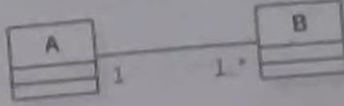


- I. True/False (0.5 pts each, write down your answer on the underline)

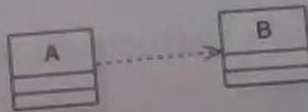
Part I: True or False (0.5 pts each, write down your answer on the underline)

F1 (T/F) To implement the class diagram below in code, a list of type A objects must be placed inside the class B.



```
classDiagram
    class A
    class B
    A "1" -- "1" B
```

F2 (T/F) If the class diagram below has been implemented in code, the following must be true at runtime: When an instance of class A is created, it keeps a reference to class B.



```
classDiagram
    class A
    class B
    A ..> B
```

F3 (T/F) A Class Diagram shows the flow of communication between the running objects of the system, driven by the use cases of the system.

F4 (T/F) Class operations is exclusively devised by sequence diagram.

F5 (T/F) For purpose of code reuse, inheritance is always the ideal approach.

Part II: Multiple Choice(0.5 pts each, write down your answer on the underline)

- II. Multiple choices (0.5pts each)

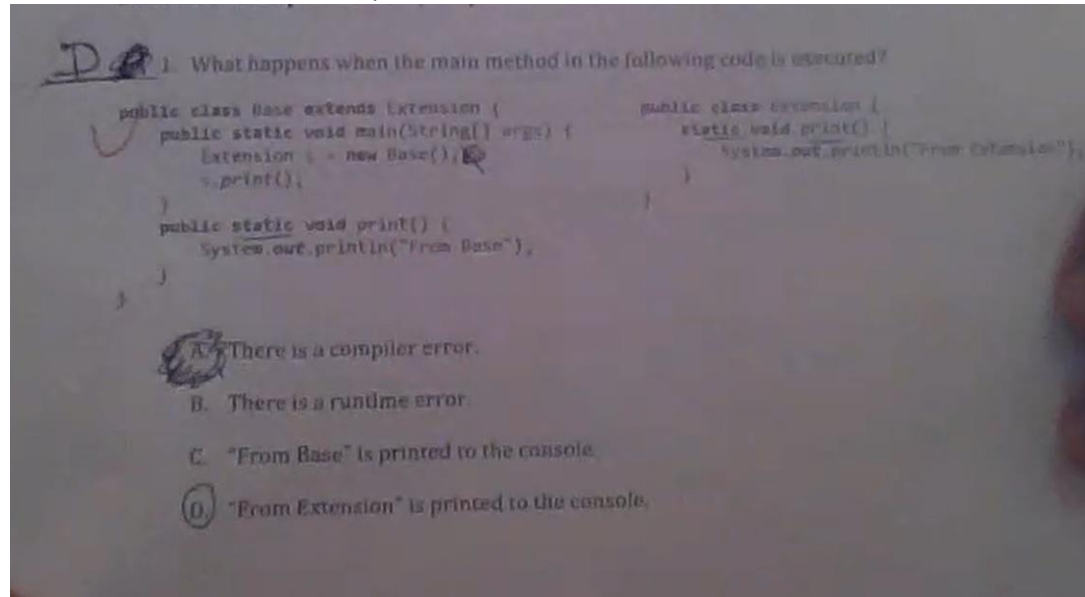
1. What happens when the main method in the following code is executed?

```
Base.java > ...
1 public class Base extends Extension {
2     public static void main(String[] args) {
3         Extension s = new Base();
4         s.print();
5     }
6
7     public static void print() {
8         System.out.println("From Base");
9     }
10 }
11

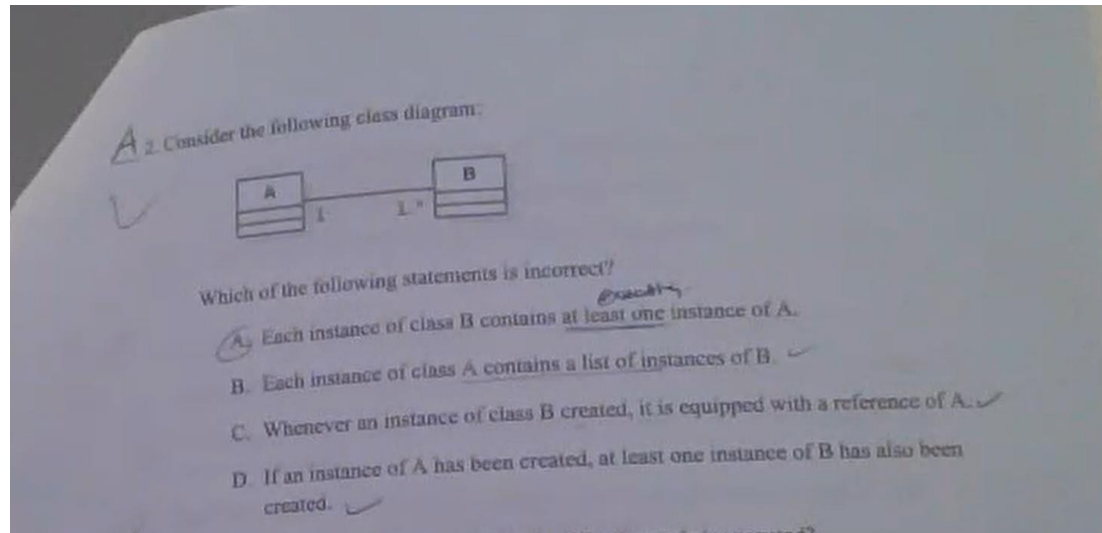
Extension.java > Extension
1 public class Extension {
2     static void print() {
3         System.out.println("From extension");
4     }
5 }
```

