Aufgabe 7:

Beschreibung der Aufgabe:
Es war geforderte eine rechenintensive mathematische Operation (z.B. Check auf Primzahl) in multiple Threads auszulagern. Hierbei sollten Java Future Objects genutzt werden.

Beschreibung der Lösung:
ExecuteablePrimFinder implementiert ein Callable und beinhaltet die Logik zum Check einer Primzahl in einem bestimmten Bereich. Z.B., Checke ob 80 durch eine Zahl im Bereich von 2 – 20 teilbar ist ….

Die statische Methode isPrim der Klasse prim, erstellt einen Executorservice mit multiplen ExecuteablePrimFinder und führt in jedem FutureObejct die Methode des ExecuteablePrimFinder aus, um festzustellen ob eine Primzahl in einem bestimmten Bereich einen Teiler hat.
Anschließend wird geprüft, ob alle Ergebnisse verfügbar sind, diese werden zusammengeführt und anschließend wird das Gesamtergebnis ausgeben.

Verständnisfragen:

1. Threads haben keinen Rückgabewert, Futures schon
2. Man kann auf die Variable nur via Get zugreifen. Get ist blockierend, im Falle das der Task noch ausgeführt wird. Man kann also den Zustand der Variable während der Laufzeit nicht abfragen. Wahrscheinlich null.
3. Futures haben ein Standard Interface und sind leichter zu benutzen (get, cancel)
4. Eine Klasse die das Ausführen von Futures managet. So kann man einfach und komfortabel einen Pool von Threads / Tasks verwalten.
5. Callable hat im Gegensatz zu Runnable einen Rückgabewert. Eine Implementierung des Callables macht dann Sinn, wenn man für die Ausführung seines Threads einen Rückgabewert benötigt.