CS401 – Modern programming practice

Midterm Review Points

Structure of the Exam

The midterm will consist of three parts,

1. Theory part
2. Design part
3. Coding part

- Theory part includes – multiple choice, fill in the blanks, Concept, Debugging, finding output
- Design(diagrams) – Draw use case diagram, class model, sequence daiagram, 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 Unidirectional and Bidirectional with one to one, one to many & many-many), Inheritance, Interface, applying factory method pattern and template pattern.

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

Lesson-6-JavaFX will not be covered.

Important points for the MPP Exam

1. The Midterm examination held on **1/25/2020 – Saturday Morning in the same class room.**
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]. Then arrange the laptop, mouse and chair.
5. Mobile should be turned off. You are not allowed to keep the mobile with you. No wearable technologis(Smartwatch). So bring backpack to keep your belongings. Keep the backpack infront of the dias.

Focus the following question from each lesson.

Lessons 1 and 2

1. Discovering classes from a problem statement
2. What is use case?
3. Difference between analysis and design
4. Differences between association,dependency, inheritance
5. Convert Associations, dependencies in Java code

6. Difference between one-way, two-way associatons
7. Properties as attributes, properties as associations
8. Association adornments: role name, multiplicity
9. Association class – how is an association class represented in a more detailed design?
10. 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

11. Good uses of inheritance vs bad uses
12. Inheritance rules ✓
13. Order of Execution.(Static intitalization, Instance initialization, constructors etc.,)
14. IS-A and LSP principles
15. Bad Stack example (IS-A) ✓
16. Benefits of inheritance
17. Rectangle-square problem (Violate LSP) ✓Because of setters
18. EnhancedHashSet problem – inheritance violates encapsulation Ripple Effect
19. Principle: Design for inheritance or prevent it(P2I)
20. How to replace inheritance by composition in a design and in code
21. Inheritance vs Composition

Skills

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

Lesson 4

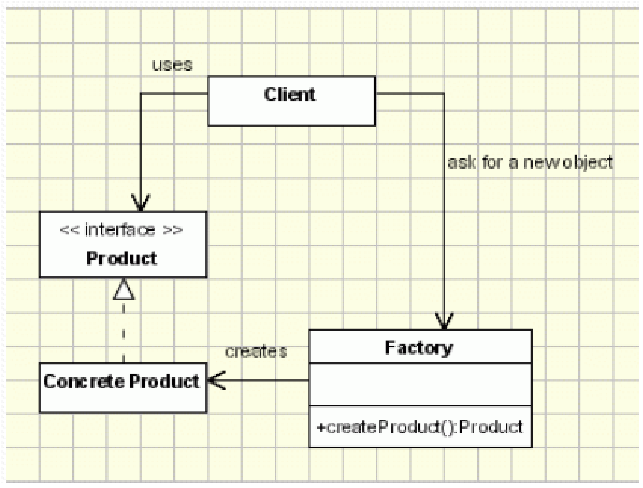
22. Syntax of sequence diagrams – Inlusion of actors, use of objects, use of activation bars; how to show looping; how to show message passing and self-calls; return arrows.
23. Centralized control vs distributed control in sequence diagrams
24. Syntax of object diagrams; purpose of object diagrams
25. The meaning of delegation
26. The meaning of polymorphism and late binding
27. The reason why static, private, and final methods have early binding & know about late binding
28. The template method design pattern. Recall how it was used in the exercise on calcCompensation(homework) and in the DataParser example in the slides.
29. Open-Closed Principle
30. 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.
- 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

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



34. The simple factory method pattern
35. Know several examples of these patterns and what they illustrate. [Refer Slide :17(Advantages) – Lesson -5]
36. Singleton design using factory method pattern
37. The "Diamond Problem" for languages with multiple inheritance
38. Benefits of using interfaces
39. Refactoring / extending a design using interfaces.
40. Abstract class vs Interfaces
41. What does Program to the Interface mean? What are some reasons to follow this principle?
42. What is the Evolving API Problem?
43. 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 Part

- 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