MPP Midterm Review

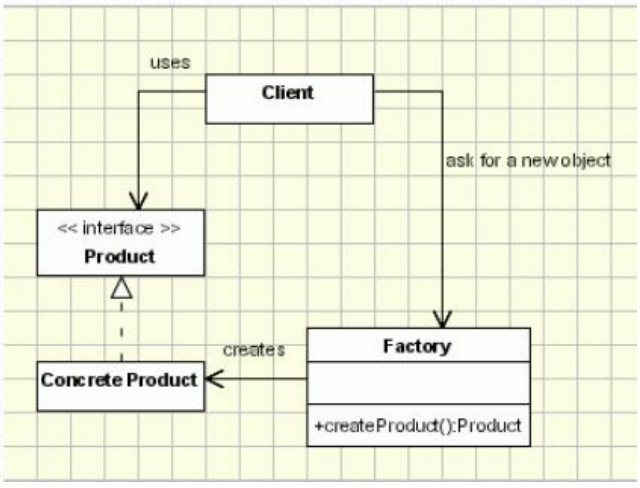
Corazza

Structure of the Exam

- 35% theory – true/false, multiple choice, short answer
- 30% UML – class, sequence, object diagram(s) based on a problem statement
- 35% code – converting UML to code and related problems (like converting inheritance to composition)
- JavaFX will not be covered

Review Points

1. Topics to know
    a. UML diagrams, particularly class and sequence diagrams
    b. OO principles and their application in solving software problems
    c. Be able to translate a UML class diagram into Java code
    d. Be able to convert an inheritance relationship into a composition relationship in Java
    e. Be familiar with the order of execution when a subclass (in an inheritance relationship) is instantiated.
    f. Know how to use polymorphism to solve programming problems
    g. Know the syntax rules and best practices for creating class, sequence, and object diagrams
    h. Know the Object Creation Factory UML diagram and be able to explain it



    i. Be able to implement polymorphism using interfaces (instead of a superclass)
2. UML
    a. Name 4 kinds of UML diagrams and describe what they are used for
    b. Name 4 types or variants of the concept of an association
    c. Difference between association and dependency
    d. Differences between association, aggregation, and composition

    e. What is an association class? What is it used for?
3. OO Paradigm
    a. Name 4 OO principles
    b. What are some differences between the OO approach to programming and procedural programming?
    c. What is the difference between analysis and design?
    d. What are propagation and delegation?
    e. What is the Open-Closed Principle?
    f. What is late binding? Which qualifiers can be used to force early (static) binding?
    g. What is the "diamond problem" that arises in a language in which multiple inheritance is supported?
4. Composition vs Inheritance
    a. Name two benefits of using inheritance
    b. Name two criteria that should be used to check whether inheritance is appropriate
    c. Name one potential problem that can arise in using inheritance
    d. Describe a situation in which using composition solves a problem that arises in using inheritance
    e. Discuss Bloch's principle: *Either design for inheritance or prevent it*.
    f. Give an example to show how composition can be used with inheritance to introduce more flexibility
5. Interaction diagrams
    a. Name 3 kinds of interaction diagrams
    b. What is a lifeline in a sequence diagram? An activation bar? What is the difference?
    c. What is the difference between "centralized control" and "distributed control" in a sequence diagram? When should you use each?
    d. What benefit does an object diagram provide that is less available from a sequence diagram?
6. Abstract classes vs Interfaces
    a. What are the differences between abstract classes and interfaces (in java 7)?
    b. What are the similarities?
    c. Give an example from the Java libraries to illustrate how interfaces and abstract classes can be used together effectively
    d. Name some advantages of using interfaces
    e. What is the principle "Program to the Interface"? What are some reasons to follow this principle?
    f. What is the Evolving API Problem?
7. Object Creation Factory pattern
    a. Give an example from the Java libraries where this pattern has been used
    b. What are some benefits of using a factory method for object construction rather than a constructor?
    c. What are some potential disadvantages?
    d. What is a *parametrized* factory method? Give an example.