Datenstrukturen PVL 4- SS2023

Professur Softwaretechnik

5| 2023

Administratives

Abgabe Termin

05.06.2023 - 23:59

Abgabe

Ihre Abgabe darf entweder in Java oder Python geschehen. Nutzen Sie für die Abgabe das Code Template aus dem OPAL-Bereich "Prüfungsvorleistung 4". Nichtbeachtung des Templates kann zu Punktabzügen führen! Sie dürfen zusätzliche Klassen und Methoden anlegen, um die Aufgabe zu lösen.

Laden Sie alle .java oder .py Dateien, die zum Ausführen Ihres Codes nötig sind, im OPAL-Bereich "Prüfungsvorleistung 4" hoch. Hinweis für Java: Von uns vorgegebene Interfaces brauchen Sie nicht mit abzugeben.

Erlaubte Klassen

Sie dürfen Klassen und Methoden aus der Standardbibliothek von Java bzw. Python nutzen. Bitte benutzen Sie keine Bibliotheken von Dritten.

Information zu Plagiaten

Es ist **erlaubt**, Lösungen aus dem Internet in Ihre Abgabe zu integrieren. Bitte **markieren** Sie jeglichen Code, der nicht von Ihnen selbst stammt, indem Sie einen Kommentar mit der Quelle des Codes (URL ist ausreichend) an die entsprechende Stelle setzen. Dies soll verhindern, dass Ihre Lösung fälschlicherweise als Plagiat erkannt wird.
Da es sich hierbei um eine Prüfungsvorleistung handelt, ist es notwendig, dass Sie die Aufgaben selbständig bearbeiten. Daher werden alle Abgaben untereinander auf Plagiate überprüft. Sollten Plagiate erkannt werden, gilt die Abgabe von **sämtlichen** beteiligten Parteien als ungültig.

Kompilierbarkeit und Clean Code

Es werden nur Abgaben gewertet, welche sich in einem ausführbaren Zustand befinden!
Achten Sie außerdem auf die Leserlichkeit des Codes. Unverständliche oder unangemessene Bezeichner von Klassen, Variablen, etc. können zu Punkteabzügen führen.

Fragen

Da es sich um eine Prüfungsvorleistung handelt, ist es wichtig, dass jeder Teilnehmer Zugang zu allen notwendigen Informationen hat. Deswegen bitten wir Sie, Fragen direkt ins Forum, im Thread "PVL 4: Fragen" zu stellen. Bitte beachten Sie, dass Sie ihre Fragen möglichst frühzeitig stellen. Kurz vor der Deadline können wir keine rechtzeitigen Antworten mehr garantieren.

Aufgabe

Der Fokus dieser PVL soll auf der Datenstruktur der Prioritätswarteschlange liegen.

Prioritätswarteschlangen werden häufig in Scheduling-Anwendungen benutzt. Ein Beispiel, wo solch eine Anwendung benutzt wird, wäre ein Betriebssystem. In Betriebssystemen existiert (grob vereinfacht) die Abstraktion Prozess. Für unsere Zwecke definieren wir einen Prozess einfach als ein kleines, nicht unterbrechbares Programm. Die Aufgabe des Betriebssystems ist nun die Planung, wann welcher Prozess ausgeführt wird.

In unserem Fall soll ein Prozess aus einer ”arrivalTime”, ”executionTime” und ”priority” bestehen. Das Betriebssystem soll die Möglichkeit haben, auf ”kernelNumber” Prozessoren zuzugreifen (so können mehrere Prozesse parallel durchgeführt werden). Die ”arrivalTime” beschreibt den diskreten Zeitpunkt, ab wann ein Prozess gestartet werden kann. Die ”executionTime” beschreibt, wie viele diskrete Zeitschritte der Prozess benötigt, um durchgeführt zu werden. Die ”priority” gibt an, welcher Prozess präferiert werden soll. Eine höhere Zahl bedeutet dabei, dass der Prozess Vorrang hat. Sollten mehrere Prozesse bereit zur Ausführung sein und ein Prozessor zur Verfügung stehen, so soll der Prozess mit der höchsten Priorität ausgeführt werden.

Für die Abstraktion des Prozesses steht das Interface ”Process” zur Verfügung. Implementierten Sie dafür folgende Methoden:

```
1 Konstruktor(int pid, int arrivalTime, int executionTime, int priority)
```

Implementieren Sie einen Konstruktor in ihrer entsprechenden Sprache, welcher die Attribute der Klasse basierend auf den übergebenen Werten setzt.

```
1 getPID()
```

Welche die PID zurückgibt.

```
1 getArrivalTime()
```

Welche die ”arrivalTime” zurückgibt.

```
1 getExecutionTime()
```

Welche die ”executionTime” zurückgibt.

```
1 getPriority()
```

Welche die ”priority” zurückgibt.

Für die Anwendung steht das Interface ”OperationSystem” zur Verfügung. Implementieren Sie dieses.

```
Konstruktor(int kernelNumber)
```

Implementieren Sie einen Konstruktor in ihrer entsprechenden Sprache, welcher die Anzahl der nutzbaren Kernels setzt.

```
createProcess(int arrivalTime, int executionTime, int priority)
```

Sie bekommen die nötigen Parameter, bis auf die PID, zur Erzeugung eines Prozesses. Ihre Implementierung soll die Zuweisung der PID durchführen. Eine PID soll im Intervall [1, 255] liegen. Es soll stets die niedrigste PID vergeben werden. Geben Sie die PID zurück. Sollte keine PID mehr verfügbar sein, geben Sie -1 zurück. Speichern Sie sich den Prozess für den weiteren Verlauf.

```
deleteProcess(int PID)
```

Sie bekommen eine Zahl aus dem Intervall [1, 255]. Sollte es einen Prozess geben, der diese PID besitzt, löschen Sie diesen und geben Sie die PID frei. Wurde ein Prozess gelöscht, geben Sie ”Wahr” zurück, ansonsten ”Falsch”.

```
execute()
```

Führen Sie die gespeicherten Prozesse aus (startend beim Zeitpunkt 0, positiv fortlaufend). Benutzen Sie alle Prozessoren und arbeiten Sie alle Prozesse so schnell wie möglich ab. Die Ausführung gilt als beendet, wenn alle Prozesse ausgeführt wurden.
Geben Sie eine Liste aus Listen zurück. Diese soll zu jedem Zeitpunkt einen ”log”- Eintrag darüber erstellen, welcher Prozessor welchen Prozess ausführt. Sollte kein Prozess ausgeführt werden, nutzen Sie die 0, sonst die PID des Prozesses. Die Liste soll folgendes Format haben:

```
[
    [ProcessInKernel[0], ProcessInKernel[1], ..., ProcessInKernel[n - 1]]
]
```

Dabei soll am Index i der Liste der i-te Zeitpunkt beschrieben sein.

Beispiel

```
OperationSystem scheduler = new OperationSystem(2);
scheduler.createProcess(0, 3, 10); //pid 1
scheduler.createProcess(0, 2, 10); //pid 2
scheduler.createProcess(2, 1, 10); //pid 3
scheduler.createProcess(5, 4, 10); //pid 4
scheduler.createProcess(5, 3, 90); //pid 5
scheduler.createProcess(9, 1, 10); //pid 6
scheduler.deleteProcess(6); //true
scheduler.createProcess(5, 1, 80); //pid 6

List<List<Integer>> result = scheduler.execute();
//[
//[1,2],
//[1,2],
//[1,3],
//[0,0],
//[0,0],
//[5,6],
//[5,4],
//[5,4],
//[0,4],
//[0,4]
//]
```