BDSA Assignment 1 Group 48

Anton Marius Nielsen,
Frederik Lukas Aguilar Svendsen
Maj Frost Jensen

September 2021

1 C#

1.1 Generics

In the first method, the 'where' constrain defines that T have to implement the IComparable<T> interface.
The second method states that the input T is of type U, and U has to implement the IComparable interface. This means that there's no real difference, since a string is of type string and implements IComparable, which means it would work in both functions.

1.2 Iterators & Regular Expressions

Link to the GitHub repository:
`https://github.com/Maroka-chan/BDSA_Assignments/tree/Ass_01`

2 Software Engineering

2.1 Exercise 1

It means that knowledge acquisition isn't a linear process. Knowledge can be acquired at any time in the process and shouldn't happen one step after another. An example could be that the client comes to find out that something previously agreed on, wouldn't work for their needs. This new piece of information could potentially mean that the entire system needs to be changed. Therefore, software engineers should always be ready for such big changes and potentially having to start all over.

2.2 Exercise 2

The first two are system design decisions, as they are about decomposing the system into smaller subsystems, and deciding on hardware.

The third is made during requirement elicitation, where the client and developer define the purpose of the system.

2.3 Exercise 3

The first two times the term account is used, it is used as an application domain. Here we hear what we will have to work with, and what some of the bigger issues are. This is part of the clients problem.
The last two times the account is used, it is used as a solution domain. Here we are on the developers point of view and hear about how we might solve the clients problem.

2.4 Exercise 4

Unlike aircraft and bridges, software such as word processors can end up going through constant change under their development time. Unless something is found to be seriously wrong with the design of aircraft or bridges, you wouldn't stop those developments. Software applications can constantly run into problems, and could end up having to start over.

2.5 Exercise 5

Functional:

- Enable a traveler to buy weekly passes
- Be written in Java

Nonfunctional:

- Easy to use
- Always be available
- Provide phone number to call when it fails

2.6 Exercise 6

The purpose of modeling is to create models which represent a system, to be able to answer questions about said system.
In software engineering modeling helps us understand the environment where the system will operate. Modeling also helps us understand how we could solve problems with our systems.