"System Design"
– *Initial Steps & Modelling [01]*

1 Homework ("Pre-Exam-Task")

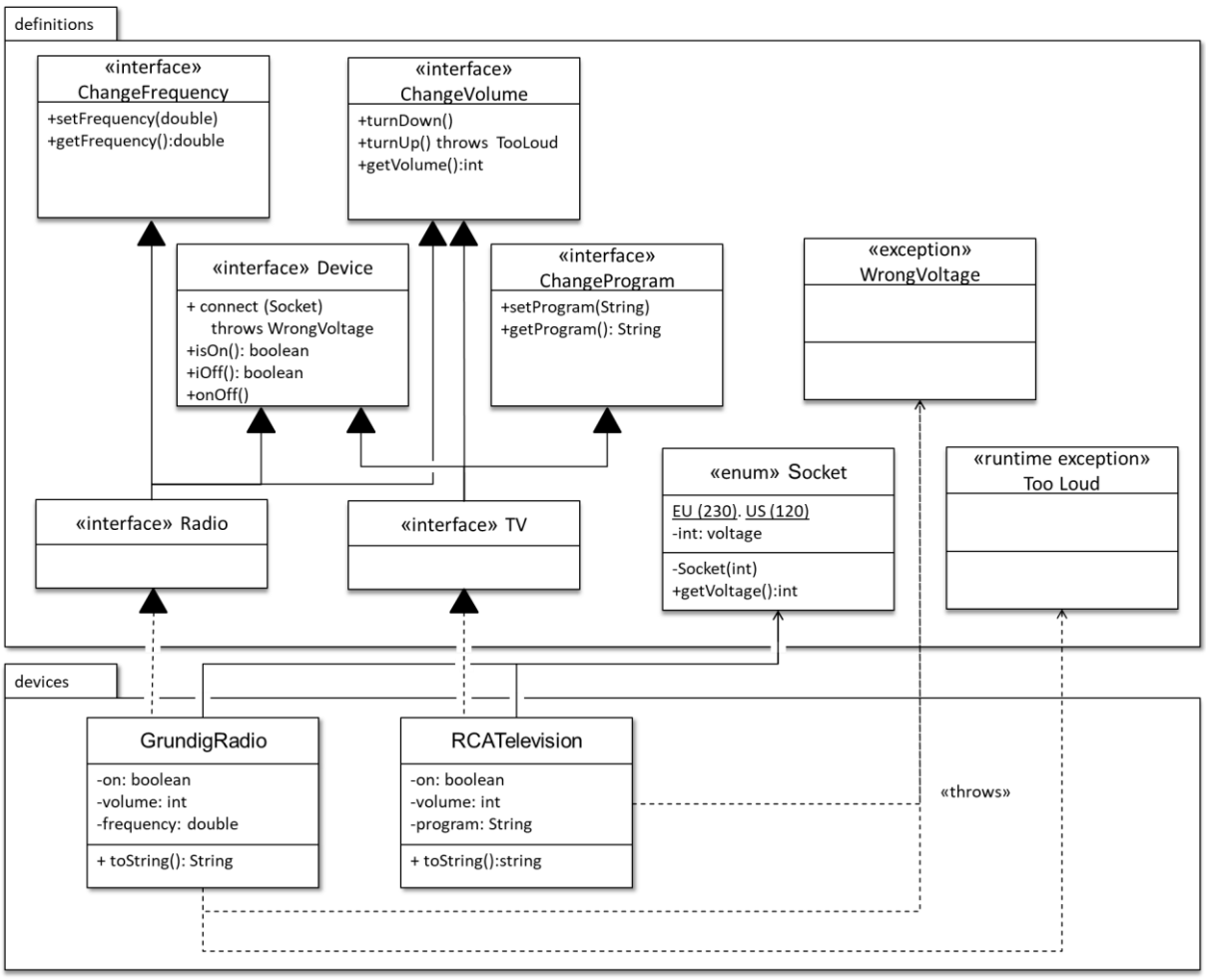
This task is part of the pre-exam-requirements. The filled-out project form should be uploaded to Moodle until 2nd November 2020, 8.00.

For a project you have done during your bachelor course or during your job come up with the project description as proposed in the lecture.

Fill out the project form you find in Moodle (either use the word template or the pdf) and upload it again in Moodle.

2 UML Modelling

A system for the simulation of radios and televisions is to be implemented. The following UML diagram is given:



Realize the UML diagram according to the conventions presented in the lecture. Unless otherwise stated, the usual implementations are to be made for constructors, getters, and setters.

