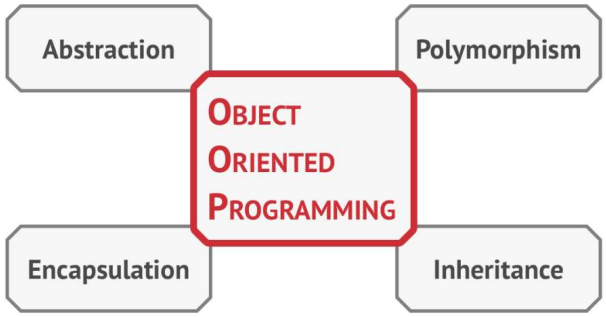
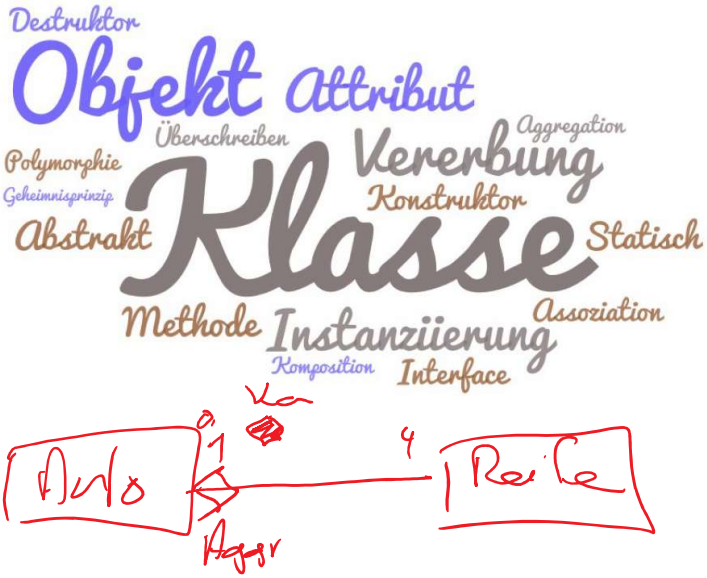
Objektorientierte Programmierung

oop

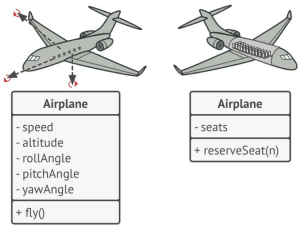
OOP beruht auf den 4 Prinzipien

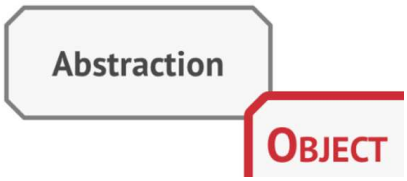




Prinzipien

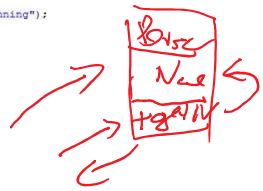
OOP beschränkt sich auf die relevanten Elemente eines Objektes
Und bildet daraus dann die entsprechenden Klassen (abstrahieren != abstract)

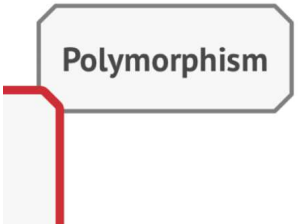


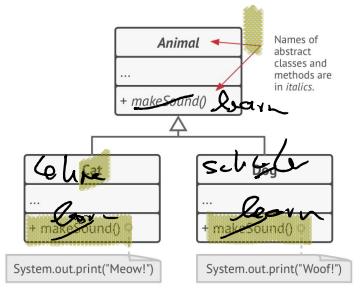



```
public class Dog extends Animal {

    private int numberOfLegs;
    private boolean hasOwner;

    public Dog() {
        numberOfLegs = 4;
        hasOwner = false;
    }

    public void makeDogBark() {
        Dog d = new Dog();
        d.bark();
    }

    private void bark() {
        System.out.println("Woof!");
    }

    public void move() {
        System.out.println("Running");
    }
}
```

