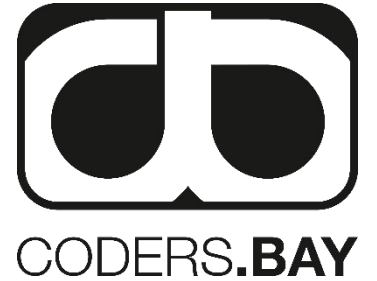


OBJEKTORIENTIERTE PROGRAMMIERUNG

OOP

INHALT



- public, static, final
- OOP Paradigmen
- Objekt
- Klasse
- Attribute (Variablen)

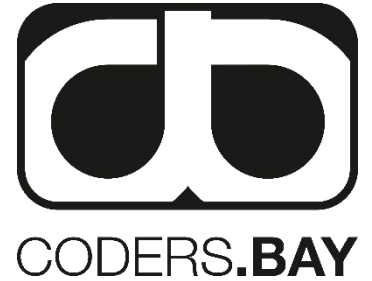
- Methoden
- Scope
- Konstruktor(en)
- Ausblick

GRUNDLAGEN OOP



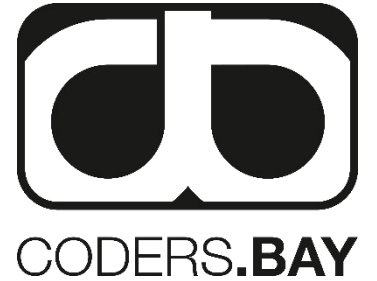
CODERS.BAY

PUBLIC, STATIC, FINAL, ...



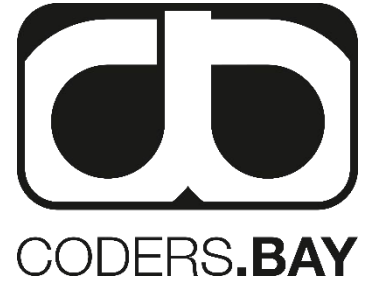
- Schlüsselwörter
- `public` → öffentlicher Zugriff auf Variable / Methode
- `static` → statische Variable / Methode → aufrufbar ohne vorher ein Objekt zu erzeugen
- `final` → Konstanten, Wert nicht veränderbar
- `private` → „private“ Variable / Methode, nicht außerhalb sichtbar

OOP PARADIGMEN



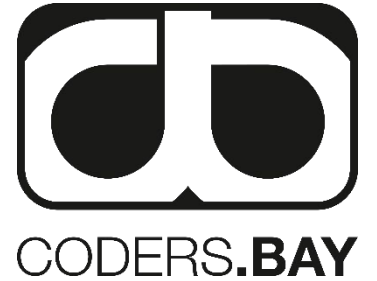
- alles ist ein Objekt → alle Klassen haben als Basisklasse `Object`
- Objekte kommunizieren durch das Senden und Empfangen von Nachrichten
 - welche aus Objekten bestehen
- Objekte haben ihren eigenen Speicher
- jedes Objekt ist die Instanz einer Klasse
 - welche ein Objekt sein muss
- die Klassen beinhalten das Verhalten aller ihrer Instanzen

OBJEKT



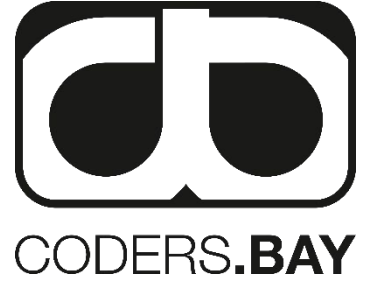
- kapselt Variablen und Methoden zu einer Einheit
- Eigenschaften:
 - Verhalten
 - Identität
- (Software)Objekt simuliert und abstrahiert reales Objekt

KLASSE



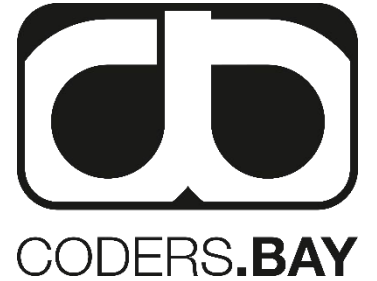
- beschreibt Struktur der Objekte
- Konstruktoren zur Initialisierung der Objekte
- jedes Objekt ist Instanz genau einer Klasse
- hat eigene Methoden
- unendlich viele Klassen

