LTAT.05.003 – Software Engineering

Written Exam – 08 January 2019

Start: 14:15 – End: 16:45

Important Notes:

- You are only admitted to this exam, if you have at least 30 marks from the lab assignments.
- The exam is open-book and open-laptop. Web browsing is allowed, but you are neither allowed to use e-mail clients or Instant Messaging clients nor to share any information "live" with anybody inside or outside the exam room.
- This document (question sheet) contains 6 pages (including the cover page). Please check that you have received 6 pages.
- At the end of the exam you must submit both the question sheets and your answer sheets. To avoid that any of your solutions get lost, make sure to write your name (and student ID) on each sheet of paper that you submit.
- Write clearly. Answers that are illegible cannot be counted as correct answers. Only answers written in English will be marked.
- To answer Part 1, use the separately distributed answer sheet. Answers given on the question sheets will not be marked!
- To answer Parts 2 and 3, use the separately distributed blank paper. Answers given on the question sheets will not be marked! Also, please number the pages on your answer sheets.
- At the end of the exam you must return the problem sheet. If you take the question sheets with you (out of the exam room), this will be considered academic fraud (cheating) and treated accordingly.
- Total exam marks: 30 (equivalent to 30% of total course grade). You must get at least 10 marks in this exam to not fail the course.

==========

PART 1: Multiple-Choice Questionnaire (10 marks)

PART 2: Open Questions (10 marks)

PART 3: Constructive Tasks (10 marks)

Total: 30 marks (=100%)

PART 1: Multiple-Choice Questionnaire (10 marks)

Important: For Part 1, please check boxes on the separate answer sheet. Read carefully before you answer and observe instructions carefully!

The following questions (up to question Q-08) have exactly one correct answer, thus, you must check exactly one answer box on the separate answer sheet. If you think that more than one answer is correct, choose the one answer that seems to be most correct/suitable/relevant.

Question Q-09 at the end of Part 1 can but must not have exactly one correct answer. Please read related instructions carefully.

Q-01 (1 mark): Which of the following indicate a waterfall approach to software development is not appropriate?
Answer choice:
A: Business is volatile
B: There are few requirements
C: Business requirements are stable
D: Legal regulations require completing requirements analysis before starting software design

Q-02 (1 mark): What are User Story Points typically used for?
Answer choice:
A: To estimate project effort
B: To estimate project quality
C: To estimate project duration
D: both A and C are correct answers

Q-03 (1 mark): Which of the following is a non-functional requirement of a web-based application?
Answer choice:
A: When the user clicks a "read me" link, the color of the link should change from blue to pink
B: When the user clicks a "read me" link, the next page should be opened within 5 seconds
C: When the user clicks a "read me" link, the mouse over should show the target page title
D: When the user clicks a "read me" link, the read me page should load correctly

Q-04 (1 mark): Which of the following are the key elements of use case diagrams?
Answer choice:
A: People, computer
B: Actors, use cases
C: People, classes, objects
D: Uses, use cases

Q-05 (1 mark): Which of the following statements about code refactoring is correct?
Answer choice:
A: Refactoring does always change the program design
B: Refactoring is a test activity
C: Before refactoring a component, a test suite for this component must be in place
D: Refactoring is done in order to add new functionality

Q-06 (1 mark): Which of the following statements is correct?
Answer choice:
A: Regression testing can efficiently be done manually
B: Equivalence class partitioning is a white-box testing technique
C: Black-box testing techniques exploit knowledge about the code that is tested
D: White-box testing techniques exploit knowledge about the code that is tested

