Visibility 1

Gegeben ist der folgende Code.

```
public class JMM1 {
    private static int value = 0;
    private static boolean ready = false;

    public static void main(String[] args) {
        new Thread("T1") {
            public void run() { value = 77; ready = true; }
        }.start();

        new Thread("T2") {
            public void run() {
                if (ready) { System.out.println(value); }
            }
        }.start();

    }
}
```

Der Thread T1 setzt die Variablen `value` und `ready` und Thread T2 gibt den Wert von `value` aus falls dieser bereits initialisiert (`ready`) ist.

Fragen:

- Welche Werte können von Thread T2 ausgegeben werden?
  Geben Sie alle möglichen Werte an.

- Welche Werte können von Thread T2 ausgegeben werden, falls die Variable `ready` als volatile deklariert wird?

  ```
  private static volatile boolean ready = false;
  ```

  Versuchen Sie ihre Aussage zu beweisen.

Visibility 2

Gegeben ist der folgende Code. Beachten Sie dabei, dass nur die Variable p als volatile deklariert ist.

```
public class JMM2 {
    private static class Person {
        private String name;
        public String getName() { return name; }
        public void setName(String name) { this.name = name; }
    }

    private static volatile Person p = null;

    public static void main(String[] args) {
        new Thread("T1") {
            public void run() {
                p = new Person();
                p.setName("Meier");
            }
        }.start();

        new Thread("T2") {
            public void run() {
                if (p != null) { System.out.println(p.getName()); }
            }
        }.start();
    }
}
```

Da die Variable p als volatile deklariert ist, sind die Änderung, die Thread T1 auf p ausführt, in Thread T2 sichtbar.

Fragen:

- Welche Werte können von Thread T2 ausgegeben werden?
  Geben Sie alle möglichen Werte an.

- Wie muss das Programm geändert werden damit sichergestellt ist, dass Thread T2 garantiert den Namen „Meier“ ausgibt falls Thread T1 die Variable p vor dem Lesen von T2 bereits gesetzt hat?

Visibility 3

In diesem Beispiel betrachten wir eine einfache Stack-Implementierung:

```
class Stack {
   private Object[] data = new Object[10];
   private int top = 0; // Points to the next free slot

   public synchronized void push(int x) {
      data[top++] = x;
   }

   public synchronized Object pop() {
      Object res = data[--top];
      data[top] = null;
      return res;
   }

   public int size() {
      return top;
   }
}
```

Die beiden Methoden `push()` und `pop()` sind synchronisiert, da sie beide die gemeinsame Variable `top` verändern. Wenn die beiden Methoden nicht synchronisiert wären, dann wäre der Stack bald inkonsistent (wie wir das in der zweiten Unterrichtswoche gesehen haben).

In dieser Aufgabe betrachten wir jedoch die Methode `size()`. Diese greift ebenfalls auf die Variable `top` zu. Muss diese Methode ebenfalls synchronisiert werden?

Fragen:

- Überlegen Sie sich, ob eine Synchronisation für die Methode `size()` zwingend nötig ist. Welche Garantien erfüllt Java, falls Sie die Methode `size` als `synchronized` deklarieren?

- Ist die Methode, so wie sie in obigem Beispiel formuliert ist, korrekt? Falls nein, wie könnte der Code korrigiert werden, ohne dass die Methode `size` als `synchronized` deklariert wird.