

CS401 MPP - Midterm

Dr. Muhyieddin Al-Tarawneh

Name: _____

Student Id: _____

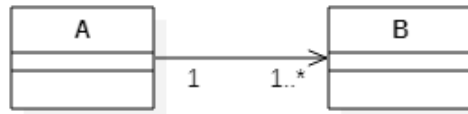
Part I.1 (4)	Part I.2 (6)	Part I.3 (6)	Part II.1 (5)	Part II.2 (5)	Part II.3 (10)	Part II.4 (6)	Part II.5 (8)

Part I: Short Answer

1) Write 'True' or 'False' for the following statements:

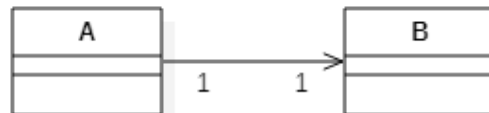
(4 points – each 1)

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



1.2) A Use Case is a sequence of steps performed by a user, interacting with the system, () for the purpose of achieving some goal.

1.3) If the class diagram below has been implemented in code, it would be possible to () navigate from class B to class A.



1.4) If class 'C' was implementing two interfaces, both had the same method name and () same signature, but no implementation in both. No compile errors will occur when class 'C' overrides the method that is in both interfaces with the same signature?

2) Circle the correct answer:

(6 points – each 2)

2.1) Given the following codes, which of the following is correct regarding the relationship between Employer and Gardener? Circle one answer.

```
public class Employer {
    public void employ() {
        Gardener gardener = new Gardener();
        gardener.garden();
    }
}
```

```
public class Gardener {
    public void garden() {
        //do gardening
    }
}
```

- A. There is a dependency from Employer to Gardener
- B. There is a one-way association from Employer to Gardener
- C. There is a two-way association between Employer and Gardener
- D. Not possible to determine from the shown code

2.2) Given the following codes, what would it output when the main method is executed?

```
public class Supreme {
    static void print() {
        System.out.println("Supreme");
    }
}

public class Super extends Supreme {
    static void print() {
        System.out.println("Super");
    }
}
```

```
public class Sub extends Super {
    public static void main(String[] args) {
        Supreme s1 = new Sub();
        Super s2 = new Sub();
        Sub s3 = new Sub();
        s1.print();
        s2.print();
        s3.print();
    }
}
```

- A. Super
Super
Super
- B. Supreme
Supreme
Supreme
- C. Supreme
Super
Super
- D. Supreme
Super
Supreme

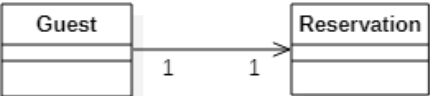
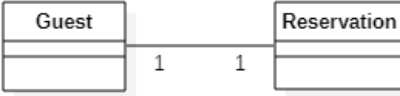
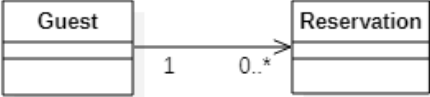
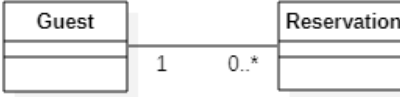
2.3) Given the following codes, which of the following UML diagrams correctly models the relationship between Guest and Reservation classes?

```
public class Guest {
    private String name;
    private List<Reservation> reservations;

    public Guest(String name) {
        this.name = name;
        reservations = new ArrayList<Reservation>();
    }
    public Reservation addReservation(LocalDate dateOfResv) {
        Reservation resv = new Reservation(dateOfResv);
        reservations.add(resv);
        return resv;
    }
    public String getName() {
        return name;
    }
    public List<Reservation> getReservations() {
        return reservations;
    }
}
```

```
public class Reservation {
    private LocalDate reservationDate;

    //package level access
    Reservation(LocalDate resvDate) {
        this.reservationDate = resvDate;
    }
}
```

- A. 
- B. 
- C. 
- D. 