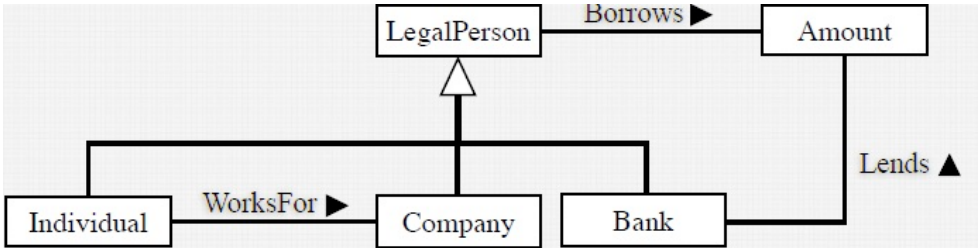
Q-07 (1 mark): Which of the following statements applies best to the code smell "Speculative Generality"?
Answer choice:
A: Developers over-generalize their code in an attempt to predict future needs
B: A variable, parameter, method, code fragment, class, etc. is not used anywhere
C: Redundancy in the naming of variables, methods and classes
D: Classes with fields and getters and setters and nothing else

Q-08 (1 mark): Which of the following is characteristic for a situation where SCRUM is typically applied?
Answer choice:
A: Task-boxing
B: Waterfall thinking
C: Stable requirements
D: Time-boxing

The following question (question Q-09) can have more (or less) than one correct answer. You must check all correct answer choices to get full marks. You get partial marks, if you check some of the correct answer choices. You will get a penalty, if you check an incorrect answer choice. You don't get a penalty, if you miss a correct answer choice. Overall, the lowest possible mark you can get is 0, i.e., even if everything you check is wrong, you won't get a negative mark.

Q-09 (2 marks): The figure below shows an initial (incomplete) sketch of a domain model for the following situation:

- An individual can work for one or several companies
- A car is owned by an individual, a bank, or a company
- Banks give loans (loan = amount of money) for buying cars
- A loan can be secured against a car



The domain class 'Car' is missing in the model. If you add the missing class 'Car' to the model, with which other class(es) should it be associated?
Answer choice:
A: LegalPerson
B: Bank
C: Individual and Company
D: Amount

PART 2: Open Questions (10 marks)

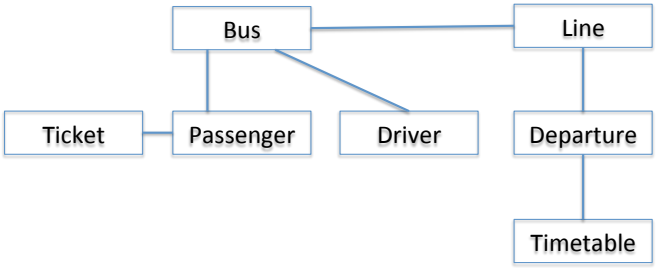
Note: Please give your answers on the provided answer sheet(s) and state clearly to which question number each answer refers. Answers that have no question number stated will not be marked. Don't forget to write your name on each separate answer sheet.

Q-10 (3 marks): Assume that a list of requirements contains the following two user stories:
(US1) As a hotel receptionist, I want to see the list of rooms not yet booked.
(US2) As an administrator, I can indicate folders not to backup so that my backup drive isn't filled up with things I don't need saved.

Answer the following questions a) to d):
a) Which of the two user stories is of better quality? Justify your answer.
b) What are the three elements of a complete user story? Explain each part briefly
c) What is the purpose of the 'why'-part in a user story?
d) For whom and how is the 'why'-part useful?

Q-11 (4 marks): The figure below shows an initial (incomplete) sketch of a domain model for the following situation:

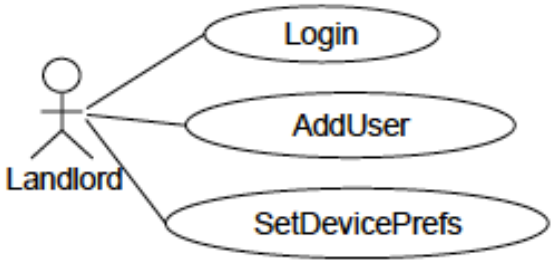
- A bus can have passengers and a driver
- A passenger must have one ticket
- Passengers can be adults, children, seniors, or students
- A bus serves a line
- A line has departures (with times and places)
- A timetable has departures (with times and places)



To Do (only use the information given above):
a) Say which domain class(es) is (are) missing and add them to the domain model
b) Place reasonable multiplicities at the ends of the association lines shown in the figure above
c) Where should the following attributes be placed: Price, Place, Time, Employee Number, License Plate, Capacity, Person ID, Number
Hint: Each attribute shall be placed in exactly one domain class.

