DIT184 Software Analysis and Design

Software Engineering Division, D&IT

August 29th 2018

Time:	08:30 - 12:30
Place:	Lindholmen
Teacher:	D. Stikkolorum, M. Chaudron
Max. score:	100
Grading scale:	G=55, VG=75
Exam. aids:	dictionary

The examination consists of 2 parts:

1. 2 theory (T1 and T2) questions related to the course topics, and

2. 5 modelling problems (P1 – P5) related to the case of a single system
    - Some problems build on each other; it's recommended to consider them in order.

Practicalities

Start each question/problem on a new sheet. (Sub-problems, may be put on the same sheet).
Only write on the front of each sheet.
Label each sheet with

- your *anonymous code (provided by the attendant).*
- the *problem number, i.e., T1,T2,P1, P2 ...*

Before handing in:

- sort your sheets in problem order, and
- enumerate sheets as 1,2,3,....,
- check that your anonymous code is written on each sheet.
- explain your answers so that the graders can understand it.
- **write down/memorize your anonymous code** (to check anonymous exam results later)

Good Luck!

Camping Reservation System - Check Me In

Camping is a very popular way of celebrating the holidays and is done by people all around the globe. For a lot of people one of the advantages of camping is that they don't have to reserve a spot on a camping place. Normally they would just check by phone if there is a spot available once they are approaching a camping place.

A little problematic for camping places is that this can lead to large queues in front of the reception building. This is the place were the clients have to check in and pay. To overcome these large queues a camping reservation system is going to be developed: we call this the **CheckMeIn** system.

Figure 1 - an illustration of the Check Me In system

CheckMeIn is capable of a number of things such as checking free spots, reserve the spot and handling of payments. On the campsite the system controls a gate that is secured with a pin code. Figure 1 shows an schema of the system.

Checking in

When a client arrives at a campsite he/she goes to a terminal (the terminal is just seen as a touch screen - an input device for the system, it does not contain software). On the screen the client selects the preferred language and starts the check in procedure by touching the 'check me in' button. The system shows a map of free camp spots (depended on the amount of nights the person wants to stay). If there are no available spots the system shows a message and ends the procedure.

When a free spot is chosen, **CheckMeIn** asks for personal information such as the client's home address, email, tent-size and the number of persons that want to stay. The next step, is payment. The payment screen asks for credit card information and when the client approves the payment is done. After the payment is done, the client receives a code for the gate. This is printed on paper through the terminal and is sent by email to the client.

Entering and leaving the campsite

Entering and leaving is done by entering the pin code at the gate. If payment was in order the client is able to enter or leave the campsite. If not, the gate stays closed.

In case of problems the client can always go to the reception of the campsite. The receptionists can manage all clients reservations.

The gate itself is an external system to the **CheckMeIn** system.

Checking out

Check out is done automatically after exceeding the check out time of 12:00 (noon) on the last camping day. After that time the pin code is invalid and the spot is set to 'free'.

An early check out (before 12:00) can be done by walking to one of the terminals and touching the 'Check me out' button. After entering the pin code and confirming the check out, the system sets the camping spot to 'free'.

Checking out with the terminal can also be used when the client is too late (after 12:00). After paying the extra costs the client is checked out and can leave the camping place within half an hour. After that time the pin code is invalid again and the spot is set to 'free'.

Extra system functionalities

The systems api enables clients to use a mobile app to use the following functions online (via internet): check in and checkout (including payment), showing available spots at particular dates. The mobile app is seen as an external system.

System Requirements

For the development of the CheckMeIn system some requirements are written down:

Application (system) requirements

- In the future, the system should also be able to support other kinds of accommodation than described in the case text. In this way the app could be used for other types of camping places (for example with prepared tents or bungalows). Also, the support of more types of accommodation makes the system useful for other companies
- The application uses an attractive graphical user interface.
- In general the software should be maintainable, extensible and changeable.

Hardware requirements

- The input/output is done with a touch screen terminal.
- The system is connected to the internet

Technical software requirements

- The program language for implementing the system is Java (object-oriented language)
- The system uses a couple of predefined software classes:
    - **GateController**, is responsible for sending signals that open or close the gate.
    - **PaymentController,** is handling communication with the various banking systems.

    - **Canvas**, the GUI module that is responsible for showing the graphical components of the application (such as touchbuttons and text-messages).
      The Canvas consists of several implementations of:
        - **GraphicalComponent**, a reusable class that is responsible for handling events for graphical components. **TouchButton** and **MessageArea** are specialisations of this class.