- A. There is a compiler error.

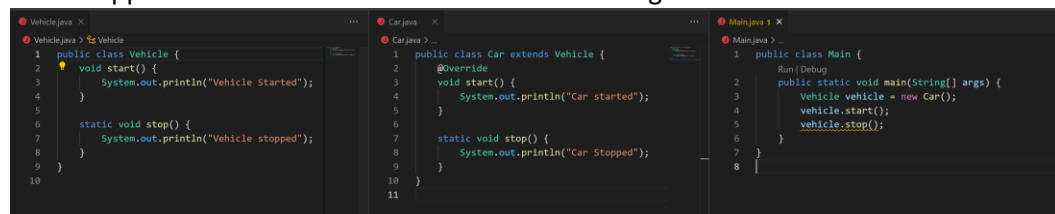
- B. There is a runtime error
- C. "From Base" is printed to the console
- D. "From Extension" is printed to the Console



2.



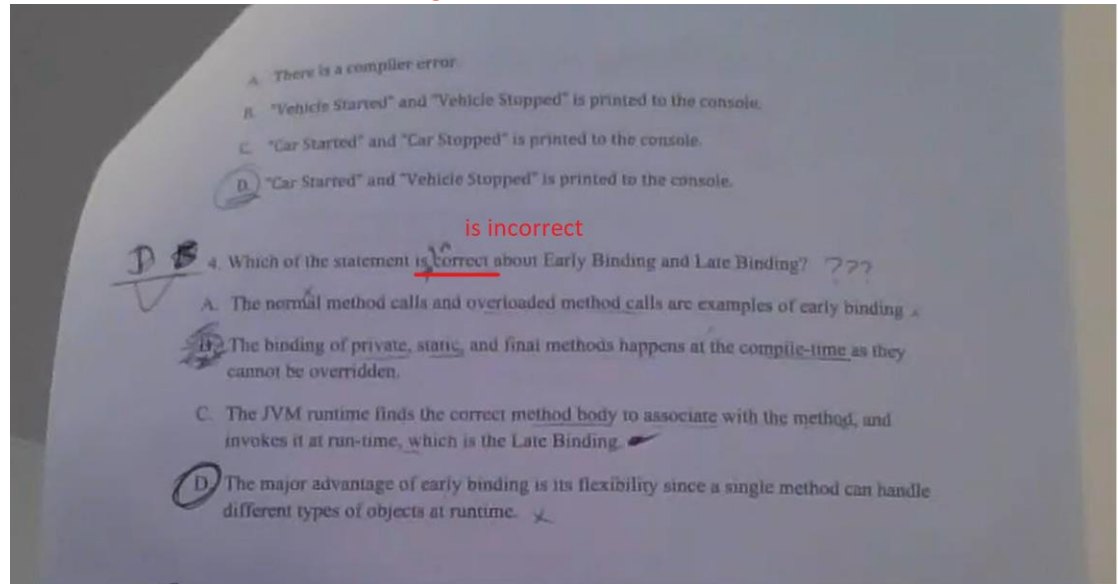
3. What happens when the main method in the following code is executed?



