— Übungsblatt 5 (P) —

Aufgabe 16 (P) (Threads)

Schreiben Sie eine Konsolenanwendung, die als einfaches Stoppuhr-Programm eingesetzt werden kann. Die `main`-Methode soll

- auf das Betätigen der Eingabetaste warten,
- den Start-Zeitpunkt in einem `Date`-Objekt festhalten und im Konsolenfenster ausgeben,
- ein Thread-Objekt für die Anzeige der Uhrzeit erzeugen und starten,
- auf das Betätigen der Eingabetaste warten,
- den Stopp-Zeitpunkt in einem `Date`-Objekt festhalten und im Konsolenfenster ausgeben,
- den Uhrzeit-Anzeige-Thread anhalten und
- die gemessene Laufzeit als Differenz von Stopp- und Start-Zeitpunkt bestimmen und im Konsolenfenster ausgeben.

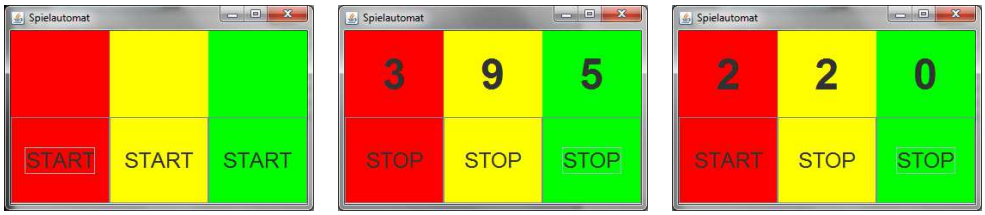
In der selbst zu schreibenden Klasse für das benötigte Thread-Objekt soll die Methode `run` so implementiert werden, dass in einer Endlosschleife in jedem Durchlauf jeweils der aktuelle Zeitpunkt bestimmt, dieser formatiert ausgegeben und eine Sekunde pausiert wird. Die Schleife bzw. der Thread soll mittels des internen Abbruch-Flags angehalten werden können.
Die Ausgabe des Programms soll etwa wie folgt aussehen:

```
Stoppuhr starten mit Eingabetaste!
Startzeitpunkt: Fri Jan 11 14:02:40 CET 2019
Stoppuhr anhalten mit Eingabetaste!
14:02:40
14:02:41
14:02:42
Stoppzeitpunkt: Fri Jan 11 14:02:43 CET 2019
Gesamtlaufzeit: 2743 ms
```

Dabei soll zur Laufzeit des Programms die Zeitanzeige im Sekundentakt aktualisiert werden.

Aufgabe 17 (P) (Frame mit Threads)

Entwerfen Sie einen einfachen Spielautomaten in Form eines Frames unter Verwendung von Labels, Buttons und Threads. Der Automat soll folgende Gestalt bzw. Benutzungsmöglichkeit haben.



Der Frame soll also beim Start die im linken Bild dargestellte Form mit unterschiedlich farbigen Spalten haben. Der bzw. die Benutzer(in) kann durch Druck jeder START-Taste die jeweils darüberliegende Zufalls-Ziffern-Anzeige zum Laufen bringen (mittleres Bild: die Ziffern verändern sich ständig).

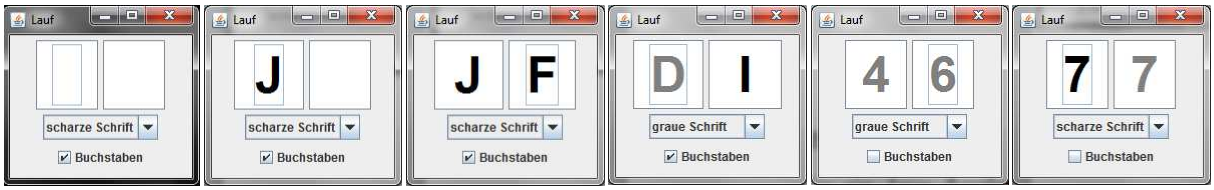
Mittels der gleichen Taste (nunmehr STOP-Taste) können die Zufalls-Generatoren wieder gestoppt werden (rechtes Bild: linke Ziffer steht, die anderen laufen weiter und können mit den entsprechenden STOP-Tasten angehalten werden).