    - There should be a class that has a role as central control unit

      The remainder of the design is left to the developer and must be obtained by text analysis (such as: reservation, client, etc.).

T1. Software Analysis and Design (15)

This course is called ‘Software Analysis and Design’. Based on the reading material of the course and the lecture slides:

a. What is the ‘Problem Domain’ ?
b. What is the ‘Solution Domain’ ?
c. How do Problem Domain and Solution Domain ***differ*** from each other? *Hint: which things are in the Problem Domain that are not in the Solution Domain and vice versa.*

T2. Taxonomy (15)

This list describes a taxonomy of animals (it appears in a book by J.L. Borges). Point out 3 things that you think are wrong with this taxonomy. Explain your answer (hint: explain from the perspective of which properties you think should hold for a good taxonomy)

1. those that belong to the Emperor,
2. embalmed ones,
3. those that are trained,
4. suckling pigs,
5. mermaids,
6. fabulous ones,
7. stray dogs,
8. those included in the present classification,
9. those that tremble as if they were mad,
10. innumerable ones,
11. those drawn with a very fine camelhair brush,
12. others,
13. those that have just broken a flower vase,
14. those that from a long way off look like flies.

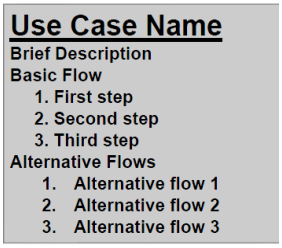
The following questions relate to the CheckMeIn system case.

P1. Use Case based requirements (15)

a) Make a use case diagram and include at least two sensible use cases in addition to the use case '*Check In*'. The use case '*Pay Reservation*' is initiated from '*Check In*''.

Next to the main use cases, the answer should at least use one include- or extend use case.

b) Give a use case description, including 1 alternative flow, of '*Check In*', following the standard notation (see figure).



P2. Problem Domain Modelling (15)

Build a domain model of the 'problem' domain presented in the description on pages 2-3 through the use of a UML class diagram. You should consider using the richness of UML (such as association names, composition- , aggregation-, inheritance relationships and multiplicities) where appropriate.

P3. Design Modelling (20)

A developer of a software development team identified classes with the use of the following CRC cards:

Campsite	
Responsibilities	*Collaborations*
show camping area show spots show route from entrance	CampSpot Canvas ....

Reservation	
Responsibilities	*Collaborations*
Create new reservation Show reservation details	CampSpot Client ….

Design the initial software structure for the **CheckMeIn** system, with the use of **1) a UML class diagram and 2) sequence diagram.** The design-level class diagram focuses on implementation aspects. The design should consider the requirements on page 3 and the classes presented by the CRC cards above.

For full score all classes, attributes, methods, and associations, association names, cardinalities, and inheritance relationships should be defined. Also coupling and cohesion are considered.

Procedure: Create 1 design class model by first constructing 2 sequence diagrams (or the other way around) for 1) *'Check In*' a client and 2) '*Open Gate' with pin code*.

The answer should consist of the final sequence- and class diagram. The diagrams should be consistent.

P4. Design Principles and Patterns (10)

a) Organize your design (class diagram P3) in a layered architecture. Draw classes without the details (attributes/operations). Use proper names for the layers. After introducing layering, it is possible that design principles are violated. Mention which principles are violated and how this should be improved in a next generation of the design. Explain your answer (don't alter the design).

b) Explain how the decorator pattern can support the application performance in the case of showing the campsite with its free and occupied spots.

P5. Behavior Modelling (10)

The gate can be in different states: CLOSED - OPENING - OPEN - CLOSING and STOPPED. When the correct pin code is entered, the gate opens (OPENING). The gate is OPEN when it reaches a 90° angle. After 20 seconds or when a car has passed ('seen' by a sensor) the gate goes to CLOSING and eventually CLOSED when it reaches a 0° angle. While opening and closing red lights on the gate arm blink.

The gate can enter the STOPPED state from closing when the sensor in the arm 'notices' an object below the arm. After 3 seconds the gates enters the CLOSING state to try again.

The class **GateController** always receives a state update.

To model the behavior: 1) consider the gate as an object. 2) describe the behavior for the gate by using a UML State Machine Diagram. The state diagram should consist of proper internal (do/) activities, transition triggers and transition effects. It should involve method calls to the relevant objects of the class design of P3. It should be consistent with the answer on P3.