CS401 MPP Midterm - Solutions

Corazza

Name:_____ StudentId:_____

Part I, 1-7 (21)	Part II, 1 (8)	Part II, 2 (8)	Part II, 3 (8)	Part II, 4 (16)	Part III SCI (3)

Part I: Theory (3 points each – 21 points total)

1. Name four kinds of UML diagrams and explain briefly what each is used for.

   Class diagram – models the static structure of the system, the classes and their static relationships

   Class diagrams basically represent the object oriented view of a system which is static in nature.

Sequence diagram – models the dynamic relationships between instances of participating classes – the flow of messages in the system

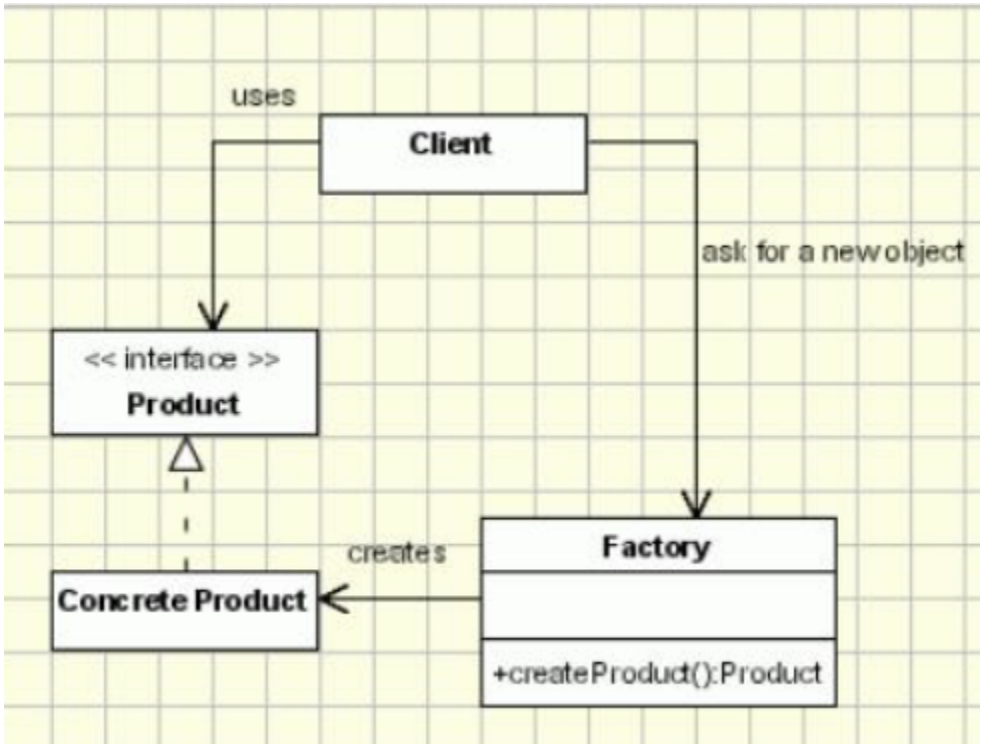
Sequence diagram is used to visualize the sequence of calls in a system to perform a specific functionality

Object diagram – models a snapshot or a state of the system at a particular time, showing the relationships and state of every object

Other diagrams mentioned in the slides

Collaboration Diagram
Activity Diagram
Use Case Diagram
Component Diagram
Deployment Diagram

2. Draw the Object Creation Factory UML diagram



3. Give an example of two classes and an association between them that is *not* a composition relationship.

   CheckoutRecord and Book

   Customer and Account

   Duck and FlyBehavior

4. Explain the difference between *analysis* and *design* in the software development lifecycle.

   analysis is concerned with understanding the problem domain, the current system, and the problem(s) to be solved – it is concerned with *understanding* and assessing *what is needed*

   design is concerned with *how* to build a solution, how to fit pieces together

5. Name two differences between interfaces and abstract classes in Java (restricted to Java SE 7 and earlier).

    - interfaces cannot have implemented methods
    - interfaces cannot have static methods --only complete abstract can be static
    - interfaces cannot have instance variables
    - a class can implement multiple interfaces --- abstract does not support multiple interface

