1 Question 1

Scrum is an incremental agile method for software engineering. The scrum method consists of multiple roles, the client, the product owner and the development team. The client is the one that wants a product, the product owner is the one that handles the communication with the client and makes sure that the team is making progress during development and the team is a group of “engineering experts” that essentially will build the product.

An example: A client wants to build a game application, he will only have contact with the product owner, which will then forward the communication to the team. The product owner creates the backlog of tasks and the idea is that the development team works independently to be more effective. The team completes a decided amount of tasks each sprint (normally between 2-3 weeks long) and the product owner validates the work with the client after the sprints to ensure the team is moving in the right direction. After a sprint the team has a retrospective where they reflect on how well the process worked during the sprint, then they keep what worked well and changed what worked bad.

This process then iterates until the product is completed.

2 Question 2

2.a

The diagram have:

1. A singleton
2. An adapter
3. A factory method

2.b

1. The singleton consist of the Bank. In the system there will only be one Bank-object which will be responsible for and contain all the customers and accounts. To ensure that there is only one Bank object the Bank contains a getInstance method that will either create a new (if no object exists) or return the already existing object.
2. The adapter consists of the StatementInterface-interface and the Statement-class. These ensure that the Account-class works together with the LgacyStatement-class (which is assumed to be a older part that is reused in this system). The adapter lets the Account class set the date in a way that it expects, instead of setting it as the older LgacyStatment-class expects.

3 Question 3

3.a

Code review is when somebody other than the author of the code reads and comments on the code. The main objective is to reduce defect rate, make cheaper bug fixes and keep the code clean preemptively to increase maintainability. The code reviews can be graded from less formal to more formal and the following list contains 4 processes listed from less formal to more formal:

1. Pair programming
2. Over-the-shoulder
3. Peer review
4. Pre-checking

3.b

A: Pair programming and Over-the-shoulder.

B: Peer review and Pre-checking.

The processes in group A will need the both the programmer and the reviewer to be at the same location while developing the code. But in the processes in group B the programming and reviewing the code is separated, and therefor the programmer and reviewer can be at different locations.

The disadvantage of the processes in group A compared with the processes in group B is that you have no benefit of distance, i.e. the programmer and reviewer needs to be at the same location.

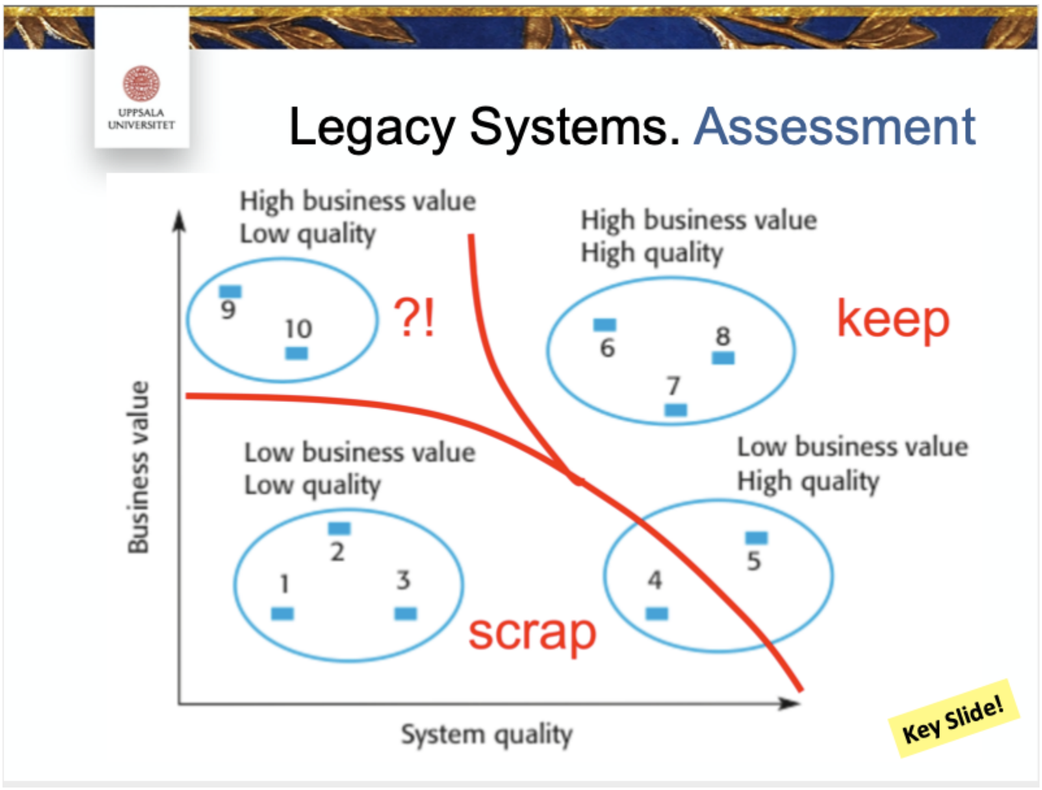
3.c

1. Pair programming is when the code is written in pair. One is typing the code and one is observing and commenting the code that's written. When the programmer writes the code s(he) "think loud" and share his thoughts on how the code will work, while the observer listen and watches for errors that the programmer might miss (since his focus is on writing the code). During this process the code review happens naturally. Advantages is that everything that is written is read by 2 people and the feedback is immediate. Disadvantages is that it's expensive and have no benefit of distance (as mentioned in 3.b).

