Aufgabe 16.4: Stack-Zugriff mit mehreren Threads

a) Es soll ein anonymes Paket aus den Dateien

- `Test.java`,
- `Reader.java`,
- `Writer.java`
- und `Stack.java`

erstellt werden.

Die Aufgabe dieser Klassen ist:

- `Writer.java`

  Die Datei `Writer.java` enthält eine Klasse `Writer`, die von der Klasse `Thread` ableitet. Der `Writer`-Thread schreibt insgesamt 100 `int`-Zahlen auf den Stack.

- `Reader.java`

  Die Datei `Reader.java` enthält eine Klasse `Reader`, die ebenfalls von der Klasse `Thread` ableitet. Der `Reader`-Thread liest die `int`-Zahlen vom Stack ein und gibt sie auf der Konsole aus.

- `Test.java`

  Die Datei `Test.java` enthält eine Klasse `Test`, die nur eine Methode `main()` zum Testen der anderen Klassen einhüllt. In der Methode `main()` soll ein Objekt der Klasse `Stack` erzeugt werden. Eine Referenz auf dieses Objekt soll an die zu erzeugenden Instanzen der Klassen `Writer` und `Reader` übergeben werden. Starten Sie daraufhin die beiden Threads `Writer` und `Reader`. Warten Sie, bis die beiden Threads ihre Aufgabe erledigt haben und beenden Sie danach die Anwendung. Die Threads sollen willkürlich auf den Stack zugreifen. Ist der Stack beim Schreiben voll, so soll der `Writer`-Thread warten. Der `Reader`-Thread liest die Zahlen einzeln vom Stack. Ist der Stack leer, so soll er warten, bis er erneut Werte lesen kann. Dieser Ablauf soll solange fortgesetzt werden, bis alle 100 Zahlen vom `Writer`-Thread auf den Stack geschrieben wurden und vom `Reader`-Thread ausgelesen wurden. Machen Sie den Stack nicht zu groß, damit die Zahlen vom `Writer`-Thread nicht auf einmal in den Stack geschrieben werden können.

- `Stack.java`

  Zum Schreiben auf den Stack dient die Methode `push()` und zum Lesen vom Stack dient die Methode `pop()` der Klasse `Stack`. Synchronisieren Sie die Methoden des Stacks, sodass es zu keinen Synchronisationsproblemen kommt.

b) Ersetzen Sie Ihre Implementierung des Stacks mit der von Java mitgelieferten Klasse `LinkedBlockingDeque` aus dem Paket `java.util.concurrent`. Passen Sie Ihre Klassen so an, dass diese Klasse anstelle der Klasse `Stack` benutzt wird. Nutzen Sie dabei die Möglichkeiten der generischen Typisierung (Generics) und programmieren Sie – wo möglich – gegen die Schnittstelle `BlockingDeque`.

Um einen blockierenden Effekt zu erzielen, darf man die Methoden `push()` und `pop()` der Schnittstelle `BlockingQueue` nicht verwenden. Verwenden Sie stattdessen die Methoden `putFirst()` statt `push()` und `takeFirst()` statt `pop()`.