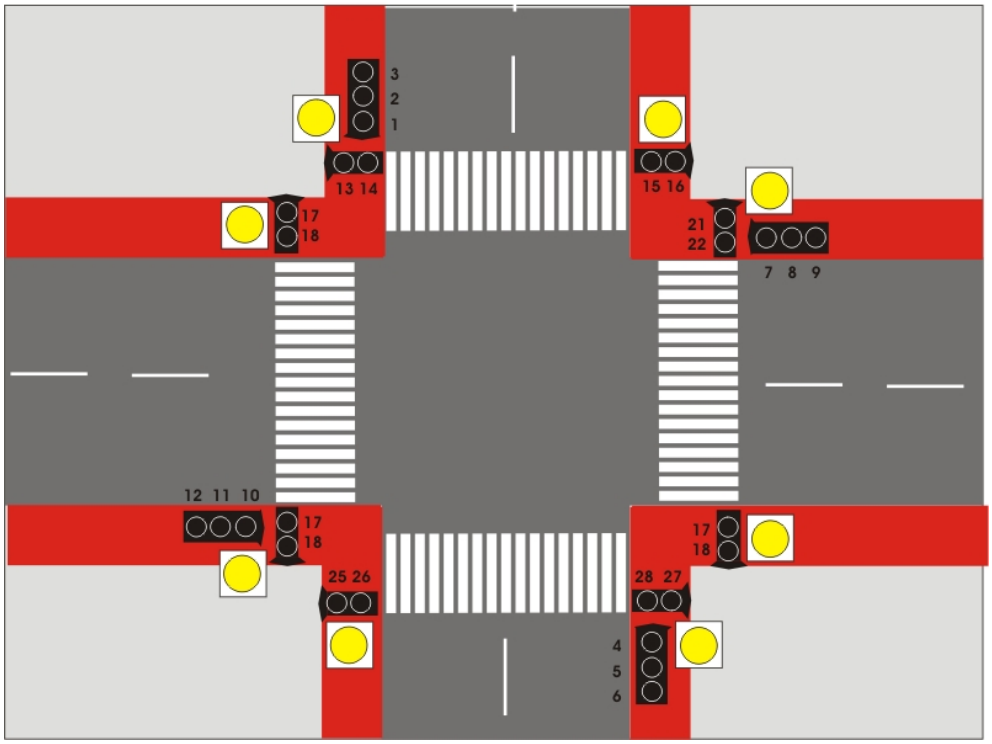
Ampelsteuerung an einer Kreuzung

Im folgenden Projekt sollen wir eine Software schreiben, die unsere Ampeln an einer Kreuzung steuern. Die Ampelanlage wird natürlich nur am Bildschirm angezeigt. Wir wollen uns aber eine real existierende Kreuzung mit 4 Autoampeln und 8 Fußgängerampeln vorstellen.

Quelle: maps.google.de (aufgerufen am 20.03.2012)

Wenn wir uns ein erstes Modell dieser Straßenkreuzung machen, dann könnte das – auf deutsche Ampeln ausgelegt – in etwa so aussehen:



Wir sehen vier Straßenampeln mit jeweils drei Lampen pro Ampel und acht Fußgängerampeln mit jeweils zwei Lampen. Die Lampen wurden durchnummeriert von 1 bis 27. In der Realität soll an einer Ecke der Straßenkreuzung ein Schaltschrank stehen, an dem die einzelnen Lampen ein- bzw. ausgeschaltet werden können. Glücklicherweise kann die „Hardware" auch mittels einer Software-Schnittstelle bedient werden. Unsere Aufgabe ist es, eine Software zu schreiben, mit der die Ampelsteuerung erfolgt.

Die gelben Kreise stellen „Druckknöpfe" dar, auf denen die Fußgänger drücken können, um eine grüne Ampel anzufordern.