KLASSE



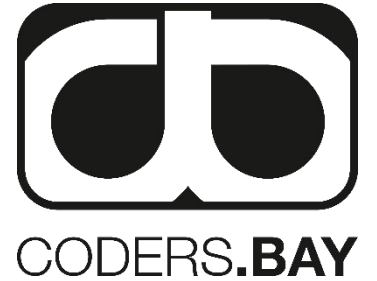
```
public class Hauptprogramm {  
    public static void main(String[] args){  
        System.out.println("Hello World");  
    }  
}
```


KLASSE



- Jede JAVA-Anwendung besitzt eine `main()` Methode
- `main()` → Einstiegspunkt
- `public` → weil von außerhalb der Klasse auf sie zugegriffen wird
- `static` → aufrufbar ohne vorher ein Objekt zu erzeugen
- `void` → weil sie keinen Rückgabewert besitzt
- `String[] args` oder `String args[]` → Array, das die vom Aufruf entgegen genommenen Kommandozeilenparameter enthält

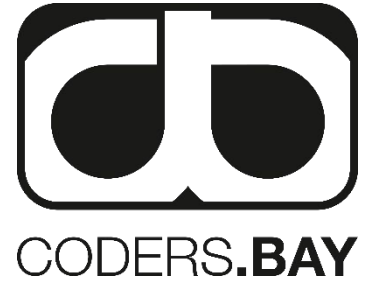
KLASSE




- `class Person` → Klasse ohne Hauptprogramm
- wird von Klasse mit Hauptprogramm aufgerufen
- neuer Typ namens Person
- beschreibt das Object → Person

```
public class Person {  
    ...  
}
```

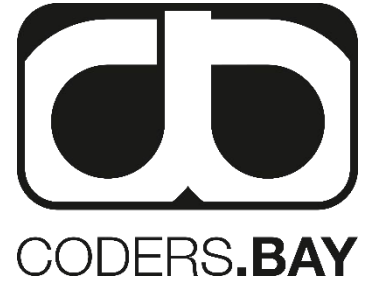
KLASSE



- eine Date, int, String, Person ... Variable enthält eine Referenz auf ein Date, int, String, Person ... Objekt
- Klasse = Typ → ihre Werte = Objekte
- **new** → erzeugt neues Objekt vom Typ int, String ...
- ```
Person p1 = new Person();
```

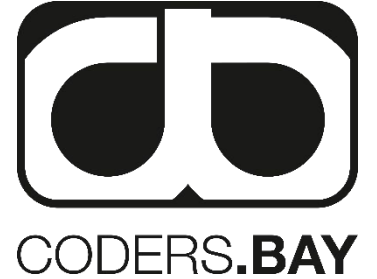

- p1 = Objekt vom Typ Person !!!

# ATTRIBUTE



- aka Variablen, alias Eigenschaften oder Felder
- eine Variable ist ein Objekt von einem gewissen Typ
  - `int`
  - `String`
  - `double`
  - ...

# ATTRIBUTE



Eigenschaften der Klasse werden in Felder / Attribute gespeichert

Konstanten sind mit dem  
Schlüsselwort `final` gekennzeichnet

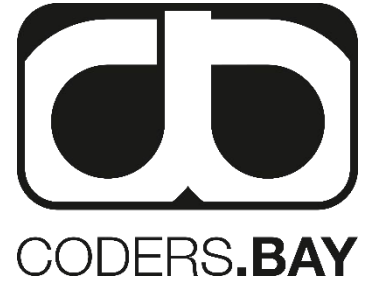
```
public class Hauptprogramm {

 /*Klassenvariablen*/
 static String lastname = "Simon";
 static String firstname = "Haidinger";
 static String adresse = "Teststraße 1, 4312 Ried in der Riedmark";
 static int age = 30;
 static String companyAdress;

 /*Konstanten*/
 static final String CB_FIRMA = "CODERS.BAY";
 static final String CB_ADRESSE = "Peter-Behrens-Platz 6";
 static final int CB_PLZ = 4020;
 static final String CB_ORT = "Linz";
 static final String CB_EMAIL = "info@codersbay.at";

 public static void main(String[] args) {
 ...
 }
}
```