3)

3.1) Given the following codes, write the expected output in the corresponding fields for each Test class, you may write 'compile error' if you assume it is. (4 points – each 1)

```
public class A {
    private int x;

    public A() {
        x = 0;
    }
    public A(int i) {
        x = i;
    }
    public void print() {
        System.out.println("class A");
    }
    public int getValue() {
        return x;
    }
}
```

```
public class B extends A {

    public void print() {
        System.out.println("Class B");
    }
}
```

<pre>public class Test1 { public static void main(String[] args) { B o1 = new B(); int r = o1.getValue(); System.out.print(r); } }</pre>	
<pre>public class Test2 { public static void main(String[] args) { A o2 = new B(); o2.print(); } }</pre>	
<pre>public class Test3 { public static void main(String[] args) { A o3 = new A(3); int k = o3.getValue(); for (int i=0; i<k; i++) { o3.print(); } } }</pre>	
<pre>public class Test4 { public static void main(String[] args) { B o4 = new A(); o4.print(); } }</pre>	

3.2) Regarding the methods of polymorphism as shown in the table below. Fill in the corresponding fields specifying the type of binding whether its Early binding or Late binding. Don't specify other words. (2 points – each 0.5)

Polymorphism method	Answer (Binding Type)
Method Overriding	
Method Overloading	
Private methods	
Final methods	

Part II: Skill Questions

1) Draw a UML class diagram consisting of three classes: *Curriculum*, *Course*, and *Lecture*. Your class diagram should reflect all the following description: (5 Points)

A curriculum contains of a name and consists of one or more courses. A course that has a name and id is composed of zero or more lectures. Every lecture is assigned to its referring course and has a specific day. All the courses of a curriculum can be accessed from the curriculum, but not the other way around. A course may have other courses as prerequisites.

//Draw the diagram in the space below...

2) Draw a sequence diagram for the method `register()` from the code below. (5 Points)

```
public static void main(String[] args) {  
    Response = BookController.Get(id) {  
        book = BookService.Get(id) {  
            book = BookRepository.FindOne(id)  
        }  
        response = createResponse(book)  
    }  
}
```

//Draw the diagram in the space below...

3) A rectangle can be created by specifying its width and length, or by specifying its width and the length of its diagonal. Here is a Rectangle class that provides two constructors that will create a Rectangle object in either of these two ways.

```
public class Rectangle {  
    private double length, width, diagonal;  
    public double computeArea() {  
        return length * width;  
    }  
    public Rectangle(double len, double width) {  
        length = len;  
        this.width = width;  
        diagonal = Math.sqrt(length * length + width * width);  
    }  
    public Rectangle(double width, double diag) {  
        this.width = width;  
        diagonal = diag;  
        length = Math.sqrt(diagonal * diagonal - width * width);  
    }  
}
```

A) As shown, there is a compile error, explain why an error is occurring. (2 points)

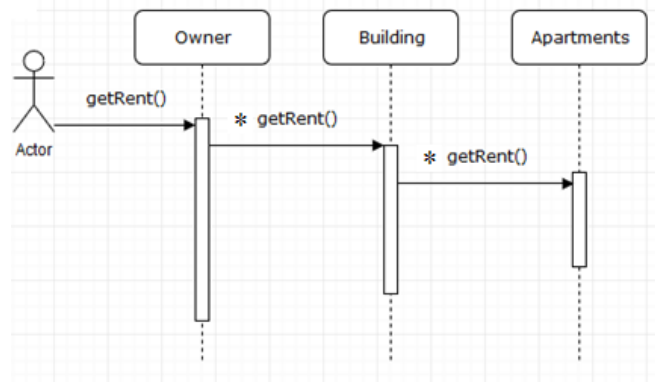
B) If an efficient Factory pattern was implemented to separate the object creation, what SOLID principle/s would it adhere? Explain why. (2 points)

C) In the space provided below, rewrite the code for Rectangle (from Part A) so that it supports the two ways of constructing a rectangle. Use a technique described in the course. (**6 points**)

```
public class Rectangle{
```


4) A system that consists of three classes Owner, Building, and Apartments.

