UNIVERSITY OF ABERDEEN **SESSION 2016/2017**

Examination in CS3028 (Principles of Software Engineering)

6 December 2016 **(15.00 – 17.00)**

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

- *Answer both questions No 1 and No 2.*
- *Use a separate answer book for each question.*
- *Each question is worth 50 marks; the marks for each part of a question are shown in brackets.*

1 (a) Explain the notion of modular program and describe the modularity principles used as a guideline in good design practice. (10)

(b) Consider a software-centric smart home system (the *Smart Home Manager*) equipped with

- a range of sensors placed both indoors and outdoors to detect a number of conditions relating to the environment and to the behaviour of its occupants (light; temperature; humidity; noxious gases; fires; flooding; appliance status; doors, windows, and gates open/closed; human presence and movement; physiological resident paramenters), and
- a corresponding range of indoor and outdoor actuators that can modify environmental conditions both inside and around the home (air conditioners; heaters; alarms; appliance controllers to swith these on or off and to manage their operations), and
- a communication infrastructure to report specific environment or occupant conditions remotely for action in specific circumstances (*e.g.*, increase temperature, close/open something, switch something else on/off, message the occupant).

The Smart Home Manager uses sensors to detect out-of-normal environmental conditions (temperature too low, front door left open, people entering a dark room, fridge short of food, homeowner likely to be sick in bed). The system uses actuators to try and rebalance such conditions automatically and the communication infrastructure to provide environment and occupant status reports, raise remote alarms, and enable remote control of the environment.

(i) Consider the above Smart Home Manager system description. Discuss the most suitable architectural pattern(s) upon which the design of such system should be based. (16)

(ii) Consider the above Smart Home Manager system description. Explain (i) whether a pluggable architecture (where a component based on a specific sensor and dealing with a specific consdition) can be added at any later time to the system) is possible and (ii) what would be the impact on the system (a) coupling and (b) cohesion. (10)

QUESTION 1 CONTINUES ON THE NEXT PAGE

(c) The *Liskov substitution principle* states that

> If, for each object `o1` of type `S` there is an object `o2` of type `T`, such that for all programs `P` defined in terms of `T`, the behaviour of `P` is unchanged when `o1` is substituted for `o2`, then `S` is a subtype of `T`.

Now consider the following class method, where the dots represent a chunk of source code that is not relevant in the context of this exam paper):

```
public void addTaxes( BasicTaxCalculator calculator, Sale sale ) {
  List taxLineItems = calculator.getTaxes( sale );
  ...
}
```

In this method:

- `BasicTaxCalculator` is a class of type `T`;
- `calculator` is an object `o2` of type `T`;
- `addTaxes` is the only program `P` defined in terms of `T`.

(i) Discuss (i) whether the above method complies with the Liskov substitution principle for any object `mycalculator` of type `SpecificTaxCalculator` which is a subtype of (*i.e.*, which extends) `BasicTaxCalculator` and (ii) why this is the case. (7)

(ii) Discuss whether the above method works unchanged for any object `somecalculator` of type `GenericTaxCalculator` where `BasicTaxCalculator` is a subtype of (*i.e.*, whether it extends) `GenericTaxCalculator`. (7)

PLEASE TURN OVER FOR QUESTION 2

2 (a) Discuss what a use case is in a software engineering context. (10)

(b) Consider the following draft use case, which focuses on patients checking in at the Accident & Emergency Unit at a major hospital for visit and treatment purposes. The use case needs to be reviewed by you in your analyst role as part of a software development team.

Use Case Name: Position patient in the visit & treatment queue

Primary Actor: A&E nurse acting as receptionist

Brief Description: This use case describes where in the ‘preliminary visit and initial treatment queue’ the system will position a patient checking in at the A&E department. In turn, this depends on the nurse’s initial assessment of the patient’s conditions in terms of *Gravity* (from ‘not serious’ to ‘extremely serious’), *Urgency* (from ‘can wait’ to ‘immediate action’), and *Tendency* (from ‘won’t change’ to ‘will get worse very quickly’).

Normal Flow of Events (actor actions and system responses):

1. The nurse initiates the action (which adds a new patient to those in the A&E queue waiting to be visited and possibly treated).
2. The system requests new patient data (name, surname, DOB, address, NOK)
3. The nurse inputs patient data
4. The system shows whether the patient is already in the hospital records. If this is the case, then the system flags any previously recorded conditions that may influence the nurse’s assessment of current patient conditions in terms of of Gravity, Urgency, and Tendency (see below).
5. The nurse selects from a closed list all those symptoms and conditions that currently affect the patient, deriving them from the patient’s narration and from previous digital records if these exist and if they are relevant in the current circumstances.
6. The nurse selects from a closed list of Gravity, Urgency, and Tendency level descriptions, giving rise to an overall *concern scenario*. This captures the current patient conditions, basing the nurse’s selections on the patient’s narration, on the nurse’s initial visual assessment of current patient conditions, and on the nurse’s taking into account previous relevant records (if any).
7. The system (which runs a decision algorithm based on data provided so far, including previous records if relevant to the current situation) provides its own assessment of Gravity, Urgency, and Tendency levels for the patient in this specific case. The algorithm can either (i) confirm the nurse’s choice of concern scenario, or (ii) suggest a higher level of concern, or (iii) suggest a lower level of concern.
8. The nurse can either (i) select the concern scenario proposed by the system (whether this matches the nurses’s own one or not), or (ii) select the concern scenario s/he had previously identified (which differs from that proposed by the system).
9. Based on the selected concern scenario, another decision algorithm computes and shows the position of the patient in the preliminary visit and initial treatment queue at the A&E department and updates such queue, recalculating actual position and estimated waiting times for all patients in the queue based on the concern scenario for each of them.
10. Use case ends.

Alternate Flow of Events:

- At (3) patient data are incomplete or in a wrong format (*e.g.*, non-alphabetical characters in name or non-existing addresses) – flag and prompt again.
- At (5) no symptoms or conditions are selected – flag and prompt again. particular item.

(i) Refer to the above use case. Discuss whether (i) the template used and (ii) the specific information provided in each use case section are actually complete. If you think this not to be the case, please state what template sections and/or what specific information items within any sections may actually be missing. (8)

QUESTION 2 CONTINUES ON THE NEXT PAGE

(ii) The general definition of functional cohesion applies to any element that contributes to a single, well-defined task.

1. Explain why cohesion — which was originally introduced and is typically used to address the quality of software components at design level — can be extended to functional requirements in general and to use cases in particular. (8)
2. Consider the above use case and discuss whether such use case is 'functionally cohesive' or whether it less cohesive than that. (8)

(iii) Consider the above use case. Even without transforming it back into a user story, it is actually possible to define a set of test cases for it in an Agile fashion.

1. Write the test cases that you would associate to the above use case if the latter were expressed as a user story. In doing so, do not be bothered at all with any issue regarding the cohesion of such use case. (8)
2. Discuss what are the implications of the specific flow of events for the two black-box testing techniques often used to derive and specify test cases, namely *equivalence partitioning* and *boundary value analysis*. (8)

END OF PAPER