中山大学软件学院 **2007** 级软件工程专业

《 系 统 分 析 与 设 计 》 期 末 考 试 试 题 （ A ）

(考 试 形 式 ： 开 卷   考 试 时 间 ： 2 小 时)

警 示 　《中山大学授予学士学位工作细则》第六条

考 试 作 弊 不 授 予 学 士 学 位

方向：_____   姓名：_____   学号：_____

一、 选择题（30 marks, each 3 marks。注意：每题的正确答案可能不止一个。答案完全正确给 3 分。选多，选错或选漏皆为 0 分）

1. The design pattern "Iterator" can be viewed as a special case of which pattern?

   A Factory Method   B DAO   C Façade   D Delegate

2. Which UML diagram describes the static perspective of a design model?

   A class diagram B state diagram C sequence diagram D communication diagram

3. What are the four phases in a Unified Process?

   A Inception, Analysis, Design, Transfer

   B Analysis, Design, Coding, Testing

   C Inception, Elaboration, Construction, Implementaion

   D Transfer, Construction, Elaboration, Inception

4. In a Design Class Diagram, which relation can be used to represent a "has-a" relation?

   A generalization B composition C dependence D inheritance

5 Which agile method belongs to Extreme Programming?

   A test driven development

   B common project workroom

   C self-organizing teams

   D regression testing

6. How many statements are true?

   (1) The actors in use cases can only be people, organizations and external systems

   (2) Assigning responsibilities is the main task of object-oriented design

   (3) UML is a methodology of object-oriented analysis and design

   A 0   B 1   C 2   D 3

7. How many statements are NOT true?

   (1) The actors of use cases can be categorized into primary actors, supporting actors and offstage actors

   (2) A use case can contains several scenarios

   (3) The main idea of MVC pattern is separating presentation logic code from business logic code

   A 0   B 1   C 2   D 3

8. How many statements are true?

(1) Pre-condition is the most important part in an operation contract

(2) Domain models are also called conceptual models, domain object models, and analysis object models

(3) Use cases are mainly used to capture non-functional requirements

    A 0   B 1   C 2   D 3

9. How many statements are true?

(1) Analysis emphasizes an *investigation* of the problem and requirements, rather than a solution.

(2) Most of the challenging, interesting, useful design work happens while drawing the UML dynamic-view interaction diagrams

(3) Use cases can only be used in object-oriented analysis and design

    A 0   B 1   C 2   D 3

10. How many statements are true?

(1) Sequence diagram and communication diagram are equivalent.

(2) In elaboration phase, we usually schedule the use cases with higher risks before the ones with lower risks.

(3) In design, we usually complete all the sequence diagrams before we start to draw the design class diagrams.
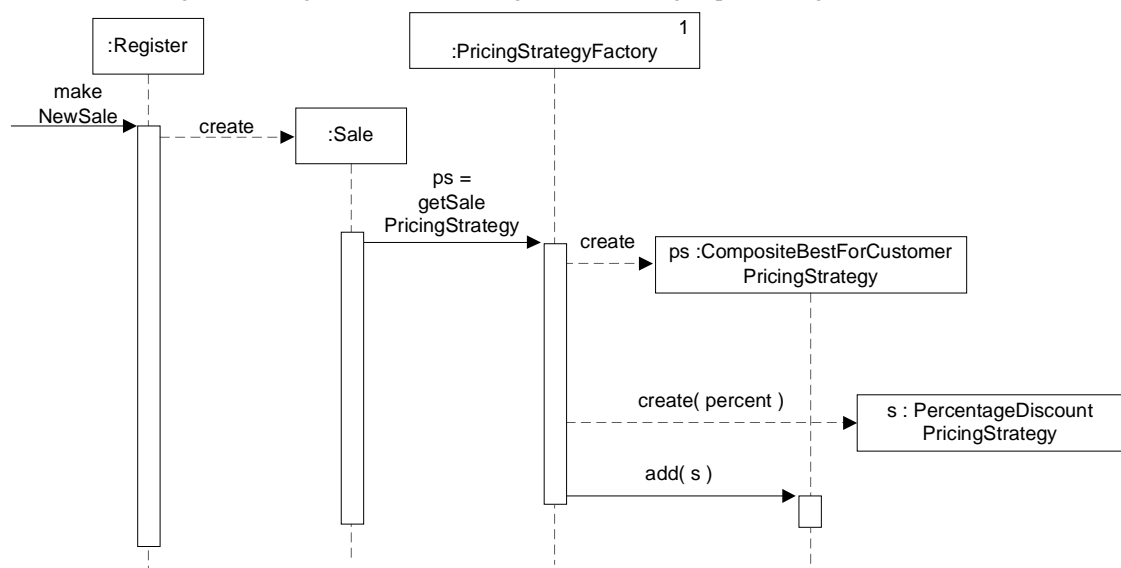
    A 0   B 1   C 2   D 3

二、 简答题 （**30 marks, each 15 marks**）

1 As we know, change in requirements is common in a software project. What things can we do to reduce the influence of change in requirements?

