S228/209

DUBLIN INSTITUTE OF TECHNOLOGY
KEVIN STREET, DUBLIN 8

BSc (Honours) Degree in Computer Science

YEAR 2

Semester 2 Examination
2013/2014

CMPU 2020

Software Engineering II

Mr R. Lawlor
Dr D. Lillis
Mr P. Collins

Wednesday 21st May
1.00 pm - 3.00 pm

Two hour exam
Attempt four out of five questions
All questions carry equal marks

1. **(a)** Discuss the pros and cons of class inheritance versus interface inheritance in building flexible and loosely coupled reusable software. Contrast C# and C++ in their support for interfaces.

(12 marks)

(b) Suppose you wish to design a class for a graphic object which could be a simple primitive object like a circle or rectangle, or a more complex object assembled from simpler ones. The subcomponents of a complex object could themselves be primitive objects of subassemblies.

Suggest a design pattern which would be suitable for this problem and describe it in UML and with some comments. Also outline its advantages and disadvantages if any.

(13 marks)

2. You are required to do some object-oriented design for a standalone restaurant software system that mainly manages bookings. The restaurant software should be able to handle advance reservations, walk-in bookings, assigning tables to reservations and so on.

(a) A layered architecture allows for separation of concerns. Explain what is meant by this. Then describe an appropriate layered architecture for the restaurant system given that it will be implemented as standalone software.

(6 marks)

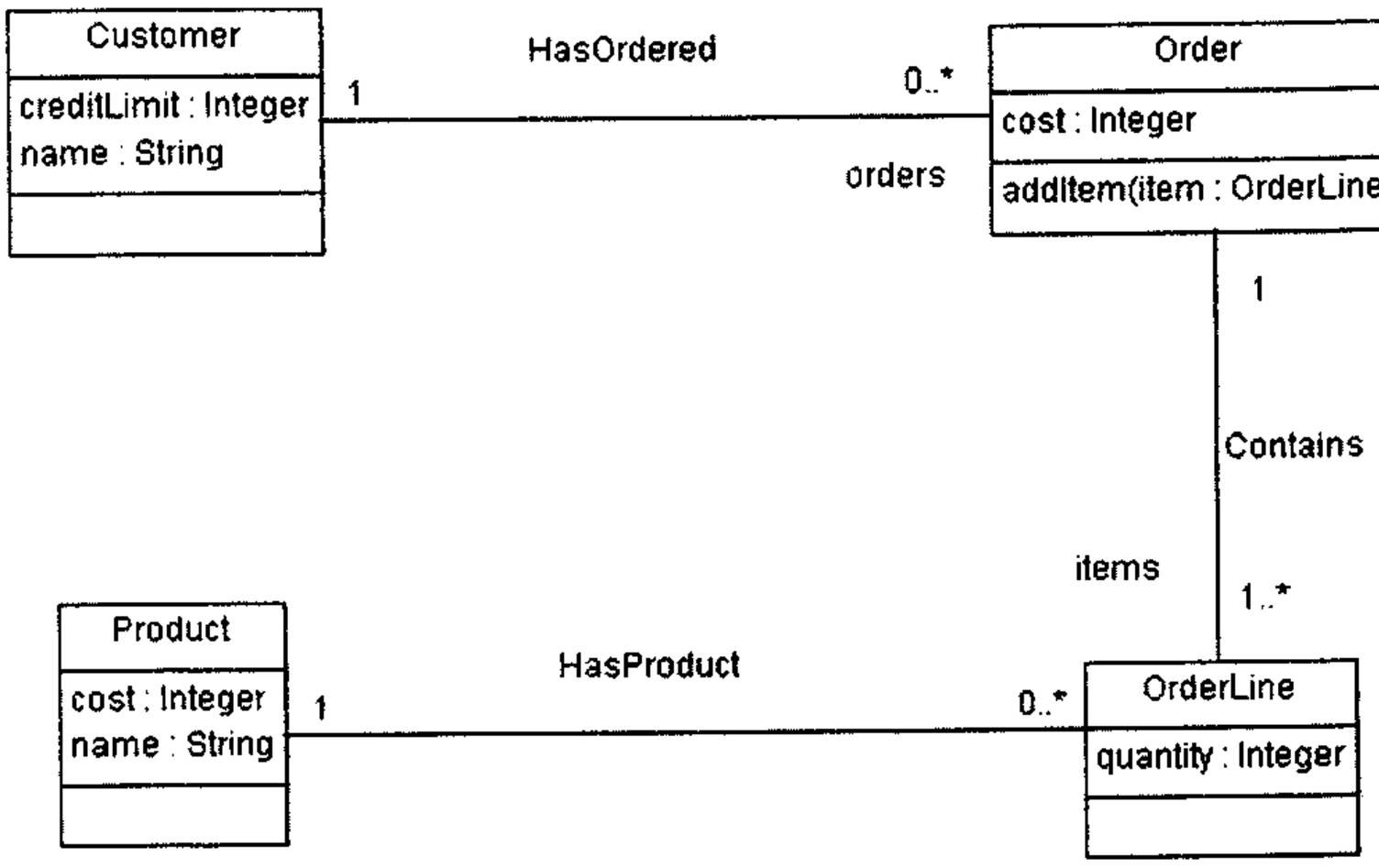
(b) Describe the structure and purpose of the Observer design pattern and in which generic situations it might be applicable. What role could it play in the design of the restaurant system?

