Final exam May 2017 шалгалтад ирсэн асуултууд бараг Final Exam (May 2015) шалгалтынхтай ижилхэн байсан ялангуяа бодлогууд нь хоёулаа ирсэн. Зарим асуултдууд жишээ нь

1. loose coupling = decoupling –г 10 pattern-ний хувьд тайлбарлах
2. Command pattern-ний оронд Composite pattern тайлбарлах
3. if/else conditional –ийн оронд хэрэглэгддэг 5 pattern бичих асуулт

Өмнөх 7 хоногт quiz аваад дараа 7 хоногт нь final авсан, midterm аваагүй. Quiz нь бусад багш нарын материалаас цуглуулсан асуултууд их байсан. Харин final нь 2015 онд авч байсантайгаа бараг ижил материал ирсэн.

ASD exam topics:

1. Proxy
2. Factory
3. Abstract factory
4. Composite
5. Chain of Responsibility
6. Command
7. Facade
8. State
9. Strategy
10. Mediator
11. Observer
12. Visitor

Important parts for each:

1. Intent
2. Structure
3. Implementation
4. Benefit

Review points

1. When you use enum type to implement a Singleton, is it considered as early or lazy instantiation?
2. What is the intent of the Adapter pattern?
3. What are the participants of Adapter pattern?
4. What is the intent of the Proxy pattern?
5. What is the Motivation of the Proxy pattern?
6. Compare the above two design patterns (what is similar and what is different?)
7. What is the intent of the Template Method?
8. How does the Template Method help us reduce code duplication?
9. What is a hook operation in the Template Method?
10. Explain some important details when implementing a Template Method
11. What is the intent of the Prototype Pattern?
12. When would the Prototype Pattern work better compared with using the constructor to create an object?
13. The Prototype Pattern hides concrete product classes from the client, thereby reducing dependencies the client puts on these classes. True or False?
14. What is the intent of the Factory Method?
15. What are the participants of the Factory Method Pattern?
16. What is good about using the Factory Method Pattern?
17. What is the intent of the Abstract Factory?
18. What are the participants of the Abstract Factory?
19. Advantages of disadvantages of the Abstract Factory?
20. Both the Factory Method and Abstract Factory help us remove if/else statements from client code. True or False?
21. What is the intent of the iterator pattern?
22. The Iterator pattern allows a client program to traverse different aggregate data structures without having to change code. True or False?
23. For the Iterator pattern, which participant is responsible to keep track of the current position in the traversal of the aggregate? (Client, Aggregate, Concrete Aggregate, Iterator, Concrete iterator)
24. With the iterator pattern, which participant is responsible to create an iterator instance?
25. When do you consider using the Composite design pattern? (Applicability)
26. What is the intent of the Chain of Responsibility pattern?
27. Draw the structure of the Chain of Responsibility pattern?
28. Compare the Proxy and the Chain of Responsibility.
29. What is the intent of the Command Pattern?
30. Draw the structure of the Command Pattern.
31. What is the responsibility of the Invoker?
32. What is the responsibility of the Receiver?
33. What is the intent of Façade design pattern?
34. Façade does not add functionality but only simplifies the use of interfaces of underlying subsystems. True or False?
35. With the Façade pattern, the client is not allowed to use the subsystems directly. True or False?
36. With the Façade pattern, it reduces client’s dependencies on subsystems. True or False?
37. What is the Intent of the State pattern?
38. What is the Intent of the Strategy pattern?
39. Compare the State and Strategy pattern.
40. Compare the State and Strategy pattern. (provide at least 2 points for similarity and difference)

Similarities:

* 1. Both help us remove if/else statements from code.
  2. Both allow runtime change of behavior.
  3. Both are behavioral.
  4. Both decouple the client from the underlying service layer.

Differences:

1. Intent is different: Strategy lets algorithms vary from the client; while State allows an object to change behavior when its internal state changes.
2. Strategy is stateless while state is stateful.
3. Behavior change is caused by internal state change (State), environment change (Strategy)

CS428

Software Development with Design Patterns  
Final Exam (May 2015)

Name

Student ID

READ ALL QUESTIONS CAREFULLY BEFORE YOU ANSWER THEM.

PART I: INFORMATION QUESTIONS

I. True/False Questions. (15 points) Mark T or F as appropriate.

1. The Command Pattern allows us to easily add concrete command classes without affecting client code.

2. In the Strategy Pattern, a strategy object stores state information across different calls from

the client.

