DEGREE EXAMINATIONS — 1996

COMP 462

Object-Oriented Paradigms

Time Allowed: THREE 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.*

continued...

1. [30 marks]

The subject of the course is *object*-oriented paradigms, despite the prominence of the concepts *class* and *inheritance*. In fact, there are object-oriented languages without classes and languages without inheritance, but the concept of *object* is common to all languages that claim to be object-oriented. Discuss why the concept of object has proved so popular.

2. [60 marks]

Long answers: Discuss any **two** of the following topics (each worth 30 marks). Your discussion need only refer to object-oriented languages discussed in the course, but you may refer to other languages if you wish.

(a) Various languages claim to be "pure" object-oriented programming languages. Chambers offered the following explanation as to what this means:

> All data are objects, and message passing is the only way to manipulate objects. Even instance variables are accessed solely using message passing. This purity offers the maximum benefit of object-oriented programming, allowing code to manipulate an object with no knowledge of (and hence no dependence on) its underlying representation or implementation.

Based on this quote, discuss what you think is generally meant when it is claimed that a language is "pure" object-oriented.

(b) Encapsulation is regarded as an important feature because it reduces the cost of developing and maintaining software. Discuss the relevance and importance of encapsulation in the object model. You should include discussion on what encapsulation is, and what agreement (if any) there is about how encapsulation is provided in different object-oriented languages.

(c) Polymorphism in object-oriented languages generally means that the actual code that is executed is determined at run-time. Discuss the different ways that polymorphism is provided in object-oriented languages.

(d) In Smalltalk, all classes fit into one hierarchy with the class `Object` at the top whereas C++ has many small hierarchies. Compare and contrast the advantages and disadvantages of the different approaches. Your answer may refer to approaches taken by other languages.

continued...

3. [60 marks]

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

(a) Smalltalk was designed around a small set of rules, two of which were:

- everything is an object
- all objects belong to a class

What language feature was a consequence of these rules? Explain your answer.

(b) Object-oriented languages generally provide two ways to build classes out of existing classes, namely *composition* (also known as aggregation) and *inheritance*. Explain what these are, and discuss why they are both needed.

(c) Type systems in different object-oriented languages have different notions of what type checking means. For example, Emerald uses *contravariance* whereas Eiffel uses *covariance*. Explain the difference between these two notions and discuss the relevance of such notions to object-oriented paradigms.

(d) Many object-oriented languages have automatic memory management (garbage collection). Is this (or should it be) inherent in object-oriented programming? Justify your position.

(e) Some languages that claim to be object-oriented do not have inheritance, but instead have *delegation*. Explain what delegation is and compare it to inheritance.

(f) A common argument against providing multiple inheritance is that it introduces ambiguity, since the same name may be used by more than one ancestor class. Discuss the different ways that languages that have multiple inheritance deal with the ambiguity problem.

(g) Many languages provide mechanisms that allow encapsulation to be "broken", for example friends in C++, "friendly" fields in Java, and feature access control in Eiffel. Explain why the designers of these languages felt such mechanisms were needed.

(h) There were two languages discussed in class (CLOS and Cecil) that provided *multi-methods*. What problem are multi-methods intended to solve?

(i) Many object-oriented database management systems (OODBMS) have, as well as attributes and methods, the concept of *complex values*. What are complex values and why are they considered important in the OODBMS world?

****