Lies Dir folgenden Beitrag durch:

Drücken sinnlos! Viele Fußgängerampeln automatisch geschaltet BILD enttarnt den Nix-passiert-Knopf

22.02.2011 — 00:18 Uhr
Von Merle Schlesselmann

Hamburg – **Knopf drücken, auf Grün warten, gehen. So funktioniert das an der Ampel. Oder auch nicht!**

Viele Signalknöpfe sind nämlich völlig wirkungslos. Ob der Fußgänger drückt oder nicht, die Ampel schaltet im immer gleichen Takt um.

BILD enttarnt den Nix-passiert-Knopf!

Die Schäferkampsallee in Eimsbüttel. Auf Höhe Moorkamp führt eine Fußgängerampel über die Straße. Wird der „Anforderungstaster" gedrückt, wird die Fußgängerampel nach 55 Sekunden grün. Drückt man ihn nicht – ebenfalls! Auf die Sekunde genau gleich. Ebenso ist es an der Kreuzung Wandsbeker Marktstraße/Wandsbeker Allee und am Jungfernstieg vorm Streits.

Matthias Schmitting vom ADAC erklärt: „Zu Stoßzeiten muss der Verkehr fließen. Deshalb werden Ampeln, z. B. an Hauptverkehrsstraßen, im Berufsverkehr automatisch geschaltet. Das gilt auch für Fußgänger-Signale. Anforderungstaster haben dann keinen Einfluss."

Fast 5000 gelbe Kästen sind an Hamburgs 1266 Ampelanlagen montiert. Jedes Gerät kostet rund 200 Euro. Gesamtwert: 1 Million Euro. Rausgeschmissenes Geld?

„Nein, denn die Anlagen wechseln die Betriebsart je nach Verkehrsaufkommen", so Helma Krstanoski von der Stadtentwicklungsbehörde. „Zudem gibt es auch Anlagen, bei denen Fußgänger nur nach Anforderung Grün bekommen."

Quelle: http://www.bild.de/regional/hamburg/verkehr/druecken-sinnlos-16095024.bild.html aufgerufen am 20.03.2012

Wenn wir mit unserer Software fertig sind, dann soll es je nach Tageszeit verschiedene Modi geben, z. B. „Nachtmodus", „Normal", „Hauptverkehrszeit", „Anforderungsmodus".

Zunächst wollen wir uns die Hardware näher betrachten.

1. Öffne das Projekt „Strassenkreuzung01“. Erstelle eine Klasse „Hardwaretest“. Die Hardware ist vorhanden und hat eine Schnittstelle zu unseren BlueJ-Programmen:

Hardware

+void: lampeAn(int);
+void: lampeAus(int);
+void: pause(int) // wartet Millisek.
+Boolean: istGedrueckt(int) //Taster
+void: unlockTaster(int) //Taster aus

Wir erreichen die Hardware über ein Referenzattribut `hardware`, das fest in unserer BlueJ-Programmumgebung vorhanden ist. Welche Lampen du ein- und ausschalten kannst, entnimmst du dem Schaltplan.

2. Modelliere eine Klasse „Lampe“

Wir wollen nun weggehen von der Hardware und uns die Objekte der Hardware näher betrachten. Wir wollen dies von den kleinsten Einheiten aus machen (Bottom-Up-Methode), weil wir dann das Funktionieren der einzelnen Objekte sofort ausprobieren können. Eine andere Möglichkeit wäre, sich zuerst die Ampeln als Objekte vorzunehmen und dann nach und nach deren Innenleben zu modellieren (Top-Down-Ansatz).

Lampe
+Lampe(int);
-int nummer;
+void: an(); +void: aus();

Das kleinste Objekt, das funktionieren soll, ist die einzelne Lampe in der Ampel, die eine rote, gelbe oder grüne Farbe besitzt. Um die Farbe müssen wir uns aber nicht kümmern, da die ja in „hardware“ festgelegt ist. Die Lampe soll über die Methoden `public void an();` und `public void aus();` verfügen. Im Konstruktor der Klasse Lampe benötigen wir eine ganze Zahl als Parameter, um dem Attribut „nummer“ einen Wert zuweisen zu können. Die Nummer der Lampe soll dann der Nummer der hardware-Lampe entsprechen. Auf dem Schaltplan sind die Nummern bei den einzelnen Ampeln angegeben.

