Kontrollfragen A

1. Zu welcher Klasse gehört die Methode wait()?
   Object

2. Nennen Sie einen wichtigen Unterschied zw. Thread.sleep() und wait().
   sleep(): der Thread ist die ganze Zeit „running“
   wait(): ist „idle“, d.h. der Thread wartet und rechnet nichts

3. Was gibt es beim notify() bzw. notifyAll() zu beachten?
   notify() resp. notifyAll() wird nicht gespeichert, d.h. später kommende Threads die auf ein notify() warten werden nicht „released“.
   Die Benachrichtigung der notify/notifyAll-Methoden wird nicht gespeichert, wenn der wait-Pool leer ist, d.h. wenn Threads nachher wait aufrufen, bleiben sie im wait-Pool. ➔ Deadlock Gefahr.

4. Warum dürfen wait() und notify() bzw. notifyAll() nur in einem geschützten Bereich, d.h. nur innerhalb eines synchronized-Blocks, aufgerufen werden?
   Der Zustand kann sich in der Zwischenzeit bereits wieder geändert haben, z.B. in einer while-Schlaufe.
   Bei einem synchronized Block wird der Lock freigegeben und somit kann sich der Zustand in dieser Zeit nicht mehr ändern.

5. Wie fair ist der Object Wait-Pool, bzw. in welcher Reihenfolge kommen die Threads aus dem Wait-Pool?
   Nicht fair! (unconditional fair)

Kontrollfragen B

1. Beschreiben Sie kurz das Singleton Entwurfsmuster in Ihren eigenen Worten.
   Ein Singleton stellt sicher, dass nur eine Instanz einer Klasse erzeugt wird und stellt einen globale Zugriffsmöglichkeit zu dieser Instanz zur Verfügung.

2. Wieso hat eine Singleton Klasse nur private Konstruktoren?
   Damit von „extern“ kein Objekt erzeugt werden kann.

3. Wieso muss eine Singleton Klasse final sein?
   Damit keine Unterklassen mit einem „public“ Konstruktor erstellt werden können (welche dann wiederum ein Objekt der Oberklasse erstellen könnten)