TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS — 2005

MID-YEAR

COMP 462
OBJECT-ORIENTED PARADIGMS

Time Allowed: 3 Hours

Instructions:

- *Read each question carefully before attempting it.*
- This examination will be marked out of **120** marks.
- Answer all questions. Each question has the same value, and should take approximately 30 minutes to answer.
- You may answer the questions in any order. Make sure you clearly identify the question you are answering.
- Many of the questions require you to discuss an issue, or to express and justify an opinion. For such questions, you will be assessed on your answer, the *evidence* you present, and any *insight* based on these.
- Some of the questions ask for examples from object-oriented languages. Your answers need only refer to object-oriented languages discussed in the course, but you may refer to other languages if you wish.
- Non-electronic foreign language-English dictionaries are permitted.

Question 1. Object-Oriented Design [20 marks]

Responsibility Driven Design (RDD) uses Class-Responsibility-Collaboration cards (CRC Cards) as part of its design process.

(a) [5 marks] Breifly describe **how** RDD employs CRC Cards.

(b) [5 marks] What advantages do CRC cards have over other methods that can produce similar kinds of designs (for example, drawing UML class diagrams from scratch)?

(c) [10 marks] How can patterns be used with RDD and CRC cards? Discuss the advantages and disadvantages of using patterns with CRC cards.

You may consider Gamma et al. style Design Patterns, or/and other kinds of software patterns.

Question 2. Methodologies and Metrics [20 marks]

(a) [10 marks] How can software metrics be incorporated to improve the quality of software produced using the Rational Unified Process (RUP)?

You may consider the different artifacts to which you could apply metrics, and the RUP phases where metrics would be appropriate.

(b) [5 marks] What would you have to do differently to apply **heuristics** rather than metrics within the RUP?

(c) [5 marks] Briefly describe an heuristic that could be applied to improve Use Cases. Give an example of a small use case before and after applying the heuristic.

Question 3. Design and Meta-Design [20 marks]

A meta-level (reflexive) API is like a object-oriented framework for a language's implementation. Using reflection, programmers can access (and perhaps modify) the implementation of the programs they write.

(a) [10 marks] Do you think that using meta-level facilities is appropriate for general purpose programs? Why or why not?

(b) [10 marks] An object-oriented framework can provide its own meta-level API to allow programmers to access (and perhaps modify) the implementation of the framework. Describe the benefits and costs of such an API. Discuss how a framework's reflexive API is related to the reflexive API of the language in which it is written.

Question 4. Extreme-Oriented-Aspect-Programming [20 marks]

Discuss how Extreme Programming could be used to develop software in an Aspect-Oriented programming language.

You may wish to consider if there are features of Aspect-Oriented programming that would work particularly well (or particularly badly) with Extreme Programming, and also if there are features of Extreme Programming that would work particularly well (or particularly badly) with Aspect-Oriented programming.

Question 5. Language Design [20 marks]

You have been hired to help design a new object-oriented programming language by a company located in Mountain View, Callifornia.

This language is supposed to be easy to learn, to let programmers design programs that model the real world, and to be fast and powerful enough that people choose to write useful programs in it.

You have to decide which of the following features to include in your language.

(a). Single Subclassing as found in Smalltalk, Simula (and C#)

(b). Multiple Subclassing as found in C++ or Eiffel

(c). Subtyping (as in Java) — or alternatively, as in Smalltalk.

(d). Generics (also known as Parametric Polymorphism) as in Java 5, Eiffel, or C++.

(a) [4 marks] List these four language features in the order you would choose to have them in the language, from most important to least important.

(b) [16 marks] For each feature, say whether (or not) you would choose to include that feature in the language, and why (or why not) you have chosen to include that language feature.

Question 6. Object-Orientation [20 marks]

> *Object-Orientation as found in Smalltalk, Simula, C++, and Java, is really based on two principles:*
>
> *(a). A class can inherit from any other class.*
>
> *(b). An object can have a reference to any other object.*
>
> *Given these two principles, it is not in the least surprising that most object-oriented programs turn in to* ***"Big Balls of Mud****". The only surprise is that it sometimes takes so long.*
>
> Attrib. Thomas J. "Tad" Peckish.

Discuss.

****