1. Was passiert wenn ein Thread *nicht* durch die Methode `start()` gestartet wird, sondern anstelle dieser Methode direkt die Methode `run()` aufgerufen wird?

2. Betrachten Sie die nebenstehende *Klassendeklaration* bei welcher die Methoden `run()` und `start()` *überschrieben* werden. Wieso wird ein Objekt dieser Klasse nicht wie ein Thread ausgeführt? Begründen Sie Ihre Entscheidung…

```
public class ThreadTest extends Thread
{
   @Override
   public void run() {
      // Meine Anweisungsfolge, die
      // parallel ausgeführt werden soll
      …
   }
   @Override
   public void start() {
      // Methode wird überschrieben
      run();
   }
}
```

3. Die mitgelieferten Klassen `PrimeFactorTool` und `PrimeFacotToolMain` ermitteln von den Zahlen im Intervall *[10000, 10250]* nacheinander die *Primfaktorzerlegungen* und geben sie am Bildschirm aus.

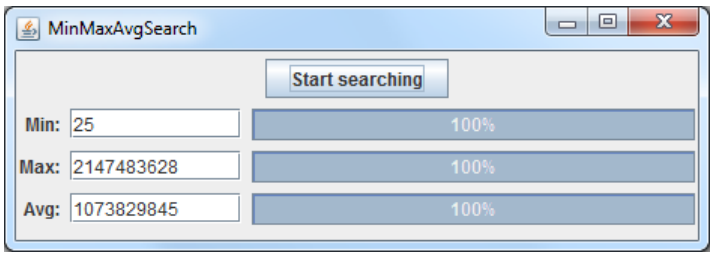
   Sie sollen nun diese Ermittlung *nebenläufig programmieren*, so dass gleichzeitig für mehrere Zahlen ihre Primfaktorzerlegung ermittelt wird und damit die Ermittlung beschleunigt wird. Dabei dürfen Sie von der Klasse `PrimFactorTool` zwar die Klasse `ThreadPrimeFactorTool` ableiten, in dieser neuen Klasse aber *keine neuen Methoden* hinzufügen. Zum Schluss dürfen Sie auch im *Hauptprogramm* nur folgende Änderungen machen:

Vorher:	Änderung:
`PrimeFactorTool pt =` `  new PrimeFactorTool();`	`PrimeFactorTool pt =` `  new ThreadPrimeFactorTool();`
`for (int num=MIN_NUM; num<=MAX_NUM; num++)` `  pt.printPrimeFactors(num);`	`for (int num=MIN_NUM; num<=MAX_NUM; num++)` `  pt.printPrimeFactors(num);`

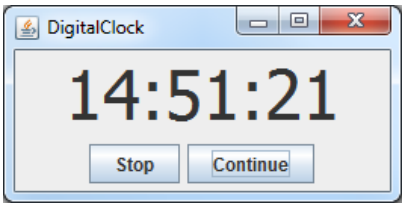
4. Erstellen Sie eine Klasse mit dem Namen `FindMin` – abgeleitet von der Klasse `Thread` – welche in der Methode `run()` das *Minimum* eines ihr bekannten `int`-Feldes sucht. Das Ergebnis der Suche soll in einem Textfeld ausgegeben werden. Beachten Sie dabei, dass das Feld Werte zwischen 0 und `Integer.MAX_VALUE - 1` enthält.

   Auf ähnliche Art und Weise sollen Sie die Klassen `FindMax` zur Suche des *Maximums* und `FindAvg` zur *Ermittlung des Mittelwertes* programmieren.

   Entwerfen Sie eine einfache Benutzerschnittstelle, welche vor Threadstart automatisch ein int-Feld mit 10.000.000 zufälligen Werten zwischen 0 und `Integer.MAX_VALUE - 1` füllt. Dann sollen per Knopfdruck alle drei Thread gestartet werden, der *Fortschritt* der Threads jeweils angezeigt werden und zum Schluss das Ergebnis ausgegeben werden.



5. Programmieren Sie die Klasse `DigitalClockLabel` abgeleitet von `JLabel`, welche das Interfache `Runnable` implementiert und im Ausgabetext der Basisklasse `JLabel` im *Sekundentakt* die *aktuelle Uhrzeit* ausgibt. Diese Klasse soll die Methoden `setStopped()` und `getStopped()` zur Verfügung stellen, durch welche die Aktualisierung der Uhrzeitausgabe *angehalten* werden kann. Programmieren Sie die Klasse so, dass im angehaltenen Zustand möglichst wenig Prozessorleistung benötigt wird.




```
public class DigitalClockLable extends JLabel implements Runnable
{
   …
}
```

   Implementieren Sie noch die *Benutzerschnittstelle*, welche Ihre Klasse `DigitalClockLabel` verwendet.