TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS – 2017

TRIMESTER 2

SWEN 423

OBJECT-ORIENTED PARADIGMS

Time Allowed: THREE HOURS

OPEN BOOK

Permitted materials: Any materials except electronic devices may be brought into the examination.

Instructions: Answer all questions
This examination will be marked out of **180** marks.
Read each question carefully before attempting it.
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 or explain 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.
Your answers need only refer to topics discussed in the course, but you may refer to other topics if you wish.

Question	Topic	Marks
1.	Simulacra and Simulation	30
2.	Libraries, Frameworks, and Patterns	30
3.	Object-Orientation	30
4.	Systems Design	30
5.	Types	30
6.	Java and Javascript	30
	Total	180

1. Simulacra and Simulation **(30 marks)**

   (a) **(10 marks)** Explain the slogan *"All programming is simulation"* .

   (b) **(10 marks)** How do CRC Cards relate to (or rely on) simulation?

   (c) **(10 marks)** Does this slogan apply to other kinds of programming (procedural, functional, logic, scripting...)? Explain why or why not.

2. Libraries, Frameworks, and Patterns **(30 marks)**

   Aristotle wrestled with the problem that *"there could not have been a first egg to give a beginning to birds, or there should have been a first bird which gave a beginning to eggs; for a bird comes from an egg."*

   Patterns can be used to create frameworks; libraries and frameworks can be analysed to create patterns; patterns can be used to describe libraries or frameworks; some libraries seem to be frameworks (like the Java Collections Framework); some frameworks seem to be libraries or patterns or even applications (like Hotdraw).

   (a) **(8 marks)** Define:

      1. "applications"
      2. "libraries"
      3. "frameworks"
      4. "patterns"

   (b) **(12 marks)** Describe the relationships between "applications", "libraries", "frameworks", and "patterns"

   (c) **(5 marks)** Would you recommend that a team adopts extreme programming to build frameworks? Explain why or why not.

   (d) **(5 marks)** Would you recommend that a team adopts the Self programming language to build frameworks? Explain why or why not.

3. Object-Orientation **(30 marks)**

   Smalltalk, Self, Eiffel, and Newspeak are **pure** object-oriented languages, while Java, C++ and Python are **hybrid** object-oriented languages.

   (a) **(10 marks)** Give three examples of how the programming experience of an hybrid language differs from programming in a pure language.

   You should use brief code snippets to illustrate the examples, showing both hybrid and pure versions; those code snippets can be written in any language syntax.

   (b) **(10 marks)** Describe how a pure object-oriented language contributes to modular extensibility and interoperability

   (c) **(10 marks)** Which kind of language — pure or hybrid — can be implemented most efficiently? Explain why.

4. Systems Design **(30 marks)**

(a) **(10 marks)** In an object-capability system, permissions may be shared by:

1. Parenthood
2. Endowment
3. Introduction
4. Initial conditions

Give brief code snippets to illustrate each of these four mechanisms (in any language syntax), and explain what they are.

(b) **(5 marks)** Explain why Smalltalk *is not* an object-capability language.

(c) **(15 marks)** Dan Ingalls identifies sixteen design principles behind Smalltalk. These principles include:

Personal Mastery: If a system is to serve the creative spirit, it must be entirely comprehensible to a single individual.

Storage Management: To be truly "object-oriented", a computer system must provide automatic storage management.

Uniform Metaphor: A language should be designed around a powerful metaphor that can be uniformly applied in all areas.

Classification: A language must provide a means for classifying similar objects, and for adding new classes of objects on equal footing with the kernel classes of the system.

Reactive Principle: Every component accessible to the user should be able to present itself in a meaningful way for observation and manipulation.

Operating System: An operating system is a collection of things that don't fit into a language. There shouldn't be one.

Natural Selection: Languages and systems that are of sound design will persist, to be supplanted only by better ones.

Choose the *three* principles that contribute most to the security and reliability of a system's design, and explain the contribution of each principle.

5. Types **(30 marks)**

The Grace programming language does not support overloading. This means a Grace object can only ever have one method of any given name, and static argument type declarations never affect which method is called.

In Grace some class `C` can implement an interface `I` if and only iff:

1. **Method Inclusion:** `C` implements every method declared in `I`.
2. **Covariant Return Types:** The return type for a method in `C` is a subtype (i.e. more specific) than the return type for the corresponding method in `I`.
3. **Contravariant Argument Types:** The argument types for a method in `C` are supertypes (i.e. more general) than the corresponding argument types for the corresponding methods in `I`.

(a) **(15 marks)** For each of the three rules above, give a small code snippet (in any syntax) to illustrate what can go wrong if the rule is broken, and briefly explain what goes wrong in your code.

(b) **(5 marks)** Java's and C♯'s built-in arrays break rule 3: an array's operations argument types are covariant in the type of the array's elements. Explain what Java and C♯ implementations do to ensure that covariant arrays work correctly.

(c) **(10 marks)** Do you think object-oriented languages should treat subtyping and subclassing as two different relationships, or should subclassing and subtyping be tightly coupled together? Justify your opinion.

6. Java and Javascript **(30 marks)**

(a) **(10 marks)** The following code outlines the core of an implementation of "`eval`" in a class-based language called "FooL".

```
1  FooLObject eval(FooLExpression e) {
2    FooLObject receiverObject = eval(e.receiver());
3    List<FOoObject> argumentObjects = new ArrayList<>();
4    for (a : e.arguments()) {
5      argumentObjects.add(eval(a));
6      }
7    return apply(
8       lookup(e.methodName(), receiverObject),
9       receiverObject, argumentObjects);
10 }
11
12 FooLMethod lookup(String name, FooLObject rcvr) {
13   Map<String, FooLMethod> m = rcvr.getFooLClass().getMethodMap();
14   if (!m.contains(name)) {throw new FooLException(‘‘Method Not Found’’);}
15   return m.get(name);
16 }
```

Explain how you would change the implementation of the `eval` or/and `lookup` methods to change FooL from a class-based language to a prototype-based language. What other main changes would be required in FooL's implementation?

(b) **(20 marks)** Discuss the following quotation with reference to **Java** and **Javascript.**

> Java is a tragedy, but Javascript is a joke.
>
> Thomas J. "Tad" Peckish.

Amongst other topics, you may wish to consider some of the following:

1. Classes vs. Prototype- based languages
2. Static vs. Dynamic types
3. Single vs. Multiple inheritance
4. Overriding vs. Overloading
5. Field declarations vs. Assignments
6. Generics
7. JSON

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