6. What is the Evolving API Problem?

    The Evolving API Problem is the problem that it is not safe to add new methods to an interface in a system that has already been released, because the presence of new methods will break existing implementations

7. Triangles and rectangles are examples of *polygons.* A polygon is obtained by joining together one or more line segments. The *length* of a polygon is the sum of the lengths of the line segments it is built from. For example, a triangle whose side lengths are 3, 4, 7 has *length* 3 + 4 + 7 = 14.

    Use this general computational method for computing length, together with the template design pattern, to complete the code below so that it is possible to polymorphically compute the lengths of `Triangles` and `Rectangles` and any other kind of polygon. You may add code to any of the classes shown. (Hint: Do *not* override the `computeLength()` method. Use the template pattern instead.)

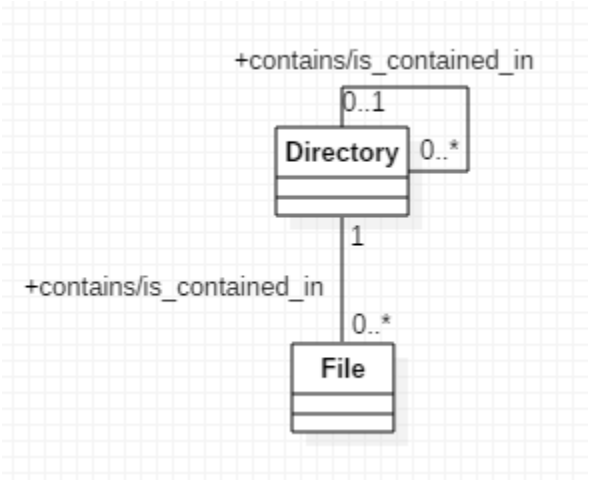
```java
public abstract class Polygon {
    public double computeLength() {
        double sum = 0;
        for(double d : getLengths()) {
            sum += d;
        }
        return sum;
    }
    abstract public double[] getLengths();
}
```

```java
public class Rectangle extends Polygon {
    double length, width;
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    @Override
    public double[] getLengths() {
        return new double[]{length, length, width, width};
    }
}
```

```java
public class Triangle extends Polygon {
    private double side1, side2, side3;
    public Triangle(double side1, double side2, double side3) {
        this.side1 = side1;
        this.side2 = side2;
        this.side3 = side3;
    }
    @Override
    public double[] getLengths() {
        return new double[]{side1, side2, side3};
    }
}
```

Part II: Applications (40 points)

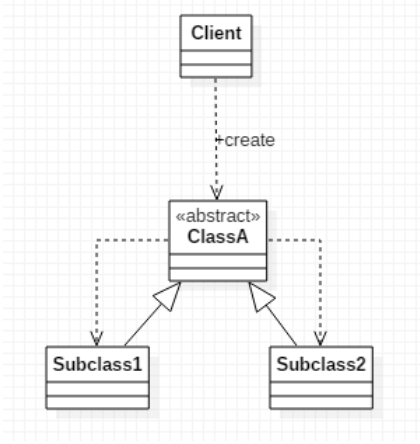
1. [8 points] In a Windows file system, a directory may contain files and other directories. Represent these concepts with classes Directory and File in a class diagram and show association relationships. You do not need to specify attributes or operations in these classes. Associations should be provided with multiplicities and names. NOTE: If A is a subdirectory of B and B is a subdirectory of C, it is <u>not</u> true that A is a subdirectory of C.