2 Describe the similarity and difference between GRASP and GoF Design Patterns

三、 画图题 （**20 marks, each 10 marks**）

1 Draw the Design Class Diagram (DCD) according to the following sequence diagram

2 Draw the corresponding sequence diagram according to the following diagram
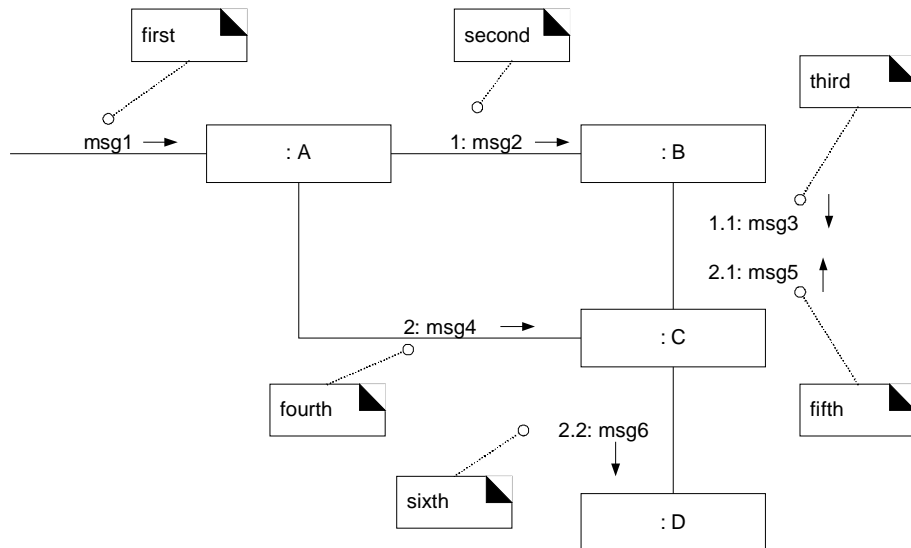
四、 案例分析 （20 分，本题两小题，任选一小题做答。如果两题都答了，只有第一小题给分）

1. Give a design of a software component that allows the user to manipulate files and directories. The files and directories are organized in a tree. The requirements are:

    1. A Client needs information about a file system.

    2. The model of the file system is composed of items.

    3. Each item is either a file or a directory.

    4. Every item has a name.

    5. Every directory has contents, which consists of some number of items (files or directories).

    6. Every file has a size which is an integer (indicating the number of bytes in the file).

    7. Every item has a method called *totalSize* that returns an **integer** result. *totalSize* shall return the total of the sizes of all items that lie anywhere at or below this item in the file system tree.

Tasks for this question

(a) Draw a UML class diagram showing how to use the Composite pattern to model this situation. Assume that a simple recursive traversal of the tree is used to implement the *totalSize* operation (that is, this operation plays the role of the Operation or DoIt of the Composite design pattern) Include:

    • the classes CLIENT, ITEM, FILE and DIRECTORY;

    • inheritance relationships;

    • association and aggregation relationships, with multiplicities and labels;

    • pattern roles for all the classes; and

    • all attributes and routines (otherwise known as fields and methods) including *totalSize*.

    • Label abstract classes and features with an asterisk (*).

Note：You do not need to mark access control/visibility of features.

(b) Write a pseudo-code algorithm for each version of the *totalSize* method. (That is, write a separate implementation of *totalSize* for each concrete class that has it.)

(c) Draw a sequence diagram to describe how *totalSize* operates in a tree with one directory that contains two files.

2   In an enterprise system, the sale processing data is stored in the ERP database (Oracle) and the financial data is stored in (another) finance database (MySQL) . In the future, these two databases may be merged in one DB. In the end of every month, we need to read sale processing data of current month and import it into the finance database so as to generate a financial report.

The procedure can be written as the following precede code

```
Public void tranferSaleInfo()
{
    //1 open connection of the ERP's DB
    //    (as conn_erp)
    //2 open connection of the finance DB
    //    (as conn_finance)
    //3 read sale data through conn_erp
    //4 write sale data into finance db through
    //    conn_finance
    //5 close connections
}
```

The interface of a database connection is as follow

```
Interface Connection
{
    void openConnection();
    void closeConnection();
}
```

Please apply design patterns to given a solution, so that the code in transferSaleInfo can keep unchanged when the types of databases change in the future.