Gehen Sie wie folgt vor:

- Schreiben Sie eine Klasse `ColorRunLabel`, die von `JLabel` erbt und als Thread lauffähig ist. Statten Sie diese Klasse mit den nachfolgend beschriebenen Elementen aus.
  - Der Konstruktor soll den Hintergrund des Labels auf die ihm übergebene Farbe setzen und eine fettgedruckte Arial-Schrift in der Größe 50 Punkt für die zentrierte Beschriftung des Labels festlegen.
  - Eine Methode `start()` soll das ausführende Label-Objekt mit einem Thread verbinden, diesen starten und ein Flag setzen, das anzeigt, dass der Thread läuft.
  - Die Methode `stop()` soll das Flag, das anzeigt, dass der Thread läuft, zurücksetzen.
  - Die Methode `run()` soll in einer Schleife (solange das Flag gesetzt ist) jeweils
    - mittels der Methode `Math.random()` und geeigneter Skalierung Zufallswerte zwischen 0 und 9 (einschließlich) erzeugen,
    - diese auf dem Label anzeigen und
    - unter Einsatz der `sleep`-Methode 10 ms lang pausieren.
- Schreiben Sie eine Klasse `StartStopButton`, die von `JButton` erbt. Statten Sie diese Klasse mit den nachfolgend beschriebenen Elementen aus.
  - Der Konstruktor soll den Hintergrund des Buttons auf die ihm übergebene Farbe setzen, eine Standard-Arial-Schrift in der Größe 25 Punkt für die zentrierte Beschriftung des Buttons festlegen und den Button mit dem Schriftzug START beschriften.
  - Die Methode `isStart()` soll genau dann den Wert `true` als Ergebnis zurückliefern, wenn das ausführende Button-Objekt mit START beschriftet ist.
  - Die Methode `switchText()` soll die Button-Beschriftung umschalten, d. h. anstelle von START soll STOP zu lesen sein und umgekehrt.
- Schreiben Sie eine Klasse `AutomatFrame`, die den Spielautomat-Frame realisiert. Statten Sie diese Klasse mit den nachfolgend beschriebenen Elementen aus.
  - Drei `ColorRunLabel`-Variablen dienen der Anzeige der Zufallsziffern.
  - Drei `StartStopButton`-Variablen dienen der Darstellung der zugehörigen Tasten.
  - Im Konstruktor werden die benötigten Label- und Button-Objekte (z. B. in rot, gelb und grün) erzeugt und diese gemäß einem geeigneten Layout dem Frame hinzugefügt. Außerdem wird jede Taste mit einem `ActionListener`-Objekt verbunden.
  - Die zugehörige Ereignis-Behandlung soll für jedes der drei Button-Objekte über ein und dieselbe innere Klasse realisiert werden, die das Interface `ActionListener` implementiert. Bei der Erzeugung des Listener-Objektes soll die entsprechende Taste sowie das zugehörige Label dem Konstruktor übergeben werden. In der `actionPerformed`-Methode soll mit Hilfe der Methoden `isStart` und `switchText` der Taste sowie `start` und `stop` des Labels vernünftig auf einen Tastendruck reagiert werden. Das heißt, wenn die Taste die Bedeutung START hat, soll der Label-Thread gestartet werden, andernfalls soll dieser gestoppt werden.

Aufgabe 18 (T & P) (Frames mit Threads)

Erstellen Sie einen Java-Frame, der zwei Button-Objekte sowie ein ComboBox- und ein Checkbox-Objekt beinhaltet und die folgende Gestalt bzw. Benutzungsmöglichkeit haben soll:



Der Frame soll also beim Start die in Bild 1 dargestellte Form haben. Durch Druck eines Buttons kann dessen Zufalls-Buchstaben-Anzeige zum Laufen gebracht werden (Bild 2 und 3, die Buchstaben verändern sich ständig). Mit der gleichen Taste können die Zufalls-Generatoren auch wieder gestoppt werden. Durch Änderung des ComboBox-Objekts kann anstelle einer schwarzen eine graue Schrift gewählt werden (Bild 4). Durch das Entfernen des Hakens in der Checkbox kann der Zufalls-Buchstabe durch eine Zufalls-Ziffer ersetzt werden (Bild 5). Alle diese Änderungen werden jedoch immer erst aktiv, wenn die entsprechende Taste wieder gedrückt wird (Bild 6, linke Taste wurde wieder aktiviert).

