Exercises on Chapter 11

Gustav Cedersjö

SlowBox

The class `SlowBox` is a synchronized container that on `get` and `set` puts the executing thread to sleep for a specified delay.

```
public class SlowBox<T> {
    private T content;
    private final long delay;

    public SlowBox(T content, long delay) {
        if (delay < 0) {
            throw new IllegalArgumentException("delay");
        }
        this.content = content;
        this.delay = delay;
    }

    public synchronized T get()
            throws InterruptedException {
        Thread.sleep(delay);
        return content;
    }

    public synchronized T set(T newContent)
            throws InterruptedException {
        Thread.sleep(delay);
        T oldContent = content;
        content = newContent;
        return oldContent;
    }
}
```

Exercise 1

Create a program that creates a `SlowBox` and a bunch of tasks that use the box. The program does not have to do anything useful. Measure the time it takes to execute the program with a single-threaded `Executor` versus a multi-threaded `Executor`.

Exercise 2

Modify the class `SlowBox` with the technique called "narrowing lock scope" described in the book and do the same measurements as in Exercise 1.