1. Es soll ein kurzes Testprogramm erstellt werden, welches die *Zustände* in denen ein Thread sein kann analysiert und ausgibt. Schreiben Sie dazu zuerst folgende Thread-Klassen mit den angegebenen Aufgaben:

`MyThread`

Der Thread soll abwechselnd eine *Sekunde lang schlafen* und danach eine beliebige, lange Berechnung durchführen (z.B. eine `while`-Schleife eine Milliarde mal durchlaufen). Dieser Thread soll von außen abgebrochen werden können (**HINWEIS:** `interrupt()`).

`MyThreadStateAnalyser`

Dieser Thread soll im *Halbsekundentakt* den *Status* von `MyThread` ausgeben. Sollte dieser Thread erkennen, dass `MyThread` fertig abgearbeitet wurde, dann soll er auch seine Arbeit beenden.

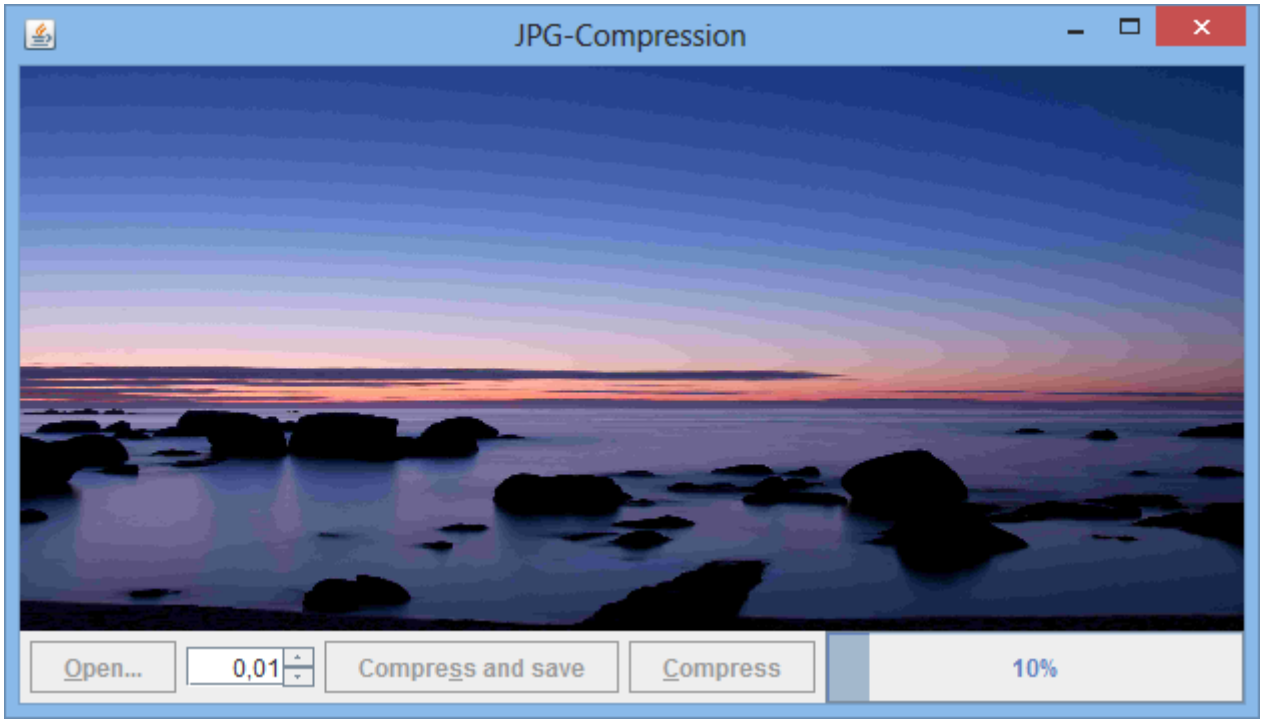
```
NEW
TIMED_WAITING
TIMED_WAITING
RUNNABLE
RUNNABLE
TIMED_WAITING
RUNNABLE
RUNNABLE
TIMED_WAITING
RUNNABLE
TERMINATED
```

`MyThreadTerminator`

Dieser Thread hat lediglich die Aufgabe *nach fünf Sekunden* den Thread `MyThread` zu beenden. Hat er diese Aufgabe erledigt, dann soll er sich auch beenden.

