# „Fertige“ Ampelanlage in BlueJ

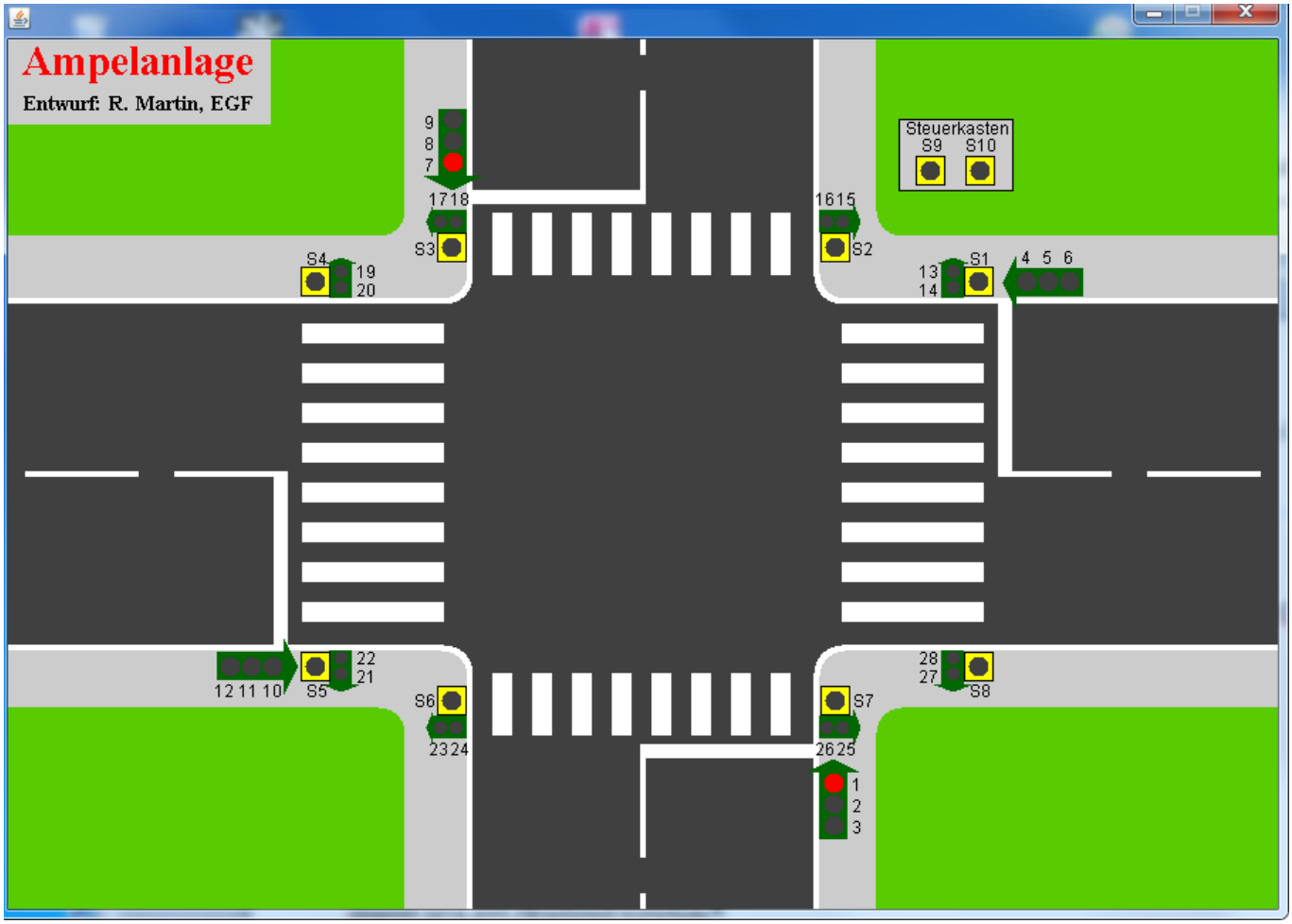


Liebe Kolleginnen und Kollegen,

beim Workshop „CrossRoads und BlueJ“ am 29.2.2012 in Münchberg kam mir die Idee, das CrossRoad-Board als Klasse nachzubilden und sie in BlueJ so zu benutzen, als wäre dies die Hardware.

Herausgekommen ist eine „fertige“ Ampelanlage, die Auto- und Fußgängerampeln sowie Schalter zur Bedienung zur Verfügung stellt.

