Merkstoff zu Datenstrukturen Schlange und Stapel

Die Datenstruktur (Warte-)Schlange→ Queue

In der Informatik ist eine Queue eine häufig eingesetzte Datenstruktur. Sie ist eine Liste mit einer beliebige Menge von Objekten.
Es wird nach dem **First In – First Out-Prinzip** (deutsch zuerst hinein – zuerst hinaus, kurz **FIFO**) gearbeitet.

Klassendefinition

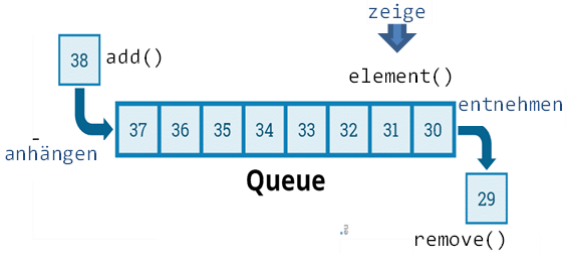
```
import java.util.Queue;
import java.util.concurrent.ConcurrentLinkedQueue;

public class SchlangeExample {
Queue<Integer> schlange;      //statt <Integer> kann beliebiger Objekttyp verwendet werden

public SchlangeExample1()
{
schlange = new ConcurrentLinkedQueue<>();
}                        }
```

Methoden

Objekt hinten anhängen	`queue.add(Objekt)`
Objekt entnehmen	`queue.remove(Objekt)`
Zeige Objekt -> Listenobjekt	`queue.element()`
Queue ist leer? -> boolean	`queue.isEmpty()`
Zahl der Objekte -> integer	`queue.size()`



Die Datenstruktur Stapel → Stack

Ein Stack (Stapel, Kellerspeicher) ist eine Liste von Objekten, bei der die Objekte nach dem **LIFO**-Prinzip (**Last in- First out**) bearbeitet werden. D.h., das zuletzt eingefügte Element wird als Erstes wieder entfernt.

Klassendefinition

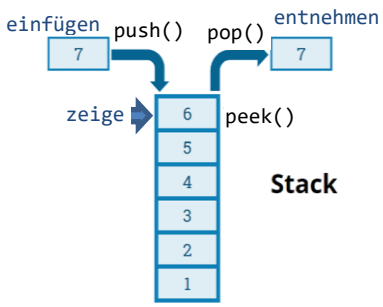
```
import java.util.Stack;

public class StapelExample1 {
Stack<Integer> stapel;        //statt <Integer> kann beliebiger Objekttyp verwendet

public StapelExample1()
{
stapel = new Stack<>();
}                        }
```

Methoden

Objekt oben einfügen	`stack.push(Objekt)`
Objekt entnehmen	`stack.pop(Objekt)`
Zeige Objekt -> Listenobjekt	`stack.peek()`
Queue ist leer? -> boolean	`stack.isEmpty()`
Zahl der Objekte -> integer	`stack.size()`



Die Datenstruktur Stapel-oder-Schlange → Deque

Ein Deque (Double-ended queue, ausgesprochen "Deck" – eine deutsche Übersetzung gibt es nicht) ist eine Liste von Objekten, bei der die Objekte sowohl auf der einen als auch auf der anderen Seite eingefügt und entnommen werden können.
Ein Deque kann sowohl als Queue (FIFO) als auch als Stack (**LIFO**) verwendet werden.

Klassendefinition

```
import java.util.*;

public class SchlaStaExample1 {
Deque <Integer> schlasta;      //statt <Integer> kann beliebiger Objekttyp verwendet

public SchlaStaExample1()
{
schlasta = new ArrayDeque<>();
}                          }
```

Methoden

Objekt vorn anhängen	`queue.addFirst(Objekt)`
Objekt hinten anhängen	`queue.addLast(Objekt)`
Objekt vorn entnehmen	`queue.removeFirst(Objekt)`
Objekt hinten entnehmen	`queue.removeLast(Objekt)`
Zeige erstes Objekt -> Listenobjekt	`queue.getFirst()`
Zeige letztes Objekt -> Listenobjekt	`queue.getLast()`
Queue ist leer? -> boolean	`queue.isEmpty()`
Zahl der Objekte -> integer	`queue.size()`