Aufgabe 1

Beantworten Sie die Kontrollfragen A und B auf den Folien 21 und 40 des Inputs zu OOP9, falls diese noch nicht beantwortet wurden.

Siehe PRG2_OOP9_KF.docx / PRG2_OOP9_KF.pdf

Aufgabe 2

Zwei Threads sollen ein Point-Objekt verändern. Die Threads belegen x und y immer gleich, und immer dann, wenn sich die Koordinaten unterscheiden, soll es eine Meldung geben:

```
import java.awt.Point;
public class DefPoint implements Runnable
{
  private Point p;
  public DefPoint(Point p)
  {
    this.p = p;
  }
  public void run()
  {
    int x = (int) (Math.random() * 1000), y = x;
    while (true) {
      p.x = x;
      p.y = y;
      int xc = p.x;
      int yc = p.y;
      if (xc != yc) {
        System.out.println("Aha: x=" + xc + ", y=" + yc);
      }
    }
  }
  public static void main(String[] args)
  {
    Point p = new Point();
    new Thread(new DefPoint(p)).start();
    new Thread(new DefPoint(p)).start();
  }
}
```

Würden Belegung und Auslesen in einem Rutsch passieren, dürfte überhaupt keine unterschiedliche Belegung von x und y zu finden sein.

Fragen:

- Was stellen Sie fest?

  Aha: x=976, y=2

  Aha: x=976, y=2

  Aha: x=2, y=976

  Aha: x=976, y=2

  Aha: x=2, y=976

  x und y Werte stimmen nicht überein…

- Wie erklären Sie sich das Programmverhalten?

  Die zwei Threads überschreiben gegenseitig die Variablen

- Wie sieht die Korrektur aus?

```
int xc;
int yc;
synchronized (p) {
    p.x = x;
    p.y = y;
    xc = p.x;
    yc = p.y;
}
```

Aufgabe 3

Gegeben ist das folgende kleine Programm. Es misst die Zeit, die einige Methodenaufrufe der Klasse StringBuffer benötigen. Die Zeitmessung ist zwar nicht genau, gibt jedoch einen Eindruck, wie viele Millisekunden die for n Schlaufe benötigt.

import java.util.Date;

```
public class SynchTest1
{
  public static void main(String[] args) {
    StringBuffer sb = new StringBuffer();
    Date now = new Date();
    long sec = now.getTime();
    for (int n = 0; n < 1000; n++) {
      for (char ch = 33; ch < 128; ch++)
        sb.append(ch);
      for (int i = 0; i < sb.length(); i++)
        sb.charAt(i);
    }
    now = new Date();
    long tot = now.getTime()-sec;
    System.out.println("Fertig nach " + tot);
  }
}
```

Führen Sie das Programm ein paar Mal aus und merken Sie sich die "gemessenen" Zeiten.

~2'800

Anschliessend synchronisieren Sie die for n Schlaufe, wie folgt:

```
public class SynchTest2
{
  public static void main(String[] args) {
    StringBuffer sb = new StringBuffer();
    Date now = new Date();
    long sec = now.getTime();
    synchronized(sb) {
      for (int n = 0; n < 1000; n++) {
        for (char ch = 33; ch < 128; ch++)
          sb.append(ch);
        for (int i = 0; i < sb.length(); i++)
          sb.charAt(i);
      }
    }
    now = new Date();
    long tot = now.getTime()-sec;
    System.out.println("Fertig nach " + tot);
  }
}
```

Fragen:

- Was stellen Sie fest? Achtung: Sie sehen unterschiedliche Effekte, je nach Java Version 1.5 und früher oder 1.6.

  Dauer: ~10'200

  Es dauert viel länger….

  Version? Habe nur 1.6…

- Wie erklären Sie sich das Programmverhalten mit der Java Version 1.5 und früher?

  ???

- Wie erklären Sie sich das Programmverhalten mit der Java Version 1.6?

  ???