Übung 04: Heap / Priority Queue

Abgabetermin: 11.05.2010 10:15

Name: _____ **Matrikelnummer**: _____

Gruppe: ❒ G1 (Wolfinger) ❒ G2 (Wolfinger)

Aufgabe	Punkte	gelöst	abzugeben schriftlich	abzugeben elektronisch	Korr.	Pkte
Aufgabe 04.1	18	❑	Java-Programm Testfälle und Ergebnisse	Java-Programm	❑	
Aufgabe 04.2	6	❑	Heaps in Binärbaum bzw. Array-Darstellung	-	❑	

Aufgabe 04.1: Prozesse in Heap verwalten (18 Punkte)

a) Implementieren Sie eine Prozessverwaltung als Prioritätswarteschlange (Heap). Das Ordnungskriterum ist die Priorität eines Prozesses. Eine echte Prozessverwaltung regelt die zeitliche Ausführung mehrerer Prozesse in einem Betriebssytem. In dieser Übungsaufgabe verwalten wir anstatt echter Prozesse Objekte der Dummy-Klasse *Process*.

Die Klasse *Process* hat folgende Schnittstelle:

```
public class Process {
  public Process() { ... }
  public int getId() { ... }
  public String toString() { ... }
}
```

- *int getId()* liefert eine Zahl die den Prozess eindeutig identifiziert; vergeben Sie für jedes neue Objekt der Klasse *Process* eine fortlaufende Nummer; der erste mit *new Process()* erzeugte Prozess hat die Id 1, der zweite die Id 2, usw.
- *String toString()* liefert eine Zeichenkette im Format "Prozess #%03d", wobei die Id des Prozesses als Argument eingesetzt wird

Implementieren Sie die Klassen *ProcessManager* und *ProcessData* mit folgenden Schnittstellen. Die Klasse *ProcessData* speichert die Priorität zu einem Prozess. Implementieren Sie den Heap keinesfalls als Baum, sondern mit Hilfe der Klasse *java.util.ArrayList*:

```
public class ProcessData {
  public ProcessData(Process p, int priority) { ... }
  public Process getProcess() { ... }
  public int getPriority() { ... }
}

public ProcessManager implements Iterable<ProcessData> {
  private ArrayList<ProcessData> heap;
  public ProcessManager() { ... }
  public ProcessManager clone() { ... }
  public void insert(Process p, int priority);
  public ProcessData peek() { ... }
  public Process poll() { ... }
  public int size() { ... }
  public Iterator<ProcessData> iterator() ( ... }
  public Iterator<ProcessData> getUnsortedEntries() { ... }
}
```

- *ProcessManager clone()* erstellt eine flache Kopie der Prozessverwaltung; flache Kopie bedeutet, dass die ArrayList kopiert wird, aber nicht die darin enthaltenen *ProcessData*-Objekte
- *void insert(Process p, int priority)* fügt den Prozess *p* in die Prozessverwaltung mit der Priorität *priority* ein und stellt die Heap-Ordnung her
- *ProcessData peek()* liefert den Prozess mit der höchsten Priorität ohne ihn aus der Prozessverwaltung zu entfernen

- *Process poll()* liefert den Prozess mit der höchsten Priorität und entfernt ihn aus der Prozessverwaltung
- *int size()* liefert die Anzahl der verwalteten Prozesse
- *Iterator<ProcessData> iterator()* liefert einen Iterator mit allen Prozessen, sortiert nach Priorität in absteigender Reihenfolge
- *Iterator<ProcessData> getUnsortedEntries()* liefert einen Iterator für alle Prozesse ohne Sortierung; der Iterator liefert die Prozesse in jener Reihenfolge in der sie in der *ArrayList* gespeichert sind

b) Implementieren Sie in einer Klasse *ProcessManagerTest* Hilfsmethoden für einen Testtreiber mit folgenden Funktionen:

- *static void addRandomProcess()* erzeugt einen Prozess mit zufälliger Priorität zwischen 1 und 999, gibt dessen Daten auf der Konsole aus und fügt ihn in die Prozessverwaltung ein

```
Process #001 with priority 984
Process #002 with priority  94
Process #003 with priority 528
Process #004 with priority 666
Process #005 with priority  25
Process #006 with priority 894
Process #007 with priority 236
```

- *static void printFormatted()* visualisiert den Heapinhalt als Baum auf der Konsole

```
      984
   666   894
 94  25 528 236
```

- *static void reinsert()* entfernt den Prozess mit höchster Priorität und fügt ihn mit neuer zufälliger Priorität zwischen 1 und 999 erneut ein

```
ProcessManager pm = ...
Process p = pm.poll();
pm.insert(p, (int) (Math.random() * 999));
```

```
      894
   666   528
 94  25 236 250
```

c) Implementieren Sie mit den Methoden der Klasse *ProcessManagerTest* mehrere Testmethoden. Variieren Sie dabei die Anzahl der verwalteten Prozesse sowie die Reihenfolge der Operationen. Ein Beispiel:

```
ProcessManager pm = ...
for(int i=0; i < 7; i++) {
  addRandomProcess();
}
printFormatted();
for(int i=0; i < 3; i++) {
  reinsert();
  printFormatted();
}
for(int i=0; i < 7; i++) {
  printFormatted();
  Out.println(repeat(pm.peek().getPriority() + " polled");
  pm.poll();
}
```

Diese Testmethode soll z.B. folgende Ausgabe erzeugen:

```
Process #001 with priority 413
Process #002 with priority 413
Process #003 with priority 705
Process #004 with priority 920
Process #005 with priority 918
Process #006 with priority  74
Process #007 with priority 864

      920
   918   864
413 705  74 413
Polled:     Process #004 with priority 920
Reinserted: Process #004 with priority 825

      918
   705   864
413 413  74 825
```

```
Polled:     Process #005 with priority 918
Reinserted: Process #005 with priority 804

      864
   705   825
413 413  74 804
Polled:     Process #007 with priority 864
Reinserted: Process #007 with priority 421

      825
   705   804
413 413  74 421


      825
   705   804
413 413  74 421
825 polled

      804
   705   421
413 413  74
804 polled

      705
   413   421
 74 413
705 polled

      421
   413   413
 74
421 polled

  413
 74 413
413 polled

  413
 74
413 polled

 74
74 polled
```

Abzugeben ist: Java-Programm, Testergebnisse

Aufgabe 04.2: Einfügen und Löschen in Heaps (6 Punkte)

Gegeben ist folgendes Array:

0	1	2	3	4	5	6	7	8
	T	**S**	**O**	**G**	**R**	**I**	**N**	**A**

a) Stellen Sie den durch das Array beschriebenen Heap als Binärbaum dar.

b) Fügen Sie die nachfolgend gegebenen Schlüssel nacheinander in den Heap ein und zeichnen Sie den Heap nach jedem Schritt als Binärbaum. Markieren Sie die Schlüssel im Heap die beim Einfügen vertauscht wurden.

D I B B U K I M

c) Stellen Sie den Heap aus b) wieder als Array dar und entfernen Sie den höchstprioren Schlüssel. Wiederholen Sie das Entfernen 3x und zeichnen Sie nach jedem Schritt den Heap als Array. Markieren Sie die Schlüssel die beim Entfernen vertauscht wurden.