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)**

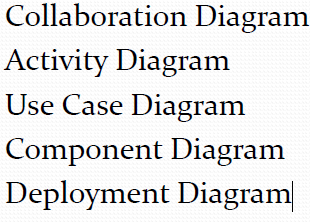
1. Name four kinds of UML diagrams and explain briefly what each is used for.

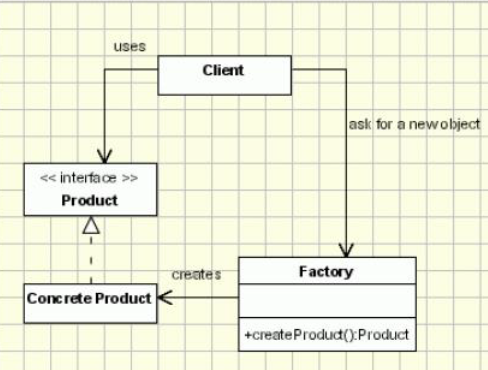
Class diagram – models the static structure of the system, the classes and their static relationships

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

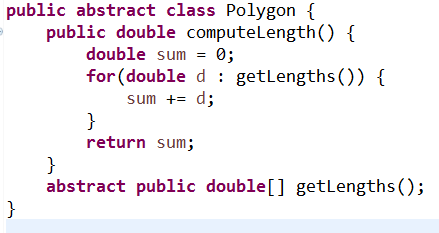
Sequence diagram – models the dynamic relationships between instances of participating classes – the flow of messages in the system

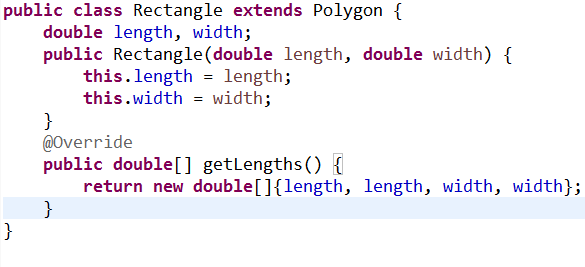
**Sequence diagram is used to visualize the sequence of calls in a system to perform a specific functionality**

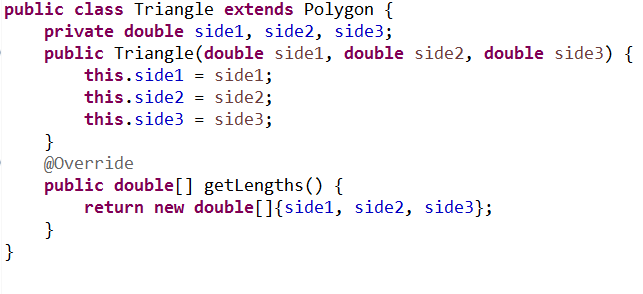
Object diagram – models a snapshot or a state of the system at a particular time, showing the relationships and state of every object  
  
Other diagrams mentioned in the slides  


1. Draw the Object Creation Factory UML diagram   
     
   
2. 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
3. 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
4. Name two differences between interfaces and abstract classes in Java (restricted to Java SE 7 and earlier).

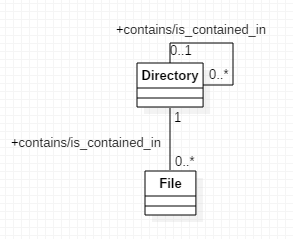
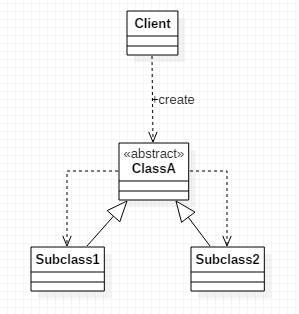
* interfaces cannot have implemented methods
* interfaces cannot have static methods --only complete abstract can be static
* interfaces cannot have instance variables
* a class can implement multiple interfaces --- abstract does not support multiple interface

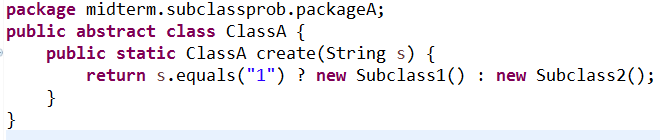
1. What is the Evolving API Problem?  
     
   The Evolving API Problem is the problem that it is not safe to add new methods to an interface in a system that has already been released, because the presence of new methods will break existing implementations
2. Triangles and rectangles are examples of *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.)  
     
