Übung 8

Entwurfsmuster „Fabrik“ und „Einzelstück“

1. Entwerfen Sie nach dem **Factory**-Muster zwei Pizzerien („Pizza-Fabriken“), die die Bestellung einer Pizza unterschiedlich handhaben, z.B. **`PizzaFabrikItalienisch`** und **`PizzaFabrikAmerikanisch`**. Beide sollen Objekte vom Typ **`Pizza`** erzeugen (Sie können auch mehrere Varianten, z.B. **`SalamiPizza`**, **`Calzone`** usw. vorsehen), aber mit unterschiedlichen Saucen oder Teigvarianten (eine italienische Pizza hat üblicherweise einen dünneren Boden als eine amerikanische, dafür gibt es bei der amerikanischen Variante Käse im Rand).

2. Verwenden Sie das Entwurfsmuster **Singleton**, um einen „Ereignis-Protokollierer“ als Klasse **`Logger`** zu generieren, der

   (a) bei seiner Instanzierung eine Datei zum Schreiben öffnet,

   (b) Nachrichten von Objekten über eine Methode **`log(String nachricht)`** entgegennimmt und in die Datei schreibt.

   (c) Zum Testen können Sie das folgende Programm verwenden, das 100 Prozesse gleichzeitig startet, die parallel die **`log()`**-Methode aus einer Instanz von **`Logger`** aufrufen.

```
import java.awt.*;

public class Main {
 public static void main(String[] args) {
  Logger l = Logger.getInstance();

  System.out.println("main() gestartet.");
  l.log("main() gestartet.");

  for(int i=0; i<100; i++) new Process();

  System.out.println("main() endet.");
  l.log("main() endet.");
 }
}

// Unser Prozess, der x mal parallel gestartet wird.
class Process extends Thread {

 public void run() { // Prozess läuft los.
  // (Einzige) Logger-Instanz holen
  Logger l = Logger.getInstance();
```

```
    // Damit man auch was sieht: Jeder Prozess bekommt ein Fenster
    Frame frame = new Frame(this.toString()); // Ein Fenster
    frame.setSize(200,100); // Größe ändern (breite, höhe)
    frame.setLocation((int)(Math.random()*800),
                      (int)(Math.random()*600));
    frame.setVisible(true);

    // In Log-Datei schreiben (synchronized: Immer nur einer)
    l.log("Neuer Prozess " + this.toString() + " gestartet.");

    // "ungefähr" 10 Sekunden warten
    try { sleep((int) (10000.0 * Math.random())); }
    // Nichts tun, falls Unterbrechung
    catch(InterruptedException ie) {}

    frame.dispose(); // Fenster (für immer) schließen
    l.log("Prozess " + this.toString() + " beendet.");
  }

  public Process() {
    // start() aus Basisklasse Thread aufrufen, startet auch run()
    start();
  }
}
```

Warum ist für solche Aufgaben der Einsatz von **Singleton**s sinnvoll?