DIT184 Software Analysis and Design: Exam

Date	June 4th 2021
Time	8:30-12:30 (plus an extra 30 minutes to upload your solutions until 13:00)
Place	*online*
Teacher	Birgit Penzenstadler
Max. score	100 – distributed to the 6 questions as detailed after each question.
Grading scale	G=50, VG=75
Exam. aids	slides, assigned readings

We grade these exams anonymously – please do not write your names on them!

Exam questions: This exam is composed of six exam questions, one for each learning objective identified for this course. The points are denoted after each question. The point distribution can give you an indicator of how much time to spend on each question.
The next page shows the case system description you work with. There may be several case system descriptions for this exam randomly assigned to disincentivize any illegal collaboration. We will run similarity checks on handed in solutions, so please invest your time in developing your own solution. You got this on your own.
We know the time is tight in comparison to the amount of time you could spend on these design questions, so set yourself a timer and remember that we want to see that you understand how to design that certain type of model and its typical elements.

Clarification questions: I'll be available for questions via *Canvas discussion forum* (some questions are frequent, so I answer them for everyone) and *email* at birgitp@chalmers.se but be mindful of how you spend your time, because it'll take me time to write answers one at a time and I can only clarify what I am asking for. I will be available on Zoom as well if really needed.

Uploading the exam: After the official four hours of working time, you have an additional extra 30 minutes to upload your exam. In case you have handwritten diagrams, scan them with CamScanner or an equivalent app and make sure you integrate everything into a single PDF file. This always takes a few minutes more than most expect, so take the time for it instead of trying to improve your solutions until the very last minute. The working time is 4 hours, the additional 30 minutes are give so you can focus in the first four, and then scan and upload calmly.
You can only upload one PDF file on Canvas for this exam. After you upload, please double-check that the file isn't corrupted by displaying it in your own browser again.
In case you experience technical difficulties with Canvas, you can email me the exam before the cut-off time at 13:00.
All the best, you got this, and I look forward to your solutions!

***Case system*: Plack**

Context: According to scientific studies (Smart people et al.), inefficient communication is one of the biggest time wasters a company can foster. In the fast-growing software industry, this waste is easily the difference between a success story and a failed start-up. Plack is a communication service for organizations that want to share information efficiently, securely, and (most importantly) with ease. Say goodbye to time-wasting emails, say hello to the new era of communication!

Features:

- Plack enables users to create purpose-driven **workspaces**. Once created, users can invite other team members to join the workspace by sending an invitation email. The invited team member can join the workspace by accepting this invitation. Finally, the owner of the workspace can delete it.
- Plack empowers efficient communication via topic-relevant **channels**. Beyond the few automatically generated channels, new private or public channels can be created by the workspace owner. Private channels require explicit invitation to the workspace members, while public channels are visible to all workspace members. Finally, the channel purpose and description can be set by the creator to direct the flow of correspondence, and the channels can be either archived or deleted.
- Plack users can post **messages** into their channels or directly to other workspace members. The messages can contain emojis, mentions to other Plack channels and specific members (which in turn triggers notification emails to the referred users). Another brilliant feature is that Plack messages can be edited by the sender after-the-fact, which is an essential tool for precise asynchronous communication. Important messages can be pinned (and unpinned) to create a short-cut to critical information. To maintain a coherent message board, threads can be used to directly reply to any message.
- Any Plack user can **share** files, code snippets, and links in their channels or with individual members. The shared files can be downloaded by the recipient members.
- Plack users can set reminders for themselves, as well as for other workspace members or even channels. A tailored message will be sent to the appropriate channel/user by the Plackbot.
- An advanced **search functionality** allows the users to search the Plack workspace. The search bar provides the user with some tips on how to search through the workspace content, e.g., by looking for all the messages from some user in some channel. Or by querying for all the files that were sent into some channel before the end of May 2021.
- Who still uses the fan-bursting Slype? Plack provides its users basic **calling** and **screen sharing** abilities, free of charge, and with high-performance results.