3. Modelliere eine Klasse „Fussgaengerampel“

Eine Fußgängerampel soll aus zwei Lampen „zusammengesetzt“ werden. Wir brauchen also zwei Objekte der Klasse Lampe und nennen die Referenzattribute „rot“ und „gruen“. Implementiere die Klasse und erstelle die Methoden `public void gruen(); public void rot();` und `public void aus();`, mit der die Ampelzustände geschalten werden können. Ergänze die Klasse um ein Attribut `String zustand` und eine sondierende Methode, mit der der aktuelle Zustand der Fußgängerampel ermittelt werden kann. Der Konstruktor verlangt hier ebenfalls einen Parameter vom Datentyp `int`, mit dem in unserer Ampelanlage die Nummer der roten Lampe der Fußgängerampel angegeben wird. Wie du leicht erkennen kannst ergibt sich die Nummer der grünen Lampe aus der roten, indem du eins addierst.

Fussgaengerampel
+Fussgaengerampel(int);
-Lampe: rot; -Lampe: gruen; -String: zustand;
+void: gruen(); +void: rot(); +void: aus(); +String: getZustand();

Probiere es aus: Erstelle mehrere Fußgängerampeln und schalte ihre Lampen auf rot und grün.

4. Modelliere die Klasse „Strassenampel“

Die Straßenampel besteht aus drei Lampen und genau so werden wir uns auch diese zusammensetzen. Wir bauen also drei Objekte der Klasse Lampe zur Straßenampel zusammen. Dies gelingt uns durch Implementierung von drei Referenzattributen „rot“, „gelb“ und „gruen“. Implementiere die Methoden `public void gruen();` `public void rot();` `public void gelb();` `public void rotgelb();` und `public void aus();`, mit denen die Ampelzustände geschalten werden können. Ergänze auch hier die Klasse um ein Attribut `String zustand` und eine sondierende Methode, mit der du den aktuellen Zustand der Straßenampel ermitteln kannst. Erstelle zudem eine Methode `public void weiterschalten();`, mit der bei mehrmaligem Aufrufen der Methode nacheinander die Ampelphasen rot – rotgelb – grün – gelb – rot usw. aufgerufen werden. (Falls du nicht weißt, wie Straßenampeln schalten, musst du mal wieder mit offenen Augen durch deine Stadt gehen.)

Strassenampel
+Strassenampel(int);
-Lampe: rot; -Lampe: gelb; -Lampe: gruen; -String: zustand;
+void: rot(); +void: gelb(); +void: gruen(); +void: aus(); +void: weiterschalten(); +String: getZustand();

5. Erstelle ein Klasse „Steuerung“

In der Klasse sollen zunächst vier Objekte der Klasse „Strassenampel“ gesteuert werden. Wähle die vier Straßenampeln aus dem Plan aus und vergebe für sie Namen „s1“, „s2“, „s3“ und „s4“. Trage die Namen der Ampeln in deinen Plan ein, um den Überblick zu behalten. Erstelle vier Referenzattribute mit der gleichen Bezeichnung. „s1“ und „s2“ sollten gegenüberliegende Straßenampeln sein, weil die beiden immer gleich geschaltet werden. Man kann sich die beiden dann besser merken. Wie immer müssen die Objekte im Konstruktor mit der Anweisung `new` erzeugt werden.

In der Steuerung sollst du nun eine Methode `public void start()` programmieren, die zwei Ampeln auf „grün“ schaltet, während die anderen beiden Ampeln „rot“ zeigen. Nach fünf Sekunden sollen die Ampeln umschalten, so dass am Ende die anderen beiden Ampeln „grün“ zeigen und die ersten beiden „rot“. Dies soll sich ständig wiederholen. Verwende die Methode `public void pause(int)` der `hardware`, um die Pausen in die Steuerung einzufügen. Überlege Dir vorher ein Zustandsübergangsdiagramm für diese Methode.

Erstelle eine Methode `public void nachtmodus()`, bei der die gelben Lampen der beiden an der vertikal verlaufenden Straße stehenden Straßenampeln im Abstand von einer Sekunde gleichzeitig blinken.