Advanced Software Design Coursework - 1

7SENG004C

Name: Devaka Lakpriya Dabare
UoW ID: **w1956508**

December 2022

Table of Content

Introduction **3**

Designing Class Diagram 3

Step 1: Identify the class names 3

Step 2: Identify attributes and methods of each class 3

Park Class 4

Museum Class 6

Step 3: Generalization 7

Implementation **9**

Folder Structure 9

Main Class 9

Sample Test Data 10

Result 11

Introduction

As the coursework describes, a tour company in London wants to build an application to make it easier for people to find out what they can see and do around London. The aim of the part of the system you are to design is to have a list of attractions, and, for each attraction, find out the price of admittance (if any), the opening time, and the closing time. The aim of the part of the system you are to design is to have a list of attractions, and, for each attraction, find out the price of admittance (if any), the opening time, and the closing time.

The attractions available to start with are parks, museums, and theaters. Parks have free admittance, some museums (but not all) have free admittance, and theaters charge. Opening times and closing times depend upon the particular attraction.

Designing Class Diagram

Step 1: Identify the class names

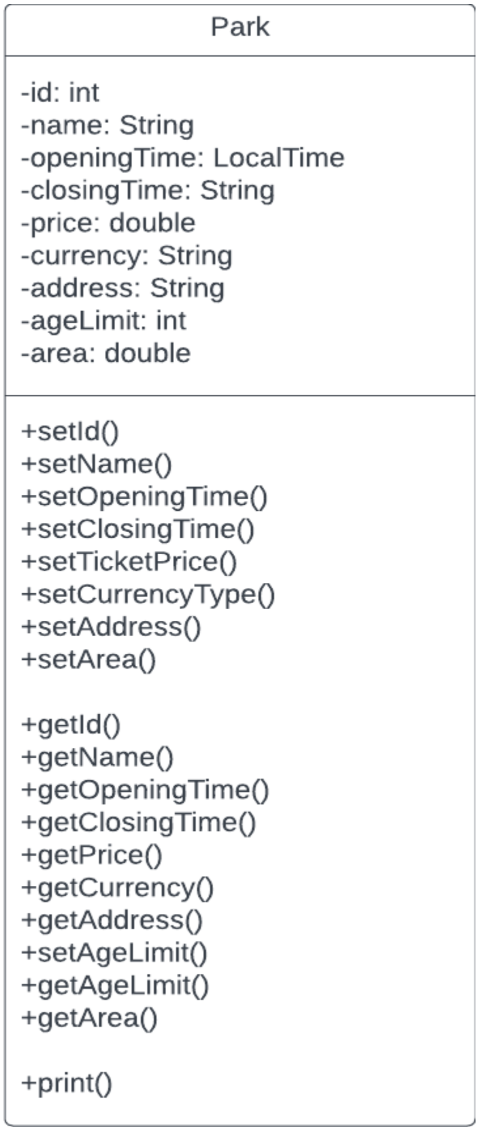
As per the description, I can identify 3 main classes Park, Museum, and Theatre. Each class has its unique as well as some common key attributes and operation.

Step 2: Identify attributes and methods of each class

Then I summarized key attributes and operations below. Some of them I identify from the description and I added some other attributes as well

Park Class

According to the coursework introduction, I identify some attributes and methods that can be in the Park class below



Park

-id: int
-name: String
-openingTime: LocalTime
-closingTime: String
-price: double
-currency: String
-address: String
-ageLimit: int
-area: double

