CS401 – Modern programming practice

Midterm Review Points

Structure of the Exam

The midterm will consist of three parts,

1. Theory part [35 %]
2. Design part [40 %]
3. Coding part [25 %]

- Theory part includes – multiple choice, fill in the blanks, Concept, Debugging, finding output
- Design(diagrams) – Draw class, sequence, object diagram and finding relationship(Inheritance,association,dependency) based on a problem statement
- Code – converting UML into Java code and related problems (like show the association, converting inheritance to composition, applying OO principles and design pattern(factory method pattern)

Portion : Lesson 1 – 5. [Reference lecture slides, demo codes & Home work]

Lesson-6-JavaFX will not be covered.

<u>Important points for the MPP Exam</u>

1. The Midterm examination held on 07/08/17 – Saturday Morning.
2. The midterm will be timed. It will begin at 9.45 am and will end at 12:00 noon.
3. Midterm should be closed book.
4. Bring Pencil/Pen, Eraser and necessary things. You are responsible to keep your writing desk neat and clean. [Use waste paper to keep the pencil sharpened dust].
5. Mobile should be in Silent or Swich off mode. You are not allowed to keep the mobile. So bring backpack to keep your belongings. Keep the backpack infront of the dias.

<u>Lessons 1 and 2</u>

1. Discovering classes from a problem statement
2. Difference between analysis and design
3. Differences between association,dependency, inheritance
4. Associations, dependencies in code
5. Difference between one-way, two-way associatons
6. Properties as attributes, properties as associations
7. Association adornments: name, role, multiplicity
8. Aggregation vs composition vs association
9. Reflexive associations

Skills:

- Create a class diagram with attributes, operations, associations, based on a problem statement
- Translate a class diagram into Java code

Lesson 3

10. Good uses of inheritance vs bad uses
11. Inheritance rules
12. Order of Execution
13. IS-A and LSP principles
14. Bad Stack example (IS-A)
15. Benefits of inheritance
16. Rectangle-square problem (Violate LSP)
17. EnhancedHashSet problem – inheritance violates encapsulation
18. Principle: Design for inheritance or prevent it(P2I)
19. How to replace inheritance by composition 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

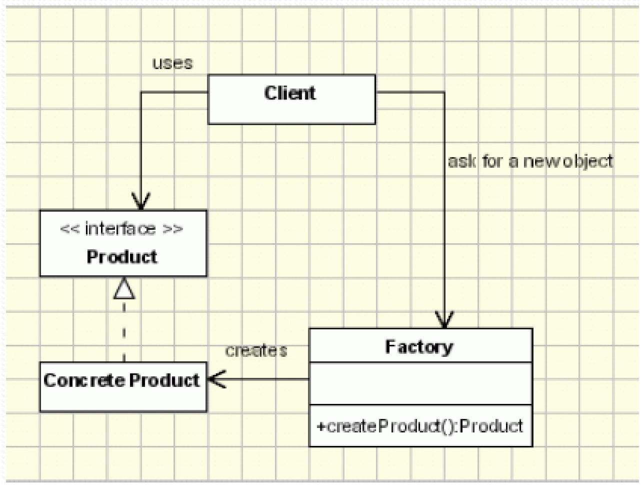
20. Syntax of sequence diagrams – use of activation bars; how to show looping; how to show message passing and self-calls; return arrows, use notes to show the decision making(if-else)
21. Using fragments of a sequence diagram starting from a reference point(an actor should be shown); introducing UI and Controller classes to model full use cases;
22. Centralized control vs distributed control in sequence diagrams
23. Syntax of object diagrams; purpose of object diagrams
24. The meaning of delegation and propagation
25. The meaning of polymorphism and late binding
26. The reason why static, private, and final methods have early binding & know about late binding
27. The template method design pattern. Recall how it was used in the exercise on calcCompensation(homework) and in the DataParser example in the slides.
28. Open-Closed Principle
29. Give an example from the Java libraries to illustrate how interfaces and abstract classes can be used together effectively. Example ArrayList class implements 6 interfaces and has one parent.

Skills

- Create a sequence diagram based on a use case description.
- Converting Java code to a sequence diagram and vice versa.
- Create an object diagram, given information about a system of objects and their attributes.
- Understanding of polymorphism and its benefits.
- Use the template method pattern to solve a design problem.

Lesson 5

30. Differences between abstract class and interface (in Java 7)
31. UML notation for abstract classes and interfaces
32. The Object Creation Factory pattern (know the diagram and what it means)



33. The simple factory method pattern
34. Know several examples of these patterns and what they illustrate. [Refer Slide :16(Advantages) – Lesson -5]
35. Singleton design using factory method pattern
36. The “Diamond Problem” for languages with multiple inheritance
37. Benefits of using interfaces
38. Refactoring / extending a design using interfaces
39. What does Program to the Interface mean? What are some reasons to follow this principle?
40. What is the Evolving API Problem?
41. Give an example from the Java libraries where factory pattern has been used.

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, avoid multiple constructors(Triangle) and useful in singleton design
- Turn a class into an *immutable* class.

Coding

- Be able to convert UML to Java.
- Be able to implement factory method pattern
- Be able to implement singleton design pattern
- Be able to implement template pattern
- Familiar with the syntax of defining class, Inheritance, abastract class and interfaces.
- Able to know how to use OO principles in the implementation