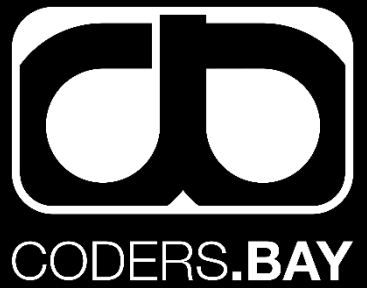
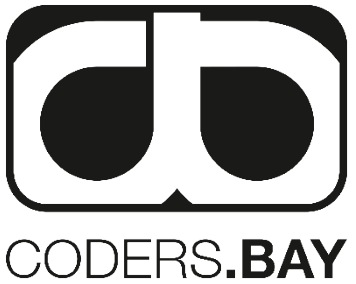


KLASSEN & OBJEKTE

Objekt Orientierung

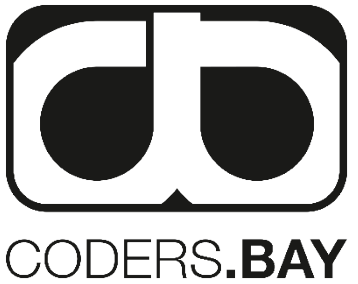
Objektorientierung



Alle Überlegungen die der Programmierung dienen orientieren sich an Objekten.

Objekte können Repräsentationen von realen Dingen sein - ein Haus, ein Tisch, eine Tür, oder etwas nicht physisches - eine E-Mail, ein Spiel, ein Newsletter.

Auf die wichtigen Dinge konzentrieren

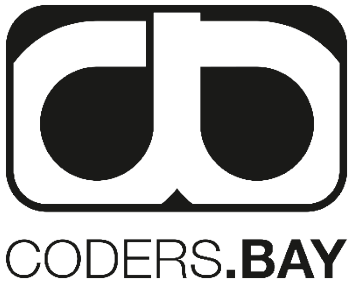


Beim Modellieren/Designen von Objekten ist es wichtig zu wissen welche Aspekte im Moment wichtig für dein Programm sind und sich nicht in Details zu verlieren.

Objekte können Eigenschaften und Fähigkeiten(Methoden) haben und aus Teilen(Attribute) bestehen. Diese Teile können wiederum wieder Objekte sein.

Ein Objekt kapselt Eigenschaften und Fähigkeiten.

Exercise Time!

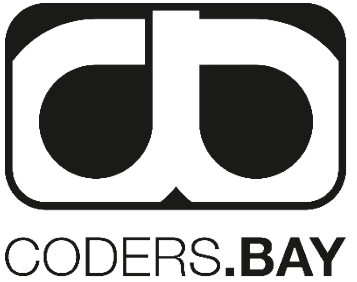


Modellieren wir eine Registrierkasse in einem Supermarkt.

Wir sollen die Möglichkeit haben:

- Produkte zu scannen
- Produkte zu stornieren
- eine Rechnung zu drucken

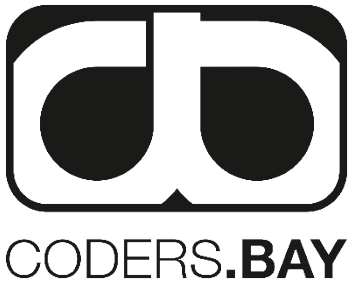
Registrierkassenbeispiel



Invoice
products: Product[] totalAmount: Double
print() balance()
Product
price: Double name: String

CashRegister
products: Product[]
scanProduct(product: Product) removeProduct(product: Product) generateInvoice(): Invoice