3. Abstract Factory makes exchanging product families easy.

4. In the State Pattern, State information can be represented by an instance variable or by the

type of the concrete state object.

5. When we use the Strategy pattern, we will have to expose complex, algorithm-specific data

structures to the client.

6. We can use the Mediator Pattern to simplify complex interactions among many objects.

7. Mediator Pattern centralizes control into the Mediator class, which compromises loose

coupling among the communicating objects.

8. In the Command Pattern, a client can send many different requests into a queue from which

these requests can be retrieved/executed one by one when it is good time to do so.

9. The Visitor Pattern introduces flexibility but may compromise encapsulation as a negative

consequence.

10. Factory Method eliminates the need of your client code to know about the concrete product

types.

1. Single/Multiple Choice Questions. (15 points)

11. Next to the name of each pattern, please write the letter of the description below that best matches it: (15 points)

Observer Pattern State Pattern Visitor Pattern

Command Pattern Strategy Pattern Mediator Pattern

Bridge Pattern Abstract Factory Factory Method

1. Provides different implementations of the same behavior. At runtime, one of these implementations is chosen based on different time and space trade-offs.
2. Allow new operations on a fixed set of target objects to be defined without modifying the target objects, by having operations and objects perform a “handshake” to determine how they should interact.
3. A family of related product objects is created and they are designed to be used together.
4. Defines a one-to-many relationship between an object and other objects that are notified about changes in its state.
5. Centralizes responsibility in a class that oversees how a set of other objects interact.
6. Allows us to pass objects around as actions to perform (also considered as object-oriented replacements for procedural language callback functions).
7. Instantiates the selected class and returns it as an instance of the parent class type.
8. Each subclass implements a behavior associated with a state of the Context.
9. Avoid permanent binding between abstraction and implementation by favoring composition over inheritance.
10. Encapsulate a request as an object, thereby letting you parameterize clients with different requests.

III. Short Answer Questions. (30 points). Give enough detail for each question. Being too skimpy will not earn you full points even though you may have the complete knowledge. You may want to give more information than “enough detail” if you are not sure.

1. (5 points) Explain HOW to implement ‘undo’ operations with a Command Pattern? And HOW to roll back a transactional operation with it. (you need to list all key points and use diagrams to explain if necessary)
2. (6 points) Which of the 9 design patterns (you see in Question 11) help us remove the if/else statements? List at least 5 patterns and for each one explain HOW.
3. (5 points) What is Double Dispatch in Software Engineering? How does it work? Use both natural language and Java code to explain.
4. (9 points) Explain how the 9 patterns listed in Question 11 promote loose coupling?
5. (5 points) Explain how each of the following 3 patterns helps us reduce subclassing? - Mediator, Factory Method, Abstract Factory. Please use whatever is helpful in your explanation - diagrams, code or comments.

PART II: PROGRAMMING QUESTIONS

1. (20 points)

You are developing a library system for a Public Library. All items that its members (adults and children) can check out are books, magazines and media items. Loan periods for everything are 4 weeks. A fine is charged for each overdue item. Details are listed below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Overdue Fines | Book/Day | Magazine/Day | Media/Day | Maximum Fine |
| for Adults | $0.10 | $0.15 | $0.20 | $10.00 |
| for Children | $0.05 | $0.05 | $0.05 | $3.00 |

You now need to develop the FineCalculator class with the Visitor Pattern. You are also responsible for all other necessary classes/interfaces to make your FineCalculator work.

1. Draw the UML diagram for your solution with all necessary participants, responsibilities and relationships clearly specified.
2. Write Java code that shows all necessary attributes/methods with implementation logic for your design. (No need to provide test code).
3. (20 points)

You are developing an application that deals with the following workflow -

1. LOCAL - Create a user record in the local database. (via DBFa9ade).
2. REMOTE - Create a payment account on a remote server. (via BillPayFa9ade).
3. REMOTE - Submit credit card info for payment method details (via BillPayFa9ade).

