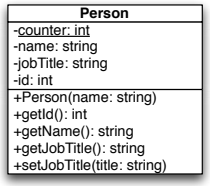
Objektorientierte Programmierung mit C++ (WS 2010)

Dr. Andreas F. Borchert, Tobias Brosch
Institut für Angewandte Informationsverarbeitung
Universität Ulm

Blatt 3: Abgabetermin 10. November 2010

1 Eine Klasse Person

Schreibt eine Klasse `Person`, welche folgendem UML-Klassendiagramm entspricht.



Teilt die Deklaration und die Implementierung wie üblich in zwei Dateien Namens `Person.h` und `Person.C`. Achtet dabei darauf, die Methoden, welche den Objekt-Zustand nicht ändern (in diesem Fall die get-Methoden) mit `const` zu deklarieren:

```
class SampleClass {
public:
        void myFunction() const {

        }
};
```

2 Pointer

Variablen, die Ihr in C++ anlegt, liegen irgendwo im Speicher. Dieses „irgendwo“ lässt sich herausfinden, indem man sich einen Zeiger oder englisch Pointer auf das Objekt geben lässt:

```
Person personFritz("Fritz");
```



```
Person* pointerToFritz = &personFritz;
```

Nun steht in `pointerToFritz` vom Typ `Person*` die Addresse, an der `personFritz` im Speicher lokalisiert ist. In gleicher Weise können wir uns auch die Addresse des Pointers holen:

```
Person** pointerToPointerToFritz = &pointerToFritz;
```

Und so nutzen wir das Objekt, auf welches ein Pointer zeigt (ein vorgestellter Asterix dereferenziert):

```
Person copyOfFritz(*pointerToFritz); // C++ style-initialization
// or: same as above, but other syntax
Person copyOfFritz = *pointerToFritz;
// note: this would be an assignment
// the above have been initializations
Person copyOfFritz;
copyOfFritz = *pointerToFritz;
// but back to Pointers: Let's call a method:
std::string nameOfFritz = (*pointerToFritz).getName();
// or operator->
std::string nameOfFritz = pointerToFritz->getName();
```

Wir können natürlich auch mehrere Pointer anlegen, die auf das selbe Objekt verweisen und einen Pointer eines Objekttyps auch auf ein anderes Objekt zeigen lassen:

```
Person personFritz("Fritz");
Person personRalf("Ralf");
Person* pointerToFritz1 = &personFritz;
Person* pointerToFritz2 = &personFritz;
Person* pointerToRalf = &personRalf;
std::cout << pointerToFritz1->getName() << std::endl;
pointerToFritz1 = pointerToRalf; // pointerToFritz1 points to Ralf
std::cout << pointerToFritz1->getName() << std::endl;
```

Durch das Schlüsselwort `const` lässt sich spezifizieren, dass etwas nicht verändert werden darf. Gleiches funktioniert bei Pointern:

```
const Person* personPointer;  // nobody may change
personPointer = &personFritz; // Fritz through this pointer
personPointer->getName(); // ok - const method
personPointer->setJobTitle("Chef"); // error
// nobody may change this Pointer
Person *const personConstPointer = personPointer;
// but you can change the object:
personConstPointer->setJobTitle("Student");
// error: try to change const-Pointer
personConstPointer = pointerToRalf;
```

```
// error: try to declare const-Pointer without initialization
Person *const personConstPointer;
// const Pointer to const Person:
const Person *const constPointerToConstPerson = personPointer;
```

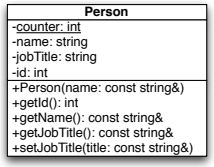
Aufgabe: Probiert diese Dinge aus! Seht euch die Fehlermeldungen an, die Ihr bekommt, wenn Ihr die mit error gekennzeichneten Zeilen ausprobiert.

3 Referenzen

In Methoden übergibt man häufig Objekte. Um nicht das ganze Objekt kopieren zu müssen, bietet es sich an Zeiger zu verwenden. Alternativ kann man *Referenzen* verwenden:

```
Person fritz("Fritz");
Person ralf("Ralf");
// like a const Pointer to a (non-const) Person
// but without the "strange" syntax
Person &fritzRef = fritz;
std::cout << fritz.getJobTitle() << std::endl;
Person &fritzRef2; // error - like uninitialized const Pointer
// this does not change the reference!
fritzRef = ralf; // this tries to assign ralf to fritz!
// a reference to a const-Person
const Person& ralfRef = ralf;
ralfRef.getName(); //ok
ralfRef.setJobTitle(); // error - non const-method called
```

Aufgabe: Probiert diese Dinge aus! Seht euch die Fehlermeldungen an, die Ihr bekommt, wenn Ihr die mit error gekennzeichneten Zeilen ausprobiert. Ändert anschließend eure Klasse aus Aufgabe 1 wie folgt ab:



- Legt eine Datei mit dem Namen `main.C` an, welche drei Person-Objekte erzeugt. Setzt den Job-Title von allen drei Objekten.

- Auf Person 1 soll ein const-Pointer angelegt werden, auf Person 2 ein Pointer auf ein const-Person-Objekt und auf Person 3 eine Referenz. Nachfolgende Aufgaben sollen erledigt werden, indem Ihr die Zeiger bzw. die Referenz verwendet!
- Gebt von allen drei Personen den Job-Title aus.
- Verändert von zwei Personen den Job-Title.
- Gebt von allen drei Personen erneut den Job-Title aus.

Submitted die drei Dateien `main.C`, `Person.h` und `Person.C` (die kompiliert und ausgeführt werden können):

```
submit cpp 3 main.C Person.h Person.C
```

Viel Spaß!