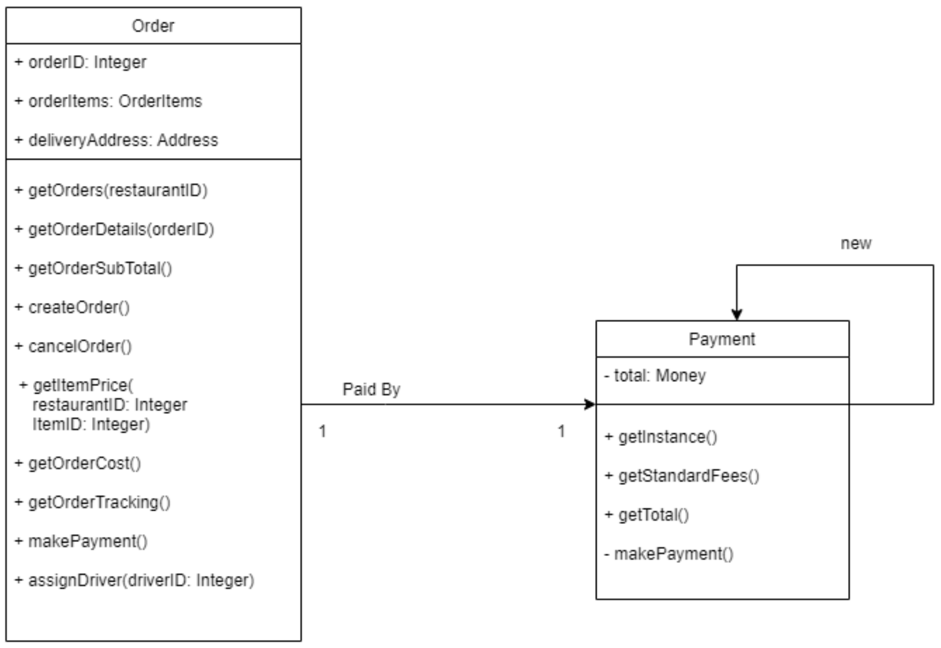
Assignment 2

Date: 03/16/2023

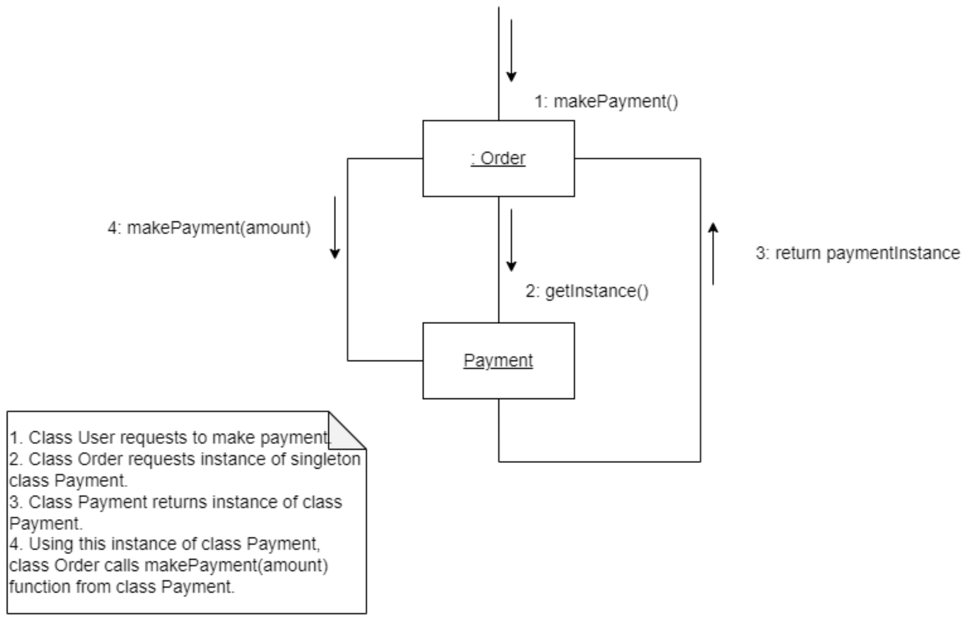
California State University Long Beach

1. Singleton design pattern

a. Class Diagram

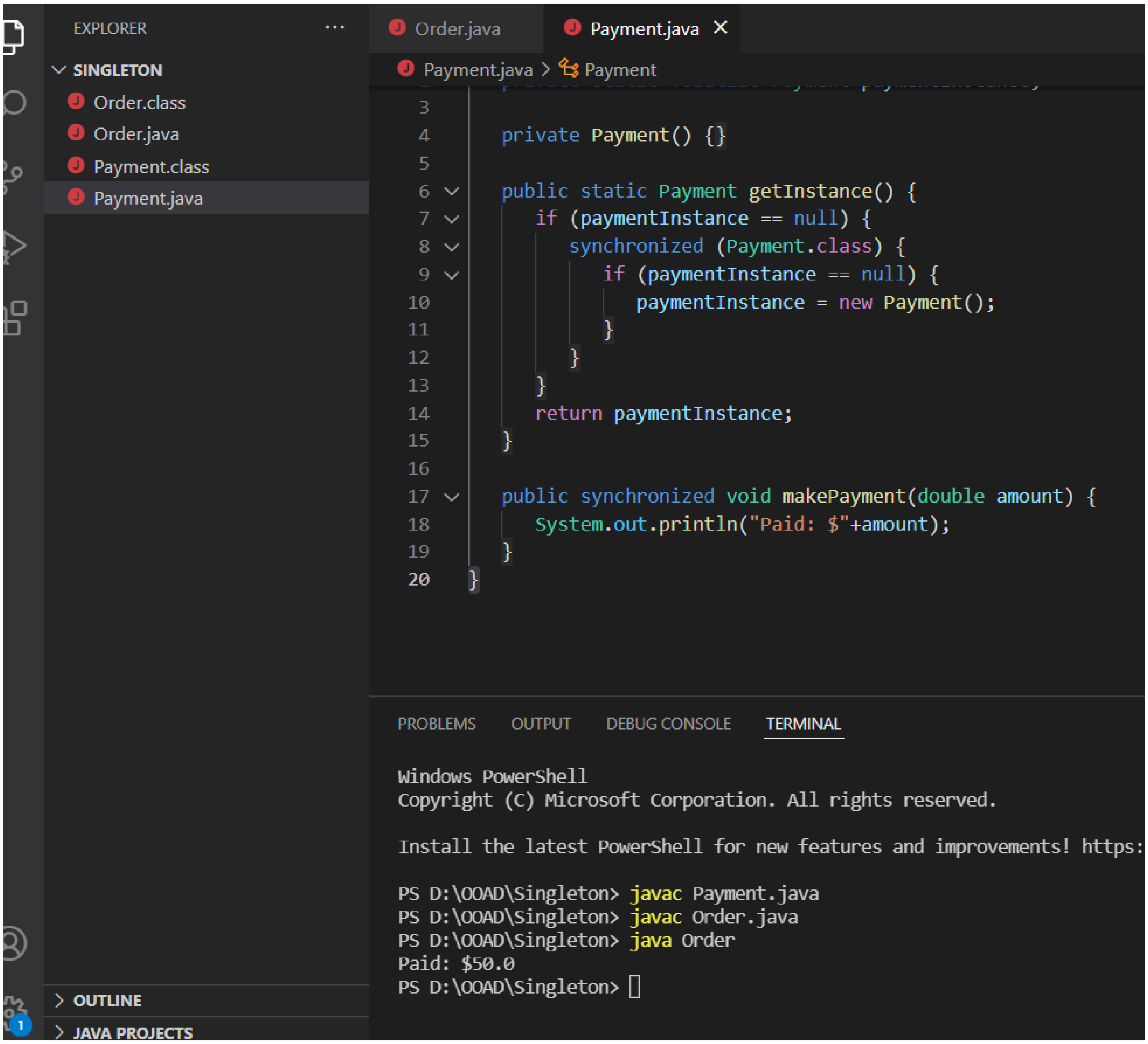


b. Collaboration Diagram


1: makePayment()
: Order
4: makePayment(amount)
2: getInstance()
3: return paymentInstance
Payment
1. Class User requests to make payment.
2. Class Order requests instance of singleton class Payment.
3. Class Payment returns instance of class Payment.
4. Using this instance of class Payment, class Order calls makePayment(amount) function from class Payment.

