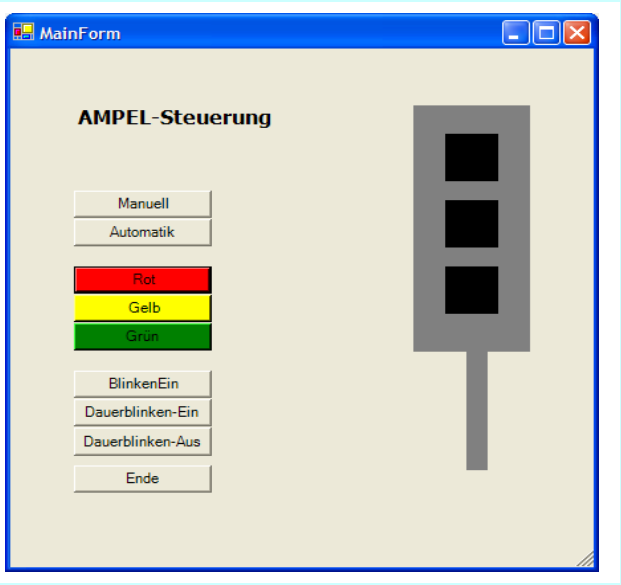
1.7. Weitere Aufgaben

1.7.1. Aufgabe: SwingAmpel (HU)



Es soll eine primitive manuelle Schaltung für eine Straßenbaustellenampel erstellt werden. Ein Mann/Frau soll die Ampel per Hand steuern können.

Die Ampel hat die drei Lampen Rot, Grün und Gelb und zum Schalten für jede Lampe einen Button.

Wenn z. B. die rote Lampe leuchtet, müssen natürlich dann die beiden anderen ausgeschaltet werden.
Beim Programmstart sind alle Lampen schwarz.

Übung 1.0: Die Ampel erstellen
☑ Bauen Sie aus Panels und Buttons die "Ampel" und Steuerung zusammen!
☑ Erzeugen Sie durch Klick auf die Buttons (Rot, Gelb, Grün) die Ereignismethoden und implementieren Sie den Code!

Übung 1.1 – Mehrfaches Blinken: Schleifen (Zählschleife, vor- und nachprüfende Schleife)

BlinkenEin
Als Erweiterungen soll eine Blinkschaltung für die gelbe Lampe eingebaut werden. Dazu wird ein neuer Button: **jButtonBlinkenEin** (Text: BlinkenEin) eingebaut.
Das Blinken lässt sich mit einer Schleife realisieren, die dafür sorgt, dass die gelbe Lampe im Wechsel 'an' oder 'aus' gezeigt wird (mit einer kleinen Pause dazwischen). Dies soll z.B. 5 mal passieren.

Hinweis:
final int intAnzahl =5;

```
.....
for (int i=0; i < intAnzahl; i++){
        ....
        java.lang.Thread.sleep( zeit in Millisekunden)
        ....
}
```

1.7.2. Hinweis: Threads

Problem: java.lang.Thread.sleep(1000);
Da durch dieses Sleep der aktuelle Thread (hier die GUI Oberfläche) schlafen gelegt wird, können keine Buttons gedrückt werden. Es können auch keine Panels eingefärbt werden.

Lösung: Thread
Deswegen muss man einen eigenen Thread erstellen, der parallel zur möglichen Usereingabe, die Panels einfärbt und dann auch noch genügend lang schläft.

☑ Zunächst definieren wir eine Variable namens runBlinkenEin vom Typ java.lang.Runnable.

```
java.lang.Runnable runBlinkenEin= new java.lang.Runnable()
{
        public void run()
        {
                for (int i=0; i < intAnzahl; i++)
                {
                        jPanelGelb.setBackground(Color.yellow);

                        try{
                                java.lang.Thread.sleep(1000);

                                jPanelGelb.setBackground(Color.black);

                                java.lang.Thread.sleep(1000);

                        }catch (InterruptedException err){
                                        err.printStackTrace(System.out);
                        }
                }//for
        }//run
};
```

☑ Wenn nun der Button **jButtonBlinkenEin** gedrückt wird, dann wird mit

```
....
public void actionPerformed(java.awt.event.ActionEvent e) {

        java.lang.Thread blinkenEinThread= new java.lang.Thread(runBlinkenEin);
        blinkenEinThread.start();
....
```

der oben programmierte Thread gestartet.

siehe auch:
http://www.dpunkt.de/java/Programmieren_mit_Java/Oberflaechenprogrammierung/44.html

