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

EXAMINATIONS – 2015

TRIMESTER 1

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 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.

Question 1. API Design [40 marks]

Joshua Bloch, in "How To Design A Good API and Why it Matters" talks about API design.

(a) [10 marks] "When in doubt, leave it out."
Explain what Bloch means with this catch-phrase and if you agree with him explain why that point is important, otherwise explain your different point of view.

(b) [10 marks] Joshua Bloch believes that API design and programming language design are tightly connected.
Explain what he meant (possibly with an example) and if you agree with him explain why that point is important, otherwise explain your different point of view.

(c) [10 marks] He suggests to "Design and Document for Inheritance or Else Prohibit it" and "Document self-use", that is, document how public methods are used internally. Explain why this is important; use code examples.

Be sure to mention the following points:

- How to "Prohibit" inheritance in Java.
- Show an example where version 2 of an API can break user code if self-use is not documented/considered.
- Show an example of code Designed and Documented for Inheritance.

(d) [10 marks] Joshua Bloch encourages us to make "APIs Hard to misuse". How can we make a Java API hard to misuse? What features/languages we have seen that can make well designed APIs harder to misuse compared to those written in Java?

Question 2. Traits [30 marks]

(a) [9 marks] Discuss the concept of traits and their main properties.

In some approaches using a trait induces subtyping, so traits can be used as types, sort of like interfaces. In other approaches traits are not types, and are instead a pure unit of code reuse.

(b) [7 marks] Explain advantages of having traits inducing subtyping.

(c) [7 marks] Explain advantages of having traits as a pure unit of code reuse.

(d) [7 marks] What kinds of traits operators can be supported in the "traits inducing subtyping" model and what in the "traits as a pure unit of code reuse" model.

Question 3. Multiple Inheritance [40 marks]

We have seen how introducing multiple inheritance in C++ was controversial. With the better understanding that we have now, we can distinguish two sub-problems: "Supporting code reuse from more than one source" and "Supporting multiple supertypes".

(a) [4 marks] How important and common is the opportunity of reusing code from more than one source?

(b) [7 marks] Make an example where is useful to reuse code from more than one source

(c) [4 marks] How important is the need of supporting multiple supertypes?

(d) [7 marks] Give an example where supporting multiple supertypes is important.

(Question 3 continued on next page)

(e) [9 marks] In Multiple Inheritance, supporting code reuse from more than one source seems to require supporting multiple supertypes, since the implicit parameter **this** could have been used in the original (multiple) source contexts. Consider the following code:

```
class PrettyPrinter{
  static void printParentA(ParentA p){...}
  static void printParentB(ParentB p){...}
}
class ParentA{
  void printMe(){ PrettyPrinter.printParentA(this);}
}
class ParentB{
  void printMeToo(){ PrettyPrinter.printParentB(this);}
}
class Merged extends ParentA, ParentB{}
..
Merged m=new Merged();
m.printMe();
m.printMeToo();
```

Explain what the static/dynamic types of "**this**" and parameter "p" are in this code, and what relations this has with subtyping relations.

(f) [9 marks] Being required to induce subtyping when reusing code (as shown above) is often considered a problem and a limitation of current mainstream languages. Do you agree? Supporting "traits as pure unit of code reuse" is a solution. Can you explain why? Can you explain some other possible solutions?

Question 4. Immutability [35 marks]

There are two main kinds of immutability proposed in literature:

- Shallow immutability, where a single object is immutable, but its field can point to mutable objects.
- Deep immutability, where all objects in the reachable object graph pointed by an immutable object, are immutable.

(a) [4 marks] Does Java support one of those models? if so, how?

(b) [13 marks] Explain advantages and disadvantages of both models.

(c) [8 marks] Do you think one of these models is better or simpler than the other? Explain your point of view on this topic.

(d) [10 marks] Another dimension is class-based immutability or object based immutability:

- In class-based immutability, all the instances of immutable class are immutable, and all instances of mutable classes are mutable.
- In object-based immutability, some objects are immutable and some are mutable, and there is no concept of mutable/immutable class.

Explain advantages and disadvantages of both models.

Question 5. Active libraries [35 marks]

(a) [5 marks] Describe the concept of "Active Libraries"

(b) [5 marks] Discuss some advantages of active libraries.

(c) [5 marks] Discuss some disadvantages of active libraries.

(d) [10 marks] Provide an original example of an active library.

(e) [10 marks] Generic collections can be encoded by active libraries. First explain how this can be done, then explain advantages and disadvantages of such encoding.

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