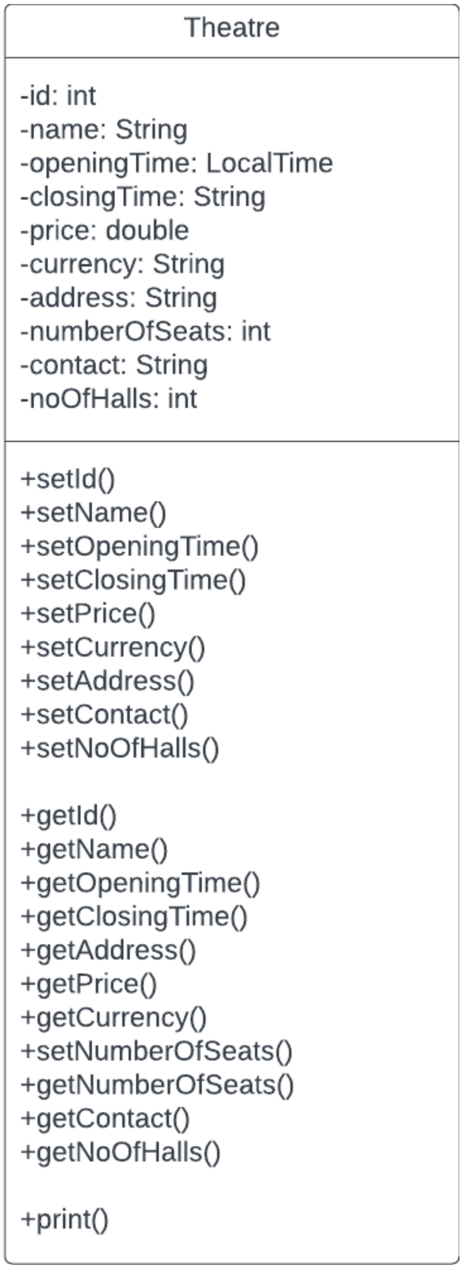
+setId()
+setName()
+setOpeningTime()
+setClosingTime()
+setTicketPrice()
+setCurrencyType()
+setAddress()
+setArea()

+getId()
+getName()
+getOpeningTime()
+getClosingTime()
+getPrice()
+getCurrency()
+getAddress()
+setAgeLimit()
+getAgeLimit()
+getArea()

+print()

Theatre Class

According to the coursework introduction, I identify some attributes and methods that can be in the Theatre class below



Theatre

-id: int
-name: String
-openingTime: LocalTime
-closingTime: String
-price: double
-currency: String
-address: String
-numberOfSeats: int
-contact: String
-noOfHalls: int

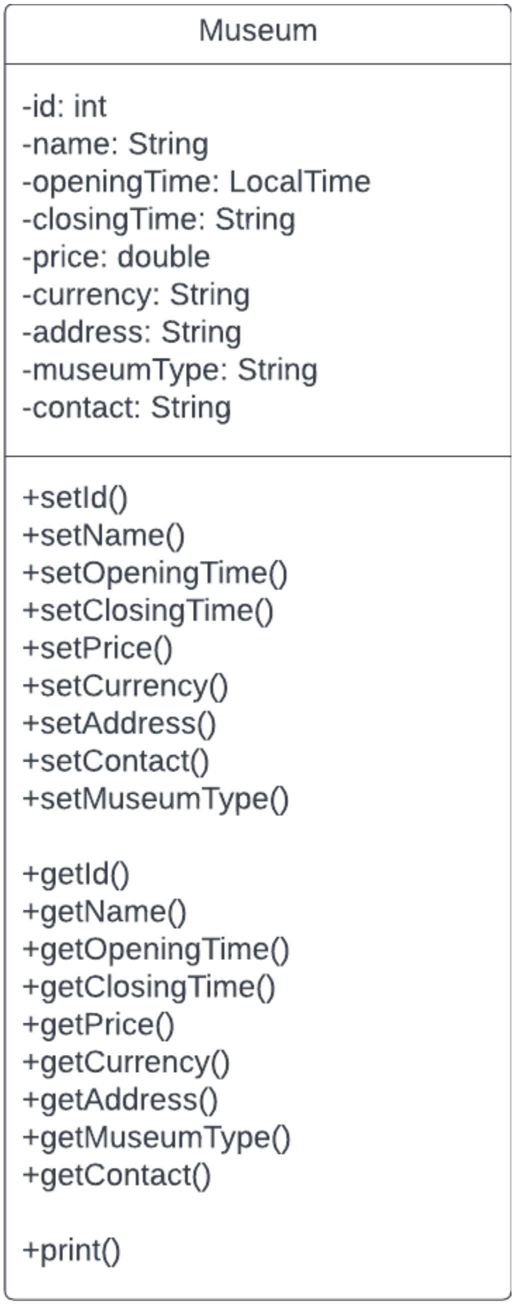
+setId()
+setName()
+setOpeningTime()
+setClosingTime()
+setPrice()
+setCurrency()
+setAddress()
+setContact()
+setNoOfHalls()

+getId()
+getName()
+getOpeningTime()
+getClosingTime()
+getAddress()
+getPrice()
+getCurrency()
+setNumberOfSeats()
+getNumberOfSeats()
+getContact()
+getNoOfHalls()

+print()