Both the DBFa9ade and BillPayFa9ade are classes locally available to you. Below are 2 interfaces

they implement respectively

public interface DBFacadeInterface{ public void save(User user); public void remove(User user);

}

public interface BillPayFacadeInterface {

public Account createPaymentAccount(User user); public void removePaymentAccount(Account account); public boolean submitCreditCard(CreditCard card);

}

During the workflow, in case step 3) fails for any user, you must rollback step 1) and step 2) to maintain data integrity. Please design/implement the above requirements with the Command Pattern. You need -

1. Draw the UML diagram for your solution with all necessary participants, responsibilities and relationships clearly specified.
2. Write Java code to implement your design illustrated above. (Need to write code to show how to roll back steps 1&2 if step 3 in the workflow fails).

Extra Credit (5 points)

After learning the 23 GoF Design Patterns during the course, you have probably developed a sense of what is import in designing robust yet flexible/extensible/reusable software that supports different business needs. Talk about your own experience, as a software designer, regarding some of the most important concerns in designing/architecting Object-Oriented software applications. You can earn a maximum of 5 points if you provide a comprehensive discussion. Do not expect full marks with less than 15 lines of text, unless they all hit some important points.

**CS428**

**Software Development with Design Patterns**

**Midterm Exam (May 2015)**

READ ALL QUESTIONS CAREFULLY BEFORE YOU ANSWER THEM.

**PART I: INFORMATION QUESTIONS**

1. **True/False Question.** (15 points) Mark T or F as appropriate.
2. In the Facade pattern clients are NOT allowed to bypass the Facade and use subsystem classes directly.
3. The Template Method is a fundamental technique for code reuse.
4. You can use the Prototype design pattern to abstract steps of construction of complex objects.
5. The composite pattern helps to make the client simple.
6. The prototype pattern provides us a simplified object copy process by cloning.
7. In Chain of Responsibility, there are multiple handlers available but the Client has the knowledge of which handler is responsible to process its request.
8. The Intent of the Adapter pattern is “provide a surrogate or placeholder for another object to control access to it”.
9. In the Singleton pattern, “early instantiation” will never cause thread-safe traversals.
10. The iterator pattern supports multiple, concurrent and thread-safe traversals.
11. The Chain of Responsibility Pattern guarantees a request will always be received and processed.
12. **Multiple Choice** (25 points) Select the best answer(s) in each case.
13. A design pattern is: (Select the best answer)
14. An algorithm used in object-oriented programming
15. A data structure used in object-oriented programming
16. A solution to a common problem in object-oriented programming
17. A class library at code level
18. Which of the following statements about the Facade Pattern are correct? (Choose 2)
19. The Facade pattern may break encapsulation of the subsystem that it means to simplify.
20. When a Facade object is used with a subsystem, the subsystem is not aware of the facade.
21. The Facade design pattern does not introduce new functionality.
22. The facade pattern is used to lower coupling between different subsystems.

13. Which of these statements are true? (Choose 2)

1. The Client is aware of the Adaptee object.
2. Adapter holds an instance of Adaptee.
3. Adapter implements the Adaptee interface.
4. Adapter implements the Target interface (interface that client expects).

14. Which of these statements are true? (Choose 2)

1. In the Prototype pattern, a Creator class is needed to manage the steps of creating an object instance.
2. The prototype pattern does not prevent class/type proliferation.
3. Use the Prototype pattern when a system should be independent of how its products are created, composed, and represented.
4. When creating a large/complex object that is resource intensive, Prototype is a good choice.

15. What are the motivations of the Prototype Pattern (Choose 2)

1. When the Client only knows that it needs an object of a certain type but does not know exactly  which one from the set of subclasses of the parent class is to be selected.
2. When we want to ensure that additional instances of the class cannot be created.
3. The Prototype Pattern allows us to make new instances by copying existing ones.
4. When your application has to create objects without knowing their type or any details of how to create them.

16. When we want to apply the Template Method Pattern? (Choose 2)

1. Define the skeleton of an algorithm in an operation, deferring some steps to subclasses.
2. When we want to choose a step-by-step approach to construct an object.
3. To localize common behavior among subclasses and place it in a common class (in this case, a superclass) to avoid code duplication.
4. When we need an algorithm that uses data that clients shouldn’t know about.

