TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VUW **VICTORIA** UNIVERSITY OF WELLINGTON

EXAMINATIONS — 2013

TRIMESTER 2

SWEN 222
Software Design

Time Allowed: THREE HOURS

Instructions:

- Closed Book.
- Total marks are 180.
- Answer all questions in the boxes provided.
- Every box requires an answer.
- If additional space is required you may use a separate answer booklet.
- No calculators are permitted.
- Non-electronic Foreign language dictionaries are allowed.

Question	Topic	Marks
1.	Design patterns 1	30
2.	Design patterns 2	30
3.	Contracts	30
4.	Functional design	30
5.	Design Quality	30
6.	Software Evolution	30
	Total	180

Question 1. Design Patterns 1

[30 marks]

(a) [8 marks] Provide a *class diagram* which describes the **Observer** pattern.

(b) Write a Java implementation of the Observer pattern. Provide code for an observer interface and a subject abstract class.

(i) [3 marks] Write the `Observer` interface here.

(Question 1 continued on next page)

(ii) [7 marks] Write the `Subject` abstract class here.

(Question 1 continued on next page)

(Question 1 continued)

(c) Design patterns often have implementation variants.

(i) [4 marks] Discuss the different choices about how to store the observers in the subject.

(ii) [4 marks] Discuss the different choices about how the subject notifies its observers, mentioning the kind of information that is transferred.

(iii) [4 marks] What if, in response to an update, some observer decides to stop observing the subject? Discuss how an implementation has to take care of this possibility.

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 2. Design Patterns 2 [30 marks]

(a) [5 marks] Factory pattern. The **Factory Method Pattern** is an object-oriented creational design pattern to allow the creation of objects (products) without specifying the exact class of object that will be created.

Explain at least one advantage of the Factory Method Pattern in software development.

(b) [8 marks] Provide a *class diagram* which describes the **Adapter** pattern.

(Question 2 continued on next page)

(Question 2 continued)

(c) [5 marks] In general, when providing a class/object diagram, would you include instances of the adapter pattern? Justify your answer.

(Question 2 continued on next page)

(Question 2 continued)

(d) [8 marks] Provide a concrete example of when/how you would use the adapter pattern. Provide some Java code to illustrate your example; feel free to use dots (...) in the non-relevant parts.

(Question 2 continued on next page)

(Question 2 continued)

(e) [4 marks] Identify where and how the adapter pattern is used in your example and what plays the different roles of the adapter pattern in your code.

Question 3. Contracts

[30 marks]

Consider the following code:

```
//@requires list!=null && elem!=null
//@ensures list.get(\result).equals(elem)
int find(ArrayList<String>list,String elem){
  for(int i=0;i<list.size(); i++){
    if(array.get(i).equals(elem))return i;
  }
  array.add(elem);
  return array.size()-1;
}
```

(a) [6 marks] Describe in natural language the meaning of the above `requires` and `ensures` clauses.

(Question 3 continued on next page)

(b) [8 marks] It is possible to use assertions to check pre and post conditions. Rewrite the method `find` so that its pre and post conditions are checked. Try to make your solution as concise as possible.

(Question 3 continued on next page)

(Question 3 continued)

(c) [8 marks] Consider now this new, more expressive, `ensures` clause:

```
//@ensures list.get(\result).equals(elem)
// && \forall i in 0..\old(list.size()) |
//            \old(list.get(i))==list.get(i)
```

Describe in natural language the meaning of this new ensure clause.

(d) [8 marks] Discuss how to adapt the code you wrote in point (b) to check this new `ensures` clause. What are the main difficulties?

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 4. Functional design [30 marks]

Briefly discuss each of the following questions:

(a) [4 marks] Why does functional design make reasoning easier?

(b) [4 marks] Why does functional design make testing easier?

(c) [4 marks] Why does functional design make parallelism easier?

(Question 4 continued on next page)

(d) Consider the flyweight pattern.

(i) [5 marks] Briefly, explain the flyweight pattern.

(ii) [5 marks] Discuss why the flyweight pattern requires flyweight objects to be immutable.

(Question 4 continued on next page)

(e) [8 marks] Discuss why functional design can lead to an increase in the memory space required, compared to an imperative approach. Use examples to illustrate.

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 5. Design Quality [30 marks]

(a) [5 marks] Discuss why *simplicity* is an important characteristic of good software design.

(b) [5 marks] What is meant by the term *coupling* in the context of software design? Is it true that a well-designed system should have low coupling? Justify your answer.

(c) [5 marks] What is meant by the term *cohesion* in the context of software design? Is it true that a well-designed system should have high cohesion? Justify your answer.

(Question 5 continued on next page)

(Question 5 continued)

(d) [5 marks] Is it true that whenever we increase the cohesion of different modules in our design, coupling between these modules automatically decreases? Justify your answer. You may use examples to illustrate as necessary.

(e) [5 marks] Discuss the purposes of using *CRC cards* in the process of software design.

(f) [5 marks] *Code review* is one of the important techniques for ensuring quality of software systems. Discuss the advantages of conducting code reviews in the process of software development.

Question 6. Software Evolution [30 marks]

Jenny has developed a library for modelling financial transactions that is used (including via inheritance) by many developers in their own projects. The following class is part of her library.

```
public class Money {
  protected int cents;

  public Money(int cents) { this.cents = cents; }

  public int getCents() { return cents; }
}
```

(a) Jenny likes to continually improve her library. For each of the following "improvements", briefly discuss how developers using her library might be affected.

(i) [3 marks] Jenny would like to rename her class from `Money` to `Cents`.

(ii) [4 marks] Jenny would like to add a method `setCents()` to her class, which allows users to mutate a `Money` object.

(iii) [5 marks] Jenny would like to add a new **`protected`** field, `dollars`, and modify her class to ensure the invariant `dollars == cents/100` is always true.

(Question 6 continued on next page)

(Question 6 continued)

(b) A common problem during development arises when software becomes a *big ball of mud*.

(i) [3 marks] Briefly, describe what a “big ball of mud” is.

(ii) [5 marks] Briefly, discuss why a “big ball of mud” is considered undesirable.

(iii) [5 marks] One recommendation is to *refactor relentlessly*. Briefly, discuss what this means.

(Question 6 continued on next page)

(Question 6 continued)

(iv) [5 marks] A common approach to dealing with a "big ball of mud" is to *keep it working* at all costs. Briefly, discuss why this can be preferable to rewriting it from scratch.

****

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.