Klassen / Objekte versperren den direkten
Zugriff auf ihre Attribute bzw auf manche Methoden
(private)

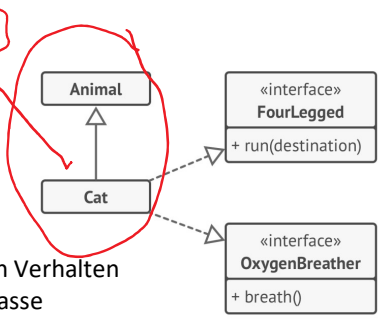






Die abstrakte Klasse Animal beschreibt
Eine Schnittstelle(interface), über die sich
Unterklassen definieren können.

Unterklassen können Implementierungen
Von Methoden der Oberklasse reimplementieren.
Das Verhalten von verschiedenen Typen von Unterklassen
kann deshalb verschieden sein.



Vererbung

Unterklassen übernehmen Verhalten
bzw. Attribute der Oberklasse

Eine Klasse kann häufig
nur von einer Oberklasse erben,
Sie kann aber viele Interfaces implementieren

Klasse - Objekt - Instanziierung

Sonntag, 5. Januar 2020 15:04

Klasse / Objekt / Instanziierung

Die **Klasse** ist der grundlegende Begriff in der OOP.

Ausgehend von der jeweiligen Problemstellung versucht man, die zusammengehörenden Informationen in einem gemeinsamen Container, der Klasse eben, zu speichern.

Die Klasse erhält einen Namen, die zu speichernden Informationen bezeichnet man als **Attribute** der Klasse.

Die Manipulation, d.h. das Schreiben, Lesen und Ändern der Daten, obliegt ebenfalls dem Verantwortungsbereich der Klasse. Neben diesen Aufgaben kann eine Klasse noch weitere Fähigkeiten besitzen. Diese Fähigkeiten werden in der Klasse durch **Methoden** definiert.

Eine Klasse besteht deshalb zumindest aus 3 Bereichen.

- Dem Klassennamen
- Der Liste der Attribute
- Der Liste der Methoden

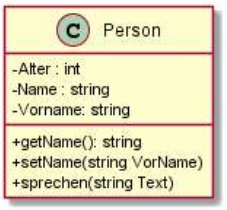
Ein typisches Beispiel für eine Klasse könnte wie folgt aussehen:

```php
<?php

class Person{
    private $Alter
    private $Name
    private $Vorname

    public function getName():string
    {
        return $this->Name
    }

    public function setName(string $VorName)
    {
        $this->Name = $VorName
    }

    public function sprechen(string $Text)
    {
        // HAs to be done
    }
}
```

Klasse

Container von

Attributen und Methoden



Neben den wirklichen Fähigkeiten, im obigen Beispiel **sprechen**, muss eine Klasse häufig über Verwaltungsmethoden verfügen, die eine Manipulation der internen Attribute ermöglichen. Man nennt diese Methode häufig **getter/setter**-Methoden. Da ein Methodenname lediglich einmal genutzt werden darf, muss man deshalb häufig 2 Methodennamen benutzen.

getter, setter

Verwaltungsmethoden

Konstruktor

Innerhalb der Klasse gibt es häufig eine spezielle Methode, den sog. **Konstruktor**. Sie hat in vielen Programmiersprachen den gleichen Namen wie die Klasse selbst und sie kann in verschiedenen Variationen vorliegen. PHP benutzt dafür die Funktion **__construct()**. Beim Erzeugen eines Objektes (s.u.) wird diese Methode als Erstes aufgerufen. Wird diese Methode nicht innerhalb der Klasse definiert, so wird ein sog. **Standardkonstruktor** benutzt. Er besteht lediglich aus dem Methodennamen ohne irgendwelche Parameter. Werden eigene Konstrukoren geschrieben, so muss der Standardkonstruktor explizit definiert werden, sonst ist er nicht mehr vorhanden.