Vervollständigen Sie die nachfolgenden Klassen `ColorRunButton` und `LaufFrame` wie folgt:

a) Die Klasse `ColorRunButton`

- Ergänzen Sie die Methode `change` zur Steuerung des Threads. Falls der Thread bereits läuft, soll lediglich das Flag `running` auf `false` gesetzt werden. Andernfalls soll das Flag auf `true` gesetzt werden und ein mit dem aktuellen Button-Objekt verbundener Thread erzeugt und gestartet werden.
- Ergänzen Sie die Methode `run`. Darin soll, solange das Flag `running` gesetzt ist, in einer Schleife
  - eine Zufallsziffer im Bereich 0,...,9 berechnet werden,
  - diese falls notwendig (abhängig vom Flag `zeigtBuchstabe`) in einen `char`-Wert im Bereich A,...,J gewandelt werden und
  - das entsprechende Zeichen als Beschriftung (Label) des Buttons gesetzt werden.

b) Die Klasse `LaufFrame`

- Vervollständigen Sie den Konstruktor so, dass
  - die benötigten Swing-Komponenten erzeugt werden und
  - die `Button`-Objekte mit entsprechenden Event-Listener-Objekten verküpft werden.
- Realisieren Sie die Ereignis-Behandlung für die `Button`-Objekte in Form einer inneren Klasse `KnopfListener`, die das Interface `ActionListener` implementiert. Bei Betätigung des zugeordneten Buttons soll jeweils folgendes geschehen:
  - Der Auswahl-Index (`SelectedIndex`) des ComboBox-Objekts wird bestimmt.
  - Die Schriftfarbe des Buttons wird gesetzt.
  - Der Zustand (`State`) der Checkbox-Markierung wird bestimmt.
  - Die Beschriftungsart (Buchstaben oder Ziffern) und der aktuelle Thread-Zustand des Buttons wird mittels der Instanz-Methode `change` des Button-Objekts gewechselt.

```
import java.awt.*;
import javax.swing.*;
public class ColorRunButton extends JButton implements Runnable {

  private boolean running = false;       // Thread läuft zu Beginn nicht

  private boolean zeigtBuchstabe = true; // Anzeige zeigt zu Beginn Buchstaben

  public ColorRunButton () {
    setBackground(Color.white);
    setFont(new Font("Arial",Font.BOLD,50));
    setText("  ");
  }

  public void change (boolean zeigtBuchstabe) {
    this.zeigtBuchstabe = zeigtBuchstabe;
    // Falls der Thread läuft, beende ihn mittels des running-Flags,
    // andernfalls, setze running-Flag, erzeuge neuen Thread und starte ihn
     .
     .
     .
     .
     .
     .
     .
     .
     .
     .
     .
  }
  public void run() {
    String text;
    while (running) {
      // Berechne Zufalls-Ziffer oder Zufalls-Buchstabe
      // und setze das Zeichen als Beschriftung des Buttons
       .
       .
       .
       .
       .
       .
       .
       .
       .
    }
  }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class LaufFrame extends JFrame {
  ColorRunButton knopf1, knopf2;
  JComboBox farbenWahl;
  JCheckBox buchstabenWahl;
  public LaufFrame() {
    // Swing-Komponenten erzeugen und einfügen
   .
   .
   .
   .
   .
   .
   .
   .
   .
   .
   .
    add(knopf1);
    add(knopf2);
    add(farbenWahl);
    add(buchstabenWahl);
  }

  class KnopfListener implements ActionListener {
    ColorRunButton crb;
    KnopfListener (ColorRunButton crb) {
      this.crb = crb;
    }
    public void actionPerformed (ActionEvent e) {
      // Auswahl-Index des ComboBox-Objekts bestimmen und Schrift setzen
     .
     .
     .
     .
     .
     .
      //  Zustand der Checkbox bestimmen und Button-Zustand ändern
     .
     .
    }
  }
}
```