Assignment 01

Benjamin Edon Hagelin, bhag
Emil Boesgaard Nørbjerg, emno
Frederik Høegh Christensen, frhc

September 14, 2022

1 Github Link

`https://github.com/Emilfranord/BDSA-assignment-01`

2 Generics

First method:

```
int GreaterCount<T, U>(IEnumerable<T> items, T x)
    where T : IComparable<T>;
```

What does the type constraint mean for the first method: The first method only have one constraint which is where T : IComparable<T>;. This given constraint defines and sets up a requirement That for GreaterCount it will be using the IComparable Interface. For both methods, we are able to receive two different types, T and U. These two types could be any type, fx integer and float, where it is required to give two types which can be the same. The constraint then adds that T must be comparable with itself. This means that two seperate instances of T needs to be comparable with eachtoher because of the IComparable interface constraint.

Second method:

```
int GreaterCount<T, U>(IEnumerable<T> items, T x)
    where T : U
    where U : IComparable<U>;
```

What does the type constraint mean for the second method: The second method have two constraints. The first is where T : U and the second is where U : IComparable<U>;. The two constrainst are in a way connected with eachother. Meaning that T needs to be comparable to U and U needs to be comparable with itself, since U is implementing the IComparable<U> interface. U is required by the constraint to be comparable to itself and since we also have a constraint stating that T implements U, then T is required to be comparable with U.

first: T comparable with T

second: T comparable with U and U comparable with U

3 Exercise 1

Nouns and verbs The following lists are of nouns and verbs, that appear in the specification.

Nouns:

1. version control system
2. changes
3. files
4. set of files
5. time
6. specific versions
7. system
8. source code
9. configuration data
10. diagrams
11. binaries
12. state
13. project
14. changes
15. problem
16. issue

Verbs:

1. want
2. records
3. can recall
4. work
5. revert
6. compare
7. see
8. modify
9. introduce

Domain The majority of the words belong to the technical domain. Several of the nouns only exist in the IT technical domain, such as 'source code', and 'binaries'. Other words come form the technical world in a less direct way.

Some of the verbs that are used, relate to the concept of time. Especially 'revert' and 'recall' belong to the time domain.

All together the words belong to the domain of version control systems.

libgit2sharp While libgit2sharp does not have classes representing 'File' or 'State', it still functions as a version control system. This is possible since the architecture of the program does not demand it. Instead of files, this implementation tracks 'GitObject' instances, be they files or something else. That way libgit2sharp, can still track the changes to files, and other data that needs to be tracked. This design was probably used to abstract the implementation away from specific files and data structures.

In this design there is not a 'state', because it is an emergent property of the system. That is, the state of the system at any given time is defined by the state of the objects, not a single object. This was there is no need for a class to handle the state of files, commits or similar, since that is handled inside the object instances themselves.

4 Exercise 3

The Coronapas App, was a software product, that the danish goverment wanted to get made. They wrote a public offering, contining the requrements for the app, and then companies could submit their bids. Therefore the app is a Customized software product, because the purchaser is in charge of the requrements.

On the other hand, Git is a Generic product. This is the case because the actors who use git, have no say in what it does, and what requrements it needs to uphold. It is also avalible for all to get, hence it is generic.