Schreiben Sie dann das *Hauptprogramm*, welches die obigen Threads anlegt, zuerst `MyThreadStateAnalyser` und `MyThreadTerminator` startet, eine Sekunde lang wartet, dann `MyThread` startet und dann auf das Ende dieser Threads wartet.

2. Schreiben Sie ein Programm, welches *JPG-Bilder komprimieren* kann. Es soll möglich sein, ein Bild zu *öffnen*, das dann angezeigt wird (**HINWEIS:** Verwenden Sie zum Anzeigen des Bildes die bereitgestellte Klasse `ImageComponent`):



Die *Granularität* des Komprimiervorgangs – ein Wert zwischen 0.01 und 0.1 – kann gewählt werden. Für das Komprimieren wird die Klasse `JPGImageCompress` bereit gestellt, welche die Komprimierung vornehmen kann. Beim Komprimieren muss die *Komprimierqualität* – ein Wert zwischen 0.0 und 1.0 – gewählt werden. Wird beispielsweise die Granularität auf 0.1 eingestellt, so sollen *elf Bilder* mit der Qualität 0.0, 0.1, 0.2, …, 1.0 komprimiert werden.

Compress and save

Wird auf den Knopf **Compress and save** geklickt, so soll für jede Komprimierqualität ein *eigener Thread* gestartet werden, der den Komprimiervorgang parallel zu allen anderen Threads durchführt und das komprimierte Bild in eine Datei im Ordner des Originalbildes ablegt. Wird beispielsweise die Ganularität auf 0.1 eingestellt, sollen elf Komprimierthreads parallel das Komprimieren und Abspeichern durchführen. Während des Komprimiervorganges müssen alle Knöpfe im Fenster *deaktiviert* werden.

Gleichzeitig mit den Komprimierthreads soll ein *weiterer Thread* gestartet werden, der im *Halbsekundentakt* kontrolliert, wie viele Threads die Komprimierung bereits beendet haben. Ausgehend davon soll er die *Fortschrittsanzeige aktualisieren*. Diese soll anzeigen, wieviel Prozent der gestarteten Komprimierthreads ihre Arbeit beendet haben.

Compress

Wird auf den Knopf **Compress** geklickt, so wird ebenfalls der Komprimiervorgang eingeleitet, die komprimierten Bilder werden aber *nicht abgespeichert*, sondern nacheinander *am Bildschirm angezeigt*, so dass beobachtet werden kann, wie sich die Komprimierqualität auf das Bild auswirkt.

Beachten Sie dabei, dass zum Unterschied zu vorher nicht alle Komprimierthreads gleichzeitig gestartet werden können, weil insbesondere bei größen Bildern nur wenige komprimierte Bilder im Speicher abgelegt werden können und ansonsten leicht ein *Heap-Überlauf* stattfinden kann.

Deshalb sollen Sie den Komprimiervorgang so programmieren, dass nur einige wenige Threads – mindestens zwei – parallel gestartet werden, die jene Bilder komprimieren, die zum Anzeigen der nächsten Bildsequenzen benötigt werden. Wichtig ist dabei, dass diese Threads die Bilder komprimieren, während dem der *Hauptthread* sich um die *Bildanzeige* kümmert. So kann die Anzeige möglicht gleichmäßig erfolgen. Der Hauptthread sollte nach Anzeige jedes Bildes etwas warten, bis er das nächste komprimierte Bild anzeigt, um ein gleichmäßig ablaufende Darstellung der Bilder zu gewährleisten.

Auch soll wiederum die *Fortschrittsanzeige* aktualisiert werden. Sie soll jetzt anzeigen, wie viel Prozent der komprimierten Bilder angezeigt wurden (**HINWEIS:** Dazu ist es nicht mehr nötig, einen eigenen Thread zu programmieren).