# DEGREE EXAMINATIONS — 1998



MID-YEAR

COMP 462

OBJECT-ORIENTED PARADIGMS

Time Allowed: 3 Hours

Instructions:

- *Read each question carefully before attempting it.*
- This examination will be marked out of **150** marks, so allocate approximately 1 minute per mark.
- Questions 2 and 3 have some choice. Do not answer more than is asked — *the extra answers will be ignored.* Cross out any solutions you do not wish to have considered.
- 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 express and justify an opinion. For such questions, you will be assessed on your *justification.*
- Many 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.

## Question 1. Compulsory question [30 marks]

A recurring concept in object-oriented languages is that of *inheritance*. Even languages that do not have an explicit notion of inheritance have something that is related. Explain the common aspects of inheritance and discuss some of the variations on inheritance. You should give examples of at least three (3) languages, including at least one with a non-standard form of inheritance.

## Question 2. Long answer questions [60 marks]

Discuss any **two** of the following topics (each worth 30 marks).

**(a)** [30 marks] *Polymorphism* is one of the defining characteristics of object-oriented programming languages. Explain what polymorphism is, give an example of its use, and discuss how it makes some aspect of software development easier.

**(b)** [30 marks] *Encapsulation* is regarded as an important concept for managing complexity in software. In Smalltalk's encapsulation model, all instance variables are private (that is, they cannot be accessed except by the object's own methods) and all methods are public (that is, they may be accessed by any other object).

Discuss how well Smalltalk's encapsulation model helps manage complexity. Your discussion should include a comparison with the encapsulation model of at least two (2) other languages.

**(c)** [30 marks] The notion of *subtypes* is used to determine whether one value can be *substituted* for another. Different object-oriented languages take different approaches to specifying when one type is a subtype (and hence, can be substituted) of another, and so have different characteristics. The three main approaches are: dynamic testing (as in Smalltalk), explicit (as in C++, Java, Eiffel), and implicit (as in Emerald). Discuss the relative advantages and disadvantages of these three approaches.

**(d)** [30 marks] Object-oriented programming is about "objects". Discuss what is generally meant by "object". Your discussion should include such things as what an object looks like, what properties it has, how one identifies what objects should appear in a system, and how objects are used to implement a system. You should use the terminology commonly used when discussing objects.

## Question 3. Short answer questions [60 marks]

Answer any **six** of the following questions (each worth 10 marks).

**(a)** [10 marks] Explain, with examples, what a "prototype"-based language is, and how it differs from a class-based language.

**(b)** [10 marks] There were two languages discussed in class (CLOS and Cecil) that provided *multi-methods*. What problem is multi-methods intended to solve?

**(c)** [10 marks] Multiple inheritance means that classes can have more than one parent. Discuss, with examples, two problems that languages providing multiple inheritance have to deal with.

**(d)** [10 marks] Discuss the support for encapsulation provided by CLOS. Your discussion should also cover why the CLOS designers chose the form of encapsulation they did.

**(e)** [10 marks] In *"Increasing Java's expressiveness with* `ThisType` *and match-bounded polymorphism"* by Kim B Bruce, he proposed the notion of *exact types*. If the type of something is specified as being exact, then it must always have a value of that type, and in particular, can never have a value of a subtype of that type. C++ can achieve a similar (although not the exact) effect by specifying that the methods for the required type be non-virtual. Discuss the advantages and disadvantages of having a feature like exact types in a programming language.

**(f)** [10 marks] In *"Designing reusable classes"* by Ralph E. Johnson and Brian Foote, the authors discussed a number of reasons why object technology helps improve reusability of code. One of the main reasons they gave for this was the support object technology gives for what they referred to as "common protocols". Explain what they meant by common protocols, and discuss whether their claim is generally applicable or just specific to the language they were using, namely Smalltalk.

**(g)** [10 marks] What are the issues associated with including *concurrency* in the Object Model? Your answer should include examples describing how this has been done in at least two languages.

**(h)** [10 marks] Briefly describe two techniques that might be used in Object-Oriented Analysis and Design to determine what classes (and behaviour) should be in an application.

**(i)** [10 marks] Java does not have multiple inheritance, but it does have the `interface` mechanism. Explain how this mechanism can be used to provide one of the benefits of multiple inheritance.

**(j)** [10 marks] Briefly discuss how object technology might be used to provide the "base" interface and "meta" interface ideas that Kiczales proposes (from either the video *"Why Black Boxes are so Hard to Reuse: A New approach to Abstraction for the Engineering of Software"* or the paper *"Foil for The Workshop on Open Implementation"*) to deal with the problems associated with Black Box Reuse.

**********