# METHODEN



- Programme > um als einziges Stück geschrieben zu werden
- Zerlegung in kleinere Anweisungsfolgen
  - logisch zusammengehören
- beliebig oft aufrufbar → Wiederverwendbarkeit
- sinnvolle Namen
  - Englisch & Deutsch nicht mischen
  - beginnen mit Kleinbuchstaben

# METHODEN



main() Methode

Methoden der Klasse

```
public class Hauptprogramm {

 /*Klassenvariablen*/
 static String lastname = "Simon";
 static String firstname = "Haidinger";
 static String adresse = "Teststraße 1, 4312 Ried in der Riedmark";
 static int age = 30;
 static String companyAdress;

 /*Konstanten*/
 static final String CB_FIRMA = "CODERS.BAY";
 static final String CB_ADRESSE = "Peter-Behrens-Platz 6";
 static final int CB_PLZ = 4020;
 static final String CB_ORT = "Linz";
 static final String CB_EMAIL = "info@codersbay.at";

 public static void main(String[] args) {
 System.out.println("##### START MAIN #####");
 printPersonData();
 companyAdress = getCompanyAddress();
 setAge();
 printPersonData();
 System.out.println("##### END MAIN #####");
 }

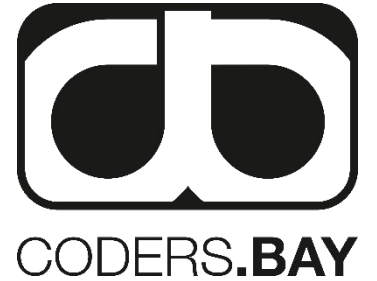
 static void printPersonData()
 {
 System.out.println("Nachname: " + lastname);
 System.out.println("Vorname: " + firstname);
 System.out.println("Adresse: " + adresse);
 System.out.println("Alter: " + age);
 System.out.println("\n");
 System.out.println(companyAdress);
 System.out.println("\n");
 }

 static String getCompanyAddress()
 {
 return CB_FIRMA + "\n" + CB_ADRESSE + "\n" + CB_PLZ + "\n" + CB_ORT + "\n" + CB_EMAIL + "\n";
 }

 static void setAge() {
 age = age + 1;
 }

}
```

# METHODEN



- einfachste Form → parameterlos
  - `printPersonData()`
- Parameter = Werte die an Methode übergeben werden
  - `sumValues(int x, int y)`
- ein und denselben Code für verschiedene Kontexte



# METHODEN ÜBERLADEN



main() Methode {

überladene Methoden {

```
public class Hauptprogramm {

 /*Klassenvariablen*/
 static String lastname = "Simon";
 static String firstname = "Haidinger";
 static String adresse = "Teststraße 1, 4312 Ried in der Riedmark";
 static int age = 30;

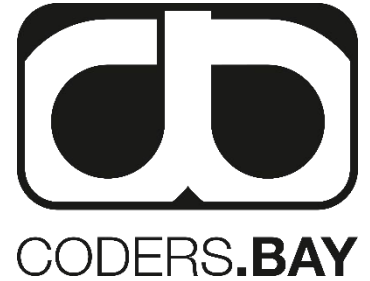
 public static void main(String[] args) {
 printPersonData();
 printPersonData("Streng");
 printPersonData("Monika", "Mustermaier");
 }

 static void printPersonData()
 {
 System.out.println("Nachname: " + lastname);
 System.out.println("Vorname: " + firstname);
 System.out.println("Adresse: " + adresse);
 System.out.println("Alter: " + age);
 System.out.println("\n");
 }

 static void printPersonData(String lastname)
 {
 System.out.println("Nachname: " + lastname);
 System.out.println("\n");
 }

 static void printPersonData(String lastname, String firstname)
 {
 System.out.println("Vollständiger Name: " + lastname + " " + firstname);
 System.out.println("\n");
 }
}
```

# FUNKTIONEN VS. PROZEDUREN



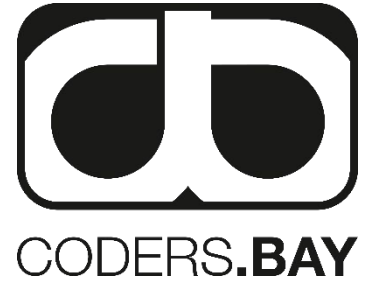
- Methoden die Wert zurückliefern nennt man Funktionen

```
static String getCompanyAddress()
{
 return CB_FIRMA + "\n" + CB_ADRESSE + "\n" + CB_PLZ + "\n" + CB_ORT + "\n" + CB_EMAIL + "\n";
}
```