- A. There is a compiler error.

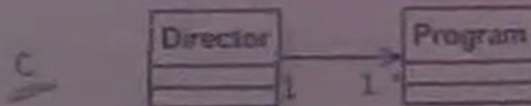
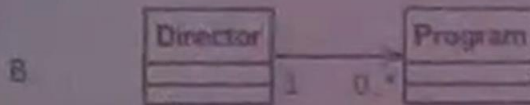
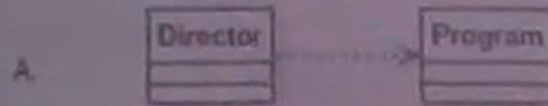
- B. "Vehicle Started" and "Vehicle Stopped" is printed to the console.
- C. "Car started" and "Car stopped" is printed to the console.
- D. "Car started" and "Vehicle Stopped" is printed to the console.

4. Which of the following is **incorrect** about Early and Late Binding? (The question was from is correct to is incorrect during exam)



5. Which of the following UML diagram correctly models the relationship between Director and Program? The code for Director and Program is shown below:

5. Which of the following UML diagrams correctly models the relationship between Director and Program? The code for Director and Program is shown below:



```

public class Director {
    private String name;
    private List<Program> programs = new ArrayList<>();
    public Director(String name, String programId) {
        this.name = name;
        addProgram(new Program(programId));
    }
    public void addProgram(Program p) {
        programs.add(p);
    }
}

public class Program {
    private String programId;
    public Program(String programId) {
        this.programId = programId;
    }
}
  
```

Part III. Short Answers(2 pts each)

```
Director.java 1 X
1 import java.util.ArrayList;
2 import java.util.List;
3
4 public class Director {
5     private String name;
6     private List<Program> programs = new ArrayList<>();
7
8     public Director(String name, String programId) {
9         this.name = name;
10        addProgram(new Program(programId));
11    }
12
13    public void addProgram(Program p) {
14        programs.add(p);
15    }
16 }
17

Program.java 1 X
1 public class Program {
2     private String programId;
3     public Program(String programId) {
4         this.programId = programId;
5     }
6 }
7
```

