Software Engineering Methods

Dr. Alberto Bacchelli
Friso Abcouwer, Rob van Bekkum, Moritz Beller,
Thijs Boumans, Aimee Ferouge, Jan Giesenberg

Final Exam

Nov 7, 2013 @ 13:45 - 15:45

First Name:	
Last Name:	
Student Number:	
NetID:	
Course code:	
Seat row:	
Seat column:	

READ THE INSTRUCTIONS CAREFULLY!
Failure to comply to any of the following instructions means *invalidation of your exam*:

1. This is a closed book exam. The use of any external resource (*e.g.*, your laptop, notes, and colleagues) is *not* allowed.
2. Write in a readable way.
3. Complete the table above with your data ('course code' refers to the code for registering your result on OSIRIS).
4. Write your name on *each* sheet of paper.
5. Write all your solutions on the official paper sheets; other papers will be ignored.
6. Once finished, put everything within an empty official sheet, and write your name and student number in large on the first page.

Correction Sheet

Exercise	Points	Result	Notes
1.1	2.5		
1.2	2.5		
1.3	3.0		
1.4	2.0		
2.1	6.0		
2.2	4.0		
2.3	6.0 (opt)		
3.1.a	3.0		
3.1.b	3.0		
3.2	2.0		
3.3	2.0		
4.1	3.0		
4.2	3.0		
4.3	4.0		
4.4	3.0 (opt)		
5.1	4.0		
5.2	2.0		
5.3	4.0		
6.1.a	1.0		
6.1.b	3.5		
6.2.a	1.5		
6.2.b	4.0		
Total	**60 (+ 9 opt)**		

Exercise 1 - Warm up (10 pts)

Exercise 1.1 - 2.5 pts

Fill in the blanks[1] on the following statements about SOLID object-oriented design principles, using one word from those in the brackets:

```
There should (1)_______[never/sometimes/always] be more than one reason for a class to
change.  Software entities should be (2)_______[open/closed] for extension and
(3)_______[open/closed] for modification.  The preconditions of a substitutable
derived class are no (4)_______[weaker/stronger] than its base class methods.
```

Exercise 1.2 - 2.5 pts

Expand the following acronym:[2] `S. O. L. I. D.`

Exercise 1.3 - 3.0 pts

Write a short paragraph (around 5 sentences) about the disadvantages of design patterns.

Exercise 1.4 - 2.0 pts

Explain what *cohesion* is in software architecture. Then, give an example of a class with high cohesion, and an example of a class with low cohesion.

[1] Use an external official sheet to write your answers, just mention the number before the blank to specify which answer you are giving. For example, you can write: *(1) = always, (2) = closed, ...*

[2] Use an external official sheet to write your answers, just mention the letter before the blank to specify which answer you are giving. For example, you can write: *S = special principle, O = other principle, ...*

Exercise 2 - Software Metrics (10 pts) *(+ 6 optional pts)*

Exercise 2.1 - 6.0 pts

Give three examples of estimation techniques that can be used to estimate the cost of a software project, along with one disadvantage for each of these techniques.

Exercise 2.2 - 4.0 pts

Why is it bad to measure programmer productivity by the LOC (lines of code) they produce? Give two arguments.

Optional exercise 2.3 - 6.0 pts

We can use the Goal-Question-Metric approach to find or prevent the occurrence of design flaws in a system. Pick a design flaw out of God Class, Method Envy and Data Class, then define the Goal for that design flaw and 3 related Questions with 1 Metric each. *Give the full name and/or a description of the metric, not just an abbreviation. Make sure to avoid overly vague metrics.*

Exercise 3 - Design Patterns - I (10 pts)

Exercise 3.1 - 6.0 pts

For each of the scenarios below, discuss: (1) which design pattern could be used, (2) why it would be a good idea to apply this pattern and (3) provide a simple class diagram that shows the applied pattern.

a. You are improving the software implementation of a machine that autonomously prepares ice creams. The machine prepares the ice creams by combining different ingredients. Among these ingredients are different flavours of ice cream (that is, strawberry, vanilla and banana) and several toppings (that is, chocolate chips, almond crisps and stroopwafel crumbs) that can be added, each with its own price. **(3 pts)**

b. As a game developer you are working on a motorcycle game in which the player can race and complete missions. The game contains several power-ups that can be picked up, each with different effects. One of the power-ups increases the speed of the motor, another power-up increases the available time for a mission in the game, and another temporarily increases the score that can be earned (a combo power-up). **(3 pts)**

Exercise 3.2 - 2.0 pts

The state pattern and strategy pattern are in some aspects very similar. Explain the most important difference between these two design patterns (concerning what the patterns are intended for) (max. 40 words for your answer).