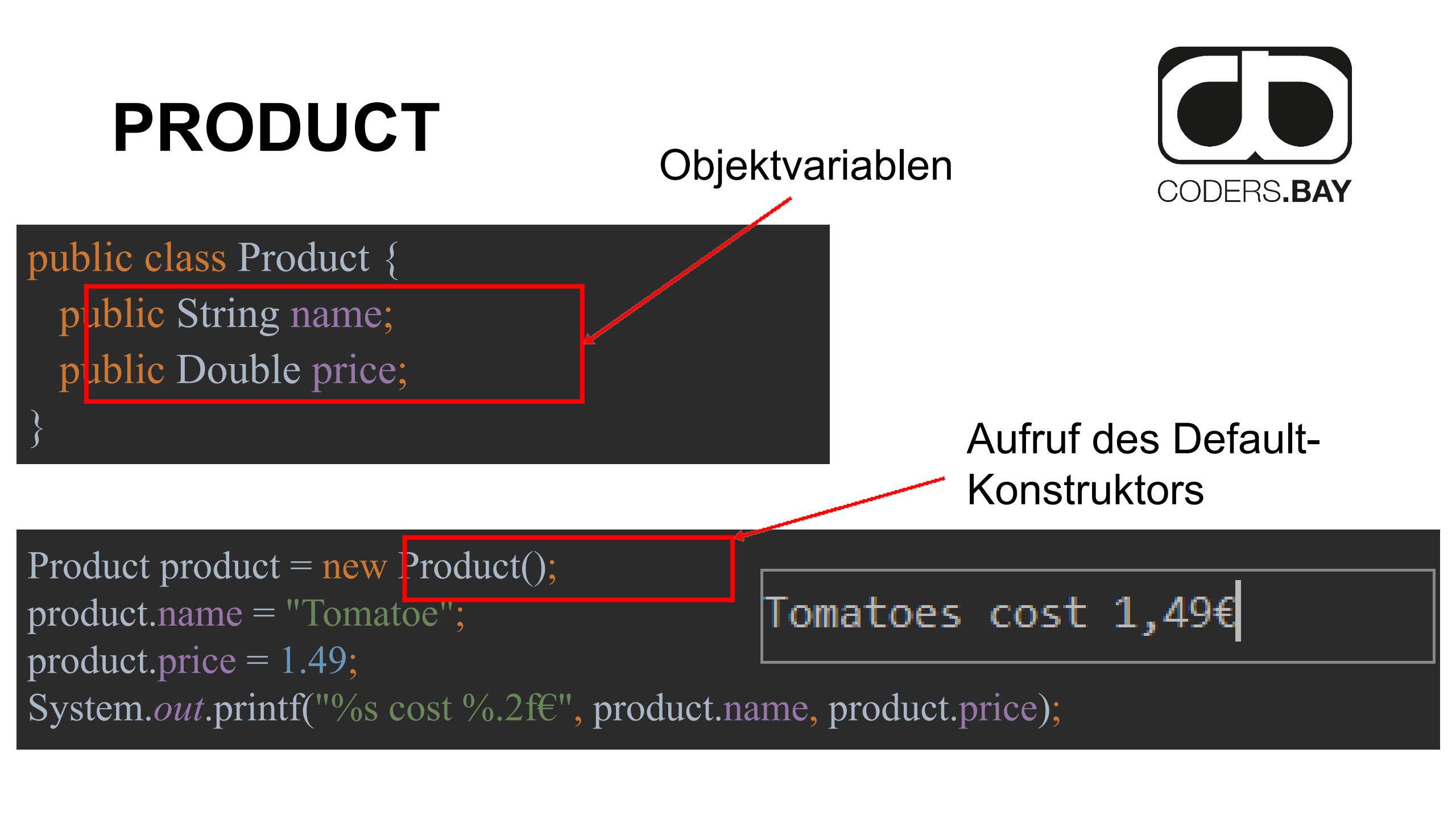
KLASSEN



Klassen sind Baupläne bzw. Schablonen für Objekte, d.h. ein Objekt ist immer eine Instanz einer Klasse.

Objekte werden mit dem **new** Keyword erzeugt.

PRODUCT




```
public class Product {
    public String name;
    public Double price;
}
```

Aufruf des Default-Konstruktors

```
Product product = new Product();
product.name = "Tomatoe";
product.price = 1.49;
System.out.printf("%s cost %.2f€", product.name, product.price);
```

```
Tomatoes cost 1,49€
```

PRODUCT

Konstruktor


```
public class Product {
    public String name;
    public Double price;

    public Product(String name, Double price) {
        this.name = name;
        this.price = price;
```

```
Product product = new Product();
```

```
Product product = new Product("Tomatoes", 1.49);
```

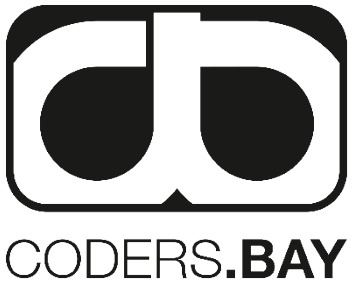
PRODUCT

CODERS.BAY

```
public class Product {
    public String name;
    public Double price;

    public Product(String name, Double price) {
        this.name = name;
        this.price = price;
    }
}
```

Mit **this** wird immer auf die aktuelle Instanz der Klasse, also das Objekt verwiesen. Über **this** kann man explizit auf Objektvariablen und -methoden zugreifen

KONSTRUKTOR



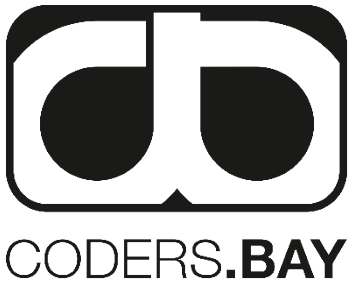
- Wenn nicht anders definiert besitzt jede Klasse einen Default-Konstruktor ohne Parameter.
- Mit dem **new** Keyword wird der Konstruktor aufgerufen und ein Objekt der spezifizierten Klasse erzeugt.
- Es kann mehrere Konstruktoren mit unterschiedlichen Parametern geben - siehe *method overloading.*

Im IntelliJ kannst du dir Konstruktoren durch Drücken von **Alt+Einfg** generieren lassen

