

**Dozent**

Prof. Dr. Thomas Vetter  
Departement  
Mathematik und Informatik  
Spiegelgasse 1  
CH – 4051 Basel

**Assistenten**

Marcel Lüthi

**Tutoren**

Pascal Mafli  
Loris Sauter  
Linard Schwendener  
Clemens Büchner  
Florian Spiess  
Jonathan Aellen  
Lukas Stöckli

**Erweiterte Grundlagen der Programmierung (45398-01)****Blatt 5****[8 Punkte]**

Vorbesprechung 22. - 26. Okt  
Abgabe 29. Okt - 2. Nov (vor dem Tutorat)

Wir empfehlen Ihnen, dass Sie im Buch “Sprechen Sie Java” bis und mit Kapitel 12 lesen (also zusätzlich Kapitel 8, 9 und 12), bevor Sie beginnen die Übungen zu lösen.

**Aufgabe 1 - Zeichenketten-Analyse****[4 Punkte]**

- (a) **Palindrom** - Schreiben Sie ein Programm `Palindrom`, das einen eingegebenen Text daraufhin überprüft, ob er ein Palindrom ist. Ihr Programm soll dazu folgende Methode implementieren:

```
public static boolean testPalindrom(String text)
```

Ein Palindrom ist ein Text, dessen Zeichen vorwärts und rückwärts gelesen die selbe Reihenfolge ergeben. Beachten Sie dabei nur die Buchstaben, nicht die Leer- und Sonderzeichen des eingegebenen Textes, und ignorieren Sie Gross- und Kleinschreibung. Ein paar Beispiele für Palindrome sind:

- Reliefpfeiler
- Lagerregal
- egale Lage
- Trug Tim eine so helle Hose nie mit Gurt?
- Eine güldne, gute Tugend: Lüge nie!

- (b) **Anagramm** - Schreiben Sie ein Programm `Anagramm`, das überprüft, ob zwei eingegebene Texte ein Anagramm bilden. Ihr Programm soll dazu folgende Methode implementieren:

```
public static boolean testAnagramm(String text1, String text2)
```

Ein Anagramm ist eine Buchstabenmordnung, zum Beispiel:

- "Die Herbstfarben" und "Herb abstreifend"
- "Dackel entlief Bassist -" und "Das laesst tief blicken."

Entfernen Sie auch hier ebenfalls Sonderzeichen, und ignorieren Sie Gross- und Kleinschreibung.

## Aufgabe 2 - Prioritätenschlange

[4 Punkte]

In der Vorlesung sind Datenstrukturen vorgestellt worden, bei denen die Reihenfolge von der Einfügeordnung abhängt. Implementieren Sie in dieser Aufgabe anhand des Queue Beispiels aus der Vorlesung eine Prioritätenwarteschlange.

Implementieren sie eine Klasse `PriorityQueue` zur Realisierung einer Prioritätenschlange, die Elemente nach Prioritäten verwaltet. Jedes Element der Schlange ist zusätzlich zu seinen Daten mit einer Priorität versehen, welche die Reihenfolge des Auslesens der Elemente aus der Schlange bestimmt. Dabei soll eine grosse Zahl einer hohen Priorität entsprechen. Die Operation `put(x)` fügt das Element `x` unter Beachtung seiner Priorität in die Schlange ein und gibt einen `Boolean` zurück je nach dem ob das Einfügen geklappt hat oder nicht. Die Operation `get()` liefert das Element mit der höchsten Priorität. Bei Elementen mit gleicher Priortität gilt FIFO (first in first out). Behandeln Sie mögliche Fehler bei `put(x)` und `get()`. Schreiben Sie ein geeignetes Testprogramm und behandeln Sie mögliche Fehler bei `put()` und `get()` (Konsolenausgabe).

*Tipp: Sie können den unten vorgegebenen Quellcode auch von der Webseite herunterladen und nutzen. Der Quellcode lässt sich jedoch nicht direkt kompilieren. Sie müssen zuerst noch fehlende Teile ergänzen.*

Listing 1: Element.java

---

```
1 public class Element {
2     private int priority;
3     private String data;
4
5     public String getData() { /* ... */ }
6     public int getPriority() { /* ... */ }
7
8     public Element(int priority, String data) { /* ... */ }
9     public String toString() { /* ... */ }
10 }
```

---

Listing 2: PriorityQueue.java

---

```
1 public class PriorityQueue {
2     static final int SIZE = 32;
3     Element[] q = new Element[SIZE];
4     int len = 0;
5
6     // Insert an element into the queue according to its priority
7     boolean put(Element x) { /* ... */ }
8
9     // Return the element with the highest priority from the queue
10    Element get() { /* ... */ }
11
12    // Return the queue length
13    int length() { /* ... */ }
14
15    // Print the queue contents
16    public String toString() { /* ... */ }
17 }
```

---