17. Which are NOT key elements in the description of a design pattern? (Choose 2)

1. Name: a label that identifies it
2. Intent: a description of the purpose of the pattern
3. Justification: a description of the benefits of using the pattern
4. Participants/Collaborators: the entities involved in the solution
5. Consequences: what happens as a result of using the pattern
6. References: where to look for more information

18. When we want to apply the Chain of Responsibility Pattern? (Choose 2)

1. When we want to handle requests with a “handle or forward” model
2. When we want to avoid coupling the sender of a request to its receiver
3. When clients will treat all objects in the chain uniformly
4. When we want to provide a uniform interface for traversing the chain of objects

19. Which is considered an example of Chain of Responsibility Pattern? (Choose 2)

1. java.awt.event.ComponentAdapter
2. The Java Servlet filter framework
3. java.awt.Toolkit
4. Java exception handling
5. javax.jms.QueueConnectionFactory

20. What are the key advantages of learning and using Design Pattern? (Choose 2)

1. Design patterns are new inventions in the domain of software design. So they will help us come up with unique design solutions.
2. They help designers reuse successful designs by basing new designs on prior experience.
3. Design patterns can be used as class libraries, thus greatly reduce development time.
4. Design patterns are abstractions over implementation, thus make communications among software designers a lot faster/easier.

**III. Short Answer Questions. (20 points)**

**21.  (5 points)** GoF divided their patterns into 3 categories. Briefly explain what they are. Also categorize the 9 patterns we have learned so far.

**22. (5 points)** Explain what a Proxy pattern is, how it works and give at least 3 examples of when you can use it.

**23. (5 points)** Briefly explain how the Composite Pattern works. List all its participants with their responsibilities.

**24. (5 points)** Talk about your experience with Design Patterns so far. In your own words, what do you think you have learned from the first 2 weeks of the course? Do you think the knowledge will help you become a better IT professional? Why?

**PART II: PROGRAMMING QUESTIONS**

In this section, there are 2 programming questions that are 20 points each. You are supposed to answer both questions that are worth a total of 40 points.

**25. (20 points)**

You are working on a framework for GUI application development. The main components that you develop include GUI widgets (buttons, textboxes, checkboxes, dropdown boxes, scrollbars, etc.) and GUI containers (a special container called Window and other regular containers). Your idea is to design the model that stores the containers and widgets in a tree structure using a Composite pattern.

The requirement is for each GUI, there is one and only one special top-level container (called a Window). All widgets and regular containers can be placed on the Window. Regular containers can hold widgets as well as other regular containers. Your design should allow adding/removing widgets onto/from containers and painting them on the screen by simply calling the paint() method on each of the widgets and/or containers.

The way it works is illustrated below -

Painting always starts from the top (the Window) by calling its paint() method first. Then below the top-level Window, if widgets are found, the widgets’ paint() methods are called one by one. If regular containers are also found, you should call the first container’s paint() method and all its children’s paint() methods before moving to the second container on the window. This process goes as deep as the tree structure until all widgets and containers are painted.

1). Design your tree structure using the Composite pattern with a structure diagram showing participants and their interfaces or responsibilities. 2) Write Java code for your implementations. (For all different widgets, you can use one Widget superclass to represent all of them.)

// YOUR CODE FOR PROB 25 GOES HERE

**26. (20 points)**

You are developing a model for displaying different contents on a client device. Suppose there are only 5 different content types (from most specific to most generic) - image, post, category, archive, and front-page. And correspondingly there are 5 different templates that can be used to display the content types. Below is a table that shows which template is good for which content types.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Image | Post | Category | Archive | Front-page |
| ImageTemplate | Yes | No | No | No | No |
| SingleTemplate | Yes | Yes | No | No | No |
| CategoryTemplate | Yes | Yes | Yes | No | No |
| ArchiveTemplate | Yes | Yes | Yes | Yes | No |
| GenericTemplate | Yes | Yes | Yes | Yes | Yes |

But the rule is you must always use the most specific template to display any content type. For example, for the ‘Category’ content type, the ‘CategoryTemplate’, the ‘ArchiveTemplate’ and the ‘GenericTemplate’ are all possible templates, according to the above table. But since the ‘CategoryTemplate’ is the most specific one among the 3, you must use this one.

1) Use the Chain of Responsibility pattern to design the model with a structure diagram, showing all the necessary participants and their relationships and key behavior.

2) Build the chain of templates to make sure the rule is always followed.

3) Implement 1) and 2) with Java code.

You can use the skeleton code for Content for your implementation

public class Content {

private String contentType; //image, post, category, archive, front-page

//...

public Content(String type) {

 this.contentType = type;

}

//…

}

// YOUR CODE FOR PROB 26 HERE