Bearbeitungszeitraum: eine Woche

Aufgabe	aI)	aII)	aIII)	aIV)	bI)	bII)	bIII)	bIV)	bV)	bVI)	bVII)
Zeit / Minuten											

a) Betrachten Sie das Beispiel der Verbraucher und Konsumenten aus der Vorlesung. Man kann sehen, dass das System (mit notifyAll) ohne Verklemmung ist, denn

- Am Anfang ist der Puffer leer und keine Threads sind im WAITING Pool des Puffers.
- Wenn ein Produzent den Puffer füllt verlassen alle Threads den WAITING Pool. Ein Konsument wird in die Lage versetzt weiter zu machen.
- Wenn ein Konsument den Puffer leert verlassen alle Threads den WAITING Pool. Ein Produzent wird in die Lage versetzt weiter zu machen.

Sagen Sie für jede der folgenden Änderungen ob die Verklemmungsfreiheit verloren geht oder nicht. Begründen Sie Ihre Antworten.

I) Drei Produzenten statt einem, und immer noch 2 Konsumenten.
II) Wie I) aber Verwendung von `notify()` statt `notifyAll()` in der `get()` Methode. Beachten Sie die Hinweise im Anhang zu diesem Aufgabenblatt. Insbesondere muss das dort beschriebene Zustandsdiagramm angefertigt werden.
III) Ein Produzent, zwei Konsumenten und Verwendung von `notify()` statt `notifyAll()` in der `put()` Methode. Beachten Sie die Hinweise im Anhang zu diesem Aufgabenblatt. Auch hier muss ein Zustandsdiagramm wie im Anhang beschrieben angefertigt werden.
IV) Vertauschen der Reihenfolge der Befehle `available = true` und `notifyAll()` in der `put()` Methode.

Hier nochmal zur Klarstallung, in welcher Aufgabe und welcher Methode notify bzw. notifyAll auftaucht:

	put	get	Zustandsdiagramm anfertigen?
I)	notifyAll()	notifyAll()	Nein
II)	notifyAll()	notify()	Ja
III)	notify()	notifyAll()	Ja
IV)	notifyAll()	notifyAll()	Nein

b) Das folgende Programm beschreibt zwei Benutzer Hänsel und Gretel, die denselben Drucker benutzen. Jeder der Benutzer druckt eine Datei mehrfach auf dem Drucker aus. Die Namen der zu druckenden Dateien werden der main-Methode der Klasse Test als Parameter übergeben. Im Folgenden bezeichnen wir das einmalige Drucken einer Datei als einen Druckjob. Hänsel soll die Datei haensel.txt mit den Zeilen
haensel 1
haensel 2
haensel 3
ausdrucken. Gretel soll die Datei gretel.txt mit den Zeilen
gretel 1
gretel 2
gretel 3
ausdrucken.

```
import java.io.*;
class Drucker {
   void druckeDatei(String dateiname) {
      try {
         BufferedReader in = new BufferedReader(
                              new FileReader(dateiname));
         String line = in.readLine();

         while (line != null) {
            // Zeile line auf dem Drucker ausgeben,
            // hier simuliert durch Bildschirmausgabe
            System.out.println(line);
            line = in.readLine();
         }
      } catch (Exception e) {
         System.out.println(
              "Eine Ausnahme ist aufgetreten.");
      }
   }
}

class Benutzer extends Thread {
   Drucker drucker;
   String dateiname;
   int anzahl;

   Benutzer(Drucker drucker, String dateiname, int anzahl) {
      this.drucker = drucker;
      this.dateiname = dateiname;
      this.anzahl = anzahl;
   }

   public void run() {
      for (int i=0; i<anzahl; i++) {
         drucker.druckeDatei(dateiname);
      }
   }
}
```

```
class Test {
   public static void main (String[] argv) {
      if (argv.length >= 2) {
         Drucker d = new Drucker();
         Benutzer haensel = new Benutzer(d, argv[0], 5);
         Benutzer gretel = new Benutzer(d, argv[1], 5);
         haensel.start();
         gretel.start();
      } else {
         System.out.println(
         "Bitte zwei Dateinamen als Argumente uebergeben!");
      }
   }
}
```

I) Anstatt Subklassen von der Klasse Thread zu bilden, kann man Threads in Java auch mit Hilfe der Schnittstelle Runnable implementieren. Formulieren Sie das obige Programm so um, dass diese Technik verwendet wird. Lassen Sie das veränderte Programm laufen, um die Lauffähigkeit zu testen.