Konstruktor

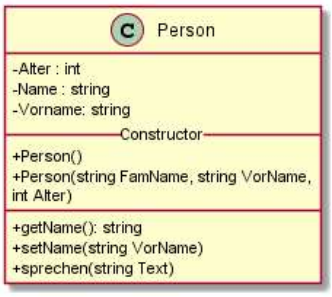
Methode, die beim Erzeugen eines Objektes als Erstes aufgerufen wird

Bemerkung

PHP kennt keine Mehrfachkonstruktoren. Man kann die damit verbundenen Absichten aber mimiken. Siehe dazu

- https://stackoverflow.com/questions/1699796/best-way-to-do-multiple-constructors-in-php
- https://www.kerstner.at/2015/03/overloading-constructors-and-functions-in-php/
- http://verraes.net/2014/06/named-constructors-in-php/

```php
<?php

class Person{
    private $Alter;
    private $Name;
    private $Vorname;

    public function __construct (){

    }

    public function __construct (string $FamName, string $Vorname, $Alter)
    {
        $this->Alter = $Alter;
        $this->Vorname = $Vorname;
        $this->Name = $FamName;
    }
```



```php
    public function getName():string
    {
        return $this->Name;
    }

    public function setName(string $VorName)
    {
        $this->Name = $VorName;
    }

    public function sprechen(string $Text){
        // Has to be done
    }
}
```

Hide

Instanziierung

Instanziierung

Erzeugen eines Objektes

Die Klasse definiert als Bauplan lediglich die Art der gehaltenen Informationen bzw. die Methoden, die zur Manipulation der Daten bzw. zur Funktionalität der Klasse notwendig sind. Doch wie kann man nun diese Klasse nutzen? Da wir ja Informationen zu einem realen Gegenstand des Systems erheben wollen, müssen wir den Bauplan der Klasse einem realen Objekt zuordnen. Nur ein reales Objekt kann Daten speichern. Dieser Vorgang wird **Instanziierung** genannt und läuft in folgenden Schritten ab:

- Deklaration einer Variablen
- Instanziierung des Objektes mit Hilfe des new-Operators in Verbindung mit dem Klassennamen.

```php
<?php

$karl = new Person()
$gerd = new Person("Gerd", "Sych", 76)
echo($gerd->Alter);
echo($karl->Vorname);
```

Hide

Der new-Operator erzeugt ein sog. **Objekt der Klasse**. Dieses Objekt ist einzigartig und ist nun in der Lage, die Informationen aufzunehmen und zu verarbeiten. Bei allen Instanziierungen wird immer ein **Konstruktor** aufgerufen. Er kann in vielen Programmiersprachen überladen sein, d.h. in verschiedenen Versionen existieren. Durch ihn ist das Objekt in der Lage, Zustände seiner Variablen bei der Erzeugung zu kontrollieren. Definiert man eigene Konstruktoren, so muss der parameterlose Standardkonstruktor ebenfalls angegeben werden, wenn man ihn zur Verfügung stellen will.

Erst nach dem Erzeugen kann man nun die Fähigkeiten des Objektes benutzen, d.h. man kann die Methoden der Klasse benutzen.

Bemerkung

Methoden werden auf Klassenebene definiert, aber auf Objektebene genutzt !

(Es gibt aber eine Ausnahme ! Welche ?)

Destruktor

Parallel zum Konstruktor gibt es den sog. **Destruktor**. Er wird durch den folgenden Aufruf beschrieben:

```php
<?php

~Klassenname()
PHP: function __destruct()
```

Das Zerstören eines Objektes wird durch das NULL-Setzen der Objektreferenz bewirkt. `$lehrer = null;`
Der tatsächliche Zeitpunkt des Zerstörens eines Objektes hängt von mehreren Faktoren ab

- Anzahl der noch gültigen Referenzen
- Tatsächliches Freigeben des Speichers durch die Garbage Collection

Es kann deshalb nicht genau vordefiniert werden, ob und wann durch das Löschen einer Objektreferenz die Destruktor-Methode aufgerufen wird.

