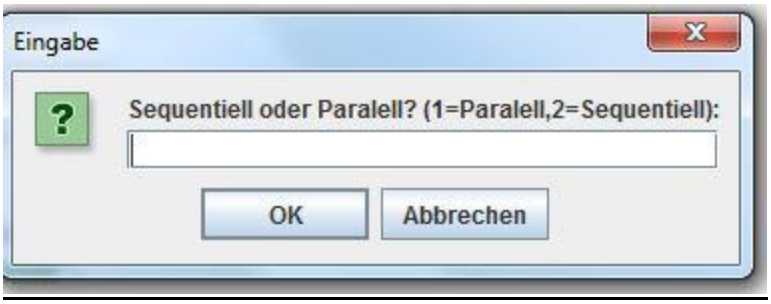
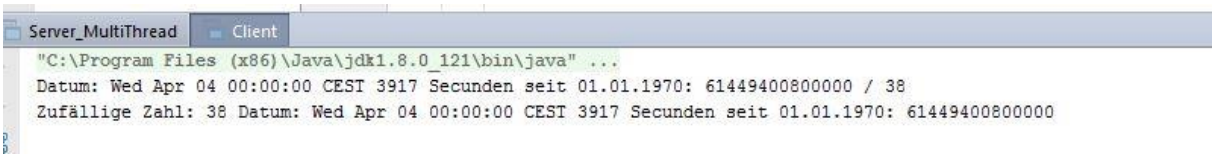
Arbeitsauftrag 6:

- <u>Aufgabenstellung:</u>
  - Realisiere eine Client/Server Anwendung in Java. Der Server stellt zwei Funktionen (getRandom() und getTime()) zur verfügen, die eine lange Berechnung simulieren sollen. Für diese Aufgabe sollten zwei verschiedene Serverversionen erstellet werden. Eine sequenzielle Version, die nur einen Thread verwendet und somit die Clients nacheinander abarbeitet und eine parallele Version die mehrere Threads verwendet und die Clients parallel abarbeitet. Teste beide Server und dokumentiere deine Beobachtungen. Welche Einschränkungen und welches Leistungsvermögen haben die Server?

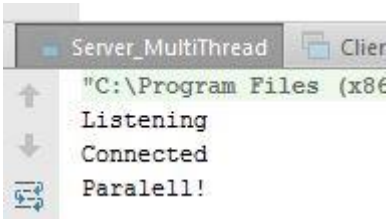
- <u>Funktionsweise:</u>
  - Man startet den Server dann den Client
  - Man wird aufgefordert Parallel oder Sequentiell auszuwählen



