

# Smart Pointer Projekt – Beschreibung der Klassen

Für den Test im Moped sind genau folgende Dateien zu erzeugen:

**license.h**

**license.cpp**

**guild.h**

**guild.cpp**

**person.h**

**person.cpp**

Zum Test dürfen Sie nur die **Basisimplementierung** mitbringen. Speichern Sie sich also gegebenenfalls einen Zwischenstand bevor Sie die Erweiterung für den Zusatzpunkt implementieren. **Abgabe der Basisimplementierung bis 28.11.21**

## 1 License

Die Klasse `License` hat folgende **Instanzvariablen**.

**string name** Name der ausstellenden Gilde.

**unsigned int salary** Lohn einer Aufgabe.

**unsigned int counter** Anzahl der bereits getätigten Aufgaben.

Die Klasse `License` hat folgende **Konstrukturen und Methoden**.

**License(string name, unsigned int salary)** Setzt Instanzvariablen. Name darf nicht leer und salary muss grösser als 0 sein. Sollte ein Parameter nicht den vorgegebenen Werten entsprechen, ist eine Exception vom Typ `runtime_error` zu werfen.

**string get\_guildname() const** Liefert den Namen der ausstellenden Gilde.

**unsigned int get\_salary() const** Liefert den Lohn einer Aufgabe.

**bool valid() const** Liefert `true`, falls maximal 3 Aufgaben erst getätigt wurden, ansonsten `false`.

**bool use()** Liefert `false`, falls die Lizenz nicht mehr gültig ist. Ansonsten wird die Anzahl der getätigten Aufgaben um eins erhöht und `true` zurückgeliefert.

**ostream& print(ostream& o) const**

Die Klasse `License` hat folgendes **Ausgabeformat**.

**ostream& print(ostream& o) const** Gibt das Objekt auf den ostream `o` aus. Format: [License for `name`, Salary: `salary`, Used: `counter`]

**operator<<** `License`-Objekte sollen zusätzlich über `operator<<` ausgegeben werden können. Der operator ist global zu überladen.

Beispiel: [License for City watch, Salary: 5, Used: 0]

## 2 Guild

Die Klasse `Guild` hat folgende **Instanzvariablen**.

**string name** Name der Gilde. (Zur Vereinfachung darf davon ausgegangen werden, dass die Namen der `Guild`-Objekte eindeutig sind.)

**unsigned int fee** Lizenzgebuehr.

**unsigned int salary** Gehalt pro Aufgabe

**map<string,shared\_ptr<Person>> members** Map von GildeMitglieder.

Die Klasse `Guild` hat folgende **Konstrukturen und Methoden**.

**Guild(string name, unsigned fee, unsigned sal, const vector<shared\_ptr<Person>>& players = {})**

Setzt Instanzvariablen, wobei `name` nicht leer sein darf, `fee` und `salary` groesser als 0 sein muessen und keine Person mehr als 1-mal vorkommt in `players`. Sollte ein Parameter nicht den vorgegebenen Werten entsprechen, ist eine Exception vom Typ `runtime_error` zu werfen.

**bool add\_member(shared\_ptr<Person> p)** Fuegt Person `p` der Gilde hinzu, falls nicht schon bereits vorhanden. Liefert `true` bei Erfolg, ansonsten `false`.

**bool remove\_member(string n)** Entfernt Person mit Namen `n` aus der Gilde. Liefert `true` bei Erfolg, ansonsten `false`.

**void grant\_license(string n)** Verkauft GildeMitglied eine Lizenz der Gilde, sofern sich Person mit Namen `n` eine Lizenz leisten kann (`fee`) und ein Mitglied der Gilde ist. Das GildeMitglied empfaengt dann die Lizenz. Sollte die Person mit Namen `n` kein GildeMitglied sein, oder sich die Gebuehr nicht leisten koennen, ist eine Exception vom Typ `runtime_error` zu werfen.

**bool offer\_job(shared\_ptr<Person> p) const** Bietet eine Aufgabe Person `p` an, welche diese auch gleich abarbeitet (`work`). Ist `p` Mitglied der Gilde, arbeitet die Person fuer das doppelte Gehalt der Gilde. Eine Lizenz ist fuer Mitglieder nicht notwendig. **Nicht** GildeMitglieder mit Lizenz, benutzen diese und arbeiten fuer das Gehalt der Lizenz. Alle anderen Personen koennen die Aufgabe nicht bekommen bzw. abarbeiten und die Methode liefert `false`. Ansonsten liefert die Methode `true`.

**ostream& print(ostream& o) const**

Die Klasse `Guild` hat folgendes **Ausgabeformat**.

**ostream& print(ostream& o) const** Gibt das Objekt auf den ostream `o` aus.

Format: [`name`, License fee: `fee`, Job salary: `salary`, {`member_name0`, `member_name1`, ..., `member_namen`}]

**operator<<** `Guild`-Objekte sollen zusätzlich über `operator<<` ausgegeben werden können. Der `operator` ist global zu überladen.