In the food delivery system, payment can be a singleton class. Here, after an order is created using the order class, the makePayment() function from Payment is called. To call this method, an instance of class Payment is requested by class Order. I have used the ‘Lazy Initialization with Double Check Locking’ method for creating a singleton pattern.

Users should pay once for every order as well as a payment can be made for one order at a time for each user. For this reason, Payment class can have a singleton design pattern.

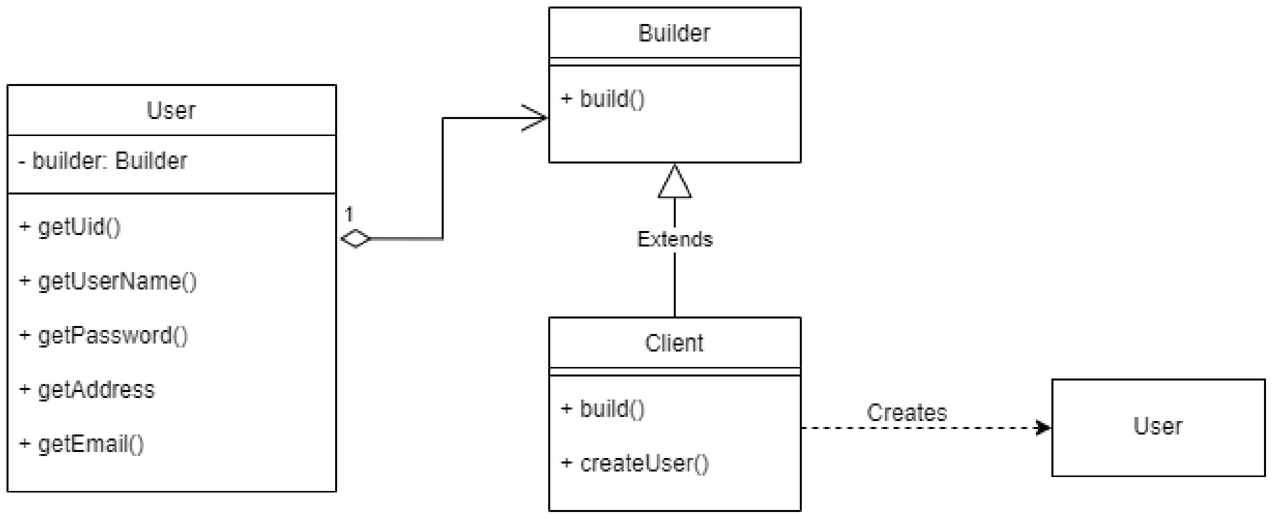


```java

    private Payment() {}

    public static Payment getInstance() {
        if (paymentInstance == null) {
            synchronized (Payment.class) {
                if (paymentInstance == null) {
                    paymentInstance = new Payment();
                }
            }
        }
        return paymentInstance;
    }

    public synchronized void makePayment(double amount) {
        System.out.println("Paid: $"+amount);
    }
}
```

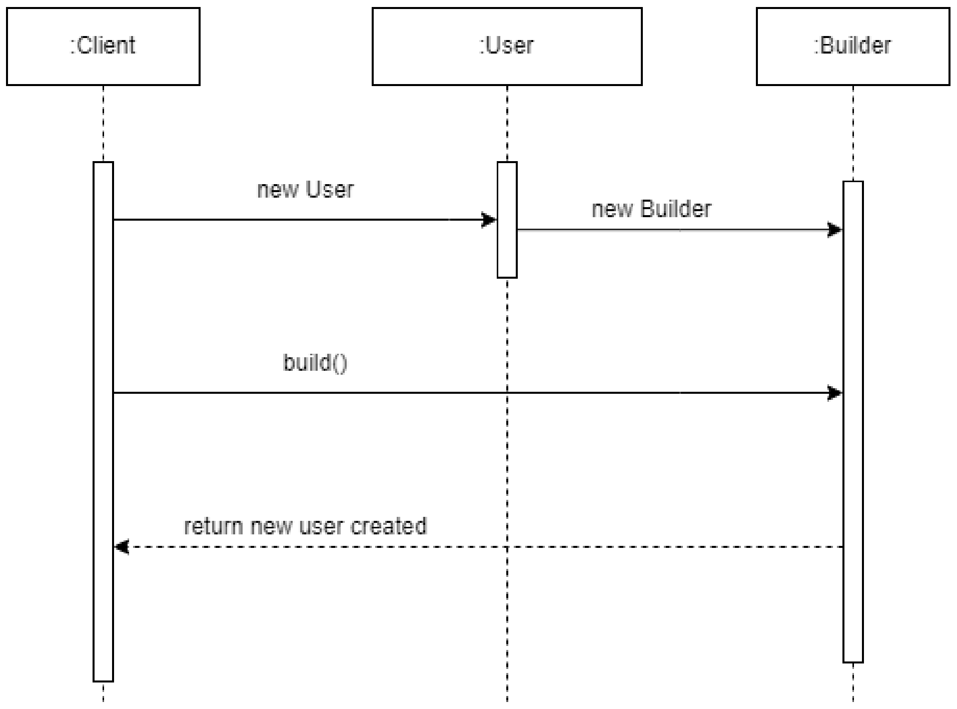
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https:

PS D:\OOAD\Singleton> javac Payment.java
PS D:\OOAD\Singleton> javac Order.java
PS D:\OOAD\Singleton> java Order
Paid: $50.0
PS D:\OOAD\Singleton> 
```

2. Joshua Bloch's builder design pattern

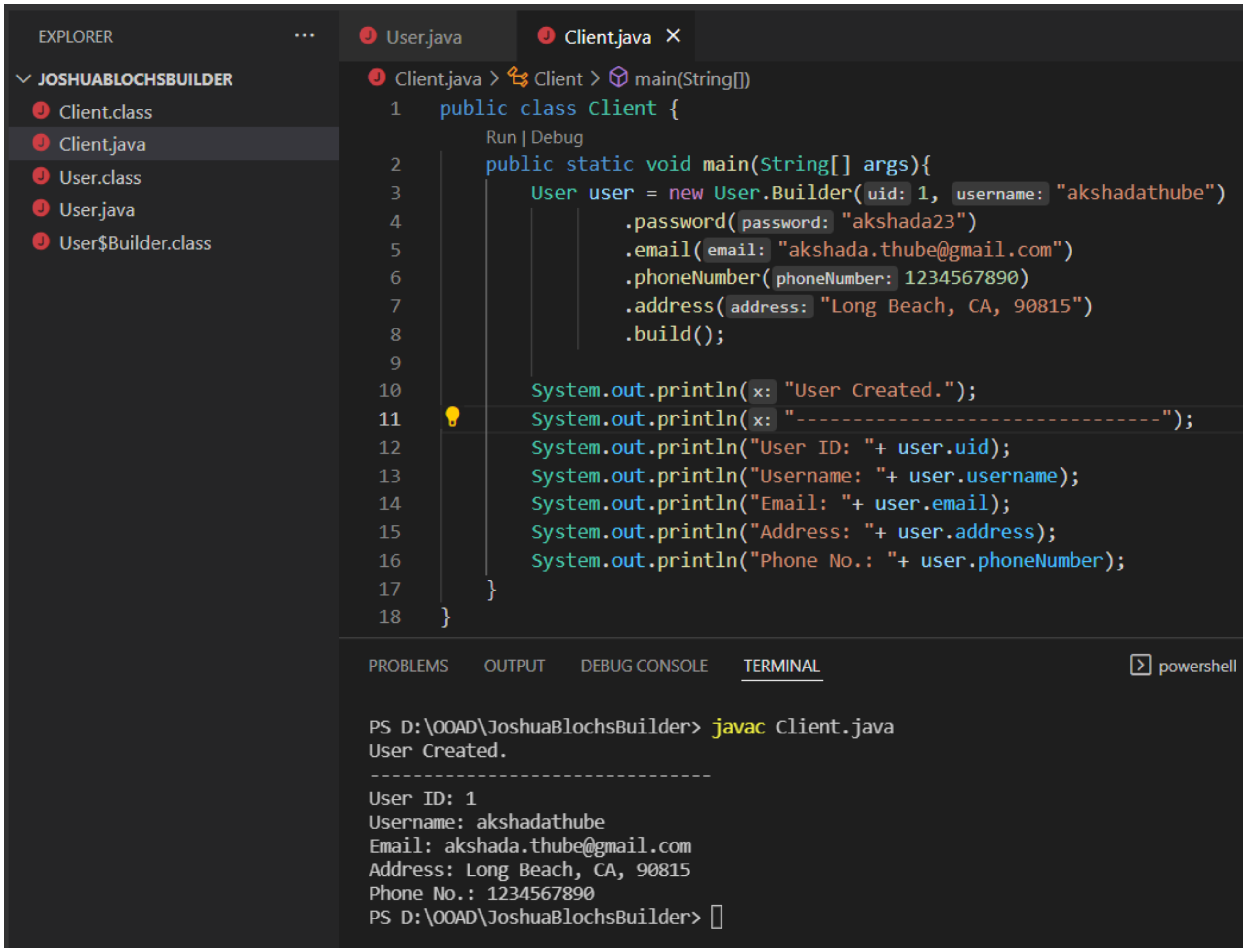
a. Class Diagram



b. Sequence Diagram