2. Over-the-shoulder is when one (or more) person is writing the code, but there is always one reviewer (usually someone experienced) available if the coder(s) need him/her. If help is needed the programmer presents the code for the reviewer and walks him/her through the code and the issue. Advantages of this method is that it's lightweight and not as expensive as Pair-programming. Disadvantages is that the reviewer only will have a short time for reflection while helping the programmers and the coder can easily over-explain which can prevent the reviewer from getting the bigger picture of the issue.

4 Question 4

4.a


UPPSALA UNIVERSITET
Legacy Systems. Assessment
High business value Low quality
High business value High quality
Low business value Low quality
Low business value High quality
?!
keep
scrap
Business value
System quality
9
10
6
7
8
1
2
3
4
5
Key Slide!

Since the streaming software system is from the 80's it can be seen as a legacy system. During the assessment of the system you have to look at it and see what quality is and what the business value is of the system. As seen by the diagram, if the assessment indicates that the business value is high and the quality is high, you should keep the system and improve it. If the quality and the business value is low, you should scrap it. But you you have a mix of either high business value & low quality or low business value & high quality you get a tougher time to decide.

The decision can than be based on things as assessment of the environment of the system, such as supplier stability, failure rates, age of hardware and OS support for libraries that are used and what the maintenance cost is. Or assessment of the system it self, such as the understandability of code, how well documented the system is, if any tests exists (data/regression/unittests), what programming language is used, and if there are any personnel skills to improve the system.

4.b

1. The first method to use is to run formal meetings once a month. During these meetings the programmers and reviewers will sit together and discuss the written code and documents. Since we're improving a legacy system there might come up large unforeseen problems, and these can easily be handled during these meetings since all the concerning parties will be participating.

2. The second method that will be used is Over-the-shoulder, since there is one expert in the team that knows the legacy systems written language very well. This person will be the reviewer and will be there for anyone that needs him during the improvement of the system.

4.c

1. **Stress test**
   The objective of this test is to find out how many concurrent connections the system can have before it begins to have a to high delay for each client.

   The test will begin with 100 client connected which will use the system in a normal manner to simulate a real user. The test will then increase the number of connected clients with 100 each 5 seconds while at the same time measuring the response time of the server for each client. The test will end when the server have a response time over 5000 ms.

2. **Integration test**
   The objective of this test is to ensure that the old component A deliverers what its API says.

   The test will run each method in the API, testing all edge cases, and normal usage, for each method and expects that the module returns what it says.

3. **Load test**
   The objective of this test is to ensure that the system can have 1000 concurrent connections and still have a response time of the server less than 100 ms.

   The test will connect 1000 clients which will use the system in a normal manner to simulate a real user. The test will be run in 30 seconds and measuring the response time of the server for each client. The expected outcome is that no client will have a higher response time than 100 ms.

5 Question 5

5.a

The requirements engineering process can be seen as a circular process that starts in the middle and creates a spiral when moving outwards. You begin at the center and in the beginning, you know nothing about the product that is about the be built, and when you move outwards you begin to know more and more, until you are at the end. Then you know everything about the system and have created the system requirement document which is the goal of the process.

When you are moving outwards in the spiral you move through the three steps of the process, requirement elicitation, requirement specification and requirement validation. You first start with the elicitation which is when *the team and the client* sit together and the team gathers information about what the client wants and what the product should be. The process then moves on to the requirements specification step where *the team* works at their part by first setting up the business requirement specification to ensure that they understand the business of the product that they are making and that the product contains a business model for the client. The process then moves on to the requirement validation step where *the team and the client* sit together and validates this document to ensure that the team is moving in the direction that the client wants.

If the team gets an OK from the client, they again move to the requirement elicitation step where the team now gathers more information about the product from the client and creates the user requirements. The process then moves the requirement specification step again where the team creates the user requirements specifications, which is then validated with the customer in the next requirement validation steps.

The process then iterates like this until you finally end up in the system requirement document, which is a full definition of the system.

5.b

Functional requirements are describing the products behavior and describes what the system should do. A non-functional requirement also describes the products behavior but will describe things that are related to the system that are not about how the system actually works. They can be seen as constrains on the system (or on the development team) that affects the developers that are creating the system. They can be requirements that wouldn't need to be there for the system to work, but the clients wants the product to be in a specific way, or that the company requires the development to move in a specific way.

So:

1. Functional requirements describe how the system will work while non functional describes constraints on the system.
2. Functional requirements are set by the client, but non-functional can also be set by other parties. Such as the developers company.