2. [8 points] Suppose `Subclass1` and `Subclass2` are both subclasses of the abstract class `ClassA`. All three classes live in `packageA`. `ClassA` is `public` but `Subclass1` and `Subclass2` both have only package level accessibility. Instances of a class `Client`, which lives in `packageB`, need to get an instance of `Subclass1` and `Subclass2` at different times. At any given time, the `Client` class knows which of these subclasses it needs, and it can indicate this by passing in one of the `String`s "1", "2"; if `Client` passes in "1", it should get back an instance of `Subclass1`; if "2", an instance of `Subclass2`.

    a. Create an optimal design to meet these requirements and represent your design in a class diagram (consisting of 4 classes – do not add extra classes or interfaces). Be sure to specify an appropriate name

for an association that exists between `Client` and any other class.



b. Implement your diagram in Java. Be sure to observe the package names and access levels described in the requirements. Set up your code so that `Client` initiates action from a `main` method. Within the `main` method, get an instance of `Subclass1` and, with a second call, get an instance of `Subclass2`.

```java
package midterm.subclassprob.packageA;
public abstract class ClassA {
    public static ClassA create(String s) {
        return s.equals("1") ? new Subclass1() : new Subclass2();
    }
}
```

```java
package midterm.subclassprob.packageA;
class Subclass1 extends ClassA {
    Subclass1() {
        System.out.println("Constructing an instance of Subclass1");
    }
}
```

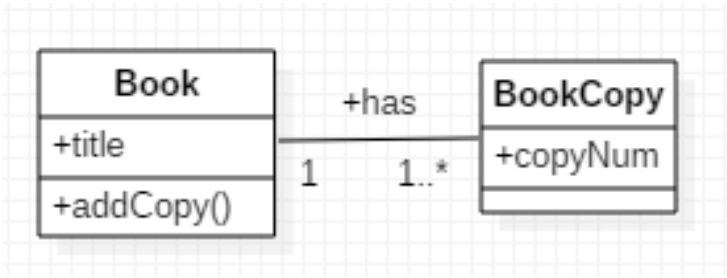
```java
package midterm.subclassprob.packageA;
class Subclass2 extends ClassA {
    Subclass2() {
        System.out.println("Constructing an instance of Subclass2");
    }
}
```

```java
package midterm.subclassprob.packageB;

public class Client {
    @SuppressWarnings("unused")
    public static void main(String[] args) {
        ClassA cl1 = ClassA.create("1");
        ClassA cl2 = ClassA.create("2");

    }
}
```

3. [8 points] In the design of a particular Library System application, the following portion of a class diagram shows the relationship between a `Book` and a `BookCopy`:



Write Java code to implement this part of the class diagram. Assume the constructor of `Book` accepts arguments title and `numCopies`, and assume the constructor of `BookCopy` accepts arguments `Book` and `copyNum`. Assume that copy numbers are consecutive, starting at 1; for instance, if a `Book` has 3 copies, the copy numbers are 1, 2, 3.

All attributes and operations shown in the diagram should be implemented in your Java code.

