Klausur im Fach Softwareentwicklung mit Java, SS08

Prüfer: Prof. Dr.-Ing. Th. Greiner

Name:

Matrikelnummer:

Semester:

Hilfsmittel: Darstellung der Sprachsyntax ohne Diagramme

Geben Sie auf allen Blättern Matrikelnummer und Name an.

1. **Aufgabe** (2 Punkte)

Erklären Sie die folgenden Begriffe:

a) Aggregation

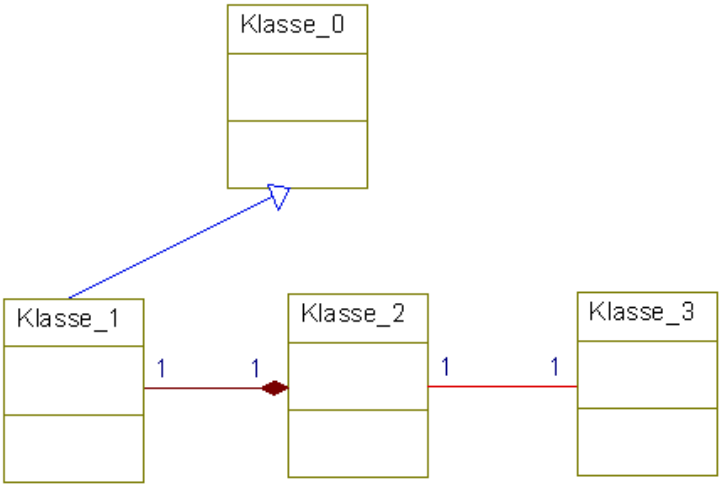
b) ArithmeticException

c) Deadlock

d) Mutex

2. **Aufgabe** (6 Punkte)

Es ist folgendes UML-Klassendiagramm gegeben:



a) Beschreiben Sie den Zusammenhang der Klassen.

b) Erstellen Sie die Java-Klassen, sodass dieser Zusammenhang wiedergegeben wird.

3. **Aufgabe** (10 Punkte)

Schreiben Sie ein Java-Programm, das 3 Threads nutzt, um auf einer gemeinsamen Integer-Variablen x zu arbeiten. Der erste Thread erhöht die Variable um 2, der zweite Thread erniedrigt die Variable um 1, der dritte Thread gibt den Wert der Variablen aus. Ist der Zählerwert 100 erreicht, werden die Threads beendet.

a) Welche grundsätzliche Schwierigkeit kann beim Verändern der Variable auftreten?

b) Erläutern Sie, ob der dritte Thread alle Veränderungen von x ausgibt. Begründen Sie Ihre Antwort.

c) Erstellen Sie das Java-Programm unter Verwendung der Klasse „Thread“, sodass sichergestellt ist, dass der dritte Thread alle Veränderungen mitbekommt.

4. Aufgabe (8 Punkte)

Es soll ein Java-Programm erstellt werden, das eine Klasse mit dem Namen „Aufgabe 4“ mit den Attributen a (integer), b (double) und c (String) realisiert.
Instanzen dieser Klasse sollen mit einer Methode „speichern()“ in einer Array-Liste abgelegt werden und mit einer Methode „lesen()“ wieder eingelesen werden können.

5. Aufgabe (5 Punkte)

Erläutern Sie kurz jede Zeile des nachfolgenden Programmtextes:

```
public class Beispiel
{
   private int wert;

   public Beispiel(int anfang)
   {
        if(anfang < 0)
             anfang = 0;
        wert = anfang;
   }

   public synchronized void k()
   {
        while(wert <= 0)
        {
             try
             {wait();}
             catch(InterruptedException e)
             {}
        }
        wert--;
   }

   public synchronized void l()
   {
        wert++;
        notify();
   }
}
```