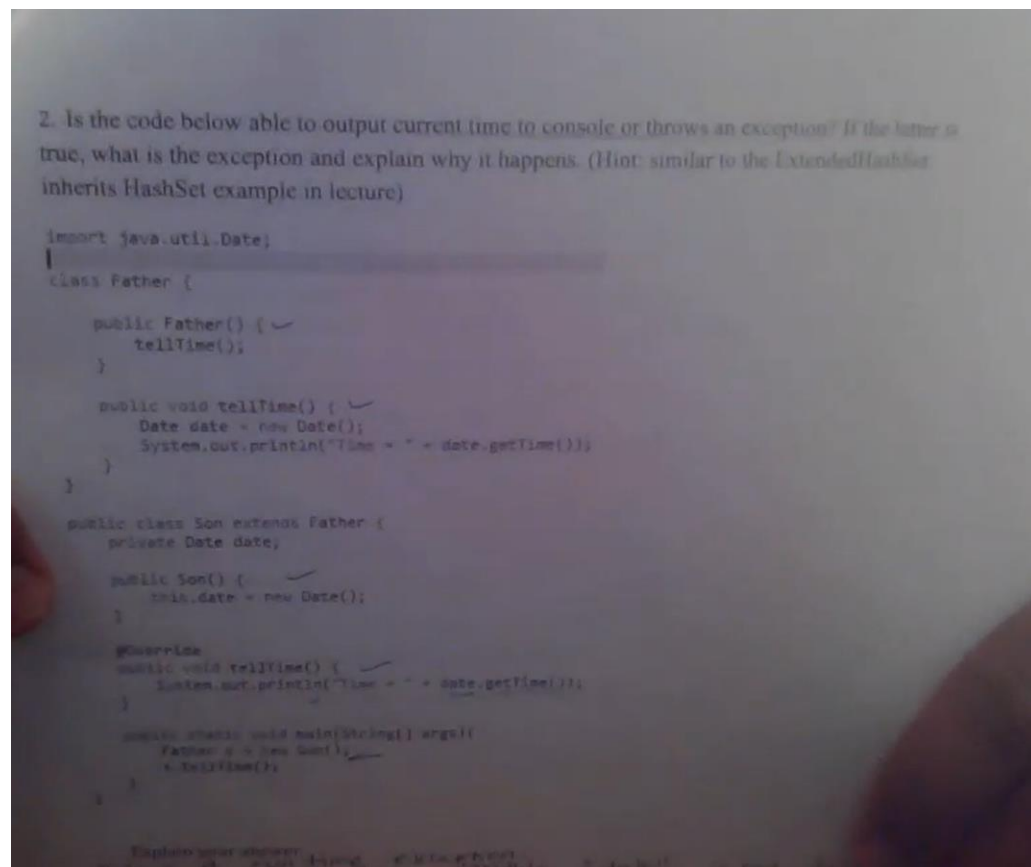
```

Father.java
1 import java.util.Date;
2
3 public class Father {
4     public Father() {
5         tellTime();
6     }
7
8     public void tellTime() {
9         Date date = new Date();
10        System.out.println("Time = " + date.getTime());
11    }
12 }
13
14

Son.java
1 import java.util.Date;
2
3 public class Son extends Father {
4     private Date date;
5
6     public Son() {
7         this.date = new Date();
8     }
9
10    @Override
11    public void tellTime() {
12        System.out.println("Time = " + date.getTime());
13    }
14
15    Run | Debug
16    public static void main(String[] args) {
17        Father s = new Son();
18        s.tellTime();
19    }
20 }

```

Explain your answer.



- Consider the following customer class. It contains instance variables of type Account LocalDate and List<Double> (The Account class is also shown.)

Explain why birthdate and thisYearSalaries should be modelled as attributes?
 Explain why checkingAccount should be modelled as associations?

3. Consider the following Customer class. It contains instance variables of type Account, LocalDate, and List<Double>. (The Account class is also shown.)

```
public class Customer {  
    private Account checkingAccount;  
    private LocalDate birthdate;  
    private List<Double> thisYearsSalaries;  
    public Customer(Account checking, LocalDate bdate,  
                    List<Double> salaries) {  
        this.checkingAccount = checking;  
        this.birthdate = bdate;  
        this.thisYearsSalaries = salaries;  
    }  
}  
  
public class Account {  
    private double balance;  
}
```

If this Customer class is modeled in a class diagram:

- Which instance variables should be modeled as *attributes*?
birthdate, thisYearsSalaries
- Which instance variables should be modeled as *associations*?
checkingAccount

Explain your answer.

IV. Skill Question

- [4pts] In the European Union, the Euro is the official currency. A particular company, [Company name], based in Europe, provides an online shopping application to their European customers. When a user selects a product from the online catalog in purchase, the system reads the quantity requested (an integer) and the productId (a string) from the user interface and creates a Product object, which is passed to the backend for processing. When a product manager for the company decides to add a new product to the system using the same application, he enters simple product information on a screen, specifying only the product name (a String) and the price per unit in Euros (an integer). The system takes the information and creates a Product object.

There is a problem in this process that has puzzled developers. The need for Product objects, as described above, leads to the use of two different Product constructors:

Product(int quantity, String productId)

Product(int price, String productName)

However, in java , it is not possible to have two constructors with the same signature. Show how to solve this problem by replacing use of public constructors with two factory creation methods. (Note, there are other ways to solve this problem, but this test question is asking for factory creation method to solve it.) Write your code in the space provided. You must show all attributes and methods that are part of your solution.

