# Neues UseCase implementieren

Hier ist eine kurze Übersicht von Klassen beschrieben die erweitert werden müssen wenn ein neuen UseCase implementiert werden soll. Es ist nur als eine simple Hilfestellung für den Entwickler gedacht und enthält keine komplette Beschreibung von allen benötigten Schritten.

### mainWindow.java

Es wird als erstes aus der main geöffnet. Hier werden die Parameter für UseCases abgefragt und weitergeleitet. Wenn ein neues UseCase integriert werden soll müsste man in der Methode *UpdateSettingsTab()*anfangen. Daraufhin muss man die Checkbox und den Parameter Bereich schreiben. Dazu könnte man sich als Beispiel die bereits existierenden Methoden angucken: *settingMenu\_add.. .*

### Brick.java

Dies ist die Klasse die Attribute eines Bricks enthält. Hier sollten Attribute die für den neuen UseCase nötig sind vervollständigt werden.

### Events.java

In dieser Klasse werden Methoden geschrieben die für einen bestimmten Event zuständig sind. Zum Beispiel: *enableTmpltCntrl(..)* oder *updateTmplPlotWidth(..)*. Die entsprechenden Methoden für den neuen UseCase sollten hier addiert werden.

### WindowController.java

In dieser Klasse werden Methoden beschrieben die eine Relation zu den GUI-Fenstern haben. Die meisten davon fungieren als ein Interface zwischen dem mainWindow und dem sensorWindow. Hier muss man ebenfalls neue Methoden hinzufügen.

### sensorWindow.java

In dieser Klasse werden die Daten von den Sensoren kontrolliert und in den entsprechenden Graphen angezeigt. Hier werden die eigentlichen Methoden für die UseCases geschrieben. Als Referenz kann man die existierenden Methoden nehmen:

* Alle update..() – Methoden
* enableSimpleCntrl(..), disableSimpleCntrl(..)

Darüber hinaus braucht diese Klasse neue Maps für die neuen Marker, siehe z.B.: *markerMapMin1Critical* und *markerMapMin1Warning… ,*die in der Methode public static void addPlot(Brick newBrick) gesetzt werden. Außerdem muss die Variable *plotControlMapNum* auf die aktuelle Anzahl der useCases gestzt werden.

Für die Anzeige der Graphen wurden ausschließlich die „jFreeChart“ – Bibliotheken verwendet.

# Neuen Sensor hinzufügen

Hier wird grob beschrieben welche Klassen man erweitern soll um mit einem neuen Sensor arbeiten zu können.

### connection.java

In dieser Klasse wurden Methoden geschrieben die für die Verbindung mit den Bricks zuständig sind. Außerdem sind hier die Listeners beschrieben die jedem Sensor hinzugewiesen sind. Als Referenz kann man die existierenden Methoden: *stopIlluminancelistener(..)* und *startIlluminanceListener(..)* nehmen. Außerdem sollte man die *stopAllSensors()* und *startAllSensors()* Methoden vervollständigen.

### constants.java

Hier werden alle benötigten statischen Daten über die Sensoren beschrieben. Hier muss jede Methode vervollständigt werden. Die BrickIds findet man auf TinkerForge – Seite.

Bei der Entwicklung bin ich davon ausgegangen dass TinkerForge nur einen Brick hat der gleichzeitig 2 verschiedene Daten-Ströme liefert. Das ist der Voltage/Current Brick. Falls Sie einen solchen dualen Brick hinzufügen möchten müssen das Programm an vielen Stellen überarbeitet werden. Da die benötigte Absprache sich immer auf die Id-Nummer des Bricks bezieht (für Voltage/Current ist das die 227) muss man an jeder Stelle wo es nach der Nummer 227 abgefragt wird die neue dazutun. Bei den einfachen Sensoren brauch man das nicht zu beachten.