Siehe dazu auch: https://stackoverflow.com/questions/2777942/php-destructor-behaviour

Statische Attribute/Methoden

Sonntag, 5. Januar 2020 15:09

Statische Attribute/Methoden

Um in OOP arbeiten zu können, sind offenbar immer zunächst Objektinstanziierungen notwendig. Dies ist aber manchmal lästig, weil wir eigentlich nur eine Funktionalität brauchen bzw. redundante Informationen speichern wollen, die für alle gleich sind.

Statisch

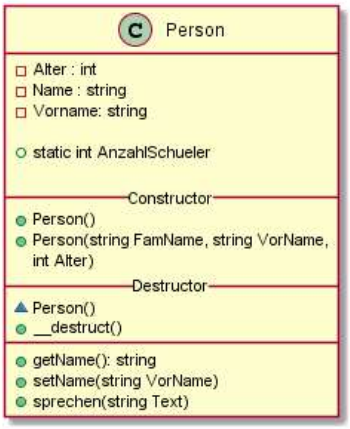
Zugriff ohne Objektinstanz

So könnte es z.B. sein, dass Schüler einer Klasse die Anzahl der Mitschüler in ihrer Klasse kennen sollen. Bei 32 Schülern würde dies bedeuten, dass 32 Objekte die gleiche Information in einer lokalen Instanzvariable halten müssten! Noch schlimmer, bei Änderungen der Schülerzahl durch Hinzukommen / Weggehen neuer Schüler müsste in allen Objekten dieser Wert geändert werden !

Für solche Fälle kennt die OOP die Möglichkeit sog. **statischer** Attribute bzw. Methoden. Diese werden bei der Klasse gehalten und durch das Schlüsselwort **static** deklariert.

Der Zugriff auf diese Werte ist sowohl über jedes Objekt als auch über die Klasse an sich möglich.

Die Aktualisierung solcher Informationen beim Zerstören solcher Objekte kann aber problematisch werden. Folgendes Beispiel verdeutlicht die Situation:




```php
<?php

    class Person
    {
        private $Alter;
        private $Name;
        private $Vorname;
        static $AnzahlSchueler=0;

        public function __construct1(){

        }

        public function __construct(string $FamName, $Vorname, int $Alter)
        {
            $this->Alter = $Alter;
            $this->Vorname = $Vorname;
            $this->Name = $FamName;

            self::$AnzahlSchueler++;
        }

        public function getName():string
        {
            return $this->Name;
        }

      public function sprechen(string $Text)
      {
          echo( $Text);
      }

  }

  $Karl = new Person("Steinam", "Karl", 20);
  echo(Person::$AnzahlSchueler);
  echo($Karl::$AnzahlSchueler);
  $Gerd = new Person("Sych", "Gerd", 32);
  echo(Person::$AnzahlSchueler);
  Person::$AnzahlSchueler = 32;
  echo($Gerd::$AnzahlSchueler);
  $Gerd::$AnzahlSchueler = 4;
  echo(Person::$AnzahlSchueler);
  echo($Karl::$AnzahlSchueler);
```