Abbildung 1 zeigt den Screenshot der Ampelanlage:




**Das Konzept:**

Man muss sich (wie bei CrossRoads) nicht mehr um den graphischen Aufbau der Kreuzung und der Ampeln kümmern, sondern kann sich sofort mit der Bedienung und der Ampelsteuerung beschäftigen. Mühsames Positionieren einzelner Lampen, Ampeln usw. entfällt.

Die komplette Steuerung erfolgt über die Klasse „Board“ auf die man als Anwender zugreifen kann. Die Klassen „View“ (enthält die Graphik), „Led“, „Button“ und „LoopThread“ werden von „Board“ benutzt bzw. angesteuert und sind prinzipiell für den Anwender unsichtbar.

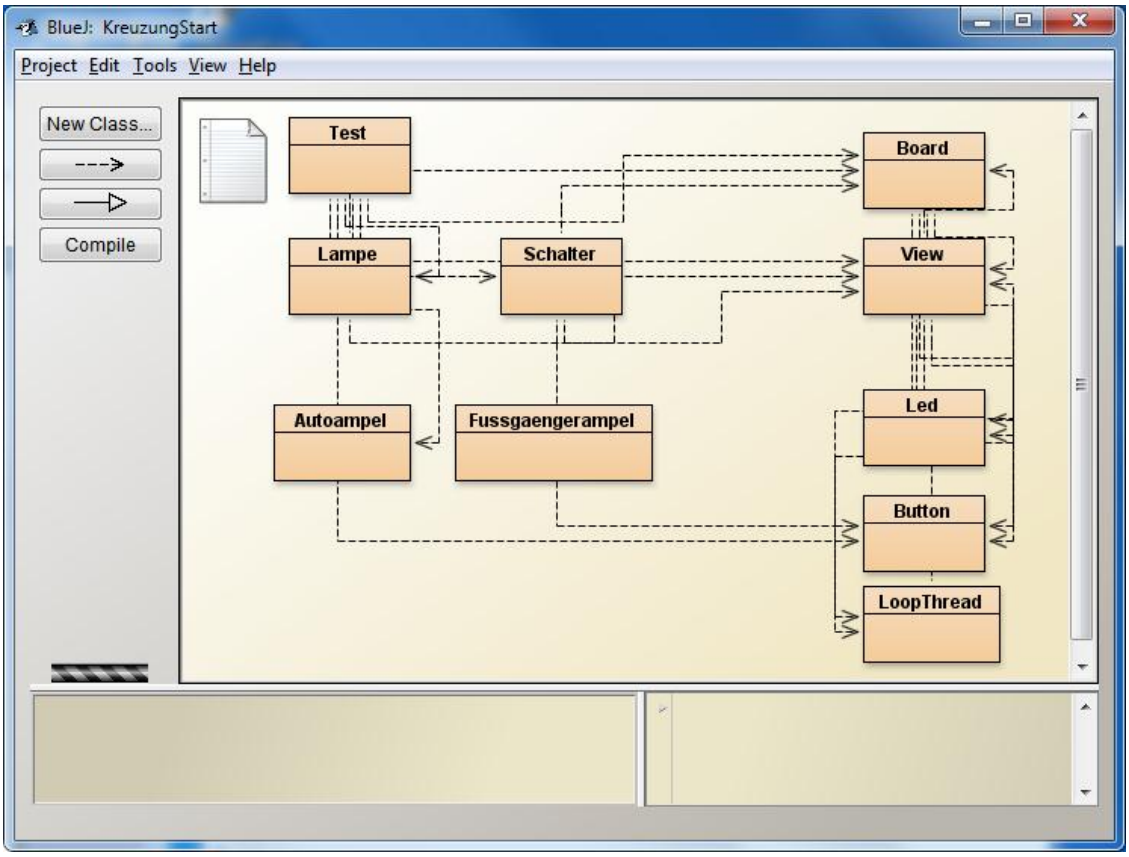
Übrigens: Sachdienliche Hinweise zum Verstecken dieser Klassen werden dankend angenommen.

Mit Hilfe einer Klasse „Lampe“ kann man auf eine einzelne Lampe (nummeriert von 1 bis 28; Index von 0 bis 27) zugreifen. Damit lässt sich jede einzelne Lampe der Ampelanlage ein- und ausschalten kann.

