MPP Midterm Review Points

The midterm will consist of questions below:

1. True/False – **10 questions** – 1 point for each question – 10 points total – Questions based on Lesson 1~5.
2. Multiple Choices – **5 questions** – 2 points for each question - 10 points total – Questions based on Lesson 1~5.
3. Programming questions – **2 questions** – 10 points for each question - 20 points total
    a. Turn Class Diagram into Java Code. See example Lab 2.
    b. Question related to Object Creation Factory
4. Design questions - **2 questions** – 15 points/20 points for each question – 35 points total
    a. (15 points) Sequence Diagram – Give you description, draw sequence diagram based on description
    b. (20 points) Class diagram – Give you description, draw class diagram based on description.

The total exam is 75 points total with **19 questions in total**.

Lessons 1 and 2

1. UML – Unified Modeling language – not a programming language, with lots of notations and symbols
2. Discovering classes from a problem statement
3. Difference between analysis and design
4. Differences between association,dependency, inheritance
5. 3 types of relationships
6. Associations, dependencies in code
7. Difference between one-way, two-way associatons
8. Properties as attributes, properties as associations
9. Association adornments: name, role, multiplicity,
10. Association matrix
11. Significance of different multiplicities in code
12. Aggregation vs composition vs association
13. Reflexive associations

Skills:

- Create a class diagram with attributes, operations, associations, based on a problem statement
- Translate a class diagram into Java code(Association, aggregation, composition, multiplicity) See Lesson 2 homework as example. For Composition, you have to master the strict version which is not allowed to create “Part” Class outside the “Whole” Class.

Lesson 3

14. Good uses of inheritance vs bad uses
15. Inheritance rules
    a. Rules for inheriting/overriding static methods
    b. Order of execution
16. IS-A and LSP principles
17. Bad Stack example
18. Benefits of inheritance
19. Rectangle-square problem
20. EnhancedHashSet problem – inheritance violates encapsulation
21. Principle: Design for inheritance or prevent it
22. How to replace inheritance by composition (Employment/Manager equals method, Stack class), or supplement inheritance with composition (Duck App), in a design and in code

Skills

- Solve a design problem by introducing composition
- Transform, in code, an inheritance relationship into a composition relationship

Lesson 4

23. Syntax of sequence diagrams – use of activation bars; how to show looping; how to show message passing and self-calls; iteration marker and interaction frame; return arrows
24. Using fragments of a sequence diagram starting from a reference point; introducing UI and Controller classes to model full use cases; when an actor should be shown and when a reference point can be used instead
25. Sequence diagrams as a way to model a use case (success scenario)
26. Centralized control vs distributed control in sequence diagrams
27. The meaning of delegation and propagation
28. The meaning of polymorphism and late binding
29. The reason why static, private, and final methods have early binding
30. The template method design pattern. Recall how it was used in the exercise on calcCompensation and in the DataParser example in the slides.
31. Open-Closed Principle

Skills

- Create a sequence diagram based on a use case description.
- Solve a problem using polymorphism.
- Use the template method pattern to solve a design problem.
- Converting Java code to a sequence diagram

Lesson 5

32. Definitions concerning abstract classes
33. Differences between abstract class and interface (in Java 7)
34. UML notation for abstract classes and interfaces
35. The Object Creation Factory pattern (know the diagram and what it means, the problems it solves)
36. Know several examples of these patterns and what they illustrate
37. The "Diamond Problem" for languages with multiple inheritance
38. Benefits of using interfaces
39. Refactoring / extending a design using interfaces (example in the slides)
40. What does Program to the Interface mean? Why is it a good practice? What are some examples?
41. What is the Evolving API Problem?

Skills

- Solve a problem using polymorphism.
- Create a factory method in a class
- Use a factory to implement a 1:1 bidirectional relationship or 1:many bidirectional relationship (check homework)

Exam Policy:

1. The exam is close-book exam, takes 2.5 hours. The exam starts at 10am, please come here 5 miuntes earlier. If you have bags, notebook, phone, etc, you **MUST** leave them in the front of class. Once everything is settled down, we'll start deliver your paper. I won't extend time if the delay is caused by putting bags, phones, etc.
2. You're allowed to bring stationery(pencil, eraser, ruler, etc).
3. You're allowed to go to restroom only once during the entire exam one by one. You must return in 10 minutes.
4. You're **NOT** allowed to
    a. Bring phone, ipad, etc. all electric products. If you say I need phone to watch the time, please bring a watch, Phone is not allowed. And I'll notify you the time every an hour.
    b. Borrow pencil, eraser, pencil sharpener etc. from your classmates during exam. Please buy what you need before exam.
    c. Go out to drink water. Please bring water if you need.
    d. Talk/discuss with classmates in any language. If you do that, I won't ask you to submit your paper. I know you won't. But I'll lower your final grade, like from A to A-.
5. If you have any questions of the exam, please ask me!!!