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

EXAMINATIONS — 2010

MID-YEAR

SWEN423
OBJECT-ORIENTED PARADIGMS

Time Allowed: 3 Hours

Instructions:

- This examination will be marked out of **180** marks.
- Read each question carefully before attempting it.
- 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. Responsible Aspects [30 marks]

(a) [10 marks] Describe how CRC cards and the notion of responsibility support object-oriented design.

(b) [10 marks] Do you think CRC cards and responsibility are specific to designing object-oriented programs, or could they be used to help design programs in other kinds of languages — e.g. procedural languages, functional languages, scripting languages, query languages, Romance languages?

(c) [10 marks] In particular, do you think responsibility could help design Aspect-Oriented programs? Explain why or why not?

Question 2. Inheriting Frameworks [30 marks]

(a) [10 marks] Compare and contrast **frameworks** and **libraries**.

(b) [10 marks] Discuss whether **subclassing** and **subtyping** are equally important to both libraries and frameworks.

(c) [10 marks] What is the impact of **multiple inheritance** (of interfaces and/or implementations), **dynamic typing** (also known as Duck Typing), and **multiple dispatch** on library and framework designs?

Question 3. Extreme Smalltalk [30 marks]

"If Smalltalk did not exist, we would have to reinvent it." (Attrib. to Thomas J. "Tad" Peckish.)

Extreme Programming (and Agile methods in general) have been caricatured as "***Smalltalk-Oriented Programming***" — that is, programming in many other languages, and on commercial projects, *as if you were programming in Smalltalk in a research lab*.

Disuss whether you think this is a fair characterisation of Extreme Programming? Why or why not? How much do you think programming languages and paradigms influence the methods and practices that are used to develop software? Conversely, how much influence do development methods have on programming languages?

Question 4. Corpus, Metrics, Mud [30 marks]

"All software degenerates into a big ball of mud if it lives long enough"

(a) [15 marks] Describe how you could use software metrics to prevent a software project ending up as a big ball of mud. Does that mean that metrics are a "Silver Bullet"?

(b) [15 marks] Could you use corpus studies of large numbers of software systems to test whether it is in fact true that "All software degenerates into a big ball of mud"? If so, explain how; if not, explain why not.

Question 5. Objective Ownership [30 marks]

(a) [10 marks] In a pure, dynamically typed Object-Oriented language like Smalltalk, any instance variable (field) of an object can point to any other object. Do you consider this an important part of the Object-Oriented paradigm, where *everything is an object*. Explain why or why not.

(b) [10 marks] In a statically typed OO language like Java or C++, fields have types. Explain how these types restrict relationships between objects. Do these restrictions make the languages less Object-Oriented? Again, explain why or why not.

(c) [10 marks] Do you think object aliasing is a fundamental part of Object-Orientation? Should it be? You may consider techniques for avoiding some of the problems caused by aliasing, or alternatively, for preventing aliasing.

Question 6. Programming Language Design [30 marks]

(a) [10 marks] According to Sun, in 1997 there were five primary goals in the design of the Java Language:

- It should be ″simple, object oriented, and familiar″.
- It should be ″robust and secure″.
- It should be ″architecture neutral and portable″.
- It should execute with ″high performance″.
- It should be ″interpreted, threaded, and dynamic″.

Discuss how well you think Java has lived up to these goals.

(b) [20 marks] Due to an accident with a Hot Tub and a Time Machine, you find yourself back before the year 2000, where you are working for Sun Microsystems and managing the development of Java. At this point in this alternative history, Java did not have generic types or reflection. You have to decide what features to include in the next release of the Java languageg. Your three assistants, Generic Gilad, Reflexive Renee, and Harry the Hacker, each argue for the following:

- Gilad says: we should add Generics. Haskell and Eiffel have generics, and they'll stop people doing casts all the time.
- Renee says: we should add Reflection. Smalltalk has reflection, and it will be useful for writing graphical user interface builder tools.
- Harry says: we shouldn't add anything. If you make Java any more complicated, all us Hackers will stop using Java and hack in untyped scripting languages instead, like Ruby, Python, and PHP.

Explain which features (if any) you choose to include in Java 2000, and why.

****