(7 marks)

(c) Show and explain in writing, with the aid of a package/class diagram and a sequence diagram, how the Observer pattern could be incorporated into the design of the restaurant system to couple application code with user interface code.

(12 marks)

3. The following class diagram is an incomplete UML specification. OCL will be used to specify some additional business rules and in order to test the specification it has been decided to model it in USE (UML Specification Environment).



(a) The business rules to be added to the specification are listed below. Show how they can be expressed in OCL.

i) Before adding any order line to an order, the quantity of the particular product has a maximum value of 10.

ii) When adding a new order line to an order with *addItem()*, the customer credit limit must be greater than or equal to the cost of the order line plus the existing cost.

iii) When a new OrderLine object *item* has been added to an order, then:

orderLines_after_operation = orderLines_before ∪ {item}

(9 marks)

(b) Briefly explain how the preconditions and postcondition in part (a) could be tested in USE with the help of !openter and !opexit.

(8 marks)

(c) Provide a rough C# implementation of the Order and OrderLine classes paying particular attention to the *Contains* association. Outline how preconditions for *addItem()* could be incorporated in the code.

(8 marks)

4. (a) Explain what is meant by *Design by Contract* (DbC).

(8 marks)

(b) Comment on DbC from the following viewpoints:

i) Precision versus Detail

ii) DbC and Quality Assurance

(9 marks)

(c) Write an appropriate contract in Spec# for a method `int ISqrt(int x)` that computes the integer square root of x.

Supposing you then implemented the method in C#, what could be done with the Spec# contract?

(8 marks)

5. (a) Describe six of the key practices of the agile methodology XP.

(12 marks)

(b) Discuss the diagram below from the point of view: Anticipatory Design versus Refactoring.

(13 marks)

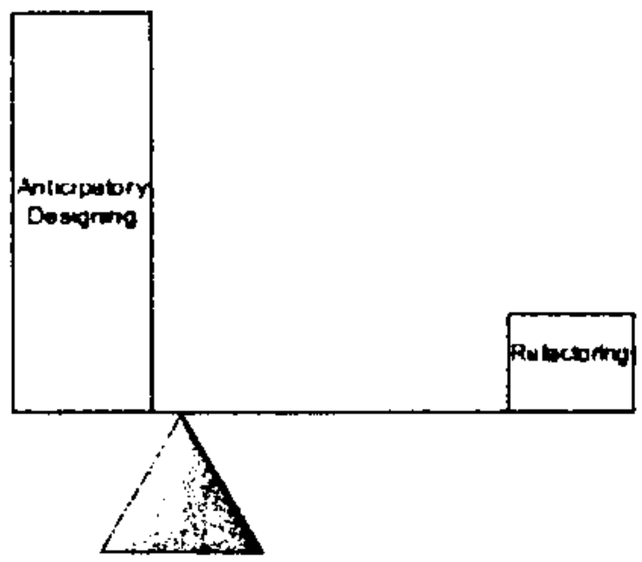


Figure 3 — Balancing design and refactoring, pre-internet.

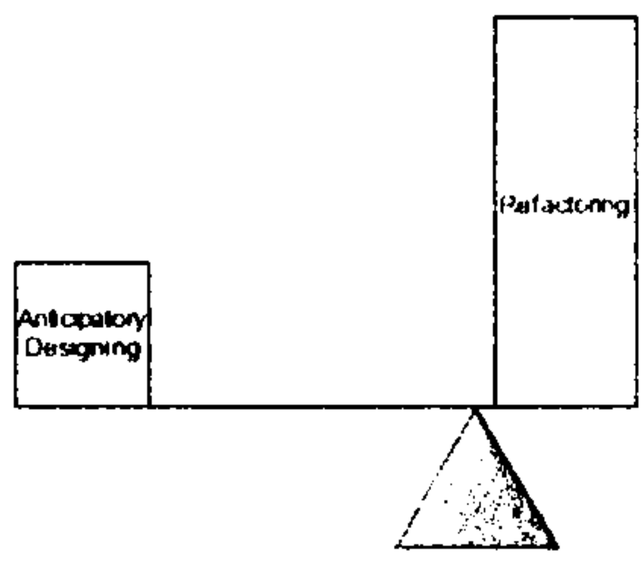


Figure 4 — Balancing design and refactoring today.