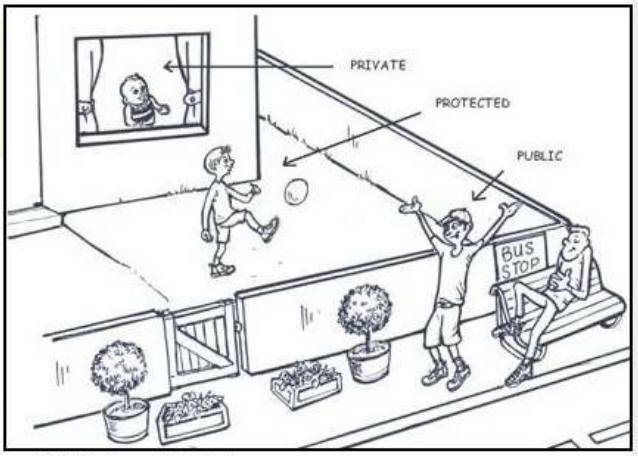
Hide

Geheimisprinzip

Geheimnisprinzip

https://stackoverflow.com/questions/4361553/what-is-the-difference-between-public-private-and-protected

Beachte https://stackoverflow.com/questions/4361553/what-is-the-difference-between-public-private-and-protected



So wie Objekte im realen Leben auch nicht jedes Geheimnis nach draußen preisgeben, so gilt dies auch in der OOP. Das Objekt sollte prinzipiell den Zustand seiner Attribute versteckt halten, d.h. keinen direkten Zugriff auf seine Attribute erlauben.

Durch das Bereitstellen von entsprechenden getter/setter-Methoden bzw. Properties kann das Objekt den Zugriff auf seine Attribute kontrollieren.

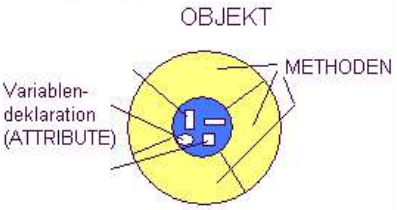
Der generelle Zugriffsmöglichkeit auf Attribute und Methoden wird in der OOP über die **Sichtbarkeiten** definiert. Diese stehen vor dem Attribut bzw. der Methode und definieren den möglichen Zugriff von außerhalb auf die Attribute und Methoden. Falls keine Aussage im Quellcode getroffen wird, gelten die default-Sichtbarkeiten der jeweiligen Programmiersprachen, die sich durchaus unterscheiden können.

Gängige Definitionen von Sichtbarkeiten sind:

- **public**: Öffentlich sichtbar, von überall aufruf- und damit manipulierbar
- **private**: Nur innerhalb des Objektes selbst benutzbar
- **protected**/internal: Nur innerhalb des gleichen Namespaces bzw. über Vererbungsmechanismen sichtbar.

In den Bildern sind die Variablen des Objektes im Zentrum, umschlossen von den Methoden. Dies deutet grafisch an, daß Variablen gegen Zugriffe von außen geschützt sind, d.h. Programme können nicht direkt auf die Variablen des Objektes zugreifen, son-dern müssen die entsprechenden Methoden auf-rufen. Dieser Schutz wird Kapselung genannt. Mit Kapselung werden oft Implementationsdetails versteckt.

Ein Objekt hat eine nach außen definierte Schnittstelle, mit der andere Objekte kommunizieren können. Das Objekt kann die privaten Informationen und Methoden ändern, ohne daß andere Objekte betroffen sind. Zudem kann ein Objekt problemlos herumgereicht werden.

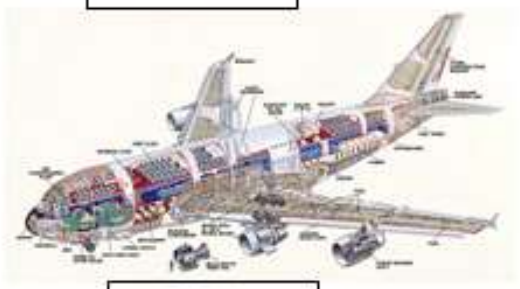


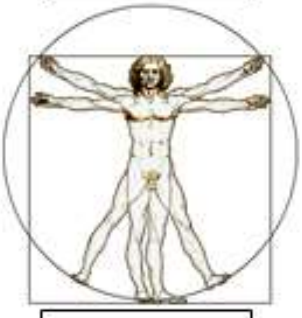
Aufgaben

Sonntag, 5. Januar 2020 15:17

A1

Weisen Sie den untenstehenden Bildern jeweils die Begriffe Klasse bzw. Objekt zu





Aufgabe: Überlegen Sie sich, welche Klassen es in Ihrer beliebten Berufsschule gibt.
Finden Sie auch einige Attribute und Methoden

