Übungsblatt 10

Willkommen zum Praktikum zu Programmieren 3. Lesen Sie die Hinweise.

Aufgabe 1. Gegeben ist eine Klasse `Canvas` mit einer Hauptroutine in `canvas.py`. Auf dem Canvas werden Objekte, die sich selbstständig bewegen, als Kreis animiert angezeigt. Diese Objekte sollten intern mit Hilfe von Threads Ihre eigene Position ändern. Die Position ist ein x-Wert und ein y-Wert zwischen 0 und 1. Damit die animierte Anzeige funktioniert müssen die Objekte die folgenden Methoden implementieren.

- `getPosition()`: gibt zwei Gleitkommazahlen $0 \leq x, y < 1$ zurück. Die Position soll sich selbstständig über Zeit ändern.
- `pause()`: Pausiert die selbständige Änderung der Position über Zeit.
- `cont()`: Fährt mit der selbstständigen Änderung der Position über Zeit fort.
- `stop()`: Beendet die selbstständige Änderung der Position über Zeit endgültig. Es findet danach kein Aufruf mehr von `pause`, `cont`, oder `stop` statt.

Dem Canvas wird die Klasse der zu erzeugenden Objekte initial übergeben. Canvas erzeugt sich Objekte bei Bedarf. Bei der Initialisierung eines Objekts sollte der Thread gestartet werden und sich in einen pausierten Zustand begeben.

Implementieren Sie in einem Modul `particle` die Klasse `Particle`, die als zu animierendes Objekt dienen kann. Verwenden Sie für die Bewegung einen Thread, der alle 10 Millisekunden die Position in x- und y-Richtung um maximal 0.01 verändert. Initial sollen die Objekte innerhalb von 0.4 und 0.6 liegen, also $0.4 \leq x, y \leq 0.6$. Bestimmen Sie die Richtung und die Größe der Veränderung mit Hilfe von Zufallsvariablen (`random`). Implementieren Sie die oben genannten Methoden. Führen Sie `canvas.py` aus. Wenn alles geklappt hat, dann sollten sich Ihre Objekte auf dem Canvas bewegen.

Aufgabe 2. Implementieren Sie eine Klasse `IntState`, die es erlaubt einen Integer-Wert um einen festen Betrag zu erhöhen oder zu erniedrigen und sich diesen zu holen. Diese Klasse soll nicht thread-sicher (threadsafe) sein. Implementieren Sie eine weitere Klasse `IntStateSafe`, die threadsafe ist und sich ansonsten gleich verhält.

Schreiben Sie ein Programm, das auf einer Instanz eines `IntState` mit 100 Threads Werte erhöht und erniedrigt. Verwenden Sie dazu 1 Million Zufallszahlen zwischen 1000 und -1000. Überprüfen Sie ob das Ergebnis korrekt ist. Geben Sie dazu aus welchen Wert Sie erwarten und welchen Wert Sie erhalten. Wiederholen Sie den Test mit `IntStateSafe` und prüfen Sie wieder. Geben Sie die jeweils verbrauchte CPU-Zeit für die Ausführung des Tests aus.

Aufgabe 3. 26 Threads generieren jeweils immer a,b,c,...,z, die an eine Liste angefügt werden. Die Threads sollen nach jedem Zeichen zwischen 0 und 0.001 Sekunden, die genaue Zeit sollte zufällig sein, warten. Dadurch soll das Thread-Umschalten wahrscheinlicher werden. Ansonsten würde wohl meist der erste Thread (a) gewinnen. Ein

weiterer Thread nimmt die Zeichen vorne aus der Liste und zählt, wie häufig ein Zeichen vorkommt. Sobald ein Zeichen x mal (erster Kommandozeilenparameter, 100 als Vorgabewert) vorkommt soll der lesende Thread sich beenden. Die Klasse des lesenden Threads hat eine Methode das letzte verwendete Zeichen, also das Zeichen das gewonnen hat, zu erhalten. Geben Sie das Gewinner-Zeichen aus. Nach Beendigung des lesenden Threads sollen sich alle anderen Threads auch beenden.

Der lesende Thread soll sich mit `wait` schlafen legen, falls keine Zeichen mehr verfügbar sind und bei neuen Zeichen wieder geweckt werden. Sie können die gemeinsame Liste in einer Klasse `Warteschlange` kapseln, dessen eine Instanz Sie allen Threads bei der Initialisierung übergeben.

Aufgabe 4. Sie können über die URL

```
http://portal.intern.mi.hs-rm.de/news/q/<maillingliste>/<jahr>/<monat>
```

alle Nachrichten der Mailinglisten `bami-allgemein`, `bami1`, `bami2`, `bami3`, `bami4`, `bami5` und `bami6` in einem Monat abrufen. Es gibt Nachrichten seit 2007. Die Monate gehen von Januar (0) bis Dezember (11). Schreiben Sie ein Programm, das alle Emails aus den Mailing-Listen liest und ausgibt.

Verwenden Sie zum Holen der Emails eines Monats jeweils einen Thread. Stellen Sie mit Hilfe von `Semaphore` sicher, dass die Anzahl der Threads, die gleichzeitig einen Monat laden beschränkt ist. Die Vorgabe ist maximal ein Thread.

Ihre Anwendung soll die folgenden Kommandozeilenparameter unterstützen:

- Option `-t threads`: Legt die maximale Anzahl gleichzeitig laufender Threads zum Herunterladen der Emails fest. Vorgabe `threads = 1`.
- Option `-m mlists`: Schränkt die zu durchsuchenden Mailinglisten ein auf solche, deren URL den Parameter enthält. Vorgabe `mlists =` (also leer).
- Option `search_term`: Schränkt die auszugebenden Emails auf die ein, die den Suchbegriff beinhalten. Vorgabe `search_term =` (also leer).

Zum Beispiel sucht

```
python search_news.py -t 5 -m bami-allgemein/2012 Grillen
```

mit 5 Threads gleichzeitig in allen monatlichen Zusammenfassungen aus dem Jahr `2012` der Mailingliste `bami-allgemein` alle Emails, die das Wort oder den Wortteil `Grillen` enthalten.

Hinweis 1. Zum Zugriff auf Daten über eine URL verwenden Sie `urllib2` und darin die Funktion `urlopen`. Zum Einlesen von Optionen können Sie `OptionParser` aus `optparse` verwenden.

`http://www.mi.hs-rm.de/~barth/hsrm/prog3`