   

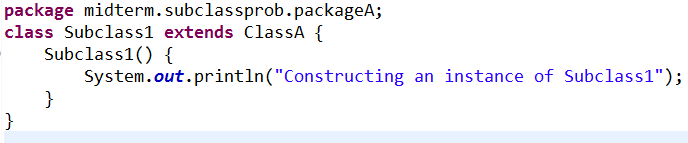


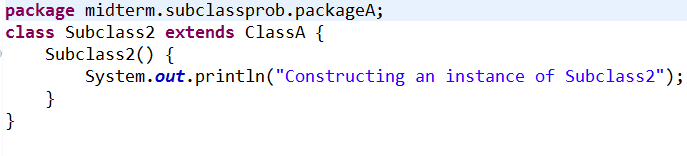


**Part II: Applications (40 points)**

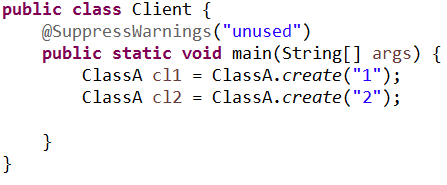
1. [8 points] In a Windows file system, a directory may contain files and other directories. Represent these concepts with classes Directory and File in a class diagram and show association relationships. You do not need to specify attributes or operations in these classes. Associations should be provided with multiplicities and names. NOTE: If A is a subdirectory of B and B is a subdirectory of C, it is not 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 Strings “1”, “2”; if Client passes in “1”, it should get back an instance of Subclass1; if “2”, an instance of Subclass2.
   1. 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.   
      
   2. 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.

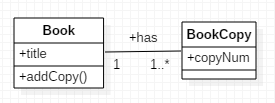




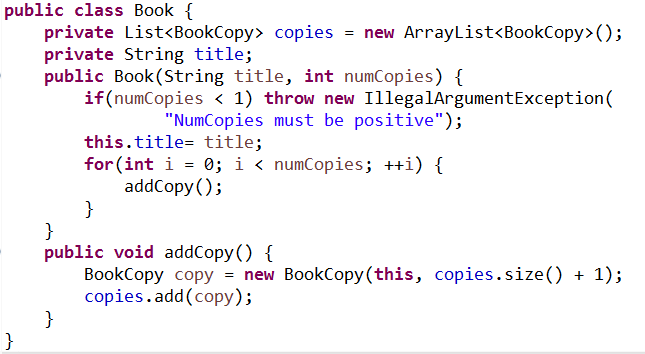


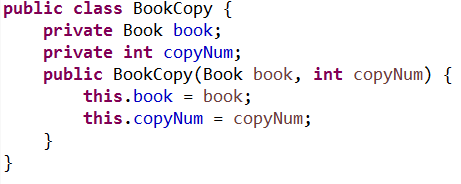


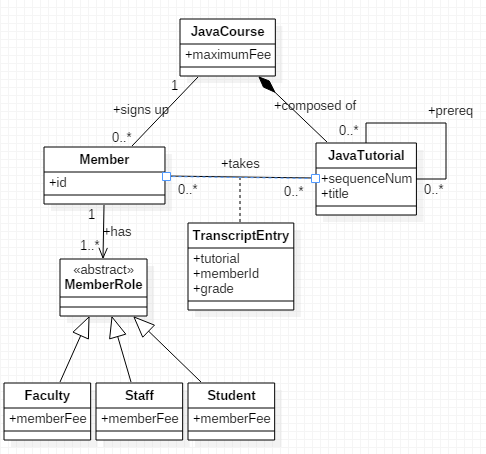
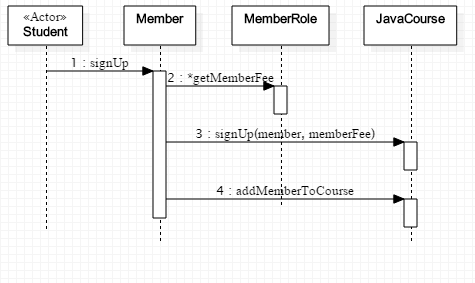


1. [8 points] In the design of a particular Library System application, the following portion of a class diagram shows the relationship between a Book and a BookCopy:  
   

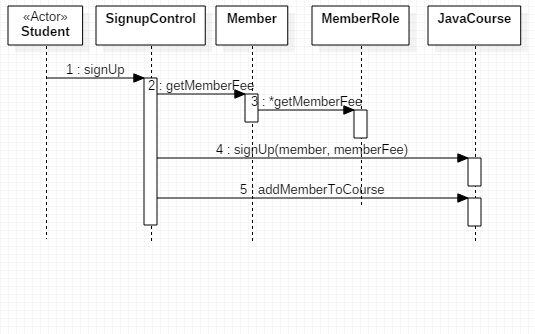
Write Java code to implement this part of the class diagram. Assume the constructor of Book accepts arguments title and numCopies, and assume the constructor of BookCopy accepts arguments Book and copyNum. Assume that copy numbers are consecutive, starting at 1; for instance, if a Book has 3 copies,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





1. [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.  
   1. 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.  
      
   2. 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.