Here Josua Bloch's builder design pattern is used to build new users objects such as username, userId, password, email, address, phone number. Also note that the class Client might be any class responsible for creating users.

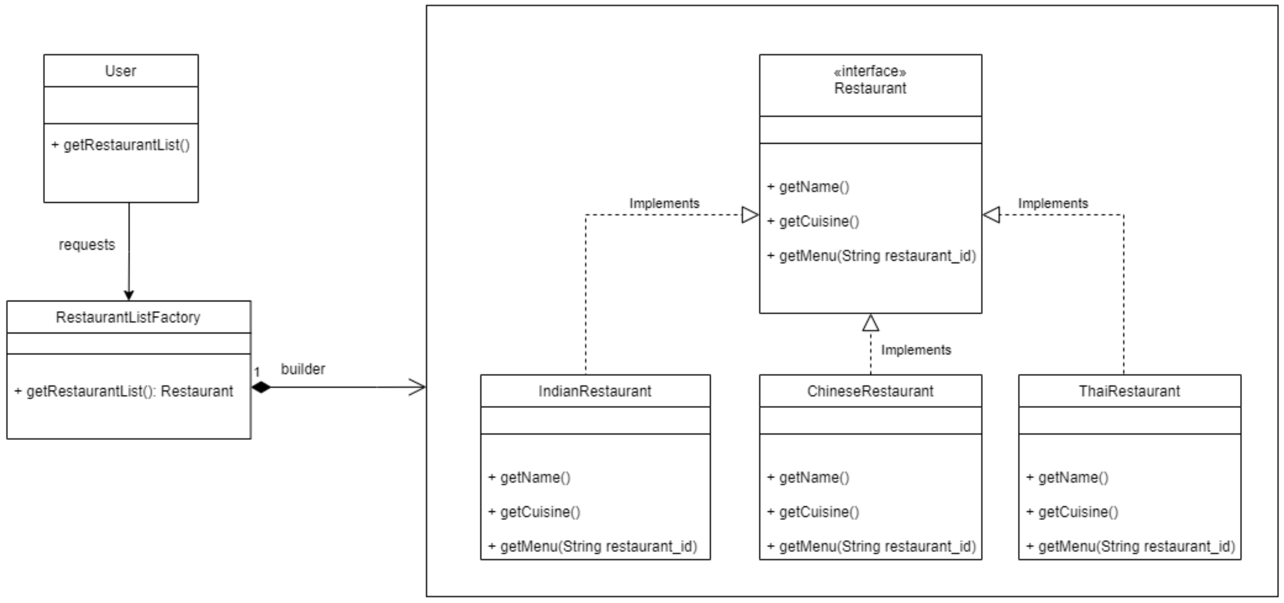