Part IV: Skill Questions

1. [4 pts] In the European Union, the euro is the official currency. A particular company Hana Product, based in Europe, provides an online shopping application to their European customers. When a user selects a product from the online catalog to purchase, the system reads the quantity requested (an integer) and the product id (a string) from the user interface and creates a Product object, which is passed to the back end for processing. When a product manager for the company decides to add a new product to the system using the same application, he enters simple product information on a screen, specifying only the product name (a string) and the price per unit in euros (an integer). The system takes the information and creates a Product object.

There is a problem in this process that has puzzled developers. The needs for Product objects, as described above, leads to the use of two different Product constructors:

```
Product(int quantity, String productId)
Product(int price, String productName)
```

However, in java, it is not possible to have two constructors with the same signature. Show how to solve this problem by replacing use of public constructors with two factory creation methods. (Note, there are other ways to solve this problem, but this test question is asking for factory creation methods to solve it.) Write your code in the space provided. You must show all attributes and methods that are part of your solution.

```
public class Product {
    private int quantityRequested;
    private int price;
    private String name;
    private String productId;
    public Product(int quantity, String productId) {
        this.quantityRequested = quantity;
        this.productId = productId;
    }
    //Adding this constructor creates a compiler error
    // public Product(int price, String productName) {
    //     this.price = price;
    //     this.productName = productName;
    // }
}
```

2. [4pts] In class we discussed how a Square can be considered to satisfy the "IS-A" relationship relative to Rectangle. However, using inherited methods of Rectangle leads to broken code (Code reproduced below.) Rewrite these two classes using composition. You must provide all constructors, getters, and setters. Code from lecture

2. [4 pts] In class we discussed how a Square can be considered to satisfy the "IS-A" relationship relative to Rectangle. However, using inherited methods of Rectangle leads to broken code. (Code reproduced below.) Rewrite these two classes using composition. You must provide all constructors, getters, and setters.

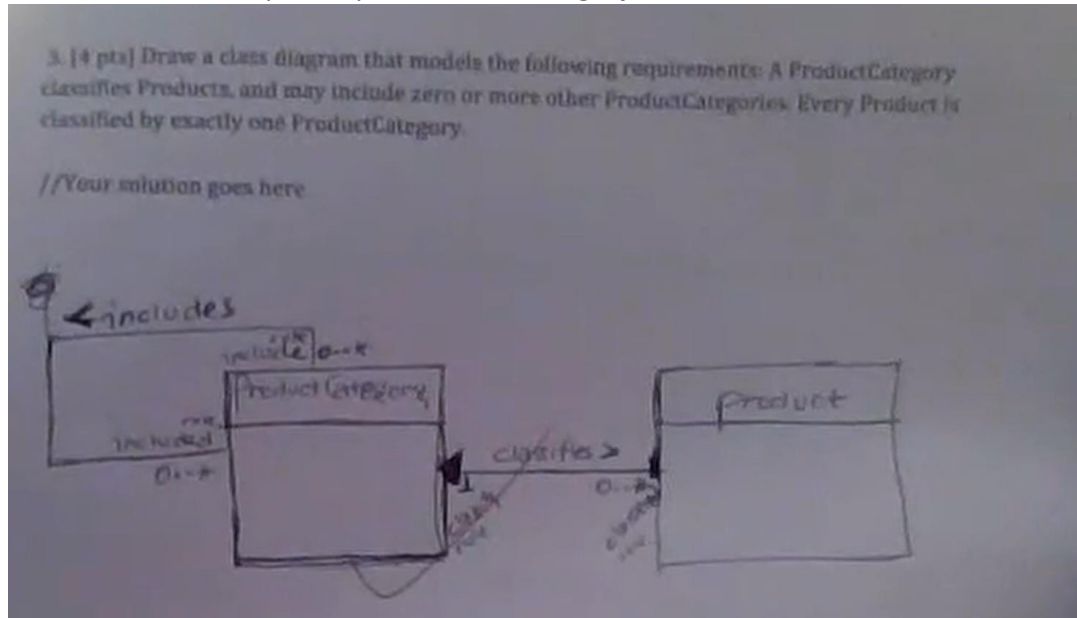
Class Rectangle {

```
private int side1, side2;
public Rectangle(int s1, int s2) {
    side1 = s1;
    side2 = s2;
}
public int getSide1() {
    return side1;
}
public void setSide1(int side1) {
    this.side1 = side1;
}
public int getSide2() {
    return side2;
}
public void setSide2(int side2) {
    this.side2 = side2;
}
public int computeArea() {
    return side1 * side2;
}
}
```

```
public class Square extends Rectangle {
    public Square(int side) {
        super(side, side);
    }
}
```