Museum Class

According to the coursework introduction, I identify some attributes and methods that can be in the Museum class below



Museum

-id: int
-name: String
-openingTime: LocalTime
-closingTime: String
-price: double
-currency: String
-address: String
-museumType: String
-contact: String

+setId()
+setName()
+setOpeningTime()
+setClosingTime()
+setPrice()
+setCurrency()
+setAddress()
+setContact()
+setMuseumType()

+getId()
+getName()
+getOpeningTime()
+getClosingTime()
+getPrice()
+getCurrency()
+getAddress()
+getMuseumType()
+getContact()

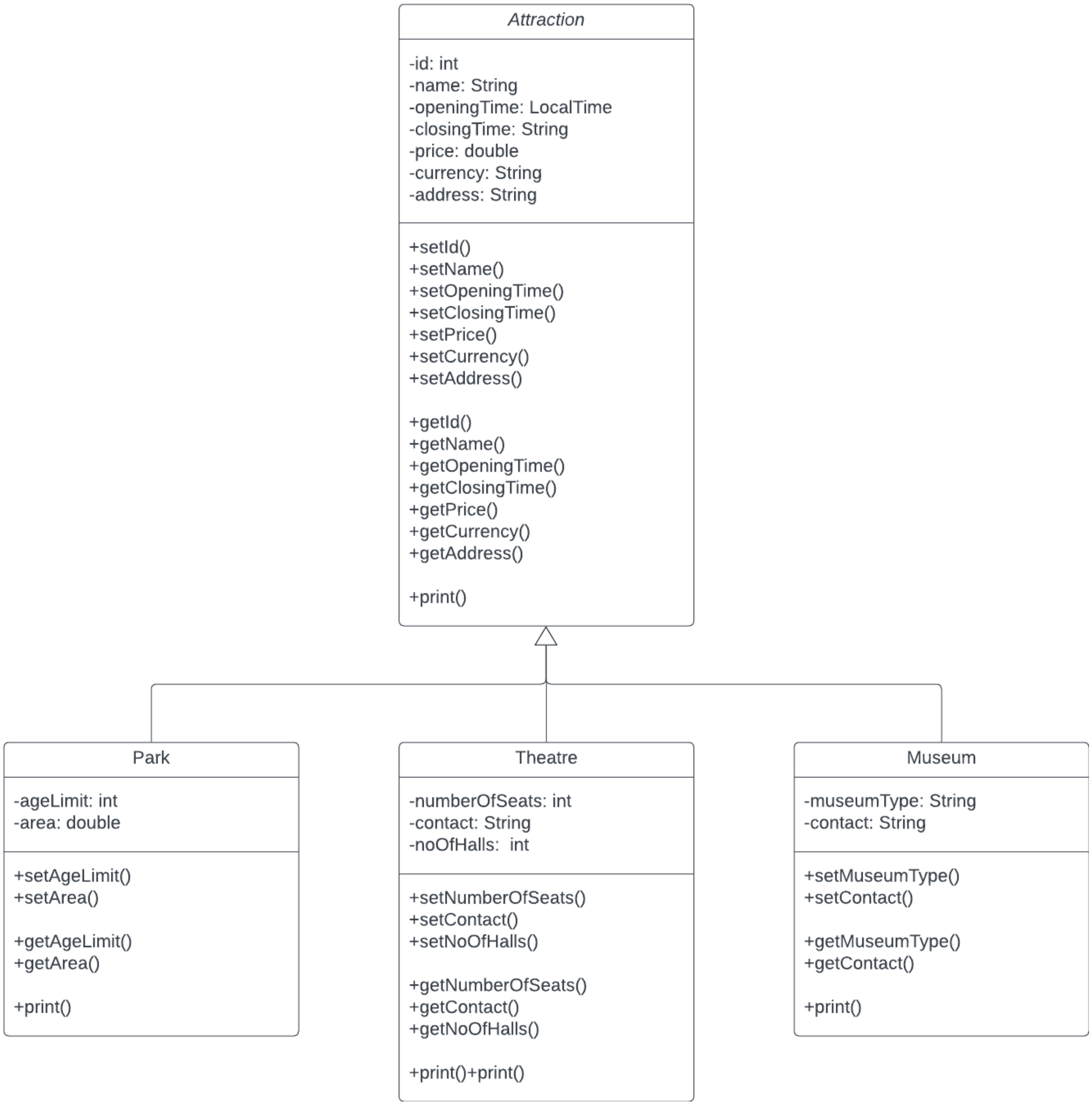
+print()

Step 3: Generalization

As I see, there are some common attributes and methods (such as id, name, openingTime, closingTime, price, currency, and address) in those classes above mentioned. Because of that, I decided to generalize those classes and create a parent class called Attraction which has all the common attributes and operations of other classes.

