Übungen zur Vorlesung Programmieren 2
FHWS – Fakultät für Informatik und Wirtschaftsinformatik

Übungsblatt 20 Threads

Aufgabe 1:

a) Schreiben Sie ein Programm, welches einen zusätzlichen Thread startet. Der zusätzliche Thread soll alle 0,5 Sekunden „Yang“ ausgeben. Der Ausgangs-Thread soll alle 0,5 Sekunden „Yin“ ausgeben.
b) Programmieren Sie Ihren zusätzlichen Thread so, dass er im Konstruktor die Nachricht entgegennimmt, die er ausgeben soll!

Aufgabe 2: (Threads, Collections)

a) Modifizieren Sie die Klasse PrimzahlThread dahingehend, dass der PrimzahlThread im Konstruktor eine Liste der zu überprüfenden Zahlen entgegennimmt.
b) In der run-Methode soll der PrimzahlThread nun alle Zahlen der Liste daraufhin überprüfen, ob Sie eine Primzahl sind.
   Anstelle des **Strings** `ergebnis` soll der Thread eine Map `ergebnis` haben. Für jede untersuchte Primzahl soll ein Eintrag der folgenden Art in der Map ergänzt werden, bspw.:

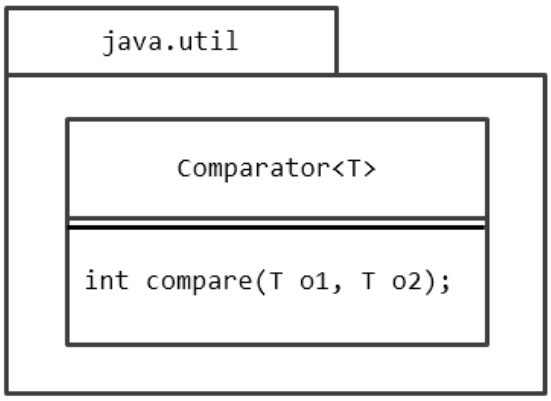
```
341 -> false
633910099 -> true
```

c) In der main-Methode sollen mehrere PrimzahlThread-Objekte gestartet werden. Dann soll der Haupt-Thread auf die Threads warten und die Ergebnisse der einzelnen Threads in einer einzigen Map zusammenlegen.
d) Die Mappings sollen wie in Aufgabe b) dargestellt ausgegeben werden.

Aufgabe 3: (Threads)

Erweitern Sie den TCPServer aus ÜB17 Aufgabe 2 so, dass er nach Annahme einer Verbindungsanfrage mit accept(), einen neuen Thread erstellt und diesem das von accept zurückgegebene Socketobjekt übergibt. Der neu erzeugte Thread soll dann die Verarbeitung der Anfrage vornehmen. Der „Hauptthread“ soll direkt auf weitere Anfragen warten.

Aufgabe 4: (Comparator, anonyme Klassen)
Ein TreeSet ist eine generische Klasse, die das Set-Interface implementiert. Das Set wird dabei als Baum abgespeichert. In dem Baum wird eine natürliche Ordnung sichergestellt, in der die Elemente (mit Hilfe der compareTo-Methode) geordnet eingefügt werden.
Soll eine andere Ordnung der Elemente hergestellt werden, ist dies durch die Implementierung des Comparator-Interfaces möglich.



Auszug aus dem Java-Doc:

„Compares its two arguments for order. Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second."

Untenstehende Klasse zeigt ein Beispiel für ein TreeSet, das die natürliche Ordnung verwendet.

```
public class Kreuzwortraetsel
{
  static String[] alleWoerter = {"Bienenschwarm", "Buch", "Bibel",
    "Beige", "Barriere", "Bein", "Beil", "Christ", "Christian",
    "Carmen"};

  public static void main(String[] args)
  {
    TreeSet<String> t1 = new TreeSet<>();
    t1.addAll(Arrays.asList(alleWoerter));
    for (String wort : t1)
      System.out.println(wort);
  }
}
```

Anmerkungen:
Mit `Arrays.asList(array)` lässt sich aus einem Array eine Liste erstellen.
Mit `t1.addAll()` lassen sich alle Elemente einer Collection in `t1` einfügen

a) Implementieren Sie einen Comparator, der es ermöglicht, die Wörter des Kreuzworträtsels zunächst der Länge nach und bei gleicher Länger nach dem Alphabet zu sortieren. Der Comparator soll einmal auf herkömmliche Weise und einmal als anonyme Klasse implementiert.
b) Was sind in diesem Beispiel die Vor- und Nachteile der Verwendung der anonymen Klasse?