**Nachdenkzettel Logging**

**Aufgabe 1:**

a)

* Wo wird geloggt?
  + Auf der Konsole
  + In dem File A1.log
* Wieviel wird geloggt?
  + Beide Loggen auf dem Level „Debug“
* Was wird geloggt?
  + Alles, was auf dem Level Debug ist, wird in das File A1.log und die Konsole geloggt.
  + Dabei wird in das File A1.log nur eine bestimme Klasse mit dem Name „VehicleManager“ geloggt, was man an dem Befehl "se2examples.core.businessLogic.VehicleManager" sehen kann.
* Wie wird geloggt?
  + Es werden zwei unterschiedliche Pattern benutzt, wie geloggt wird.
  + Logging für die Konsole:
    - pattern="%t %-5p %c{2} - %m%n"
  + Logging für das File:
    - pattern="%d %-5p [%t] %C{2} (%F:%L) - %m%n"

b)

Ja, das ist ebenfalls möglich, man müsste dies im Logger mit angeben, zum Beispiel mit:

name=“se2examples.core.businessLogic“

**Aufgabe 2:**

1. Error:
   * Um schwerwiegende Fehler auszugeben, welche kritisch für die Anwendung sind
   * Beispielsweise für die SQLException in Java, wenn ein Datenbankverbindungsfehler vorliegt
2. Info:
   * Um wichtige Informationen zum aktuellen Stand und dem Fortschritt der Anwendung auszugeben
   * Evtl wichtige Variablen und Ergebnisse ausgeben
3. Debug:
   * Detaillierte Informationen während der Entwicklung ausgeben
   * Protokollierung
   * Hauptsächlich bei der Fehlersuche zu verwenden

**Aufgabe 3:**

Ja, es könnte die Anwendung langsamer machen, da das Schreiben der Lognachrichten ebenfalls zusätzliche Ressourcen in Anspruch nehmen kann.

Wie sich ein Loggingframework optimalerweise zu verhalten haben sollte:

* + Kaum Auswirkungen auf die Leistung nehmen -> gut optimiert
  + Asynchrones Logging -> Logging wird in einem Hintergrundprozess/ Thread verarbeitet
  + Einfaches Formatieren der Lognachrichten
  + Ein gutes Ressourcenmanagement, wie das korrekte Freigeben von Speicher, sodass die Anwendung nicht langsamer oder instabil wird.