//You must write your solution in the space provided on the next page

3. [4pts] Draw a class diagram that models the following requirements. A **ProductCategory** classifies Products and may include zero or more other **ProductCategories**. Every Product is classified by exactly one **ProductCategory**.



4. [4pts] A Doubleton class is a class for which it is possible to create at most two instances – in other words, a Doubleton allows you to create 0, 1, or 2 instances, but no more. Use a (single) parametrized factory method to implement a Doubleton class. The parameter that the factory method should accept must be an integer with value 0 or 1, the parameter should be used to specify which of the two instances will be returned. Your class should be designed so that no instances are created until the factory method is called: in particular, after your class has been loaded into memory, the following scenarios should all be possible:
- Exactly two instances are created
 - Exactly one instance is created
 - No instance is created

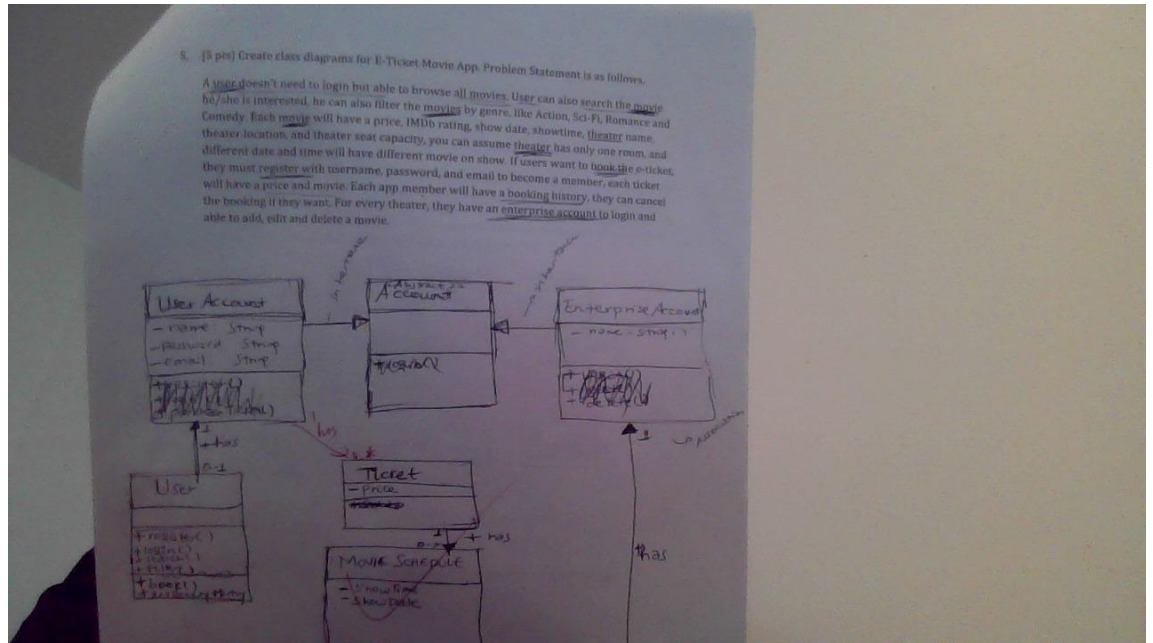
// Your work goes here.

4. [4 pts] A Doubleton class is a class for which it is possible to create at most two instances - in other words, a Doubleton allows you to create 0, 1, or 2 instances, but no more. Use a (single) parametrized factory method to implement a Doubleton class. The parameter that the factory method should accept must be an integer with value 0 or 1; the parameter should be used to specify which of the two instances will be returned. Your class should be designed so that no instances are created until the factory method is called; in particular, after your class has been loaded into memory, the following scenarios should all be possible:
- Exactly two instances are created
 - Exactly one instance is created
 - No instance is created.

