Programme Code: TU856
Module Code: CMPU2020
CRN: 22394

TECHNOLOGICAL UNIVERSITY DUBLIN

GRANGEGORMAN

TU856 - BSc. (Honours) Degree in Computer Science

Year 2

SEMESTER 2 EXAMINATIONS 2022/2023

CMPU2020 - Software Engineering 2

Internal Examiners:
Mr. Richard Lawlor
Dr. Paul Doyle

External Examiner:
Ms. Pamela O'Brien

Instructions To Candidates:

Attempt four out of five questions. All questions carry equal marks.

Exam Duration: 2 hours

Special Instructions /Handouts/ Materials Required: none

1. (a) Briefly describe and distinguish between the following types of software testing:

- unit
- integration
- functional

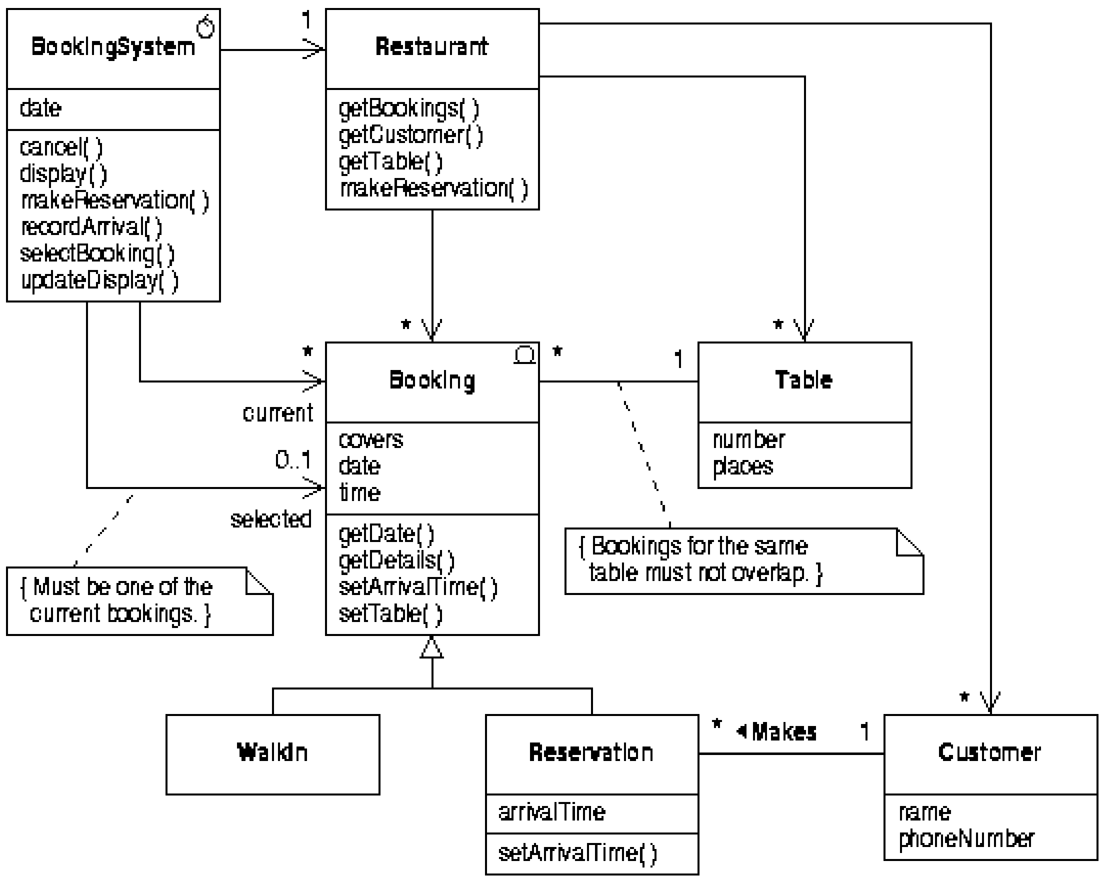
In your answer, make reference to white-box and black-box testing and mention who is responsible for writing the different levels of tests in a software development project.

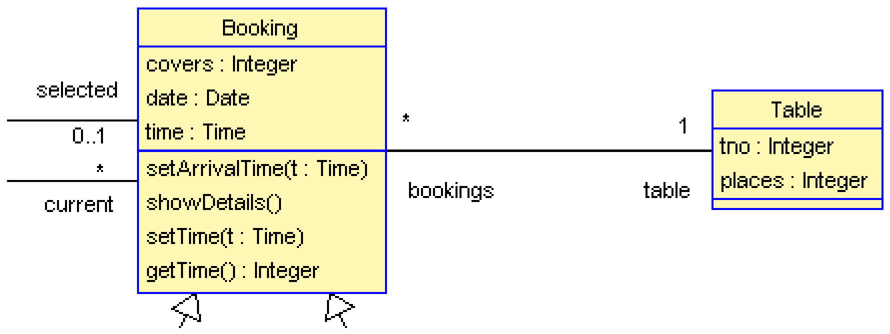
(15 marks)

(b) Describe the Adapter design pattern and distinguish between the class adapter and object adapter approaches.