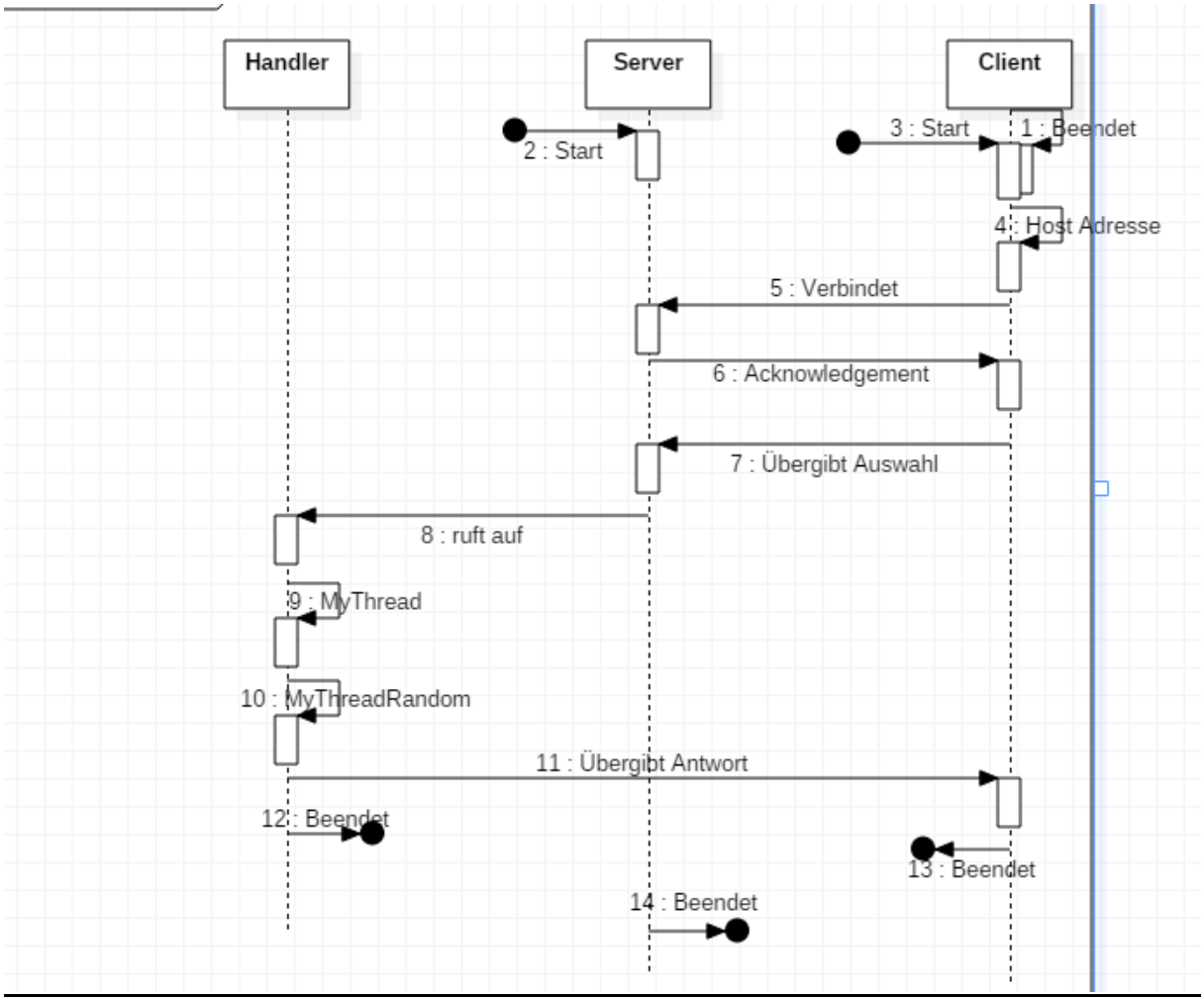
  - Server startet Handler
  - Handler führt dann je nach Auswahl getRandom() und getTime() parallel oder sequentiell aus
  - Handler übergibt dann die Daten dem Client
  - Dieser gibt sie dann auf der Konsole aus



  - Logging seitens des Servers



Zustandsdiagramm:



Anmerkung:

Als Protokoll wird die Klasse ServerMsg und die Klasse ClientMsg benutzt. Beide erben von der abstrakten Klasse Message die einen String beinhaltet der jeweils übergeben wird. In diesem String befinden sich jeweils die zu übergebenden Daten, z.B GetRandom und GetTime.

Antworten:

Die parallele Durchführung der 2 Threads ist schneller da der 2 Thread nicht auf dem Abschluss des Ersten warten muss und somit gleich starten kann. Würde es Sequentiell ablaufen müsste der zweite Thread auf die Beendigung des ersten Threads Warten und somit gäbe es eine deutliche Verzögerung. Das Ausführen paralleler Threads ist nur so lang möglich bis sie voneinander unabhängig sind. Sobald sie voneinander abhängen ist die sequentielle Abarbeitung naheliegender bzw. es ist auch möglich gewisse teile der Threads parallel laufen zu lassen und diese dann mit wait() und notify() zu koordinieren.