Context of use: Mario and Luigi have to complete a feature implementation for a Korean Barbecue Delivery Service - and now they have discovered a bug in their soon-to-be-delivered prototype!… ;) Normally, Mario and Luigi were desk mates, but now they have to do this from their own home. So they set up a Plack workspace dedicated to developing the Service, they assign some tasks for each other in the #task channel, and Mario shares the bug-report document in the #material channel. They get to work, and chat for every new commit they make in the code base.
Scenario: One hour before the deadline, Luigi discovers that the bug is still present through some untypical control flow by the program, and communicates the edge test case to Mario. He describes the context and the parameters for the control flow and sends a description with a few screenshots of what happens and how the system crashes. Mario responds with two observations and potential error sources. Luigi responds to each of the observations by checking the potential error source and commenting on the

outcome (one negative, one positive). They found the bug just in time and send a couple of animated gifs to the #celebrate channel.

Challenges:
This system faces two main challenges. One major challenge is ensuring user **privacy**. The information shared in private correspondences should never be revealed to other members, or outside the organization. The second challenge is enforcing a gentle but firm handling of **inappropriate behavior** in conversations. Think of, for example, rude comments, microaggressions, or slippery jokes taken too far.

1. Agile Development: You have a team of 5 developers that created a super successful app and now your company got a spectacular angel investor so you can hire 50 more people. (15 points)

    a. How will you transition from Agile to "Agile at Scale"?

    b. What are three challenges to keep in mind and how will you address them?

    Please write about 250 words for each a and b.

2. Explain and apply guidelines and heuristics for performing a domain analysis:
   C*onstruct a **domain model** for the case system based on the information given and mention which guidelines and heuristics you are using for that.*
   Do not forget to justify your model with a clear description of all model elements.
   (20 points)

Tip: How do you make use of each different section of the case system description for the analysis? Ensure that the descriptions are tailored to the case system, so that you show you have understood the material.

3. Analyze and design software systems using object-oriented techniques:
   a) *Create a **use case overview diagram** (use includes or extends at least once) for the case system* (10 points)

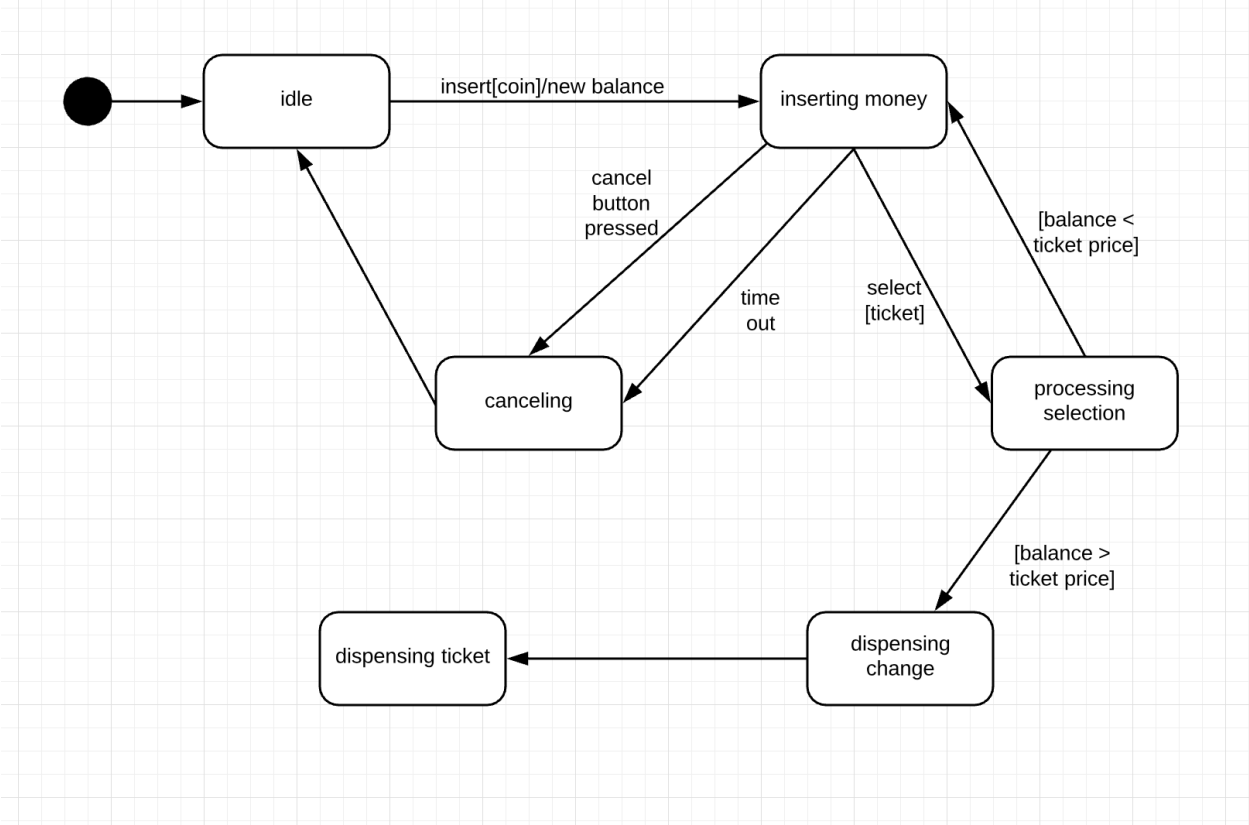
   b) *Create a **design class diagram** for the case system inspired by the domain model from question 2.* (10 points)