A2 Fahrzeugleihe

Arbeitsauftrag

- Erstellen Sie zwei Klassen (Auto, Mitarbeiter) zur Verwaltung eines Fuhrparks in einem Unternehmen

- Speichern Sie sich für das Auto die Attribute (Hersteller, Modell, Leistung, Verbrauch, Baujahr, KilometerStand)
- Die Klasse Auto verfügt über eine Methode fahren(). Diese prüft, ob die Türen geschlossen sind. Falls ja, wird der Motor gestartet, falls nein erhält der Nutzer einen Hinweis.
- Mitarbeiter können sich Fahrzeuge für eine Dienstreise ausleihen. Bei der Rückgabe müssen Sie die gefahrenen Kilometer sowie den Kilometerstand des Autos an den Chef melden.
- Erstellen Sie zunächst ein Klassendiagramm
- Implementieren Sie anschließend den Sachverhalt mit Hilfe von PHP
- Der Mitarbeiter Brenner leiht sich für eine Dienstreise nach Hamburg (700 km einfach) den BMW mit dem Kennzeichen WÜ-MA-777 aus. Der Kilometerstand ist am Ende der Fahrt im Auto zu setzen.
- **Überlegen Sie, wie Herr Brenner mit dem Objekt Auto kommunizieren kann.**

A3 HighScoreListe

Von klassischen Computerspielen kennen wir vielleicht noch die sog. Highscore-Liste. Diese stellen die höchsten in dem entsprechenden Spiel erzielten Punkte in absteigender Form dar.

Eine Highscore-Liste für ein Computerspiel verwaltet mehrere Einträge, die jeweils durch einen Spitznamen des Spielers repräsentiert sind. Die Highscore-Liste verfügt weiterhin über gängige Methoden, die das Hinzufügen von Platzierungen und die Ausgabe der gesamten Liste ermöglichen.

Aufgabenstellung

- Überlege dir ein Klassendiagramm für die beschriebenen Eigenschaften.
- Implementiere die Klasse *HighScoreEntry*, mit der wir Einträge für die HighScore-Liste erzeugen können. Die Klasse soll eine geeignete *toString()*-Methode besitzen.
- Befülle die HighScore-Liste mit entsprechenden Daten. Eine Sortierung nach Punkten muss nur für die Ausgabe erreicht werden; die interne Implementierung bleibt Ihnen überlassen.
- Aufgrund der Verwendung von PHP sollte eine Persistierung per JSON erfolgen.
- Die Liste soll maximal 10 Einträge aufnehmen. Neue Einträge mit höheren Punktzahlen müssen dementsprechend kleine Einträge entfernen.

A4 Kochrezeptverwaltung

Wir wollen eine Software für eine Koch-Website entwickeln. Das Kundengespräch liefert folgende Informtionen:
„Ein Rezept hat einen Namen sowie eine Zubereitungszeit in Minuten und besteht aus einer Menge von Anweisungen. Jede Anweisung beschreibt dabei an welcher Stelle was zu tun ist. Zudem verfügt eine Rezept über eine Menge von Zutaten. Jede Zutat gibt an, was für das Rezept benötigt wird und wie viel davon benötigt wird. Für Letzteres muss zudem die jeweilige Einheit vermerkt sein. Ein Kochrezeot enthält spezische Anweisungen zum Einstellen de Herds. Ein Backrept enthält eine spezifische Anweisung in Bezug auf das Einstellen des Backofens.“

- Erstellen Sie aus den obenstehenden Angaben ein Klassendiagramm
- Programmiere anschließend eine Testmethode, die die Einzelheiten des untenstehenden Testfalles als Objekt instanziiert und als echo()-Anweisung ausgibt.

