TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS – 2014
Trimester 1

SWEN423 OBJECT-ORIENTED PARADIGMS

Time Allowed: 3 Hours

Instructions:

- Open book.
- This examination will be marked out of **180** marks.
- Read each question carefully before attempting it.
- Answer all five 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. Be careful in how you answer: is important to both show that you *understand* the topic but also that you can *explain* it properly *using accurate terminology*.
- Your answers need only refer to the topic discussed in the course, but you may refer to other topics if you wish.
- Printed foreign language-English dictionaries are permitted.
- Non- electronic student notes are permitted, including annotated SWEN423 books.
- No calculator permitted.

Question 1. Software development process [35 marks]

(a) [10 marks] Explain why in most cases, independently of the difficulty of the task at hand, "programmers write the most complex software that could possibly work"

(b) [9 marks] What do you think can help to mitigate this problem?

(c) [8 marks] Do you think language features like dynamic code loading, traits, active libraries, aspect orientations or family polymorphism could help? Justify your answer.

(d) [8 marks] Do you think that language guarantees, like static typing, strong exception safety, immutability or ownership could help? Justify your answer.

Question 2. Subtype vs Subclassing [35 marks]

Some papers in the recent literature completely separate the concept of Subtyping from Subclassing.

(a) [9 marks] How do languages like Java and C# deal with the issue of offering Subtyping without subclassing or vice versa.

(b) [8 marks] How do languages like Pyhon, Ruby or Smalltalk deal with the issue of offering Subtyping without subclassing or vice versa.

(c) Frameworks are a fundamental tool for programmers. Subtyping and Subclassing are language features widely used when implementing and reusing frameworks. Discuss

(i) [6 marks] What is the role of Subtyping in Frameworks? Is it a fundamental language feature allowing the existence of frameworks?

(ii) [6 marks] What is the role of Subclassing in Frameworks? Is it a fundamental language feature allowing the existence of frameworks?

(iii) [6 marks] Discuss how a better separation of Subtyping and Subclassing can help to design/use frameworks.

Question 3. Reflection and Metaclasses [35 marks]

(a) [12 marks] Compare the model of Meta Classes and Reflection in Java and in Smalltalk.

(b) [12 marks] Do you think one of these models is better or simpler than the other? motivate your answer.

(c) [11 marks] What is the paper "Mirrors" about? How does it relate to the way the standard Java library treats reflection?

Question 4. Family Polymorphism [35 marks]

(a) [10 marks] Show and comment on an example of Family Polymorphism that is not present in the course text book. The example should be useful for the purpose of understanding the concept of Family Polymorphism.

(b) [10 marks] Explain what Family Polymorphism is, and what it is good for. You are encouraged to refer to your own example.

(c) [15 marks] Show why a naive type system would be unsound in the presence of Family Polymorphism.

Question 5. Designing Modular Systems [40 marks]

You are the founder of "JavaW". Your three cooperators/employers all have a high knowledge of Java, in addition: Mike is an expert in algebra and calculus, Rose is an expert in data representation and storage, and Alice is an expert in dynamic class loading, aspect orientation and domain specific languages.

You are a team manager, and you have to design a modular system implementing the features described below.

United High-School Association of New Zealand want to realize a teaching tool to teach algebra and calculus.

- Equations should be represented on the screen,
- Through an intuitive graphical user interface, students should be able to select different parts of the equation and apply different rewriting rules (there are many of such rules, around 100).
- This allows students to solve a set of problems provided by the various lecturers. Problems can be added and removed during the years.
- The results of the students should be automatically recorded on a server while the student is doing the exercise.

(a) [40 marks] Outline a high-level plan, such that could be understood by a software expert.

You should divide the problem into at least 5 different parts that can be modularly tested. For each part identify:

1 The name of the component and a short description

2 programmers involved (between you, Mike, Rose and Alice)

3 Relationship and dependency with other parts.

4 An example testing strategy.

5 The main design patterns / programming techniques involved.

* * * * * * * * * * * * * * *