The attraction class act as a parent class and the other three class as child classes of the Attraction class; all children classes will inherit all common attributes and behaviors from their parent class but could also include additional unique attributes and behaviors specific to that type of attraction as well.

After the generalization, the UML class diagram looks like the below :



Attraction

-id: int
-name: String
-openingTime: LocalTime
-closingTime: String
-price: double
-currency: String
-address: String

+setId()
+setName()
+setOpeningTime()
+setClosingTime()
+setPrice()
+setCurrency()
+setAddress()

+getId()
+getName()
+getOpeningTime()
+getClosingTime()
+getPrice()
+getCurrency()
+getAddress()

+print()

Park

-ageLimit: int
-area: double

+setAgeLimit()
+setArea()

+getAgeLimit()
+getArea()

+print()

Theatre

-numberOfSeats: int
-contact: String
-noOfHalls: int

+setNumberOfSeats()
+setContact()
+setNoOfHalls()

+getNumberOfSeats()
+getContact()
+getNoOfHalls()

+print()+print()

Museum

-museumType: String
-contact: String

+setMuseumType()
+setContact()

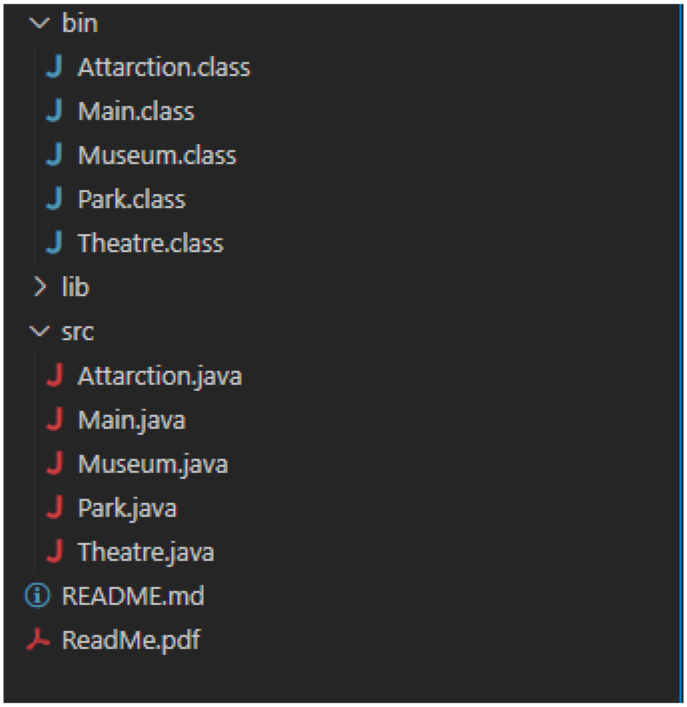
+getMuseumType()
+getContact()

+print()

Implementation

For implementation, I used java as the programming language.

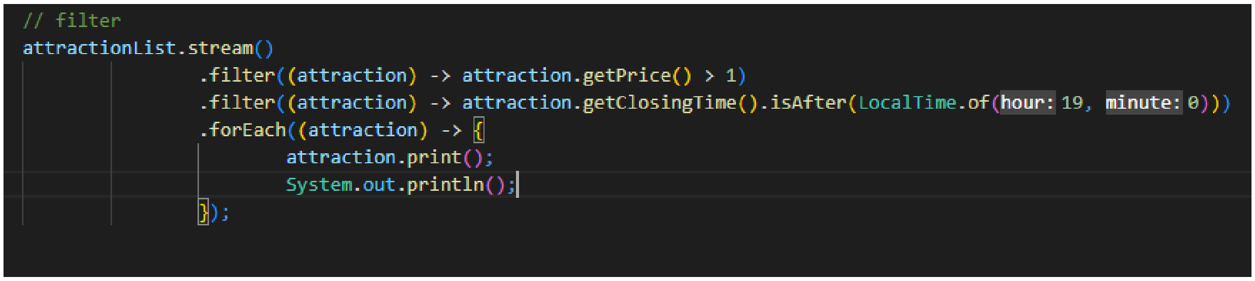
Folder Structure



```
v bin
    Attarction.class
    Main.class
    Museum.class
    Park.class
    Theatre.class
> lib
v src
    Attarction.java
    Main.java
    Museum.java
    Park.java
    Theatre.java
README.md
ReadMe.pdf
```