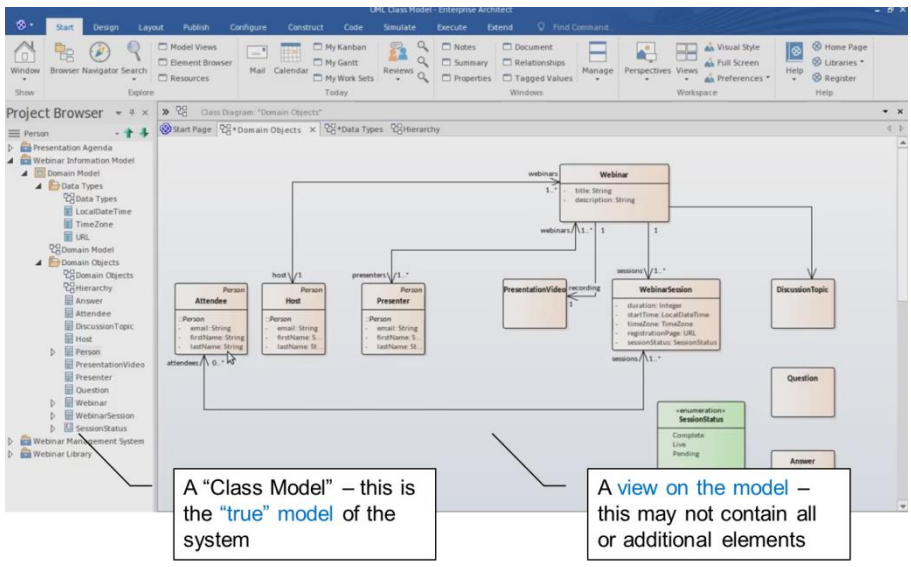
The following requirements apply to the remaining elements:

- `WrongVoltage` should be a subclass of `Exception`.
- `TooLoud` should to be a subclass of `RuntimeException`.
- The method `connect(Socket)` sets the respective field of a device with the given socket. For a `GrundigRadio` `WrongVoltage` is to be thrown when the voltage of the passed socket is not in the range 220-230. For a `RCATelevision` `WrongVoltage` is to be thrown when the voltage of the socket is not 120.
- The method `isOff()` should return `true` if the device is not on, otherwise `false`.
- The method `onOff()` turns on a connected device (i.e. a device with a connected socket) when it was off, and vice versa.
- The method `turnDown()` turns down a switched on device by one. The value should not go below 0.
- The method `turnUp()` turns up a switched on device by one. The value should not not exceed 100. For a `GrundigRadio` a `TooLoud`-Exception should be thrown, if attempted to go beyond 100.

2.1 Designing the System

Realize the design with the help of Enterprise Architect (EA). If you are not used to UML have a look at the respective slides. If you are not used to EA use the following hints.

EA looks typically like this:



The most important thing to understand is that the class model of the project browser (left) is always the true representation of the model. All diagrams are only views on the model and may leave out elements or have additional elements. If you delete an element in the diagram, it is only removed from the diagram, not from the model.

This is a typical workflow:

- Open a new project
- Add a new class model
- Import a Java-Subset for referencing these classes (the subset `java.lang` can be found in Moodle)
- Add new packages (by dragging a package from the toolbox to the diagram)
- Add new class diagrams to these packages open them
- Add classes, interfaces and enumerations (by dragging an element from the toolbox to the diagram; supress the “New-Dialog” at Start > Preferences > Objects > Edit on New)
- Add relationships between interfaces and classes
- Add attributes and methods for classes and interfaces (Properties > Details or Attributes & Operations)

Some usage hints:

- Add subclasses: drag parent class from project to the diagram, select “child class”
- For operations throwing exceptions tag the operation
  - Tagged… | new tag:
  - Tag: “throws” Value: the respective exception
- For Enum Constant Values tag the attribute
  - Tagged… | new tag:
  - Tag: “arguments” | Value: the respective argument
- Making private Enum fields
  - Remove enum stereotype
  - Make it private
  - isLiteral = false
- For selecting operations to override: Ctrl + Shift + O

2.2 Create the Code

Follow these steps:

- Open Eclipse and create a project which should contain the generated code.
- Now rename the “Class Model” to “src” – this is because this is the root of your generated package structure.
- Select the src-Package and call Code Engineering > Generate Source Code …
- Select “Child packages”
- Check “Auto Generate Files” and select the path to your project
- Generate the code

Unfortunately, not everything will be transferred correctly, so fix all errors now and add the missing functionality.

2.3 Testing your Code

A possible test method for the `GrundigRadio` might look like this (create a similar sample for the `RCATelevision`). Alternatively, you may create Junit-Test Cases.

```
public class Main {
     public static void main (String [] args) {
          Radio r = new GrundigRadio ();
          boolean ok = false;
          try {
               r.connect (Socket.US);
               System.out.println ("Successfully connected to US-Socket");
               ok = true;
          } catch (WrongVoltage e) {
               e.printStackTrace ();
          } finally{
               r.onOff();
          }
          if(!ok)
               try {
                    r.connect (Socket.EU);
                    System.out.println ("Successfully connected to EU-Socket");
               } catch (WrongVoltage e) {
                    e.printStackTrace ();
               } finally{
                    r.onOff ();
               }
          System.out.println (r);
          try {
               for(int i = 0; i < 105; i ++) {
                    r.turnUp ();
                    System.out.println (r);
               }
          } catch (TooLoud e) {
               System.out.println ("Too loud!");
               e.printStackTrace ();
          }
     }
}
```