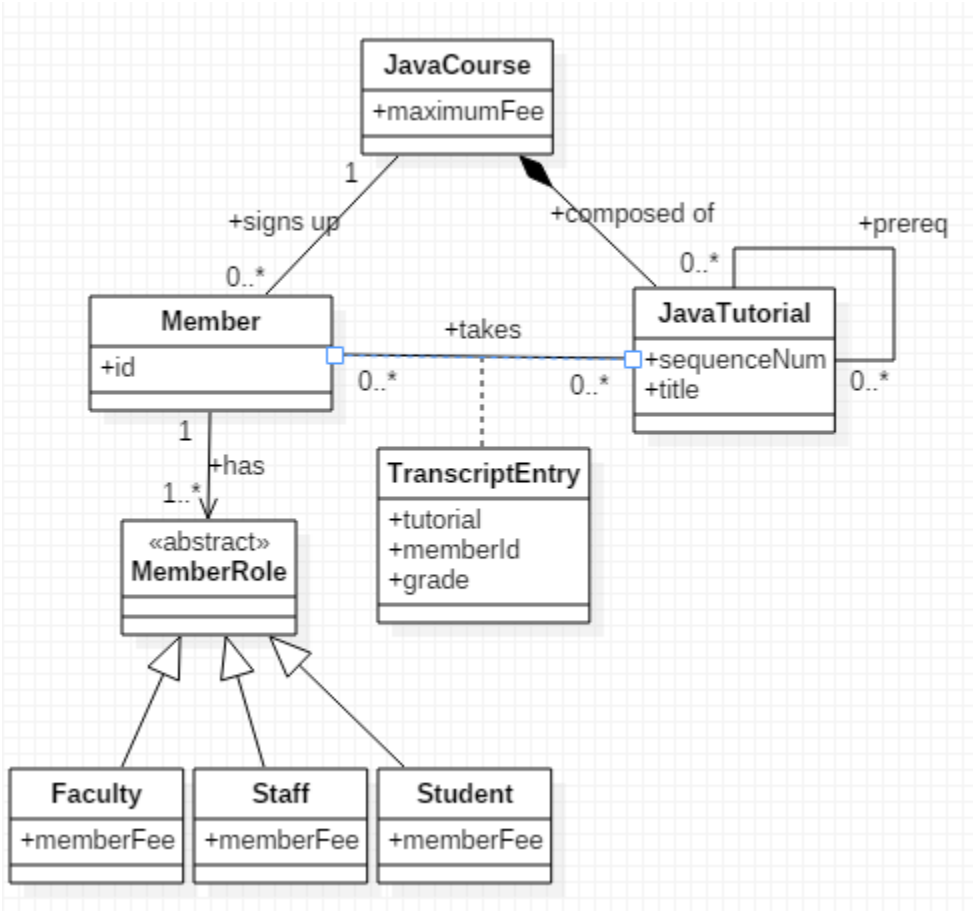
```
public class Book {
    private List<BookCopy> copies = new ArrayList<BookCopy>();
    private String title;
    public Book(String title, int numCopies) {
        if(numCopies < 1) throw new IllegalArgumentException(
                "NumCopies must be positive");
        this.title= title;
        for(int i = 0; i < numCopies; ++i) {
            addCopy();
        }
    }
    public void addCopy() {
        BookCopy copy = new BookCopy(this, copies.size() + 1);
        copies.add(copy);
    }
}
```

```
public class BookCopy {
    private Book book;
    private int copyNum;
    public BookCopy(Book book, int copyNum) {
        this.book = book;
        this.copyNum = copyNum;
    }
}
```

4. [16 points] Problem Statement: We wish to create an online MUM Java Course which is a series of Java tutorials. We allow staff, faculty, and students to become members of our Java Course. A member could be both on staff and a student, or change from staff to faculty or from a student to staff, etc. Students will pay $50 to become a member, staff will pay $25, and faculty can become a member for free. In the case of a member being a combination of types, they get the lowest price available from their combination. For example, a member who is both student and staff will pay $25. Members must complete the Java tutorials for our course in proper sequence: They can take a tutorial only if they have completed the preceding tutorials in the course. They pay their membership fee

