Hochschule Furtwangen
Fakultät Informatik
Prof. Dr. Lothar Piepmeyer

# Praktikum ‚Objektorientierte Programmierung'

## Aufgabenblatt 3

Für dieses erste Aufgabenblatt müssen Sie zunächst eine C++ Entwicklungsumgebung wie etwa CLion aufsetzen und einrichten. Wie in der Vorlesung besprochen, empfehle ich die Compiler-Flags -Wall -Wextra. Um möglichst im Einklang mit der Vorlesung zu sein, sollten Sie dafür sorgen, dass Ihr Compiler den C++14-Standard unterstützt. In keiner der Aufgaben darf der *globale* Namensraum genutzt werden.

- Definieren Sie einen eigenen Namensraum.
- Die Anweisung `using namespace` darf nicht genutzt werden.
- Alle Namen, die in Ihrem Code auftreten sind einfache und klare englische Begriffe. Überhaupt sollte Ihr Code klar und einfach sein. Gehen Sie davon aus, dass es eine einfache und klare Lösung gibt. So sind lange if-else-Kaskaden sicher nicht nötig.

Aufgabe 1:
Mit dem folgenden Code kann man ein Array mit `size` ganzen Zahlen sortieren.

```
void sort(int* numbers, int size){
    for (int i = 1; i < size; ++i) {
        int key = numbers[i];
        int j = i - 1;
        while (j >= 0 && numbers[j] > key) {
            numbers[j + 1] = numbers[j];
            j = j - 1;
        }
        numbers[j + 1] = key;
    }
}
```

a. Entwickeln Sie eine Funktion

```
void test_sort(int* numbers, int size)
```

die das Array `numbers` mit Hilfe von `sort` sortiert und mit Hilfe von `assert` testet, ob es wirklich sortiert wurde.

b. Entwickeln Sie eine Funktion

```
void test_sort()
```

die die Funktion aus a. für vier verschiedene Arrays aufruft.

Aufgabe 2:
a. Ersetzen Sie in der Klasse `Position` aus Aufgabenblatt 2 die set-Methode durch einen Konstruktor. Die set-Methode ist nicht mehr erforderlich.

b. Stellen Sie sicher, dass Ihre Klasse den folgenden Test besteht:

```
std::string mordor = "Mordor";
hfu::Position position(mordor, 47, 11);
```

```
assert(position.getName() == mordor);
assert(position.getX() == 47);
assert(position.getY() == 11);

mordor[0]='X';
assert(position.getName() == "Mordor");
```

Was ist der Sinn des letzten Tests?

Aufgabe 3:
a. Trennen Sie Klasse `Position` vom letzten Aufgabenblatt in je eine Datei für Deklaration und Implementierung. Behalten Sie den Namespace bei.

b. Ergänzen Sie Ihre Klasse um einen Methode `compare` zum sinnvollen Vergleich zweier Objekte vom Typ `Position`. Die folgenden Tests müssen bestanden werden:

```
std::string mordor = "Mordor";
hfu::Position position(mordor, 47, 11);
assert(position.compare(position)==0);
hfu::Position position_name("zordor", 47, 11);
hfu::Position position_x(mordor, 48, 11);
hfu::Position position_y(mordor, 47, 12);
assert(position.compare(position_name)<0);
assert(position.compare(position_x)<0);
assert(position.compare(position_y)<0);
assert(position_name.compare(position)>0);
assert(position_x.compare(position)>0);
assert(position_y.compare(position)>0);
```

Lesen und verstehen Sie den Test-Code, um die Funktionalität von `compare` zu verstehen.

Aufgabe 4:
Die Methode `compare` lässt sich ausgezeichnet zum Sortieren von Arrays mit Objekten vom Typ `Position` nutzen. Passen Sie die Funktionen aus Aufgabe 1 geeignet an.