Datenstrukturen
Übung – SS2020

Abgabe: 14.06. bis 24Uhr.
Abgabeort: Bitte laden Sie Ihre Klasse im OPAL-Bereich "Abgabe_Pflichtaufgabe_6" hoch. Ergänzen Sie bei Ihrer hochgeladenen Klasse die Namen (Vor- und Nachname) aller beteiligten Personen. Sie können Ihre Abgabe bis zur Abgabefrist beliebig oft löschen und erneut einreichen.

Wichtig: 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, in den Thread "Fragen zur Pflichtaufgabe 6" zu stellen.

Packen Sie Ihre Klasse in das Package "pvl6_$groupX". Sollten Sie nicht in einer Gruppe arbeiten, wählen Sie den Namen statt "pvl6_$Nachname_$Vorname".
Ihre Klasse soll den Namen "PVL6_$GroupX" bzw. "PVL6_$Nachname_$Vorname" besitzen.

Für die Abgabe: Packen Sie all Ihre Klassen in ein .zip Archiv mit dem Namen "Set_GroupX.zip"
[Anmerkung: .rar ist kein .zip]

Prüfungsvorleistung 6 – Prioritätswarteschlangen

Prioritätswarteschlangen werden häufig in "scheduling-Anwendungen" benutzt. Ein Beispiel, wo solch eine Anwendung benutzt wird wäre ein Betriebssystem. In Betriebssystemen (grob vereinfacht) existiert die Abstraktion "Prozess". Ein Prozess sei für unsere Zwecke nichts weiter als ein kleines, nicht unterbrechbares Programm. Die Aufgabe des Betriebssystems ist nun die Planung, wann welcher Prozess ausgeführt wird. Hierfür gibt es wieder diverse Modelle. 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. Dabei bedeutet, dass eine höhere Zahl als Priorität 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 dieses. Sie können Ihre Implementierung beliebig benennen.

Für die Anwendung steht das Interface **"SchedulingTask"** zur Verfügung. Implementieren Sie dieses. (In Python benötigen Sie kein Interface, doch halten Sie sich an die folgenden Methoden-Signaturen).

Sollte zu einem ein Übergabeparameter "null" sein, geben Sie "null" zurück.

Die folgenden Methoden finden Sie im Interface "**Process**", hier finden Sie eine Spezifikation für das Verhalten der Methoden, Beispiele folgen weiter unten.

Java: public ProcessImplementation(int pid, int arrivalTime, int executionTime, int priority)
Python: __init__(self, pid: int, arrivalTime: int, executionTime: int, priority: int)

Sie bekommen eine PID gegeben (Prozess-Identifikation-Nummer). Diese soll den Prozess eindeutig beschreiben. Die weiteren Parameter sind wie beschrieben aufzufassen. Weisen Sie diese dem Prozess zu.

Java: public int getPID()
Python: def getPID(self)-> int

Diese Methode gibt die „PID“ zurück.

Java: public int getArrivalTime()
Python: def getArrivalTime(self) -> int

Diese Methode gibt die "arrivalTime" zurück.

Java: public int getExecutionTime ()
Python: def getExecutionTime(self) -> int

Diese Methode gibt die "executionTime" zurück.

Java: public int getPriority()
Python: def getPriority(self) -> int

Diese Methode gibt die "priority" zurück.

Die folgenden Methoden finden Sie im Interface "**SchedulingTask**", hier finden Sie eine Spezifikation für das Verhalten der Methoden, Beispiele folgen weiter unten.

Java: public PVL6_$GroupX(int kernelNumber)
Python: __init__(self, kernelNumber: int)

Die Anzahl der Prozessoren soll festgelegt werden.

Java: public int createProcess(int arrivalTime, int executionTime, int priority)
Python: def createProcess(self, arrivalTime: int, executionTime: int, priority: int) -> int

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 stehts 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.

Java: public boolean deleteProcess(int pid)
Python: def createProcess(self, pid: int) -> bool

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".

Java: public String execute()
Python: def execute (self) -> str

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, sollten alle Prozesse ausgeführt wurden sein. Geben Sie einen String zurück, dieser soll zu jedem Zeitpunkt einen "log"-Eintrag erstellen, welcher Prozessor, welchen Prozess ausführt. Sollte kein Prozess ausgeführt werden, nutzen Sie die 0 sonst die „PID“ des Prozesses. Der String Soll folgendes Format haben:
"$Zeitpunkt[i] : $ProcessInKernel[0] $ProcessInKernel[1] … $ProcessInKernel[kernelNumber -1] \n
…."
Für alle Zeitpunkte i die Betrachtet werden müssen.

Beispiele in Java PseudoCode

```
PVL6_$GroupX scheduler = new PVL6_$GroupX(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
String result = scheduler.execute();
```

"result" hat Soll dabei am Ende folgende Form besitzen:

```
"0 : 1 2
 1 : 1 2
 2 : 1 3
 3 : 0 0
 4 : 0 0
 5 : 5 6
 6 : 5 4
 7 : 5 4
 8 : 0 4
 9 : 0 4
"
```