Übung 1.2 – DauerBlinken: Globale Zustandsvariablen und while
Wenn die Zählschleife fehlerfrei läuft, erstellen Sie bitte eine weitere Funktionalität der Ampel, die die WHILE-Schleife verwendet:
Dazu wollen wir zwei weitere Buttons einsetzen:
☑ **jButtonDauerblinkenEin** (Text: Dauerblinken-Ein)
☑ **jButtonDauerblinkenAus** (Text: Dauerblinken-Aus)

Die Ereignsiprozedur von **jButtonDauerblinkenEin** enthält eine Schleife, die keine voreingestellte Wiederholzahl hat. Man verwendet dabei eine sog. WHILE-Schleife, die jedesmal die aktuelle Abbruchsbedingung prüft. Dazu führen wir eine Zustandsvariable

boolean *boolBlinken;*

ein, die mit einem neuen Aus-Button (**jButtonDauerblinkenAus**) auf den Wert *false* gesetzt wird.

Hinweis: Definition einer globalen Variablen
Damit mehrere Ereignisprozeduren eine Information gemeinsam verwenden, kann man u.a. so genannte globale Variablen verwenden. Diese werden am Beginn des Programmes (ausserhalb von Funktionen/Methoden) definiert.

```
// Meine gloablen Variablen
boolean boolBlinken= false;
```

Die einzelnen Funktionen/Methoden können dann diese Variable auslesen bzw. einen neuen Wert setzen.

Hinweis: Synchronisation
Natürlich darf nicht jeder Thread eine globale Variable beliebig ändern. Dieses Problem der Synchronisation wollen wir aber hier nicht behandeln.

Übung 1.3 – Notabschaltung: Entscheidungen (if - then - else)
Im letzten Ausbauschritt soll ein Zähler eingebaut werden, der die Blinkvorgänge mitzählt, um bei bestimmten Situationen automatisch abzuschalten.
Wenn der Zähler den Wert 10 erreicht,
dann eine Warnung ausgegeben
sonst wenn der Wert 15 erreicht ist,
dann eine Notabschaltung ausführen
ansonsten Zählerwert anzeigen

Übung 1.4 – Automatik: Entscheidungen ()
Baustellenampel automatisieren
Eine ordentliche Ampel muss auch nachts und am Wochenende automatisch laufen können. Das ist ziemlich einfach zu programmieren. Die Schaltphasen der drei Lampen brauchen nur nacheinander mit kleinen Pausen dazwischen abzulaufen.

Die Button-Clicks werden nicht mehr manuell bedient, sondern in einer Methode vom Programm selbst wiederholt bis die Automatik abgestellt wird.
Dazu sind folgende Schritte erforderlich:
Ausbaustufe 1 : Automatik einbauen
☑ Führen Sie zwei neue Buttons ein: **jButtonAutomatik** und **jButtonManuell**
☑ Deklarieren Sie eine globale Variable mit **boolean boolAutomatik;**
mit der sich das Programm merkt, ob die automatische Steuerung gerade ein- oder ausgeschaltet ist.

☑ Erzeugen Sie eine Runnable-Variable:

```
java.lang.Runnable runAutomatik= new Runnable()
                {
                   public void run()
                   {

                        boolAutomatik=true;  //  @jve:decl-index=0:
                              while (boolAutomatik)
                              {
                                    try
                                    {
                                          // rot
                                          java.lang.Thread.sleep(2000 );

                                          //gelb
                                          java.lang.Thread.sleep(2000 );

                                          //gruen
                                          java.lang.Thread.sleep(2000 );
                                    }
                                    catch(InterruptedException err){
                                          err.printStackTrace(System.out);
                                    }
                              }//while
                   }//run
          };
```

1.7.3. Hinweis: Ereignisprozeduren direkt aufrufen

Man kann das Klick-Ereignis mittels Aufruf einer Funktion nachahmen:

```
jButtonRot.doClick(10);
```

☑ Starten Sie das Programm und testen Sie es!

Ausbaustufe 2 : Umschalten

Damit die Handsteuerung und die automatische Steuerung nicht gleichzeitig bedient werden können, sollten die Einzelbuttons deaktiviert sein.
JbuttonGelb.enabled(false);
....

Zusammenfassung:
Erstelle eine Zusammenfassung des in diesem Kapitel gelernten Stoffes in einer Word-Datei java-ampel.doc (mit kl. Programmbeispielen)
☑ Variablen, Konstanten definieren
☑ Datentypen verstehen und anwenden
☑ Globale Variable, Lokale Variable
☑ Verzweigungen
☑ Schleifen
☑ Threads