Solution pointers:

- Use case overview diagram has properly named use cases, uses correct notation
- Class diagram has names, attributes, couple of methods, relations, correct notation
- Refine by adding what is needed for implementation as software system

b) At least the elements of … should be there. Design level should also have elements for database, interface, controller in the simplest case.

4. Use tools for domain and requirements analysis, modeling, program visualization, and object-oriented program design:
   *Analyze the **UML state diagram** below and name and explain 3 errors or omissions in the diagram.* (15 points)



5. Create an UML model that is an abstract representation of the source code:
   *Reverse engineer an **activity diagram** for the pseudo code below.* (15 points)

Pseudocode (Note: incomplete on purpose to limit the scope and therefore time you need):

```
public class smartCar {
  String name = "KITT"; // name of the car, KITT by default, can be changed - if you catch the
pop culture reference, you know a lot about the 80s (or have seen the remake in 2008)

  private void setUnderbodyLighting(Color){…} // let's get in some more cliché car stuff
  private void hydraulicCarJumping(){…} // let's get in some more cliché car stuff
  private void driveTo(Destination d){…} // code omitted, treat as black box
  private void turboBooster(){…} // code omitted, treat as black box
  private String listen(){…} // code omitted, treat as black box
  private Destination analyzeDestination(String s){…} // code omitted, treat as black box
  private void say() {…} // code omitted, treat as black box
  private void goSleep() {…} // code omitted, treat as black box

  public static void main(String[] args) {
    smartCar car = new smartCar ();  // car initialized
    car.say("Hi! My name is" + car.name); // car introduces themselves

while (car.listen() != "go sleep"){ // while the command is not to "go sleep"
    car.say("Let's ride! Where are we doing?"); // ask for direction
    if (car.listen() == "drive me to"){
          if(car.analyzeDestination(car.listen()) != null){
                car.driveTo(car.analyzeDestination(car.listen()));
```

```
                car.say("Ok, I'll get you there as quickly as possible.");
        } else {
                car.say("I am sorry. I did not catch that. Could you repeat?")
        }
   } else if (car.listen() == "bounce"){
        car.say("Ok, let's go wild.");
        car.hydraulicCarJumping();
   } else if (car.listen() == "light up"){
        car.say("Yassss. Which color?");
        car.setUnderbodyLighting((Color)car.listen());\\ casting return parameter, yup, it's a hack
   } else if (car.listen() == "KITT get me out of here"){
        car.say("Coming to the RESCUE!!");
        car.turboBooster();
   } else {
        car.say("I am sorry. I did not catch that. Could you repeat?")
   }
car.goSleep();
  }
}}
```

6. Below is a component diagram of a community Barter system (think Craigslist, without the payments, i.e., no money is exchanged, only items and services). Study carefully the design and answer the following questions:

   a) Which design pattern can you spot in this diagram? (5 points)

   b) Discuss the trade-off between security (e.g., using encryption to ensure integrity of the offers) and performance. (5 points)

   c) Name one more strength and one potential weakness of this design. (5 points)