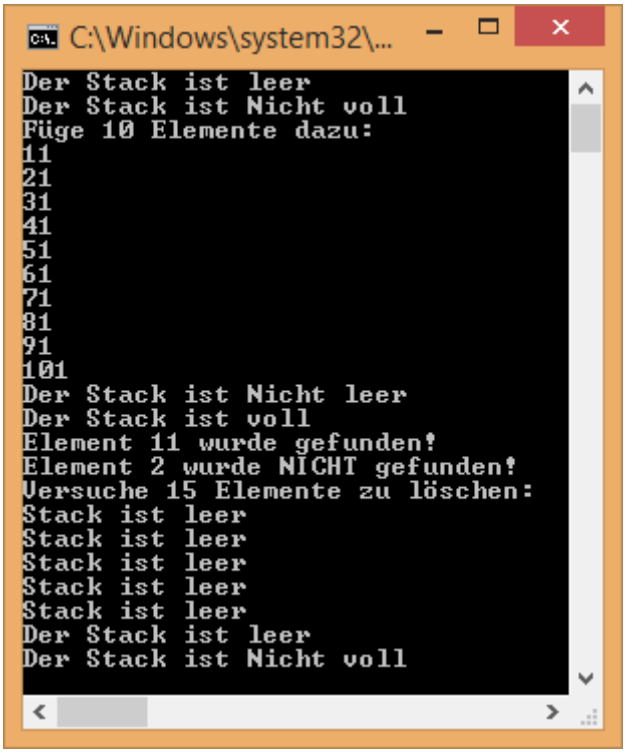
ADT - Anwendungen von Listen

(1.) Fügen Sie der Klasse Stack die Methoden peek und contains hinzu und testen Sie diese! Testen Sie auch die beiden Methoden isFull und isEmpty!



(2.) Fügen Sie der Klasse Queue die Methoden peek und contains hinzu und testen Sie diese! Testen Sie auch die beiden Methoden isFull und isEmpty! Verwenden Sie dazu die Beschreibungen in zur den beiden Klassen!

(3.) Die entsprechende Java-Klasse Queue und Stack leitet sich von java.util.collection ab und dort sind bereits die Methoden clear, containsAll, size implementiert. Kodieren Sie diese drei Methoden in der Stack Klasse und testen Sie diese in einem Rahmenprogramm!

(4.) Erstellen Sie ein Programm zum Erkennen von balancierten Klammerausdrücken(->googeln) wie (a(b))!

„(„:Lesen von „(“: push ("(")

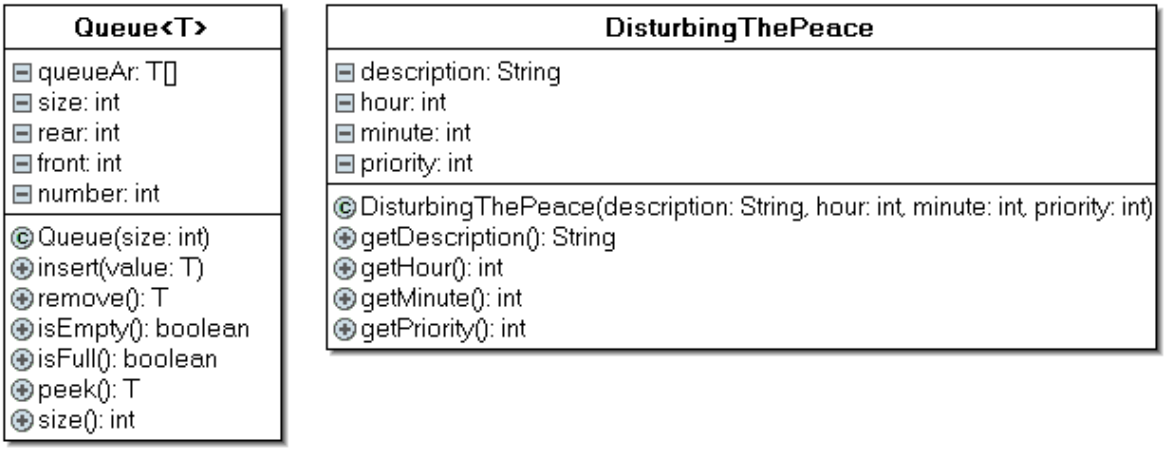
Lesen von „)“: pop ("(")

Wenn der String ausgelesen ist: Ausdruck ist korrekt, wenn Stack am Ende leer ist! Erstellen Sie dazu eine Methode:

```
public static boolean hasValidBrackets(String expr){
……..
}
```

(5.) (a.) Erstellen Sie eine generische Klasse Queue analog zur Lösung von (1.)! Orientieren Sie sich an der Vorlage der generische Stack Klasse!

(b.) Eine Polizeiwache möchte Ruhestörungen verwalten. Erstellen Sie eine geeignet Klasse DisturbingThePeace, welche die Attribute description, hour, minute (Eingang des Auftrags) und eine Priorität(1..10) enthält. Verwenden Sie die Klasse Queue, um Störungen aufzunehmen und zu löschen. Ergänzen Sie kreativ um eigene Erweiterungen!



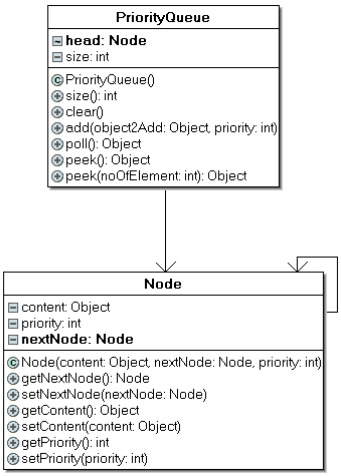
(6.) Kodieren Sie für obigen Fall eine Priorität-Warteschlange PriorityQueue, wo die Priorität nicht in der Klasse, sondern in der Queue gespeichert wird! Verwenden Sie eine modifizierte verkettete Liste!