INVOICE

```
public class Invoice {
   public Product[] products;
   public Invoice(Product[] products) {
       this.products= products;
   }
   void print() {
   }
   void balance() {
   }
}
```

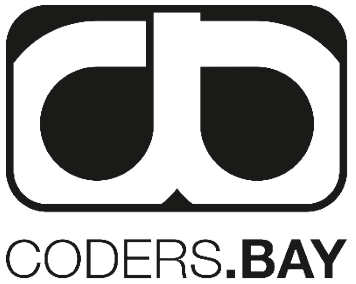
Klassen- vs. Objektmethoden



Klassenattribute bzw. -methoden sind direkt auf der Klasse verfügbar - d.h. es wird keine Instanz der Klasse gebraucht.
Sie sind mit dem **static** Keyword gekennzeichnet.

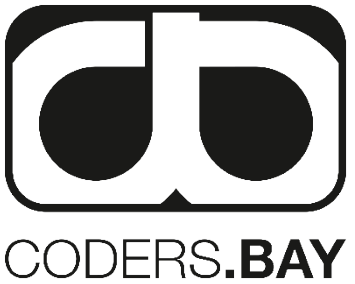
Objektattribute bzw. -methoden hingegen erfordern ein Objekt, eine Instanz einer Klasse.

CASH REGISTER



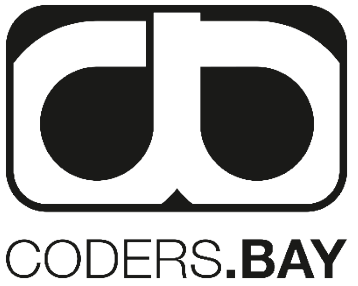

```
public class CashRegister {
    public Product[] products;
    public void scanProduct(Product product) {
    }
    public void removeProduct(Product product) {
    }
    public Invoice generateInvoice() {
        return new Invoice(this.products);
    }
}
```

Beispiel: Person mit Zähler



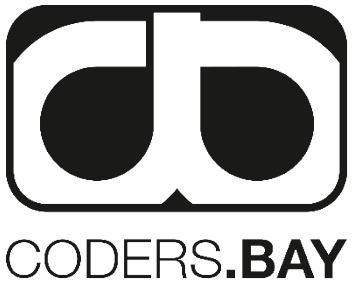

```
public class Person {
   public static int PERSON_COUNT = 0;
   public String name;

   public Person(String name) {
       this.name = name;
              PERSON_COUNT++;
   }
}
```

Beispiel: Person mit Zähler




```
Person stefan = new Person("Stefan");
System.out.printf("Welcome %s, we now have %d persons!\n", stefan.name, Person.PERSON_COUNT);
Person ines = new Person("Ines");
System.out.printf("Welcome %s, we now have %d persons!\n", ines.name, Person.PERSON_COUNT);
Person becca = new Person("becca");
System.out.printf("Welcome %s, we now have %d persons!\n", becca.name, Person.PERSON_COUNT);
Person alex = new Person("alex");
System.out.printf("Welcome %s, we now have %d persons!\n", alex.name, Person.PERSON_COUNT);
```

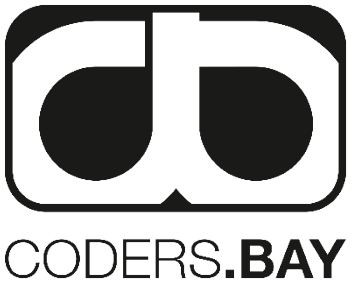
Beispiel: Person mit Zähler




```
Person stefan = new Person("Stefan");
System.out.printf("Welcome %s, we now have %d persons!\n", stefan.name, stefan.PERSON_COUNT);
Person ines = new Person("Ines");
System.out.printf("Welcome %s, we now have %d persons!\n", ines.name, ines.PERSON_COUNT);
Person becca = new Person("becca");
System.out.printf("Welcome %s, we now have %d persons!\n", becca.name, becca.PERSON_COUNT);
Person alex = new Person("alex");
System.out.printf("Welcome %s, we now have %d persons!\n", alex.name, alex.PERSON_COUNT);
```

```
Welcome Stefan, we now have 1 persons!
Welcome Ines, we now have 2 persons!
Welcome becca, we now have 3 persons!
Welcome alex, we now have 4 persons!
```

Zusammenfassung



- Dinge als Objekt betrachten
- Klassen sind Baupläne für Objekte
- Klassen kapseln bestimmte Aspekte und Fähigkeiten
  - Attribute sind Teile aus denen ein Objekt besteht, z.b besteht ein Fahrrad aus zwei Reifen, einer Lenkstange, …
  - Methoden kapseln Verhalten eines Objekts, z.b Bremse treten oder Klingel bedienen
- Objekte werden durch Aufruf des Konstruktors erzeugt
- Mit dem **static** modifier unterscheidet man zwischen Klassen- und Objektvariablen/-methoden