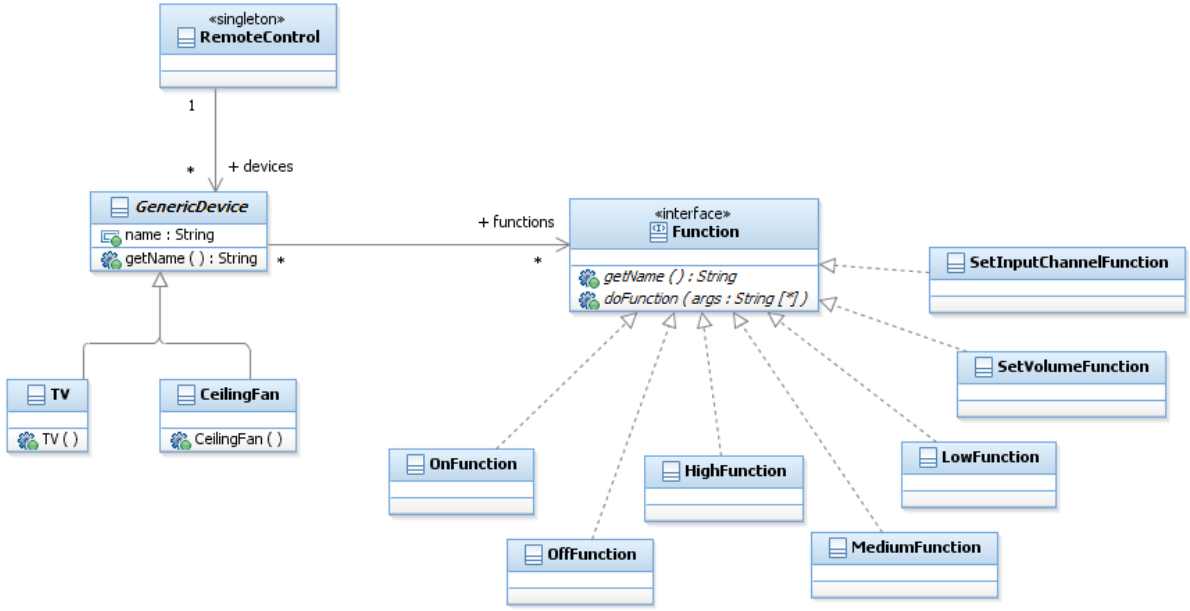
Intelliware 1.0 setup

Assignment 2

For the Intelliware 1.0 software package we came up with this design:



This means we have specific devices such as a CeilingFan or TV inheriting from a GenericDevice class. The GenericDevice class has an ArrayList with the interface class-type GenericFunction which is then filled with specific implementations of functions. The amount and type of functions are determined by the specific device classes.

There are also specific Function classes such as: On/Off, Notify, SetVolume etc.

The specific Function classes implement a Function interface, so that every functionclass has one specific method: doFunction(). This method allows a class to do what it is supposed to do. For example: turning on/off a device, setting the volume, changing the channel etc.

The method doFunction receives an undetermined number of parameters. Some functions will not use any (such as OnFunction and OffFunction), whereas others will need one or more parameters (such as SetVolumeFunction, which needs one parameter to set a particular volume).

With this design, which you could call a modular set up. It is easy to add new specific devices, supply them with existing functions or add new functions for the device and other new or existing devices to perform.

In this design, the *Strategy pattern* is easily found. It is used to add behavior (i.e. functions) to every device in a dynamic way.

Stefan Bos 500645244
Christian Aguilera 500641645
Ruben Herrebrugh 500614892

Code Snippets – IntelliWare 1.0

```
public class GenericDevice {
    protected ArrayList<Function> functions = new ArrayList<Function>();
    public boolean doFunction(String function, String value) {
        // loop through device functions to find desired function
        for (int i = 0; i < functions.size(); i++) {
                // verify if we have match
                        if (functions.get(i).getName().equals(function)) {
                                // create boolean to store result of function execution
                                boolean bResult;

                                // execute function, with or without parameter depending on input
                                if (value.isEmpty()) {
                                        bResult = functions.get(i).doFunction();
                                } else {
                                        bResult = functions.get(i).doFunction(value);
                                }
                                // display function result (success or failure)
                                if (bResult) {
                                        // print what executed function
                                        System.out.println(
                                        "function '" + function + value +
                                        "' peformed by " + name);
                                } else {
                                        System.out.println(
                                        "function '" + function + value +
                                        "' can not be executed by " + name +
                                        " (requires different arguments)");
                                }
                                // return we executed command
                                return true;
                        }
                }
        // function not found, not supported by device, print message
        System.out.println("function '" + function + "' not supported by " + name);

        // return false
        return false;
    }
}
```

Any functions can be passsed to any device however the Generic Device class will always filter out those requests that a device can't handle. The success or failure of a function will be returned.

```
public interface Function {
    public String getName();
    public boolean doFunction();
    public boolean doFunction(String value);
}

public class TV extends GenericDevice {
    public TV() {
        name = "TV";
        functions.add(new OnFunction());
        functions.add(new OffFunction());
        functions.add(new SetInputChannelFunction());
        functions.add(new SetVolumeFunction());
    }
}
```

You can create new devices and send commands directly to them. You can simulate any command being send to the device and the device class will filter out the invalid commands.

```
TV myTV = new TV();
myTV.doFunction("ON");
myTV.doFunction("SET_VOLUME", "20");
```

You could also manipulate multiple devices by using a Generic Device arraylist and a simple for each loop as illustrated below.

```
oaAllDevices.add(myFan);
oaAllDevices.add(myAlarm);
oaAllDevices.add(myTV);
for (GenericDevice device : oaAllDevices) {
        device.doFunction("OFF");
}
```

```
public class OnFunction implements Function {
    @Override
    public String getName() {
            return "ON";
    }
    @Override
        public boolean doFunction() {
            System.out.println("Turn On");
            return true;
        }
    @Override
    public boolean doFunction(String value) {
            return doFunction();
    }
}
```


Stefan Bos 500645244
Christian Aguilera 500641645
Ruben Herrebrugh 500614892