//Your work goes here:

```
public class Doubleton {
    private static Doubleton[] instances = new Array(2);
    public static Doubleton getNewDoubleton(int index)
    {
        if (index < 0 || index > 1)
        {
            return null; // Instead of throwing exception,
                        // I chose to create no instance &
                        // return null
        }
        if (index == 0)
        {
            if (instances[0] == null)
            {
                instances[0] = new Doubleton();
            }
            return instances[0];
        }
        if (index == 1)
        {
            if (instances[1] == null)
            {
                instances[1] = new Doubleton();
            }
            return instances[1];
        }
    }
}
```

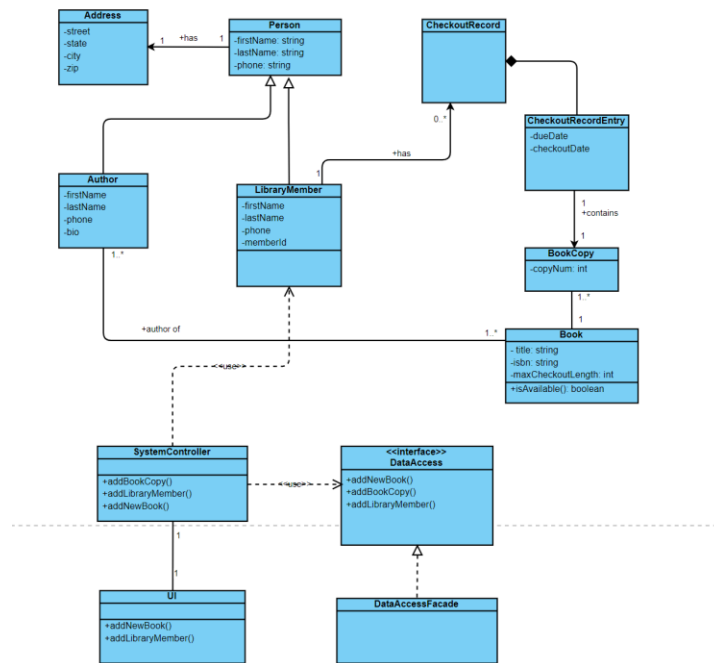
5. [5pts] Create the class diagram for E-Ticket Movie App Problem statement is as follows:
A User doesn't need to login but able to browse all movies. User can also search the movie he/she is interested. He can also filter the movies by genre, like Action, Sci-Fi, Romance and Comedy. Each movie will have a price, IMBb rating, show date, showtime, theater name, theater location, and theater seat capacity, you can assume theater has only one room, and different date and time will have different movie on show. If users want to book the e-ticket, they must register with username, password, and email to become a member, each ticket will have a price and movie. Each app member will have a booking history, they can cancel the booking if they want. For every theater, they have an enterprise account to login and able to add, edit and delete a movie.



6. [8pts – 4 pts each] Below is an updated class diagram for the Library System introduced in class as part of this week's project. (Note that getters and setters for attributes shown in the classes are assumed, but now shown.) One of the use cases for this project is Add New Book. Do the following:

- Create a Use Case description for the main flow and Add New Book and place your work in the table provided(below).s
- Use your use case description, the class diagram, and the knowledge you have about the intended architecture of the Library System to draw a sequence diagram modelling the main flow of Add New Book. All your messages must have names and be numbered, and you must use activation bars correctly. Pay careful attention to the type of association that exists between **Book** and **BookCopy** in the class diagram.

[Some operations are not visible in the photo taken nor I don't exactly remember them but something similar to what is drawn here].



//Your solution to Problem IL6.a. goes here (you may add more rows if you want)

ADD_NEW_BOOK Use Case Description: Main Flow

User Action	System Response
<p>User enters all required book details to the system</p> <p>User clicks on Add new book button in the UI</p>	<p>System checks the user has all proper permissions to add book. Validates the book details provided & presents UI to fill book details required</p>
<p>User enters all required book detail to the system</p>	<p>System validates user input and adds the new book. It also checks if a book copy exists, if it does, increment book copy count.</p>

//Your solution to Problem IL6.b. must be written on the next page