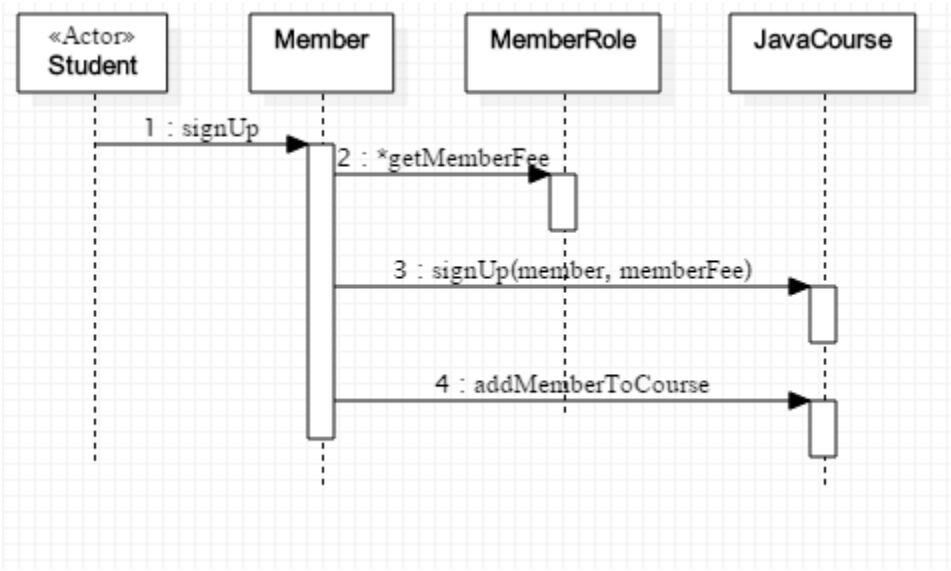
when signing up for our course.

a. Create a UML class diagram for the problem description. Show the major class attributes and methods, and label your associations. Make a note of any design assumptions you make.

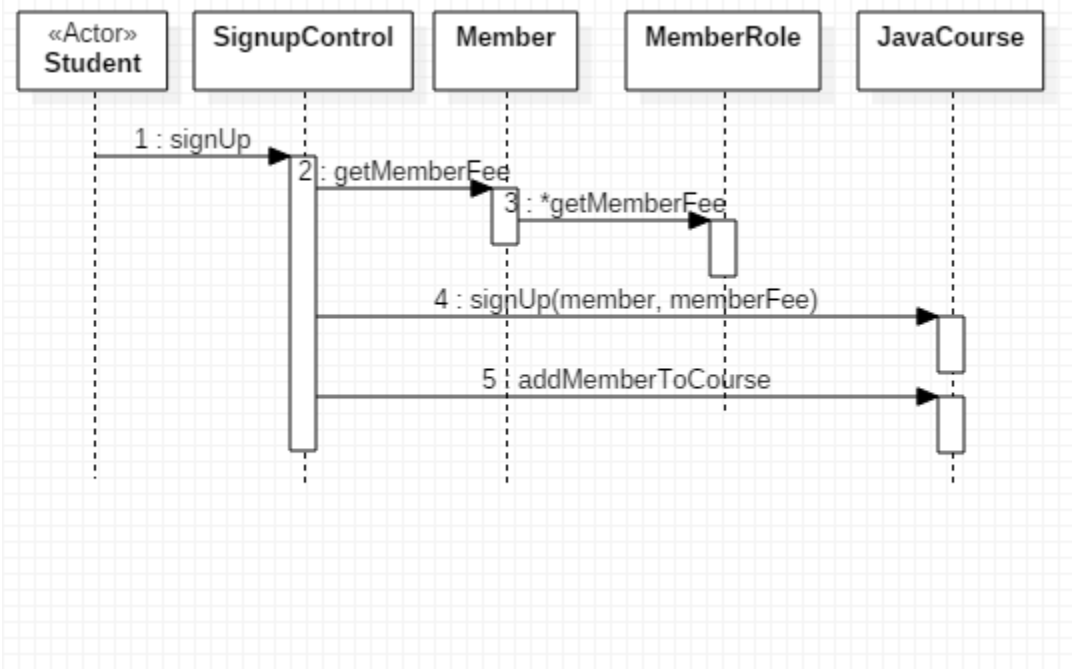


b. Create a sequence diagram for the use case of a member signing up for our Java Course.

This is acceptable:



This is also acceptable:



Part III: SCI – 3 points

Write one or two paragraphs relating a point from the course to a principle from SCI.