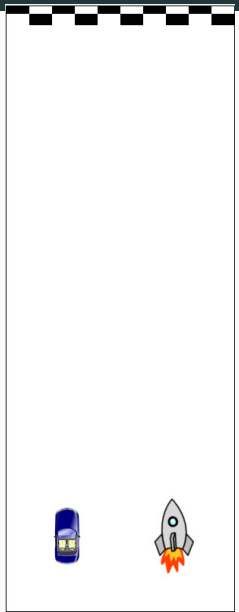
Rennbahn



Arbeitsauftrag - Modellierung 5 min

Erstelle eine Klassendiagramm (mehrere Klassenkarten) zu allen Klassen, die du beim Rennen entdecken kannst.

Greenfoot und die geheimnisvollen Knöpfe

Act

Ruft die `act`-Methode jedes Actors einmal auf.

Run

Startet eine Wiederholug und ruft in jedem Schritt die `act`-Methode jedes Actors auf.

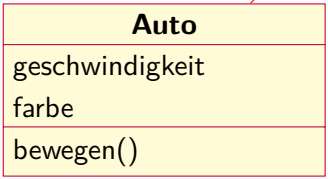
Von der Klassenkarte zum Code

Auto
geschwindigkeit farbe
bewegen()

```java
public class Auto { // Klassendefinition
  // Attribute
  int geschwindigkeit;
  String farbe;

  // Methoden
  public void bewegen() {
    ...
  }
}
```

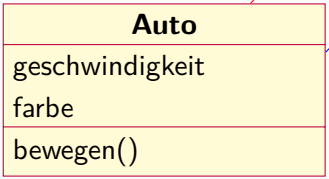
Von der Klassenkarte zum Code




```java
public class Auto { // Klassendefinition
  // Attribute
  int geschwindigkeit;
  String farbe;

  // Methoden
  public void bewegen() {
    ...
  }
}
```

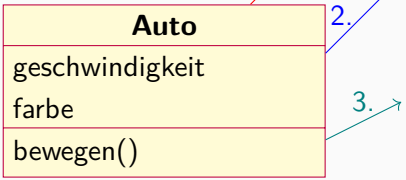
Von der Klassenkarte zum Code




```
public class Auto { // Klassendefinition
  // Attribute
  int geschwindigkeit;
  String farbe;

  // Methoden
  public void bewegen() {
    ...
  }
}
```

Von der Klassenkarte zum Code




```
public class Auto { // Klassendefinition
  // Attribute
  int geschwindigkeit;
  String farbe;

  // Methoden
  public void bewegen() {
    ...
  }
}
```

Von der Klasse zum Objekt

Auto
geschwindigkeit farbe
bewegen()

$\Rightarrow$

Von der Klasse zum Objekt

Von der Klasse zum Objekt

Wo kommen die Attributwerte her?

Von der Klasse zum Objekt

Wo kommen die Attributwerte her?

Sie müssen beim Erzeugen eines Objekts angegeben werden.

Objekte erzeugen

Objekte einer Klasse werden mit dem Schlüsselwort **`new`** erzeugt.

Objekte erzeugen

Objekte einer Klasse werden mit dem Schlüsselwort `new` erzeugt.

Der Konstruktor

- **Methode** der Klasse, die beim Erzeugen ein neues Objekt aufgerufen wird.

Objekte erzeugen

Objekte einer Klasse werden mit dem Schlüsselwort **`new`** erzeugt.

Der Konstruktor

- **Methode** der Klasse, die beim Erzeugen ein neues Objekt aufgerufen wird.
- Heißt immer wie die Klasse.

Objekte erzeugen

Objekte einer Klasse werden mit dem Schlüsselwort `new` erzeugt.

Der Konstruktor

- **Methode** der Klasse, die beim Erzeugen ein neues Objekt aufgerufen wird.
- Heißt immer wie die Klasse.
- Hat keine Angabe eines Rückgabetyps.

Objekte erzeugen

Objekte einer Klasse werden mit dem Schlüsselwort `new` erzeugt.

Der Konstruktor

- **Methode** der Klasse, die beim Erzeugen ein neues Objekt aufgerufen wird.
- Heißt immer wie die Klasse.
- Hat keine Angabe eines Rückgabetyps.

Objekte erzeugen

Objekte einer Klasse werden mit dem Schlüsselwort **new** erzeugt.

Der Konstruktor

- **Methode** der Klasse, die beim Erzeugen ein neues Objekt aufgerufen wird.
- Heißt immer wie die Klasse.
- Hat keine Angabe eines Rückgabetyps.

Auto
geschwindigkeit
farbe
...

```
public class Auto {
int geschwindigkeit; // Attribute
int farbe;
public Auto() { // Konstruktor
  geschwindigkeit = 2;
  farbe = "blau";
}
```

Arbeitsauftrag - Implementiere den Konstruktor der Klasse Auto 10 min

- Kopiere das Greenfoot-Projekt **Rennbahn** aus dem Vorlagen-Laufwerk und öffne es in Greenfoot.
- Ergänze den **Konstruktor** der Klasse `Auto`
  - Der Konstruktor soll die `geschwindigkeit` auf 2 initialisieren
- **Für Schnelle**:
  Erzeuge ein Auto und eine Rakete im Konstruktor der Klasse `Rennbahn`. Positioniere das Auto an Position (90|700) und die Rakete an Position (220|700) (`addObject(object, x, y)`).

