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

EXAMINATIONS — 2008

MID-YEAR

COMP462 OBJECT-ORIENTED PARADIGMS

Time Allowed: 3 Hours

Instructions:

- *Read each question carefully before attempting it.*
- This examination will be marked out of **180** marks.
- Answer all six 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, the assessment will take into account the *evidence* you present and any *insight* you demonstrate.
- 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. Use Cases and the Unified Process [30 marks]

(a) [20 marks] Describe how **use cases** can be used in *each* phase of the Unified Process: **Inception Phase**, **Elaboration Phase**, **Construction Phase** and **Transition Phase**.

(b) [10 marks]

"Use Cases? — they're just ***not*** *object-oriented!"*

(Attrib. to Thomas J. "Tad" Peckish.)

Discuss.

Question 2. Framework Design [30 marks]

(a) [10 marks] Describe how the **Java Collections Framework** is simultaneously both a library and a framework.

(b) [10 marks] Explain how you would use metrics and heuristics to improve the design of an existing framework.

(c) [10 marks] How would you design a new framework from scratch?
Would metrics and heuristics help?

Question 3. Responsibility and Patterns [30 marks]

CRC cards and RDD use the notion of *responsibility* to structure designs.

(a) [10 marks] How are CRC cards used in the RDD design process?

(b) [10 marks] How could *design patterns* help you to use CRC cards and RDD?

(c) [10 marks] Explain how the notion of *responsibility* could help when describing design patterns.

Question 4. Extreme Programming [30 marks]

Extreme programming defines a number of *practices* that work together to support teams building software.

(a) [10 marks] Name and describe **five** extreme programming practices.

(b) [10 marks] Could *multiple inheritance* be used to support extreme programming?
If so, how? If not, why not?

(c) [10 marks] Could *meta-programming* be used to support extreme programming?
If so, how? If not, why not?

Question 5. Object Orientation [30 marks]

(a) [5 marks] Discuss whether *static types* are an essential part of the object-oriented programming paradigm?

(b) [15 marks] What are the benefits and liabilities of including static types in object-oriented programming languages? You may discuss Smalltalk, SIMULA, Eiffel, Java, C++, C♯, C♭, Python, Ruby, or other programming languages you know.

(c) [10 marks] Explain why *co-variance* is a problem for statically typed languages, and what sensible language designers should do about it.

Question 6. Aspect Orientation [30 marks]

Aspect-Oriented Programming uses language constructs like pointcuts, advice, and introductions (or intertype declarations) to achieve *seperation of concerns*. Discuss whether Aspect-Oriented Programming will stop programs turning into *Big Balls of Mud* —- or whether it will just make things worse?

****