Q-12 (3 marks): Do the following:

1. Briefly describe the key elements contained in a use case diagram
2. Describe the steps to follow for identifying use cases
3. Explain what is wrong in the use case diagram shown below. Then say how to fix it,



PART 3: Constructive Tasks (10 marks)

Note: Please provide solution on the provided answer sheet(s) and state clearly to which task number each solution relates. Solutions that have no task number stated will not be marked. Don't forget to write your name on each separate answer sheet.

T-01 (3 marks): Assume, you have 20 user stories (US1-US20) in your project backlog. You have estimated the user stories to have a difficulty/complexity expressed in story points as follows:
- Each of US1 to US5 equals 2 story points
- Each of US6 to US10 equals 3 story points
- Each of US11 to US15 equals 6 story points
- Each of US16 to US20 equals 14 story points

To do:
Q1: If you have a team of 4 developers and weekly sprints (1 week = 5 days = 40 hours), which user stories would you be able to implement in the next sprint and achieve the highest possible value without violating your capacity (effort) constraint?
Q2: How would your result change if the following dependencies between user stories apply?
- US6 must be implemented together with US11
- US7 must be implemented together with US16
- US8 must be implemented together with US17

How much is the overall value affected in the solution of Q2?

For your calculations, assume the following:
All user stories have equal value for the end user.
In past projects, on average, one developer could implement 1% of the sum of user story points of your current project in one day. In the current project, developers have the same productivity as in the past.
We don't allow overtime.

Show all your calculations and explain all your choices. (If you just present the lists if User Stories that can be accommodated in Q1 and Q2, you will get 0 marks, even if the lists are correct.)

T-02 (3 marks): Have a look at the code snippet from a Java class 'Item' shown below. Assume that 'quantity' and 'itemPrice' are instance variables of the class 'Item'. Do the following:
(a) Point out up to three code smells in the code shown below, i.e., name the smells and explain what the problem is.
(b) Then refactor the code: Say the name of the refactoring you apply, explain how you solve the problem, and show the refactored code.

```
double calculateItemAmount() {
   double baseItemAmount = quantity * itemPrice;
   if (baseItemAmount > 1000) {
     return baseItemAmount * 0.97;
   }
   else {
     return baseItemAmount;
   }
}
```

T-03 (4 marks): The code snippet below shows a function that calculates fines for speeding when driving a car. Fines are calculated based on age of the driver (age), the excess speed (overspeed), and the number of penalty points (penaltypts) already accumulated in the driver's police record.

```
0  public static int speedingfine (int age, int overspeed, int penaltypts) {
1    int fine = 0;
2    if ((age >= 25) && (overspeed < 30) && (penaltypts < 3)) {
3       fine = fine + 100 * overspeed; }
4    if ((age < 25) || (penaltypts >= 3)) {
5       fine = fine + (200 * overspeed); }
6    if (overspeed >= 30) {
7       fine = fine + 5000; }
8    return fine; }
```

To do:
Provide one single minimum set of test cases that achieves both (a) 100% statement and (b) 100% decision coverage. Remember that complete test cases include both input values and expected output values. You must also say for each test case which lines it covers, which decisions it covers, and in which direction it covers each covered decision (i.e., 'true' or 'false').

Note: You will get a penalty, if the number of test cases is not the minimum.

You can assume that all inputs have been checked for syntactic/semantic correctness, i.e., all parameter values are of the correct type and in a reasonable range (e.g., age is in [18, 99]).

Hint: Use the following table format to present your test cases of part a) and part b):

Test case number	Age (in)	Overspeed (in)	Penaltypts (in)	Fine (out)	a) Lines covered b) Decision(s) covered (T/F)
1					
...					
...					