- Methoden die keinen Wert zurückliefern nennt man Prozeduren

```
public void printCompanyAddress()
{
 System.out.println("CODERS.BAY");
 System.out.println("Peter-Behrens-Platz 6");
 System.out.println(4020);
 System.out.println("Linz");
 System.out.println("info@codersbay.at");
}
```

# SCOPE



- Deklarierte Variablen innerhalb einer Methode sind nur in eben dieser LOKAL verfügbar
- Deklarierte Variablen außerhalb einer Methode sind GLOBAL verfügbar
- Lokale Variablen „leben“ nur während der Ausführung der Methode

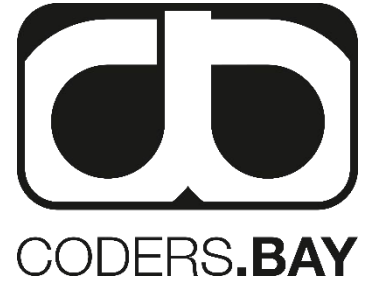
• **#WTF ???**

```
class Program {

 static void add(int x)
 {
 int sum = 0;
 sum = sum + x;
 }

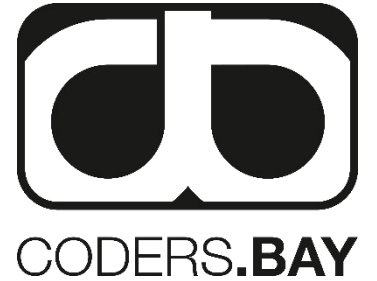
 public static void main (String[] arg)
 {
 add(2);
 add(5);
 add(25);
 System.out.println(sum);
 }
}
```

# KONSTRUKTOR(EN)



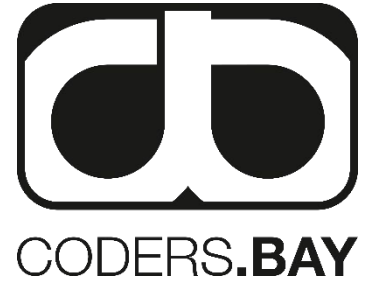
- spezielle Art von Methoden
- gleichen Namen wie die Klasse
- Konstruktor → konstruieren = erzeugen
- via Konstruktor(en) lassen sich Objekte einer Klasse erzeugen

# KONSTRUKTOR(EN)



- eine Klasse kann über viele Konstruktoren verfügen
- sie unterscheiden sich in den Eingabeparametern
- jeder Konstruktor hat einen eindeutigen Satz von Eingabeparametern
- Standardkonstruktor = parameterlos

# KONSTRUKTOR(EN)



(default)Konstruktor der Klasse {

```
public class Person
{
 /*Klassenvariablen*/
 public static String lastname;
 public static String firstname;
 public static String adresse;
 public static int age;
 public static String companyAdress;

 /*Konstanten*/
 private static final String CB_FIRMA = "CODERS.BAY";
 private static final String CB_ADRESSE = "Peter-Behrens-Platz 6";
 private static final int CB_PLZ = 4020;
 private static final String CB_ORT = "Linz";
 private static final String CB_EMAIL = "info@codersbay.at";

 Person(){
 lastname = "Max";
 firstname = "Mustermann";
 adresse = "Musterstraße 1";
 age = 20;
 }

 public void printPersonData()
 {
 System.out.println("Nachname: " + lastname);
 System.out.println("Vorname: " + firstname);
 System.out.println("Adresse: " + adresse);
 System.out.println("Alter: " + age);
 System.out.println("\n");
 }
}
```