Konstruktor

Was wenn die Attributwerte nicht immer gleich sein sollen?

Konstruktor mit Parametern

Was wenn die Attributwerte nicht immer gleich sein sollen?

$\Rightarrow$ Der Konstruktor ist eine Methode, der wir **Parameter** übergeben können.

Konstruktor mit Parametern

Was wenn die Attributwerte nicht immer gleich sein sollen?

⇒ Der Konstruktor ist eine Methode, der wir **Parameter** übergeben können.

```
new Auto(5, "gruen");
```

⇒

Konstruktor mit Parametern

Was wenn die Attributwerte nicht immer gleich sein sollen?

⇒ Der Konstruktor ist eine Methode, der wir **Parameter** übergeben können.

```
new Auto(5, "gruen");
```

⇒

a1: Auto
geschwindigkeit = 5
farbe = "gruen"

Konstruktor mit Parametern

Was wenn die Attributwerte nicht immer gleich sein sollen?

⇒ Der Konstruktor ist eine Methode, der wir **Parameter** übergeben können.

```
new Auto(5, "gruen");
```

⇒

a1: Auto
geschwindigkeit = 5
farbe = "gruen"

```
public class Auto {
    public Auto(PARAMETER) {
      geschwindigkeit = ...;
      farbe = ...;
    }
...
```

Arbeitsauftrag - `Auto` Konstruktor mit Parametern

- Ergänze den Konstruktor der Klasse `Auto`:
  - die Geschwindigkeit soll angegeben werden können
  - die Farbe soll angegeben werden können

- **Für Schnelle:** Implementiere eine neue Klasse `Bus` als Unterklasse von `Actor` mit Geschwindigkeit 3.

Zweimal der gleiche Code?

```
pubic class Rakete {
  int geschwindigkeit = 20;
  public void bewegen() {
    setLocation(getX(),
      getY() - geschwindigkeit);
  }
}
```

```
pubic class Bus {
  int geschwindigkeit = 3;
  public void bewegen() {
    setLocation(getX(),
      getY() + geschwindigkeit);
  }
}
```

Zweimal der gleiche Code?

```
pubic class Rakete {
  int geschwindigkeit = 20;
  public void bewegen() {
    setLocation(getX(),
      getY() - geschwindigkeit);
  }
}
```

```
pubic class Bus {
  int geschwindigkeit = 3;
  public void bewegen() {
    setLocation(getX(),
      getY() + geschwindigkeit);
  }
}
```

Geht das nicht besser?

Vererbung

Eine Klasse kann alle Attribute und Methoden einer anderen Klasse **erben**.

Vererbung

Eine Klasse kann alle Attribute und Methoden einer anderen Klasse **erben**.

Vererbung

Eine Klasse kann alle Attribute und Methoden einer anderen Klasse **erben**.

Man spricht: Die *Unterklasse* erbt von der *Oberklasse*.

Vererbung

Eine Klasse kann alle Attribute und Methoden einer anderen Klasse **erben**.

Man spricht: Die *Unterklasse* erbt von der *Oberklasse*.

Beispiel

*Ein Auto **erbt** von Fahrzeug* bzw. *ein Auto **ist** ein Fahrzeug*.

Vererbung in Java

Um eine Vererbung auszudrücken wird in der Klassendefinition der **Unterklasse** `extends Oberklasse` ergänzt.

Methoden und Attribute der Oberklasse können in der Unterklasse verwendet werden, als ob sie dort definiert sind.

```java
class Oberklasse {
  int zahl;

  Oberklasse() {
    zahl = 13;
  }
}
```

```java
class Unterklasse extends Oberklasse {
  // zahl ist bereits in Oberklasse definiert
 Unterklasse() {
    // Aufruf des Konstruktors der Oberklasse
    super();
    zahl = 12;
  }
}
```

Vererbung - Generalisierung

Rakete
geschwindigkeit
bewegen()

Bus
geschwindigkeit
bewegen()

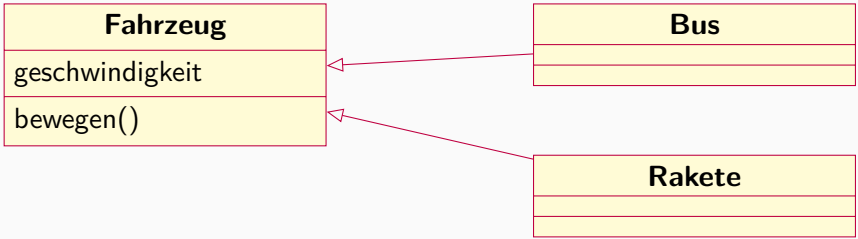
Gemeinsamkeiten mehreren Klassen in eine Oberklasse *auszulagern* nennt man **Generalisierung**.

Vererbung - Generalisierung

Rakete
geschwindigkeit
bewegen()

Bus
geschwindigkeit
bewegen()

Gemeinsamkeiten mehreren Klassen in eine Oberklasse *auszulagern* nennt man **Generalisierung**.



Arbeitsauftrag - Rest der Stunde

Tobt euch aus!

Lasst Autos explodieren!

Baut einen zweiten Spieler ein . . .