(10 marks)

2 (a) Draw a statemachine for the BookingSystem control class for the partial restaurant domain models shown below. Write the operation appropriate for each state transition and an appropriate guard condition for recordArrival() which states that the table capacity must be greater than or equal to the number of people in the booking.





(8 marks)

(b) Draw sequence diagrams for the use-cases display() and recordArrival(). Is recordArrival() linked with or dependent on any other use-cases?

(10 marks)

(c) Provide an OCL version of the two constraints shown above.

(7 marks)

3. 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) As part of the design for the restaurant software it was decided to control access to the underlying functionality by creating a single instance of the control class *BookingSystem* and passing all requests through it.

What design pattern is implicit in this?
Provide some code fragments for the class *BookingSystem* showing how this design idea can be realised.

(6 marks)

(b) Describe the structure and purpose of the Observer design pattern and in which generic situations it might be applicable.

(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)

4. (a) Explain what is meant by *Design by Contract* (DbC). Elaborate on how a contract is affected by subclassing/polymorphism.

(9 marks)

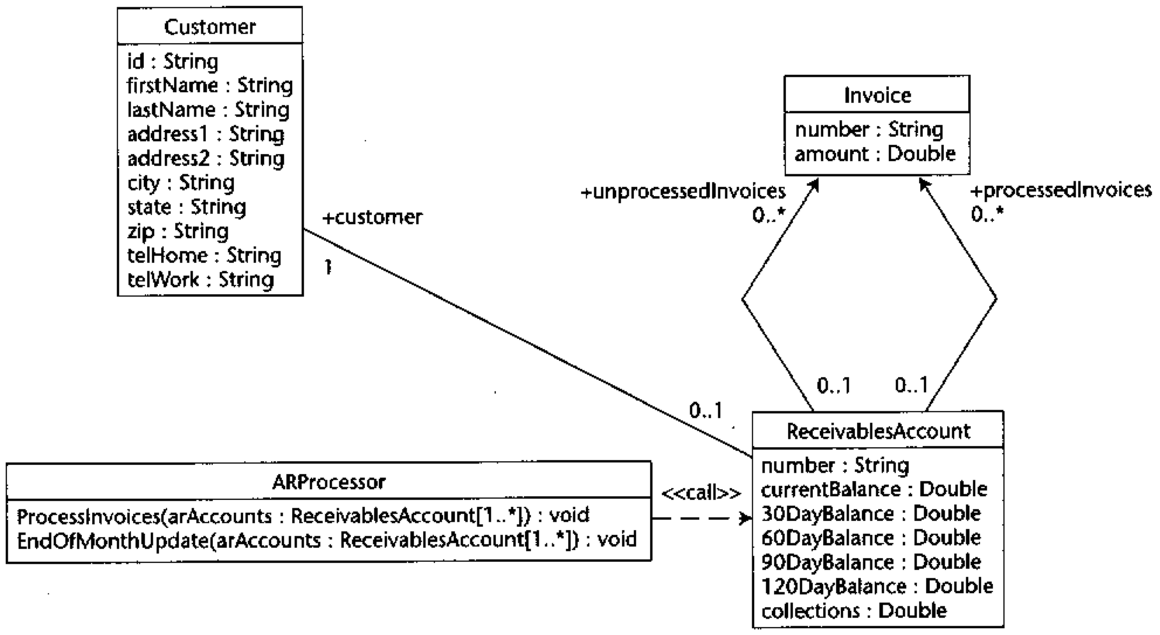
(b) Within the context of DbC, comment on benefits and obligations for both client code and provider code. Mention when exceptions might be appropriate.

(6 marks)

(c) Given the class diagram below, write an *Object Constraint Language* (OCL) contract invariant for *ReceivablesAccount* which states that no invoice can be in both *processedInvoices* and *unprocessedInvoices* collections at the same time.

Write an OCL contract that you deem appropriate to express the business logic *ProcessInvoices()* operation of the class *ARProcessor*.

(10 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)

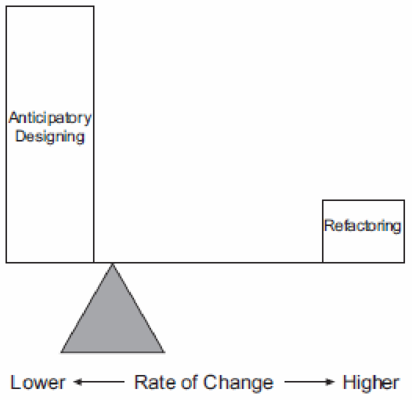


Figure 3 — Balancing design and refactoring, pre-internet.

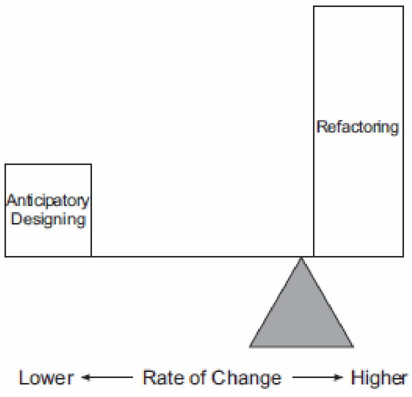


Figure 4 — Balancing design and refactoring today.