# OBJEKT ERZEUGEN



```
public class Hauptprogramm {

 public static void main(String[] args) {
 System.out.println("##### START MAIN #####");
 Person p1 = new Person ();
 p1.printPersonData();
 System.out.println("##### END MAIN #####");
 }
}
```

```
public class Person
{
 /*Klassenvariablen*/
 public static String lastname;
 public static String firstname;
 public static String adresse;
 public static int age;
 public static String companyAdress;

 /*Konstanten*/
 private static final String CB_FIRMA = "CODERS.BAY";
 private static final String CB_ADRESSE = "Peter-Behrens-Platz 6";
 private static final int CB_PLZ = 4020;
 private static final String CB_ORT = "Linz";
 private static final String CB_EMAIL = "info@codersbay.at";

 Person(){
 lastname = "Max";
 firstname = "Mustermann";
 adresse = "Musterstraße 1";
 age = 20;
 }

 public void printPersonData()
 {
 System.out.println("Nachname: " + lastname);
 System.out.println("Vorname: " + firstname);
 System.out.println("Adresse: " + adresse);
 System.out.println("Alter: " + age);
 System.out.println("\n");
 }
}
```

# KONSTRUKTOR ÜBERLADEN



• 2 Konstruktoren

```
public class Person
{
 /*Klassenvariablen*/
 public static String lastname;
 public static String firstname;
 public static String adresse;
 public static int age;
 public static String companyAdress;

 /*Konstanten*/
 private static final String CB_FIRMA = "CODERS.BAY";
 private static final String CB_ADRESSE = "Peter-Behrens-Platz 6";
 private static final int CB_PLZ = 4020;
 private static final String CB_ORT = "Linz";
 private static final String CB_EMAIL = "info@codersbay.at";

 Person() {
 lastname = "Max";
 firstname = "Mustermann";
 adresse = "Musterstraße 1";
 age = 20;
 }

 Person (String p_lastname, int p_age)
 {
 lastname = p_lastname;
 age = p_age;
 }

 public void printPersonData()
 {
 System.out.println("Nachname: " + lastname);
 System.out.println("Vorname: " + firstname);
 System.out.println("Adresse: " + adresse);
 System.out.println("Alter: " + age);
 System.out.println("\n");
 }
}
```



# KONSTRUKTOR ÜBERLADEN – OBJEKT ERZEUGEN



```
public class Hauptprogramm {

 public static void main(String[] args) {
 System.out.println("##### START MAIN #####");
 Person p1 = new Person ("Haidinger", 30);
 p1.printPersonData();
 Person p2 = new Person();
 p2.printPersonData();
 System.out.println("##### END MAIN #####");
 }
}
```

```
public class Person
{
 /*Klassenvariablen*/
 public static String lastname;
 public static String firstname;
 public static String adresse;
 public static int age;
 public static String companyAdress;

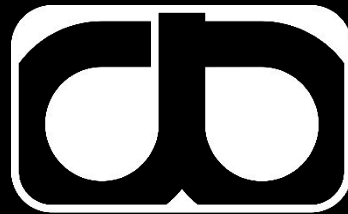
 /*Konstanten*/
 private static final String CB_FIRMA = "CODERS.BAY";
 private static final String CB_ADRESSE = "Peter-Behrens-Platz 6";
 private static final int CB_PLZ = 4020;
 private static final String CB_ORT = "Linz";
 private static final String CB_EMAIL = "info@codersbay.at";

 Person(){
 lastname = "Max";
 firstname = "Mustermann";
 adresse = "Musterstraße 1";
 age = 20;
 }

 Person (String p_lastname, int p_age)
 {
 lastname = p_lastname;
 age = p_age;
 }

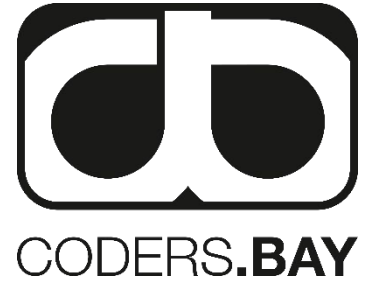
 public void printPersonData()
 {
 System.out.println("Nachname: " + lastname);
 System.out.println("Vorname: " + firstname);
 System.out.println("Adresse: " + adresse);
 System.out.println("Alter: " + age);
 System.out.println("\n");
 }
}
```

**BEISPIEL / AUFGABE**  
**TRY UNTIL ERROR → FIX ERROR :-)**



**CODERS.BAY**

# AUSBLICK



- private, protected, ...
- „getters & setters“
- Vererbung
- Polymorphismus
- Abstrakte Klassen
- Interfaces
- Rekursion

**SYSTEM.OUT.PRINTLN(„VIELEN DANK  
FÜR EURE AUFMERKSAMKEIT“);**



**CODERS.BAY**