Entsprechend kann man mit Hilfe einer Klasse „Schalter“ auf die Buttons (S1 bis S10; Index von 0 bis 9) zugreifen. Hiermit kann man Umschalter (Schalter offen bzw. geschlossen) oder Taster erzeugen.

In den Klassen „Autoampel“ und „Fussgaengerampel“ wird man jeweils 3 bzw. 2 Lampen zu einer Ampel zusammenfassen. Diese besitzen typische Methoden wie „weiterschalten“ der Ampelphase.

Eine Klasse „Steuerung“ (im Beispiel noch „Test“ genannt.) wird sich um die komplette Ampelsteuerung kümmern.




Hinweis: Am Anfang wird man aus dem BlueJ-Projekt die Anwenderklassen „Test“, „Lampe“, „Schalter“, „Autoampel“ und „Fussgaengerampel“ löschen und erste schrittweise mit den Schülern entwickeln.

**Die Methoden der Klasse „Board“**

```
Board getBoard()
```

gibt die einzige Instanz der Klasse Board zurück und erzeugt diese ggf.

```
Button getButton(int index)
```

gibt den Button mit Index index zurück

```
Led[] getListLeds():
```

gibt die Liste der Leds zurück

```
Led getLed(int index)
```

gibt die Led mit Index index zurück

```
void setLed(int index) {
```

fügt eine Led mit der Farbe rot zur Liste der Leds hinzu

```
void turnLedOn(int index)
```

schaltet die Led mit Index index an

```
void turnLedOff(int index){
```

schaltet die Led mit Index index aus

```
void pressButtonAsSwitch(int index){
```

schaltet den Zustand des Buttons mit Index index um (offen ⇔ geschlossen); Einsatz als Schalter

```
void pressButtonAsTaste(int index, int repeat)
```

schaltet den Zustand des Buttons mit Index index nicht um; stattdessen blinkt der Buttons repeat-mal kurz auf; Einsatz als Taster

```
boolean istButtonOpen(int index){
```

prüft, ob der Button mit Index index geöffnet ist

```
int getIndexButtonPressed()
```

gibt den Index des gedrückten Buttons zurück

```
void updateButtonPressed(int index){
```

wird von View aufgerufen, wenn ein Button gedrückt wurde; setzt die Variable „indexSchalter“ auf den aktuellen Index; wird z. B. in „Test“ ständig abgefragt

```
void loop(Object objekt, String methodenname)
```

startet die angegebene Methode als fortlaufende Wiederholung in einem nebenläufigen Prozess (Thread)

```
void run(Object objekt, String methodenname) {
```

startet die angegebene Methode einmalig als nebenläufigen Prozess (Thread)

```
void loopWithDelay(Object objekt, String methodenname, int millisekunden,
boolean status)
```

startet die angegebene Methode als fortlaufende Wiederholung und wartet nach jeder Ausführung für die angegebene Zeit in Millisekunden; status muss dazu auf false gesetzt werden

`void stopLoop(int index)`

stoppt den Thread mit Index index

`void stopp(int ab) {`

stoppt alle Threads ab dem Index index

`void stopLoop(String methodenname)`

stoppt den Thread mit dem angegebenen Namen

`void listLoops()`

listet alle laufenden Threads auf, die mit loop(...) gestartet wurden

`void startClock()`

startet die Stoppuhr

`long stopClock()`

stoppt die Stoppuhr und gibt verstrichene Zeit seit dem Stoppuhr-Start in Millisekunden zurück

`int getIndexSchalterGedrueckt()`

gibt den Index des zuletzt gedrückten Buttons zurück

Hinweis: Einige Methoden habe ich aus dem im Workshop vorgestellten BlueJ-Projekt „CrossRoads“ übernommen.

Hinweis: Prinzipiell können alle Klassen verändert werden, insb. die Klasse „View“, so dass man diese auf eine neue Anwendungssituation anpassen kann. Vorschläge?!

Hinweis: Das vorliegende BlueJ-Projekt darf gerne an weitere Kolleginnen und Kollegen weitergegeben und uneingeschränkt benutzt werden.

Ich wünsche viel Erfolg beim Einsatz im Unterricht und würde mich über Rückmeldungen freuen.

Rainer Martin

Ehrenbürg-Gymnasium Forchheim

Forchheim, 6.3.2012