Übung 5: Verwendung von Java-Threads

Ziel der Übung:

Diese Übung dient dazu, den Umgang mit Threads in der Programmiersprache Java kennenzulernen. Ein einfaches Java-Programm, das Threads nutzt, soll zum Ablauf gebracht werden. Außerdem soll die Benutzung der Java API kurz erläutert werden.

Java API:

API steht für „*Application Programming Interface*". Die Java-API beschreibt viele bereits vorhandene Java-Klassen. Die API ist in einem eigenen Dokument beschrieben. Dieses Dokument findet man im Web[1].

Aufgabenstellung – Teil 1:

1. Suchen Sie in der Java API nach der Klasse Thread. In welchem Package befindet sich die Klasse Thread?
2. Über welche zwei Arten kann man einen Thread erzeugen?
3. Finden Sie in der API der Thread-Klasse die Methoden start(), run() und sleep(). Lesen Sie sich deren Beschreibung durch und versuchen Sie diese zu verstehen.

Datei-, Project- bzw. Sourcen-Importierung unter Eclipse:

Um Source-Files unter Eclipse einzubinden (in den Workspace), gibt es grundsätzlich zwei Möglichkeiten:

1. File → Import → General → Existing Project into Workspace

   Diese Variante ist zu wählen, wenn man ein Java-Projekt bereits zur Verfügung hat. Dafür ist ein Verzeichnis inklusive einer .project-Datei notwendig. In diesem Fall werden sämtliche Packages usw. korrekt eingebunden und verfügbar gemacht.

2. File → Import → General → File System

   Verfügt man nur über die Source-Files ohne Projektstruktur, bietet sich der Import über das Filesystem an. Hier muss zum Verzeichnis der Source-Files navigiert und die zu importierenden Files müssen markiert werden.

Vorsicht: Beim Importieren eines bereits existierenden Projekts wird von Eclipse lediglich eine Referenz auf die Source-Files generiert, es existiert also weiterhin nur eine Version des Programms auf dem Dateisystem des Rechners. Ist eine physikalische Kopie des Projekts in den aktuellen Workspace gewünscht, muss in dem Schritt *Import* die Option c*opy projects into workspace* angewählt werden.

Hinweis: Ein eher pragmatischer Ansatz wäre, das Eclipse-Projekt manuell per Hand zu erstellen (File → New → Java Project) und die Dateien danach einfach per Drag&Drop in das Eclipse-Fenster zu ziehen.

[1] Die aktuelle Java API finden Sie bei Google mit dem Suchbegriff „*Java API*".

Aufgabenstellung – Teil 2:

Importieren Sie zunächst die zur Übung mitgelieferten Source-Files über die zuvor vorgestellte Eclipse-Import-Funktion. Es sollten zwei Java-Klassen in ein Projekt mit dem Namen *Wi-Inf_Übung 6* importiert worden sein.

1. Versuchen Sie zunächst die Java-Klassen zu verstehen und überprüfen Sie, ob sie richtig übersetzt werden.

   Was macht das Programm?

2. Bringen Sie das Programm zum Ablauf. Hierzu müssen Sie den Eclipse-Menüpunkt „Run“ benutzen.

   Prüfen Sie die Ausgaben des Programms auf die Konsole!

   Was wird ausgegeben?

3. Starten Sie das Programm erneut und betrachten Sie parallel dazu den Windows-Taskmanager. *Hinweis*: Mithilfe des Process Explorers (siehe Übung 2) sind die Informationen deutlicher und besser zu erkennen.

   Suchen Sie im Taskmanager den Java-Prozess, in dem das Programm abläuft (Name = *javaw.exe*) und stellen Sie die Anzahl der Threads während des Ablaufs fest. Hierzu müssen Sie den Taskmanager entsprechend einstellen (Spalte *Threadanzahl* ergänzen über *Ansicht->Spalten auswählen*).

   Wie viele Threads benutzt der Prozess und wie verändert sich die Anzahl der Threads zur Laufzeit?

   Wie viel CPU-Zeit und wie viel Hauptspeicher verbraucht das Programm?

4. Ersetzen Sie in der Klasse *myThreadTest* den Threadaufruf *t1.start()* durch *t1.run()*.

   Vergleichen Sie nun die Ausgabe beider Versionen bzw. betrachten Sie speziell die Threadanzahl im Taskmanager. Was ist passiert bzw. hat sich verändert?

   Warum ist das so? (Hinweis: Sind die Aufrufe nun nebenläufig (parallel) oder sequentiell?)

5. Versuchen Sie nun das Programm so zu ändern, dass die Threads über das Runnable-Interface implementiert werden.

   (*Hinweis*: In diesem Fall müssen einige statische Methoden der Klasse Thread verwendet werden. Interessant könnte folgende Methode sein: `Thread.currentThread()`. Auch den Konstruktor `Thread(Runnable target)` könnte man sich hierzu ansehen. Sehen Sie in der Java API nach, was diese Methode zurückliefert.)

6. Wie Sie nun gesehen haben, gibt es zwei Möglichkeiten Threads in Java zu erzeugen. Einmal über die Vererbungstruktur und ein anderes Mal über die Implementierung des Interfaces Runnable. Überlegen Sie, was der Vorteil der Interface-Variante sein könnte.