Main Class

I use the main class to create an example object of attraction and as per the coursework description, filter which attractions are open after 1900 and which cost less than £5 .



```java
// filter
attractionList.stream()
                .filter((attraction) -> attraction.getPrice() > 1)
                .filter((attraction) -> attraction.getClosingTime().isAfter(LocalTime.of(hour: 19, minute: 0)))
                .forEach((attraction) -> {
                        attraction.print();
                        System.out.println();
                });
```

Sample Test Data

I create some sample test data below

```java
Park park = new Park(
    1,
    "Park_1",
    LocalTime.of(10, 0),
    LocalTime.of(18, 30),
    "Park_1_Address",
    18, 0);
Park park2 = new Park(
    2,
    "Park_2",
    LocalTime.of(9, 30),
    LocalTime.of(23, 0),
    "Park_2_Address",
    18, 0);

Theatre theatre = new Theatre(
    3,
    "Theatre_1",
    LocalTime.of(12, 0),
    LocalTime.of(22, 0),
    2.0,
    100,
    "Theatre_1_Address",
    "0770699166",
    2);
Theatre theatre2 = new Theatre(
    4,
    "Theatre_2",
    LocalTime.of(11, 0),
    LocalTime.of(17, 0),
    0,
    100,
    "Theatre_2_Address",
    "0770689166",
    1);
```

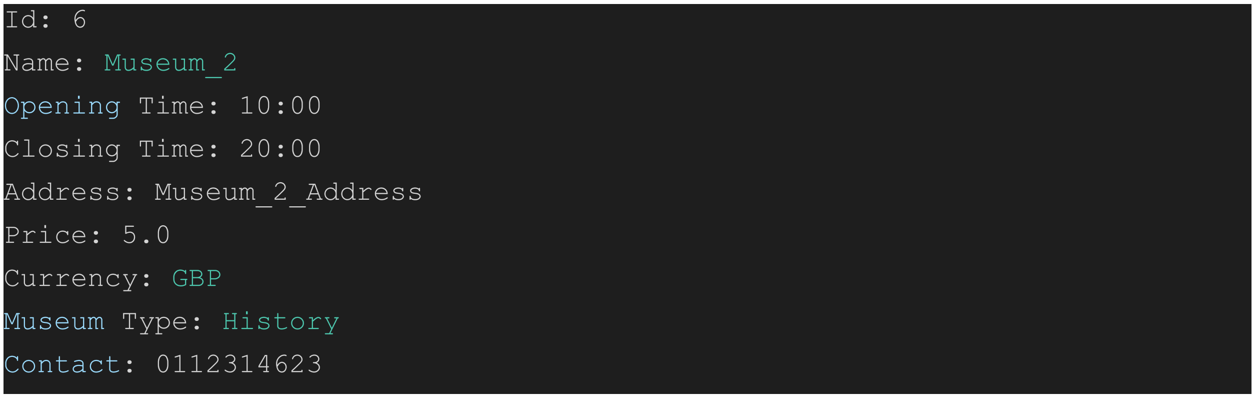
```
Museum museum = new Museum(
    5,
    "Museum_1",
    LocalTime.of(8, 0),
    LocalTime.of(18, 0),
    5.0,
    "Art",
    "Museum_1_Address",
    "0112643542");
Museum museum2 = new Museum(
    6, "Museum_2",
    LocalTime.of(10, 0),
    LocalTime.of(20, 0),
    5.0,
    "History",
    "Museum_2_Address",
    "0112314623");
```

Result

When compiling the java code and running it, it'll filter out attractions that are open after **1900** and which cost less than **£5**.

The result is displayed below:

```
Id: 3
Name: Theatre_1
Opening Time: 12:00
Closing Time: 22:00
Address: Theatre_1_Address
Price: 2.0
Currency: GBP
Number of Seats: 100
Contact: 0770699166
Number of Halls: 2
```



```
Id: 6
Name: Museum_2
Opening Time: 10:00
Closing Time: 20:00
Address: Museum_2_Address
Price: 5.0
Currency: GBP
Museum Type: History
Contact: 0112314623
```