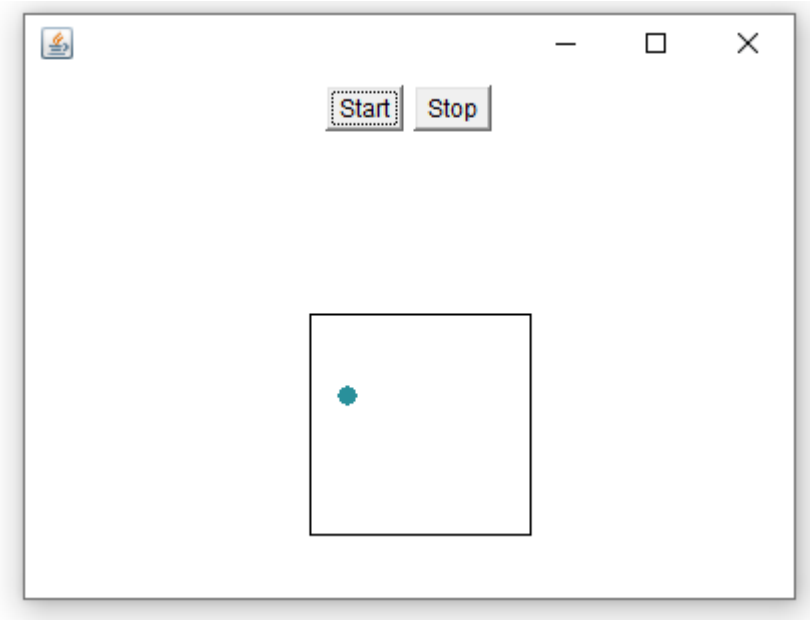
Übung: Nebenläufige Programmierung

Diese Übung soll ihnen zeigen, welche Situationen **Threads** erfordern und wie diese eingesetzt werden können. Im ersten Schritt soll durch den Aufruf der Methode **sleep** () das Programm künstlich verlangsamt werden. Im zweiten Teil fügen wir einen unabhängigen **Thread** ein, auf diese Weise ist das Userinterface nicht mehr blockiert und kann weiterhin Befehle verarbeiten.



Aufgabe 1: Klasse AWTDemo.java und Ball.java

Versuchen Sie zunächst mit einer for-Schleife in der Klasse **Ball** das Programm auszubremsen, so dass sich der Ball mit einer sinnvollen Geschwindigkeit bewegt.

Mit dieser Art der Implementierung ist die Geschwindigkeit im Wesentlichen von der momentanen Rechenleistung des Computers abhängig.

→ Rufen Sie jetzt an der Stelle der for-Schleife die sleep() Methode des aktuellen Treads auf.

```
try {
  Thread.sleep(50);
} catch (InterruptedException e) {
  e.printStackTrace();
}
```

Aufgrund einer möglichen Exception, muss ein try/catch Block verwendet werden.

Mit dieser Implementierungsart wird nun das Programm tatsächlich für 50ms angehalten und CPU-Leistung steht für andere Prozesse zur Verfügung.

Aufgabe 2: AWTThreadDemo.java und ThreadBall.java

Nun wollen wir durch einen unabhängigen Thread dafür sorgen, dass das GUI bedienbar bleibt.

- Implementieren Sie eine neue Klasse ***ThreadBall***, mit der der Ball als unabhängiger Thread gestartet werden kann.
- Die Klasse ***ThreadBall*** soll nun die Klasse Thread erweitern.
- Der Code innerhalb der Methode ***run()*** soll jetzt solange ausgeführt werden wie die boolsche Variable ***weitermachen*** auf ***true*** steht.
- Implementieren Sie weiterhin eine Methode ***stoppen()***, welche die Variable ***weitermachen*** auf ***false*** setzt.
- Das passende Frame ***AWTThreadDemo***.java zur neuen ***ThreadBall*** Klasse finden Sie bereits in der Vorlage und muss noch so ergänzt werden, dass mit dem Start und Stop Button der Thread gestartet bzw. beendet wird.

Aufgabe 3

Ergänzen Sie die Klassen ***AWTThreadDemo*** und ***ThreadBall*** in der Art, das bis zu 50 Bälle gestartet und wieder gestoppt werden können.

Aufgabe 4

Verfahren Sie nun wie in Aufgabe 3, allerdings soll dieses Mal die Klasse Ball als implementierte ***Runnable*** Schnittstelle realisiert werden und im Frame entsprechend verwendet werden.

Erstellen Sie hierzu eine Klasse **AWTRunnableDemo** und eine Klasse **RunnableBall**.