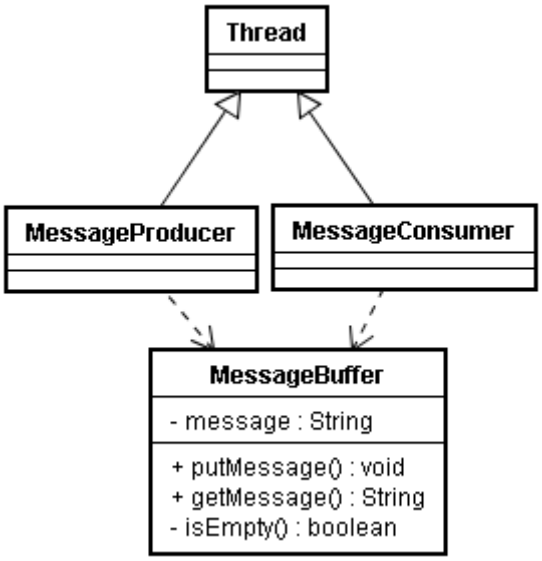
Übung 7: Producer/Consumer-Problem in Java

Ziel der Übung:

In dieser Übung soll anhand eines selbst zu entwickelnden Programms das Verständnis für Threads und für die Synchonisation in Java vertieft werden.

Aufgabenstellung:

Bei der Übung soll ein Programm entwickelt werden, welches das Producer-Consumer-Problem abbildet. Bei dem Programm wird das Prinzip eines Nachrichtenpuffers umgesetzt. Das bedeutet, ein Produzent stellt eine Nachricht (vom Typ `java.lang.String`) in einen Puffer ein. Anschließend wird ein Konsument aktiviert, der die Nachricht abholt. In nachfolgender Abbildung ist das zugehörige Klassenmodell dargestellt.



Im beiliegenden Eclipse-Projekt *Uebung7_Angabe* hat sich bereits ein Programmierer an der Lösung versucht, konnte allerdings die Klasse `MessageBuffer` mangels Zeit nicht mehr fertig stellen. Ausgeführt wird das Programm über die Klasse `RunMessageBufferDemo`.

1. Überarbeiten Sie die unfertige Klasse dahingehend, dass die Methodenaufrufe für die Put- und Get-Operation synchronisiert werden und Producer und Consumer abwechselnd aktiviert werden. Möglicherweise ist dieses Verhalten über die Nutzung der Methoden `wait()` und `notify()` realisierbar.

2. Überlegen Sie sich nun anhand des Quellcodes wie die Ausgaben auf der Konsole aussehen müssten und betrachten Sie anschließend die tatsächliche Ausgabe! Was fällt Ihnen auf und warum ist das so?

3. Versuchen Sie die Anwendung so zu modifizieren, dass die Ausgabe mit dem tatsächlichen Programmablauf übereinstimmt.

   (*Hinweis*: Die Ausgabeanweisung aus den Thread-Klassen müssen in den `MessageBuffer` verschoben und anschließend angepasst werden)

4. Passen Sie das Programm so an, dass es möglich ist, die Anzahl der Producer und Consumer jeweils in einer Variable festzulegen.

   (*Hinweis*: Es werden Thread-Arrays und einige Schleifen benötigt – siehe Übung 6)

   Überprüfen Sie nun die Korrektheit Ihrer Lösung, indem Sie die Anzahl der Producer und Consumer zunächst auf 1 setzen. (Der Programmablauf müsste identisch zu jenem aus Aufgabe 3 sein.)

5. Erhöhen Sie nun die Anzahl der Producer auf 2 und bringen Sie das Programm erneut zum Ablauf. Was passiert und warum ist das so?

   (*Hinweis:* Lesen Sie sich dabei in der API-Beschreibung der Klasse `Object` die Dokumentation zu den Methoden `notify()` und `notifyAll()` durch!)

   Versuchen Sie nun, mit Ihrem neuen Wissen, das Programm zu korrigieren und bringen Sie es anschließend erneut zum Ablauf (Producer = 2, Consumer = 1).

   Wie hat sich die Ausgabe im Vergleich zu Aufgabe 4 verändert?

   Sollte der Ablauf immer noch fehlerhaft sein, versuchen Sie auszurechnen, wieviele Nachrichten ingesamt in den Puffer eingestellt und wieviele abgeholt werden (Hinweis: loops * Anzahl Threads). Was fällt Ihnen dabei auf, und warum kann das Programm so gar nicht funktionieren? Versuchen Sie diesen Fehler zu korrigieren.

6. Warum ist in den Methoden `putMessage()` und `getMessage()` jeweils eine while-Schleife implementiert?