Testfall
Pfannkuchen (30 Minuten)
Zutaten
- 80 g Kartoffelmehl
- 80 g Maisstärke
- 3 Eier
- 400 ml Milch
- 5 EL Traubenzucker
- 5 EL Pflanzenöl

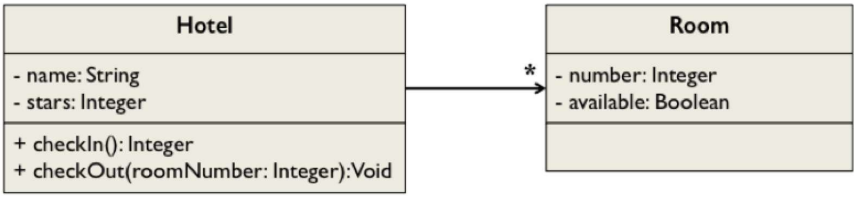
Zubereitung
1. Die Mehle vermischen und sieben

2. Eier, Zucker und Milch zugeben
3. Alles mit dem Schneebesen gut verquirlen
4. 10 Minuten quellen lassen
5. Noch einmal verrühren
6. Pfanne auf hoher Stufe erhitzen und Teig portionsweise in heißem Öl ausbacken

A5 Hotelzimmer

Beschreibung

Zur Verwaltung der Zimmerbelegung in einem Hotel wollen wir ein Programm schreiben.
Hierfür liegt uns folgende Klassenstruktur vor:



Getter- und Setter-Methoden sind nicht aufgelistet, werden aber in den entsprechenden Klassen erwartet. Für die Methoden in den *Hotel*-Klassen müssen wir zudem Folgendes berücksichtigen:

- Die Methode *checkIn()* gibt die Zimmernummern eines freien Zimmers zurück. Sind keine freien Zimmer verfügbar, liefert der Methodenaufruf 0 zurück.
- Mit der Methode *checkOut()* geben wir ein belegtes Zimmer wieder frei. Bitte darauf achten, dass nur belegte Zimmer freigegeben werden können.

```
Hotel hotel = new Hotel("Seeblick", 4, rooms);

println(hotel.checkIn());
println(hotel.checkIn());
println(hotel.checkIn());
hotel.checkOut(102);
println(hotel.checkIn());
println(hotel.checkIn());
```

sollte diese Ausgabe in der Konsole erzeugen:

```
101
102
103
102
201
```

Highscore

Mittwoch, 8. Januar 2020 08:16

A3 HighScoreListe

Von klassischen Computerspielen kennen wir vielleicht noch die sog. Highscore-Liste. Diese stellen die höchsten in dem entsprechenden Spiel erzielten Punkte in absteigender Form dar.

Eine Highscore-Liste für ein Computerspiel verwaltet mehrere Einträge, die jeweils durch einen Spitznamen des Spielers repräsentiert sind. Die Highscore-Liste verfügt weiterhin über gängige Methoden, die das Hinzufügen von Platzierungen und die Ausgabe der gesamten Liste ermöglichen.

Aufgabenstellung

- Überlege dir ein Klassendiagramm für die beschriebenen Eigenschaften.
- Implementiere die entsprechende Klasse, mit der wir Einträge für die HighScore-Liste erzeugen können. Die Klasse soll eine geeignete *toString()*-Methode besitzen.
- Befülle die HighScore-Liste mit entsprechenden Daten. Eine Sortierung nach Punkten muss nur für die Ausgabe erreicht werden; die interne Implementierung bleibt Ihnen überlassen.
- Aufgrund der Verwendung von PHP sollte eine Persistierung per JSON erfolgen.
- Die Liste soll maximal 10 Einträge aufnehmen. Neue Einträge mit höheren Punktzahlen müssen dementsprechend kleinere Einträge entfernen.