UNIVERSITY OF ABERDEEN SESSION 2015/2016

Examination in CS3028 (Principles of Software Engineering)

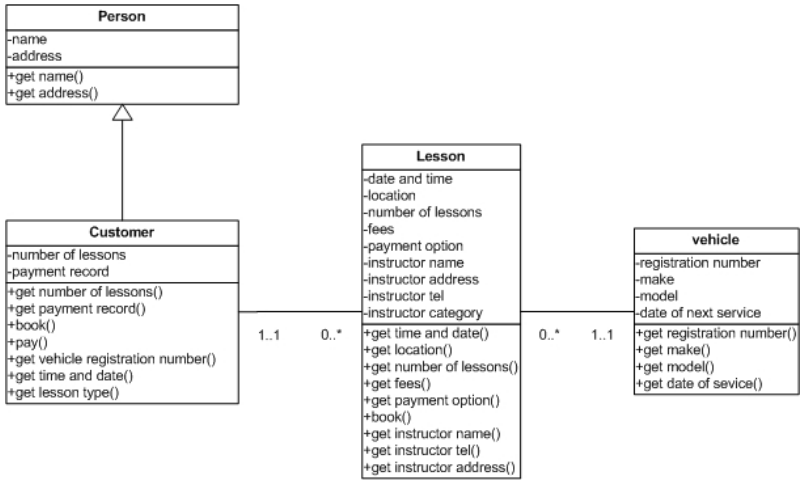
14 December 2015 (15.00 – 17.00)

Candidates are not permitted to leave the Examination Room during the first or last half hours of the examination.

- *Answer TWO questions.*
- *Use a separate answer book for each question.*
- *Each question is worth 25 marks; the marks for each part of a question are shown in brackets.*

1 (a) Discuss the difference between functional and non-functional requirements. (4)

(b) Consider the following UML class diagram, which shows some ≪entity≫ classes for a Driving school management system.



(i) Refer to the above UML class diagram. Note that class `Person` in this UML diagram must be considered as an abstract class and that (ii) information on customers, lessons, and vehicles needs to be permanently stored on a database. Discuss what ≪boundary≫ classes there should be in the storage layer. (5)

(ii) Refer to the above UML class diagram. Discuss whether coupling between classes `Person` and `Customer` is higher or lower than coupling between classes `Customer` and `Lesson`. (4)

(iii) Refer to the above UML class diagram. Discuss whether class `Lesson` is an example of functional class cohesion (which represents the highest level of class cohesion). (4)

(c) Describe white-box testing and black-box testing, explaining in what context you would use each of them. (4)

(d) Suppose that you head a small software development company and that you want to improve the quality of the software you develop. Describe the activities you will consider introducing into your organisation in order to produce higher quality software. (4)

PLEASE TURN OVER FOR QUESTION 2

2 (a) Describe the different kinds of requirements the FURPS+ model focuses on. (5)

(b) Consider the *broker architecture*. Draw a UML sequence diagram to show the interaction between `client`, `broker`, and `server` objects. (3)

(c) Consider the three-tier system architecture, which is composed of a presentation (GUI) layer, an application logic (information processing) layer, and a storage (database) layer. You may want to distribute these layers across a network, giving rise to the following two different situations:

1. The presentation layer is on one side, while the business process and the storage layer are on the other.
2. The presentation and the business process layers are on one side, while the storage layer is on the other.

Discuss the advantages and the disadvantages of each of these two distribution approaches. (5)

(d) As part of a commissioned software development project, a group of analysts has been asked to produce a Requirements Specification Document (RSD). This contains a detailed description of all the use cases identified during the requirements elicitation and determination phase. Each use case, which is described using structured text with no accompanying diagrams, contains a brief specification of use case goals, participating actors, preconditions, postconditions, expected flow of events and alternative flows of events. Despite the completeness of the textual use case description, the customer who commissioned the project argued that the analysts' specification is not complete because UML class diagrams are missing.

Discuss whether this position is acceptable or not. (4)

(e) Agile approaches clearly state that a suite of black box test cases should be attached to each User Story. What kind of black box techniques would you consider as most suitable to create a small but spectrum-covering suite of test cases? (4)

(f) Agile software development approaches put major emphasis on early code development, assuming that documentation written before actual code development (namely, requirements specification and architectural/detailed design documents) is pretty much useless. Advocates of agile methods argue that initial requirements and design are always subject to major changes during code development, which is a highly iterative process. Hence, preliminary documentation of requirements and design can be reduced to a bare minimum as it would only be used as a throwaway 'paper' prototype used to kick-start the coding process.

Discuss whether the agile operational philosophy can be accommodated within the Unified Process (UP) approach to software development. (4)

PLEASE TURN OVER FOR QUESTION 3

3 (a) Consider a computerised burglar alarm equipped with a range of sensors that detect out-of-normal environment conditions. For instance, movement detectors are used by this system to check whether an intruder is in a house. Once some suspicious movement is identified, the computer raises an alarm. Discuss the most suitable architectural pattern(s) upon which the design of the above system should be based. (4)

(b) Describe the *creator* GRASP pattern and provide an UML example of responsibility allocation using this pattern. (4)

(c) Consider the following 'simplified' use case, which describes how multiple units of the same products are scanned for sale at a supermarket till and are consequently removed from the supermarket product database.

Use Case Name: Sell multiple product units
Primary Actor: Checkout assistant
Brief Description: This use case describes how two or more units of the same product are scanned for sale by the checkout assistant during a sale and are consequently removed from the supermarket product database.
Normal Flow of Events: 1. The checkout assistant initiates the action. 2. The checkout assistant scans the product barcode. 3. The system displays product description and unit price. 4. The checkout assistant enters number of product units. 5. The system displays total cost for given product units. 6. The checkout assistant accepts the displayed product description and number of units. 7. The system commits the transaction, removing the purchased number of available product units from the corresponding product record. 8. The checkout assistant ends the action.
Alternate Flow of Events: • At (2) the product with the scanned code does not exist in the product database —- transaction aborts. • At (3) — maybe because the product is too expensive for the customer to buy — the checkout assistant aborts the purchase of that particular product. • At (6) the checkout assistant aborts the purchase of that particular lot of product units.

Discuss whether this use case provides a coherent and complete picture of the functionality it aims to specify. Explicitly describe wrong and/or missing use case parts and/or events, if any. (5)

(d) Describe any five properties of software that characterise testability. (5)

(e) You have been asked to test a program that computes body mass index from two user inputs, namely height and weight. Using black-box technique(s), design test cases required to test this program. State any assumptions you need to make in order to design the test cases. (7)

END OF PAPER