```java
public class Client {
    public static void main(String[] args){
        User user = new User.Builder(uid: 1, username: "akshadathube")
                    .password(password: "akshada23")
                    .email(email: "akshada.thube@gmail.com")
                    .phoneNumber(phoneNumber: 1234567890)
                    .address(address: "Long Beach, CA, 90815")
                    .build();

        System.out.println(x: "User Created.");
        System.out.println(x: "--------------------------------");
        System.out.println("User ID: "+ user.uid);
        System.out.println("Username: "+ user.username);
        System.out.println("Email: "+ user.email);
        System.out.println("Address: "+ user.address);
        System.out.println("Phone No.: "+ user.phoneNumber);
    }
}
```

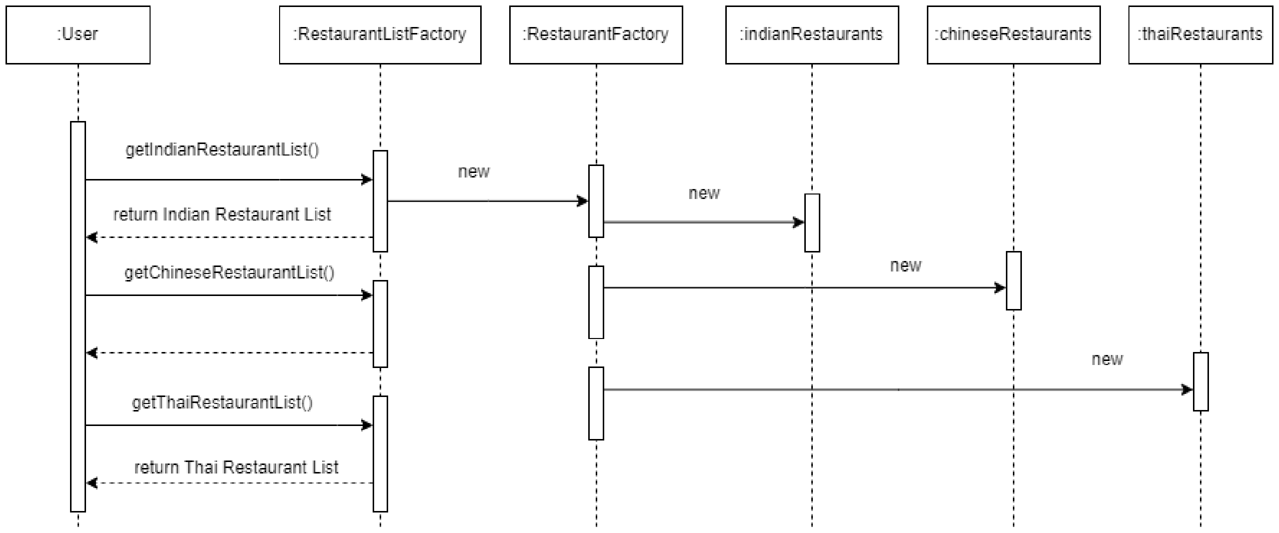
```
PS D:\OOAD\JoshuaBlochsBuilder> javac Client.java
User Created.
--------------------------------
User ID: 1
Username: akshadathube
Email: akshada.thube@gmail.com
Address: Long Beach, CA, 90815
Phone No.: 1234567890
PS D:\OOAD\JoshuaBlochsBuilder>
```

3. Factory Method

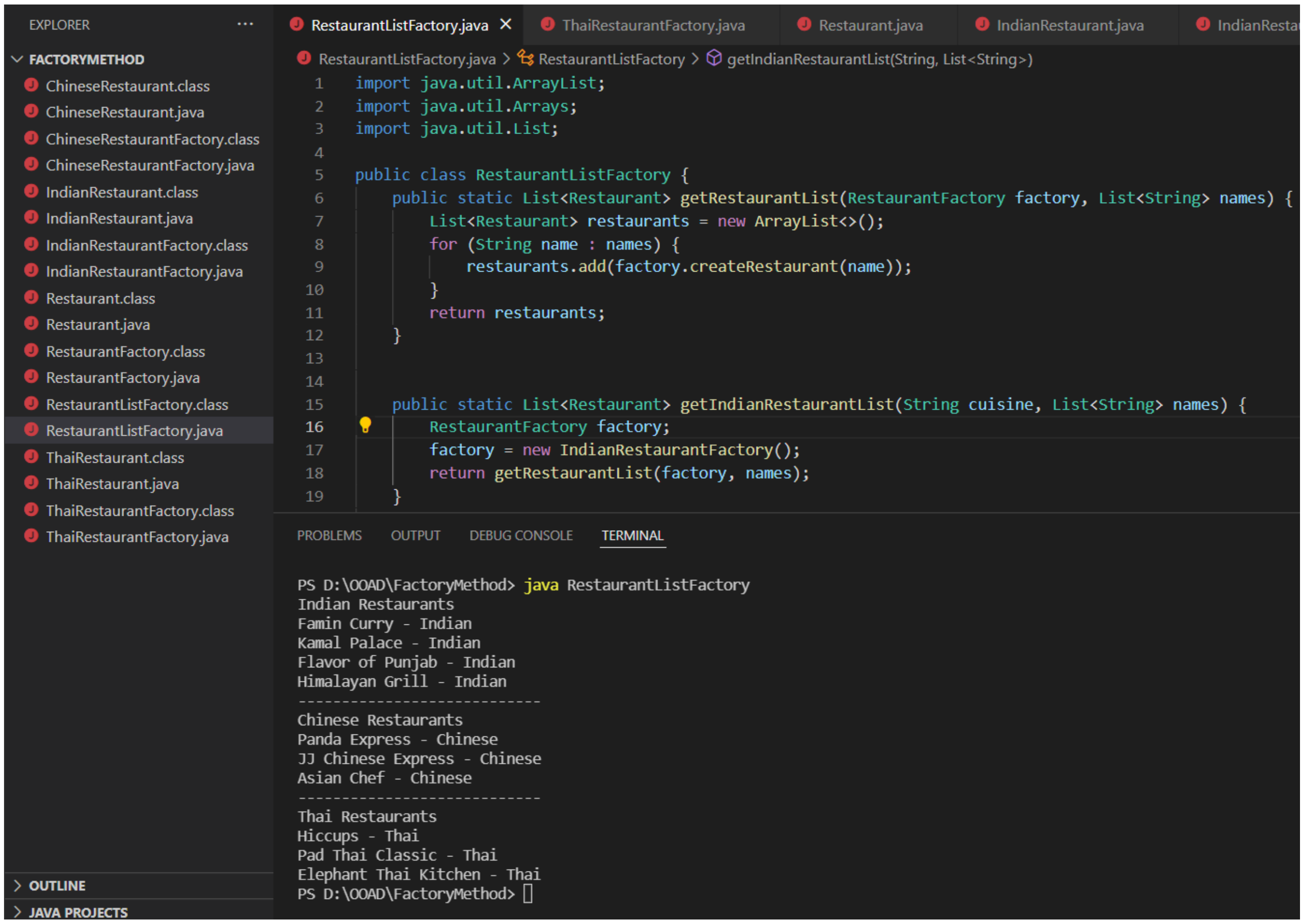
a. Class Diagram



b. Sequence Diagram



Factory method is used to implement restaurant list as per different cuisines. Users might want to look for specific cuisines. Using the factory method design pattern, the system can fetch and display specific cuisine options to the user.



EXPLORER

FACTORYMETHOD
- ChineseRestaurant.class
- ChineseRestaurant.java
- ChineseRestaurantFactory.class
