Java
Obligatory Exercises Week 21

Anders "Bongo" Bjerg Pedersen

31. maj 2006

Exercise 21

Implement a queue using the following skeleton. Remember to test your queue.

Vores FIFO (First In, First Out)-kø skal kunne et par forskellige ting: man skal kunne tilføje et nyt element bagerst i køen, tage et element forrest i køen ud og kunne læse indholdet af de køelementer, man hiver ud. Til dette bruger vi tre klasser: *Queue.java*, der indeholder methods til at udføre ovennævnte handlinger; *QueueElement.java*, der opretter og håndterer de enkelte køelementer og til sidst *QueueTest.java*, der afprøver alle funktionerne.

Programmet fungerer efter hensigten bortset fra, at det giver NullPointerException, sidste gang man prøver at deQueue et tomt køelement (null). Dette kan løses på flere måder, enten ved en while-løkke, der kører indtil køen er tom eller en simpel konstrueret fejlmeddelelse, der dukker op, hvis man prøver at deQueue flere elementer, end der er tilgængelige.

Koden til *Queue.java* er som følger:

```
public class Queue {
  private QueueElement first;   //Tre fields til en standardkø
  private QueueElement last;
  private QueueElement temp;

  public Queue() {   //Opretter en tom kø
    first = last = null;
  }

  public boolean isEmpty() {   //Returnerer true, hvis første og sidste element er tomt
    return first == null && last == null;
  }

  public void enQueue(QueueElement elmt) { //Lægger et element i køen
    if(isEmpty()) {
      first = last = elmt;
      return;
    }
    else {
      last.behind = elmt;
      last = elmt;
      return;
    }
```

```
  }

  public QueueElement deQueue() {       //Fjerner et element fra køen
    if (isEmpty())
      return null;
    else {
      temp = first;
      first = first.behind;
      if(first == last)
        last = null;
        return temp;
    }
  }
}
```

Koden til *QueueElement.java* er som følger:

```
public class QueueElement {
  public QueueElement behind;             //Angiver det efterfølgende element
  public String content;                  //Opbevarer køelementets indhold

  public QueueElement(String content) {   //Opretter et nyt køelement
    this.content = content;
    behind = null;
  }

  public String content() {               //Returnerer indholdet af køelementet
    return content;
  }

  public void setBehind(QueueElement elmt) {   //Angiver det efterfølgende element
    behind = elmt;
  }
}
```

Koden til *QueueTest.java* er som følger:

```
public class QueueTest {

  static Queue koe = new Queue();

  //Opretter 3 nye køelementer:

  static QueueElement e1 = new QueueElement("First element");
  static QueueElement e2 = new QueueElement("Second element");
  static QueueElement e3 = new QueueElement("Third element");

  public static void main(String[] args) {

    koe.enQueue(e1); //Lægger elementerne i køen
    koe.enQueue(e2);
    koe.enQueue(e3);

    //Følgende kode spytter 4 gange et køelement ud
    // og angiver derefter, om køen er tom:

    for(int i=1;i<5;i++) {
```

```
      System.out.println("Queue element no "+i+": "+koe.deQueue().content());
      System.out.println("Empty: "+koe.isEmpty());
    }
  }
}
```

Hvilket giver følgende output:

```
wl250:~/Desktop/Java/queue anders$ java QueueTest
Queue element no 1: First element
Empty: false
Queue element no 2: Second element
Empty: false
Queue element no 3: Third element
Empty: true
Exception in thread "main" java.lang.NullPointerException
        at QueueTest.main(QueueTest.java:16)
```