The **Owner** is specified with a name, telephone number, 'total rent', and may own a building or more. Each **Building** is labeled with a building number and has a 'total rent', it consists of one or more apartment. Every **Apartments** has a rent value. Each class has a method `getRent()`, if it was executed from the Owner class it should be delegated in a distributed manner to the Building class then to the Apartments as well to calculate the total rent for a specific owner. Below is a sequence diagram for this problem. Write the code for all three classes. (6 points)



5) A channel service subscription system, you are required to write the code for the mentioned classes in the class diagram below according to the following:

The customer can subscribe to one or more services, and it is possible to remove a service as well. A method in the Customer class **totalFee()** should traverse through all subscribed services for a customer and calculate the total fee. As shown in the class diagram below, there are two types of services:

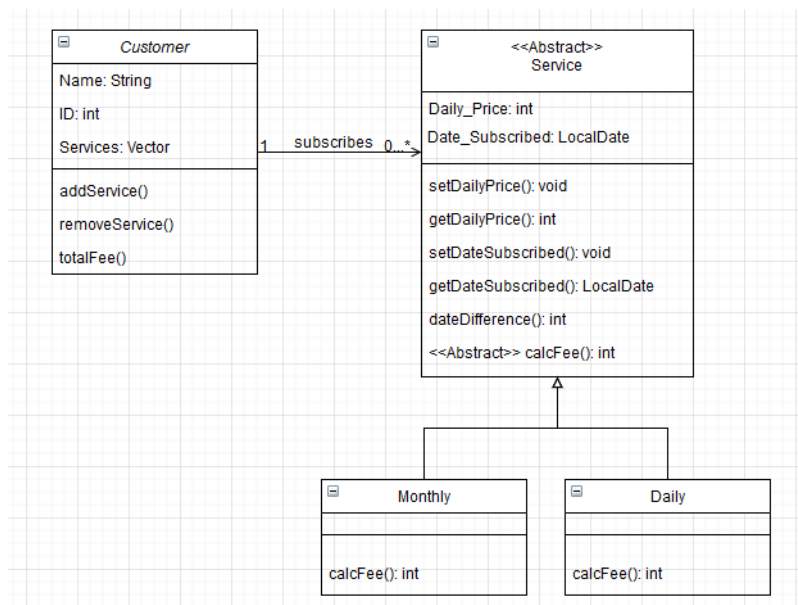
'**Monthly**', which calculates the fee by multiplying the **Daily_Price** by 30. The method **calcFee()** in the **Monthly** class applies the following:

$$\text{Daily_Price} * 30$$

'**Daily**' takes the result value of **dateDifference()** method and multiplies it with the **Daily_Price**. The method **calcFee()** in the **Daily** class applies the following:

$$\text{Daily_Price} * (\text{result value of dateDifference()})$$

The initial code for the Service class is given which contains the implementation of the **dateDifference()** method. This method will return a number based on the difference between the date subscribed and the current date. **(8 points)**



```

import java.time.LocalDate;
import java.time.Period;

public abstract class Service {

    private int Daily_Price;
    private LocalDate Date_Subscribed;

    /* This method will calculate the difference between the current data and the
    date subscribed and return a number of days */
    public int dateDifference() {
        LocalDate now = LocalDate.now();
        LocalDate subDate = Date_Subscribed;
        Period period = Period.between(subDate , now );
        int diff;
        int Month_diff = period.getMonths();           // This gets difference of in months
        int Day_diff = period.getDays();               // This gets difference of in days
        if (Month_diff !=0) diff = ((Month_diff*30)+Day_diff);
        else diff = Day_diff;
        return diff;
    }
}

```


***** Only one bonus question will be graded; you may answer both.**

Bonus Question (A): Go back to question 5, in the Monthly class the `calcFee()` method only calculates for one month, what if the result of the `dateDifference()` was more than 30, then the system should charge for more than a month. Suggest a more efficient implementation that will calculate the Monthly class even if the days exceeded 30. **(2 points)**

- *Note: only write the method for the `calcFee()`*

Bonus Question (B): In an ancient text, one reads the following:

Know that by which all this is known.

1. What does this expression mean? What is it saying? Is it some kind of SCI point?
2. Does this expression illuminate any aspect of the software engineering discipline discussed in class? Explain.

(2 points)