II) Warum wurde in Java neben dem Erben von der Klasse Thread diese zweite Möglichkeit vorgesehen, Threads zu implementieren?

III) Bei dem angegebenen Programm kann es passieren, dass ein Druckjob von Hänsel mit einem Druckjob von Gretel vermischt auf dem Drucker ausgegeben wird. Beschreiben Sie einen Programmlauf, bei dem das passiert. Führen Sie den Fall anhand der Implementierung vor.

IV) Verändern Sie das Programm so, dass sich Druckjobs nicht mehr miteinander vermischen können. Ihre Lösung soll aber nicht so restriktiv sein, dass alle Druckjobs eines Benutzers als Einheit behandelt werden. Zum Beispiel soll es möglich sein, dass erst die Datei von Hänsel einmal gedruckt wird, dann die von Gretel, dann wieder die von Hänsel usw. Begründen Sie, warum Ihr Vorschlag das Problem löst. Demonstrieren Sie die Wirkung Ihres Vorschlags anhand der Implementierung.

V) Synchronisieren Sie die beiden Threads für Hänsel und Gretel nun so, dass zunächst alle Druckjobs des einen Benutzers gedruckt werden und erst im Anschluss daran die Druckjobs des anderen Benutzers. Begründen Sie, warum Ihr Vorschlag das Problem löst. Demonstrieren Sie die Wirkung Ihres Vorschlags anhand der Implementierung. Nutzen Sie die Methode Thread.yield(), die den Scheduler zwingt, auf einen anderen Thread umzuschalten.

Anhang:
Hinweise zu der Aufgabe a) II und a) III . Für beide Aufgaben muss ein entsprechendes Zustandsdiagramm angefertigt werden. Achten sie darauf, ob bei get bzw. put notify oder notifyAll benutzt wird. Dies hat eine große Auswirkung auf das Zustandsdiagramm.

Sie müssen den Beweis über die Verklemmung mit Hilfe eines Zustandsdiagramms führen. Ein Zustand des Verbraucher / Konsumenten Problems kann graphisch so dargestellt werden:

_ _ _ L _ _

Dabei bezeichnen die ersten drei Unterstriche jeweils Plätze für drei Produzenten im Waiting Pool. Die letzten beiden Unterstriche bezeichnen Plätze für die zwei Konsumenten im Waiting Pool. Wenn sich ein Produzent und ein Konsument im Waiting Pool befinden sieht der Zustand so aus:

P _ _ L K _

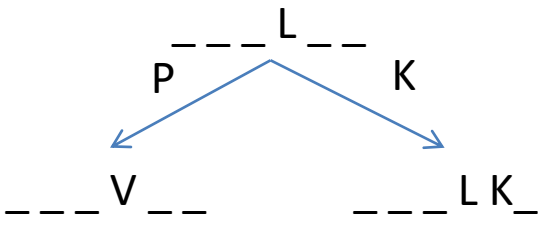
Der Buchstabe in der Mitte zeigt an, ob der Puffer leer (L) ist oder voll (V). Wenn ein Produzent erfolgreich die put Methode durchläuft könnte der resultierende Zustand z.B. so aussehen:

_ _ _ V_ _

Am Anfang befindet sich das System im Ausgangszustand:

_ _ _ L _ _

Kein Thread ist im Waiting Pool und der Puffer ist leer. Von diesem Zustand aus sind zwei Zustandsübergänge möglich: entweder ein Produzent (P) bekommt die Ausführungsberechtigung oder ein Konsument (K). Den Zustandsübergang können wir durch einen Strich zu dem neuen Zustand darstellen. Der Strich wird mit P oder K bezeichnet, je nachdem, ob der neue Zustand durch einen Produzenten oder Konsumenten erzeugt wird



Zeichnen Sie nun das vollständig Zustandsdiagramm, indem Sie alle Pfade verfolgen und die entstehenden Zustände aufzeichnen. Dieses Diagramm wird nicht riesengroß, da sich die Zustände wiederholen, d.h. die Pfeile enden auf bereits vorhandenen Zuständen.
Wann besteht eine Verklemmungsmöglichkeit? Dann wenn einer der folgenden Zustände im Zustandsdiagramm auftritt:

P P P V K K oder P P P L K K

Wenn keiner dieser Zustände auftritt, ist das Problem verklemmungsfrei.