Hinweis 1: Da diese Aufgabe für Anfänger sehr komplex ist, habe ich diese „stufenweise" gestellt!

Hinweis 2: In ähnlicher Form wurde die Aufgabe im Landesabitur Informatik an allgemeinbildenden Schulen für den GK 2015 in Nordrhein-Westfalen gestellt!

Hinweis 3: Beachten Sie mögliche Dateien in den Template-Ordner der hier zugehörigen Datei!

(a.) Erstellen Sie eine Klasse Node, in der zusätzlich das Attribut Priorität vorhanden ist und kodieren Sie analog zu der Klasse SinglyLinkedList aus „Einfach verkettete Liste" eine Klasse PriorityQueue! Verwenden Sie möglichst viele Methoden ganz oder modifiziert aus SinglyLinkedList!
Dabei haben die „neuen" Methoden die gleiche Funktion wie in der Klasse PriorityQueue von Java außer peek(int: noOfElement), welches das Element an der noOfElement-Stelle zurückgibt:



Testen Sie die Klasse mit dem Hauptprogramm:

```
public class QueueExample {
  public static void main(String[] args) {

    PriorityQueue distThePeaceQueue = new PriorityQueue();
    distThePeaceQueue.add(new String("Karnevalhelau"),2);
    distThePeaceQueue.add(new String("Karnevalgesang"),4);
    distThePeaceQueue.add(new String("Grölen"),7);
    distThePeaceQueue.add(new String("Pfeifen im Lehrerzimmer"),3);
    distThePeaceQueue.add(new String("Karnevalgesang"),5);
    distThePeaceQueue.add(new String("Knaller am Mittag"),9);
    distThePeaceQueue.add(new String("Disco um 5.00"),6);
    System.out.println("No of Elements before listing/removing:"+distThePeaceQueue.size());
    int noOfElements=distThePeaceQueue.size();
    for (int i=1;i<=noOfElements;i++) {
      String tmpDistThePeaceQueue= (String)distThePeaceQueue.peek(i);
      System.out.println(tmpDistThePeaceQueue);
    }
    System.out.println("=====================================");
    for (int i=1;i<=noOfElements;i++) {
      String tmpDistThePeaceQueue= (String)distThePeaceQueue.poll();
      System.out.println(tmpDistThePeaceQueue);
    }
    System.out.println("No of Elements after listing/removing:"+distThePeaceQueue.size());
    System.out.println("That's all guys(and girls :-) )!");
  }
}
```

(b.) Ändern Sie die Methode add ab, so dass diese Element nach Priorität in die Warteschlange einsortiert werden, allerdings hinter Elemente von gleicher Priorität!

(c.) Ändern Sie die Klasse Node und anschließend PriorityQueue so ab, dass diese generisch ist:

public class Node<T>{

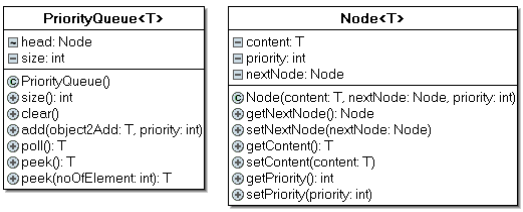
// Modifizierter Quellkode aus (b.)

}

public class PriorityQueue<T> {

// Modifizierter Quellkode aus (b.)

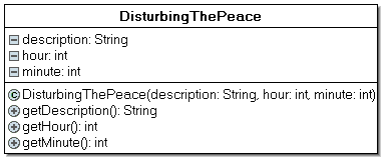
}



Ändern Sie dazu im Hauptprogramm den Aufruf wie folgt ab:

```
PriorityQueue<String> distThePeaceQueue = new PriorityQueue<String>();
…….
…….
….String tmpDistThePeaceQueue= (String)distThePeaceQueue.poll();
```

(d.) Erstellen Sie eine Klasse DisturbingThePeace, die den Anforderungen der Polizei entspricht:



Ändern Sie dazu im Hauptprogramm den Aufruf wie geeignet ab!

(e.) Ergänzen Sie die PriorityQueue-Klasse um folgende Funktionalitäten(ohne Lösung, nicht nach steigender Schwierigkeit geordnet!):

- *Ist ein bestimmtes Element in einer Warteschlange vorhanden?*
- *Eine Warteschlange soll in eine neue kopiert werden!*
- *Es soll geprüft werden, ob zwei Warteschlangen gleich sind!*
- *Zwei Warteschlangen sollten in eine kopiert werden, dabei wird die „Reihenfolge" der Objekte eingehalten; bei gleicher Priorität werden die Elemente der 1. Warteschlange bevorzugt!*
- *Ist eine Warteschlange in einer anderen Warteschlange enthalten d.h. sind alle Elemente der 1. Warteschlange eine Teilmenge der Elemente der 2. Warteschlange!*

Informieren Sie sich über die Namensgebung in Java, wählen sie geeignet Namen für Methoden und Parameter; implementieren und testen Sie diese in der Version von (b.)!

Bei „nachhaltiger“(etliche Tests!) Korrektheit ergänzen Sie die Methoden in (c.) und (d.) und testen Sie diese dort!

ADT - Anwendungen von Listen - Lösungen

(1.)- (X.) In Dateiform!