5.c

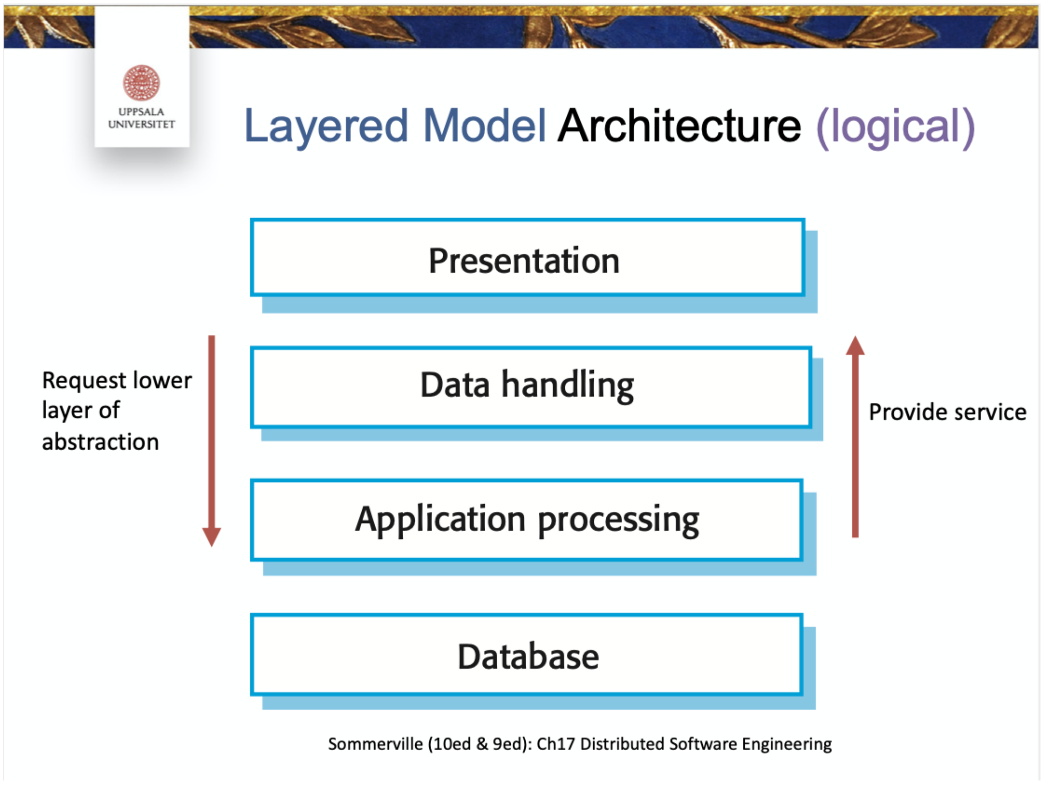
1. (Non-functional requirement coming from the company)
   During the development of the system, there can only be 4 developers working on the system to ensure that the company is making a profit during the project.
2. (Non-functional requirement coming from the client)
   When a user is renting a instrument from a another user, the payment must go through Klarna to ensure that there are safe payments in the applications.

5.d

User story:
A user must be able to search for instruments to rent in a within a specific range to ensure that the user only sees relative advertisements.
Diagram:
No time.

6 Question 6

6.a


UPPSALA UNIVERSITET
Layered Model Architecture (logical)
Presentation
Data handling
Application processing
Database
Request lower layer of abstraction
Provide service
Sommerville (10ed & 9ed): Ch17 Distributed Software Engineering

Layered model architecture is a logical architecture containing of multiple layers, where each layer is one component which is providing a specific service to the system. The components/layers can then only talk to adjacent layers, meaning that each component can only make requests/return data to the layer either above or below. Advantages of this model is that it's simple, it organizes the system and supports reuse (you can simply just remove and replace one layer as long as they have the same interface). On disadvantage is that it's have an impact on the performance, moving through all the layers is quite slow.

6.b

Yes it could be reasonable to have a 2-tier architecture in a combination with the layered architecture. In this system the client can be seen as the VR glasses and the mobile device can be seen as the server in the 2-tier architecture. The top layers (such as presentation) of the layered model will then be places in the client, the VR glasses, while the bottom layers will be places at the server, the mobile device, where the mainly computations will be done.

6.c

For this application there would be better to use a thin-client architecture. The VR glasses can be seen as the client while the mobile device is the server. The idea with this architecture is to let the server to the “heavy lifting”, i.e. do all the application processing and data management, while the client only focuses on presentation. The advantage with this model is that the client can be very simple and lightweight which in this system means that the VR-glasses can be cheaper since they don’t need as much computational power. Since most people today already have a mobile device, it’s more likely that they will buy the product if the VR glasses are cheaper.

But one issue worth mentioning when going for a thin-client is that the VR glasses and the mobile will do a lot of communication, since the glasses will ask the mobile for (almost) everything. So the network communication speed between these two devices, maybe the speed of Bluetooth, will be a bottleneck for how much data that can be sent and therefor have a impact on the quality of the product.