Exercise 3.3 - 2.0 pts

Is it possible to compound design patterns? If so, give an example of such a compound pattern and explain of which patterns it consists, otherwise explain why this is not possible.

Exercise 4 - Design Patterns - II (10 pts) *(+ 3 optional pts)*

Exercise 4.1 - 3.0 pts

Both the strategy pattern and double dispatch can be used to decide a type of behavior. Explain the differences between double dispatch and the strategy pattern.

Exercise 4.2 - 3.0 pts

Describe one advantage and one disadvantage of the *Observer* pattern (you can reference those presented in the “Holub on Patterns” appendix).

Exercise 4.3 - 4.0 pts

You are working in Java on a multi-threaded application. You are working on a class `GoodCode` that is currently tracked using a static variable inside a manager class, refactor the code below so that it uses the Singleton pattern, and write down the corresponding code.

```
class GameManager{
    static GoodCode gc = new GoodCode();
}

class GoodCode{
    int grade;

    public GoodCode(){
        grade=10;
    }
}
```

Optional exercise 4.4 - 3.0 pts

Explain *how* the open-closed principle is supported by the abstract factory class used in the abstract factory pattern.

Exercise 5 - Requirements Engineering (10 pts)

Exercise 5.1 - 4.0 pts

Fill the four empty cells in the following table:[3]

Activity name	Activity description
Feasibility study	(1)
(2)	Find out **what the system stakeholders require** from the system
(3)	**Define the requirements** in a form understandable to the customer
Requirements specification	(4)

Table 1: Requirements Engineering Activities

Exercise 5.2 - 2.0 pts

Indicate whether the next requirements (for a Frogger game) are functional (F) or non-functional (NF).

a. The game should have a multiplayer mode.

b. As an Linux user, I should also be able to play the game.

c. A log should be generated in the form of a .txt file, keeping track of the player's moves.

d. The game is won if three frogs reach the other side of the game field.

Exercise 5.3 - 4.0 pts

Evolutionary prototyping and throw-away prototyping are types of prototyping that are suitable in different situations. Please explain when to use evolutionary prototyping and when to use throw-away prototyping.

[3] Use an external official sheet to write your answers, use the numbers as a reference. For example, you can write: *(1) = A study that ..., (2) = ...*

Exercise 6 - Closing (10 pts)

Exercise 6.1 - 4.5 pts

Consider the Java code below, taken from a fictitious video game.

```
public String randomEncounter(String playerClass, int enemyHealth, int playerHealth) {

    while (enemyHealth > 0) {

        switch (playerClass) {
            case "mage":
                castFireBall ();
                break;
            case "warrior":
                swingSword ();
                break;
            case "bard":
                playSong ();
                break;
            }

            if (playerHealth =< 0) { return "Battle lost..."; }

    }
    return "Battle won!";
}
```

a. What is the amount of LOC (lines of code) in the code stated above? **(1 pts)**

b. The '`switch`' statement in the aforementioned code is a very bad design idea. How would you refactor it, so that it does not need the '`switch`' anymore? (hint: Replacing the '`switch`' with '`if`' or '`else if`' statements is *not* a good solution!) **(3.5 pts)**

Exercise 6.2 - 5.5 pts

Consider the class diagram in the figure below:

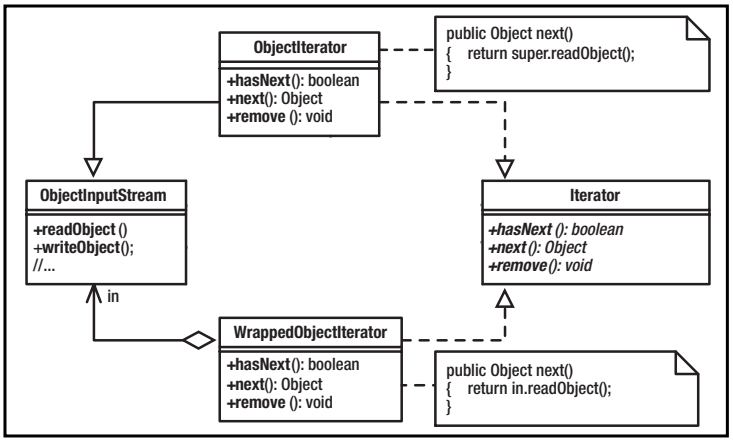

Figure 1: A class diagram of an application of a design pattern

a. Which design pattern is depicted in the diagram? (hint: Do not be misled by the name of some classes!) **(1.5 pts)**

b. What problem does it solve? Also mention one scenario (excluding the one in the above figure) in which it is useful. **(4 pts)**