- ChineseRestaurantFactory.java
- IndianRestaurant.class
- IndianRestaurant.java
- IndianRestaurantFactory.class
- IndianRestaurantFactory.java
- Restaurant.class
- Restaurant.java
- RestaurantFactory.class
- RestaurantFactory.java
- RestaurantListFactory.class
- RestaurantListFactory.java
- ThaiRestaurant.class
- ThaiRestaurant.java
- ThaiRestaurantFactory.class
- ThaiRestaurantFactory.java

OUTLINE
JAVA PROJECTS

RestaurantListFactory.java > RestaurantListFactory > getIndianRestaurantList(String, List<String>)

```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class RestaurantListFactory {
    public static List<Restaurant> getRestaurantList(RestaurantFactory factory, List<String> names) {
        List<Restaurant> restaurants = new ArrayList<>();
        for (String name : names) {
            restaurants.add(factory.createRestaurant(name));
        }
        return restaurants;
    }


    public static List<Restaurant> getIndianRestaurantList(String cuisine, List<String> names) {
        RestaurantFactory factory;
        factory = new IndianRestaurantFactory();
        return getRestaurantList(factory, names);
    }
```

```
PS D:\OOAD\FactoryMethod> java RestaurantListFactory
Indian Restaurants
Famin Curry - Indian
Kamal Palace - Indian
Flavor of Punjab - Indian
Himalayan Grill - Indian
----------------------------
Chinese Restaurants
Panda Express - Chinese
JJ Chinese Express - Chinese
Asian Chef - Chinese
----------------------------
Thai Restaurants
Hiccups - Thai
Pad Thai Classic - Thai
Elephant Thai Kitchen - Thai
PS D:\OOAD\FactoryMethod> 
```