Beispiel: [City watch, License fee: 6, Job salary: 5, {Carrot Ironfoundersson, Samuel Vimes}]

### 3 Person

Die Klasse `Person` hat folgende **Instanzvariablen**.

**string name** Name der Person. (Zur Vereinfachung darf davon ausgegangen werden, dass die Namen der Person-Objekte eindeutig sind.)

**unsigned int wealth**

**map<string,unique\_ptr<License>> licenses** Map von aktuellen Lizenzen.

Die Klasse `Person` hat folgende **Konstrukturen und Methoden**.

**Person(string name, unsigned int wealth=0)** Setzt Instanzvariablen, wobei `name` nicht leer sein darf. Sollte ein Parameter nicht den vorgegebenen Werten entsprechen, ist eine Exception vom Typ `runtime_error` zu werfen.

**virtual ~Person() = default;**

**void work(string guild)** Falls eine Lizenz fuer `guild` vorhanden ist und man diese noch benutzen kann, wird die Lizenz benutzt und die Person arbeitet fuer das auf der Lizenz gedruckte Gehalt. Sollte keine Lizenz vorhanden sein, oder nicht mehr benutzbar, ist eine Exception vom Typ `runtime_error` zu werfen.

**virtual void work(unsigned int) = 0**

**void increase\_wealth(unsigned int i)** Erhoeht `wealth` um `i`.

**string get\_name() const** Liefert `name`.

**bool pay\_fee(unsigned int i)** Sollte das Vermoegen der Person nicht ausreichen um die Gebuehr `i` zu bezahlen oder `i` 0 sein, ist `false` zu liefern. Ansonsten wird das Vermoegen reduziert um `i` und `true` returniert.

**void receive\_license(unique\_ptr<License> l)** Speichert die Lizenz `l` in der Lizenzen der Person. Sollte es bereits eine Lizenz mit dem gleichen Namen geben, ist die alte Lizenz zu zerstören.

**void transfer\_license(string l,shared\_ptr<Person> p )** Uebergibt die Lizenz mit dem Namen `l` vom `this`-Objekt an Person `p` und entfernt diese aus den Lizenzen vom `this`-Objekt. Sollte die Person (=this-Objekt) keine Lizenz mit dem Namen `l` besitzen, ist eine Exception vom Typ `runtime_error` zu werfen.

**bool eligible(string l) const** Liefert **true** falls eine Lizenz mit Namen **l** vorhanden ist, ansonsten **false**.

**virtual ostream& print(ostream& o) const**

Die Klasse **Person** hat folgendes **Ausgabeformat**.

**ostream& print(ostream& o) const** Gibt das Objekt auf den ostream **o** aus.

Format: **name, wealth** Coins,  $\{license\_0, license\_1, \dots, license\_n\}$

**operator<<** **Person**-, **Superworker**- und **Worker**-Objekte sollen zusätzlich über **operator<<** ausgegeben werden können. Der operator ist global zu überladen.

Beispiel: Josiah Herbert Boggis, 190 Coins, {[License for City watch, Salary: 5, Used: 0], [License for Thieves guild, Salary: 4, Used: 0]}

Von der Klasse **Person** werden folgende Klassen abgeleitet.

### 3.1 Worker

**Worker** ist eine arbeitende Person.

**Worker(string, unsigned int=0)** Setzt Instanzvariablen durch Konstruktor der Basisklasse.

**void work(unsigned int i)** Erhoeht **wealth** um **i**.

**ostream& print(ostream& o) const** Gibt das Objekt auf den ostream **o** aus.

Format: [Worker **Person::print(o)**

### 3.2 Superworker

**Superworker** ist ein **Person**, welche eine Gebuehr fuer jede Arbeit verrechnet.

**Superworker(unsigned int fee, string, unsigned int=0)** Setzt Instanzvariablen durch Konstruktor der Basisklasse. **fee** ist eine zusaetzliche Instanzvariable in der **Superworker**-Klasse.

**void work(unsigned int i)** Erhoeht **wealth** um **i+fee**.

**ostream& print(ostream& o) const** Gibt das Objekt auf den ostream **o** aus.

Format: [Superworker: **Person::print(o)**

## 4 Zusatzaufgabe

Erweitern Sie die Klasse `Guild` um eine Logbuch an abgearbeiteten Auftraegen, also ein Objekt durch welches ersichtlich ist welche Personen am meisten Geld erwirtschaftet haben, sei es als Gildenmitglied oder als Lizenzarbeiter. Speichern Sie hierzu `weak_ptr` auf die Personen-Objekte und erweitern und aendern Sie bei Bedarf die Klassen.

Es muss moeglich sein, sich jederzeit die Person, welche das meiste Geld erwirtschaftet hat, einer Gilde auszugeben falls eine existiert. Sollte also die Person mit dem meisten erwirtschafteten Geld geloescht werden, sollte automatisch die naechst beste Person geliefert werden.