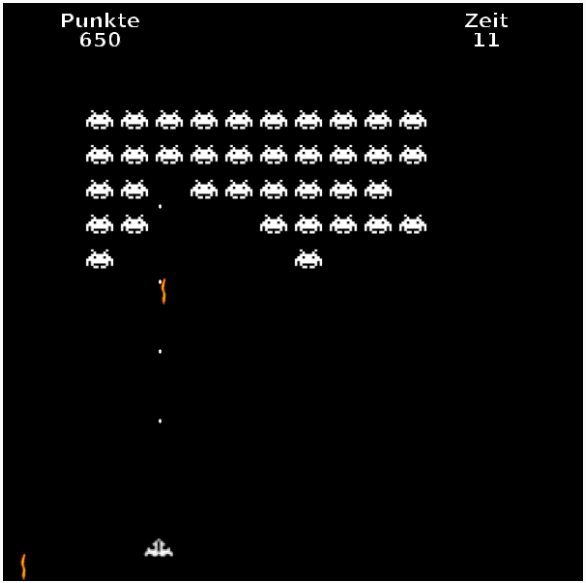
Ziel



Arbeitsauftrag - Modellierung 10 min

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

Exkurs: Wie funktioniert ein Spiel?

- Wer zeichnet das Bild?
- Wer spielt den Ton ab?
- Wer prüft ob eine Taste gedrückt wurde?
- Wer ruft die Methoden unserer Objekte auf und wann?

Exkurs: Wie funktioniert ein Spiel?

- Wer zeichnet das Bild?
- Wer spielt den Ton ab?
- Wer prüft ob eine Taste gedrückt wurde?
- Wer ruft die Methoden unserer Objekte auf und wann?

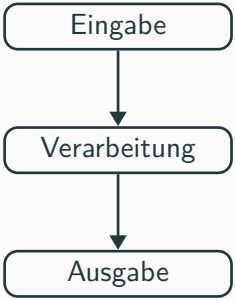
Eine **Engine** und in unserem Fall Greenfoot.

Exkurs: EVA-Prinzip

Exkurs: EVA-Prinzip

Trennung der Datenverarbeitung in:

- **E**ingabe: `input()`
- **V**erarbeitung: `update()`
- **A**usgabe: `output()`

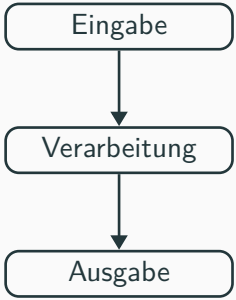


Exkurs: EVA-Prinzip

Trennung der Datenverarbeitung in:

- **E**ingabe: `input()`
- **V**erarbeitung: `update()`
- **A**usgabe: `output()`

Wie oft müssen wir die drei Schritte ausführen?



Exkurs: EVA-Prinzip

Trennung der Datenverarbeitung in:

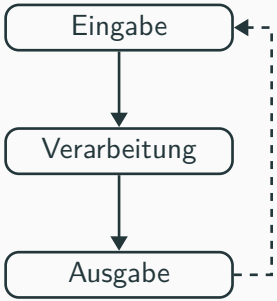
- **E**ingabe: `input()`
- **V**erarbeitung: `update()`
- **A**usgabe: `output()`

Wie oft müssen wir die drei Schritte ausführen?

Solange das Spiel läuft!

EVA-Loop

```
solange Spiel aktiv
 input()
 update()
 output()
```



`Update()` - der spannende Teil

Greenfoot hat zwei wichtige Klassen:

- World - Die Spielwelt
- Actor - Ein Objekt, das Teil der Welt ist und etwas tun kann

Ein Actor

Damit Grennfoot ein Objekt etwas tun lassen kann, muss es ein `Actor` *sein*.

Wie geht das? (Genaueres beim nächsten Mal)

- Die Klassendefinition muss `class Klassenname extends Actor { ... }` enthalten
- Die Klassen muss die Methode`public void act() { ... }` definieren

Greenfoot und die geheimnisvollen Knöpfe

Act

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

Run

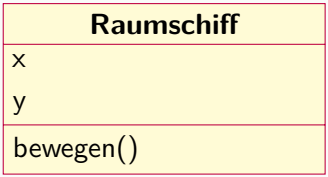
Startet einen **EVA**-Loop und ruft während dem `Update`-Schritt die `act`-Methode jedes Actors auf.

Arbeitsauftrag - Implementiere die Klasse Raumschiff

- Kopiere das Greenfoot-Projekt **SpaceInvaders** aus dem Vorlagen-Laufwerk und öffne es in Greenfoot.
- Bearbeite den Arbeitsauftrag **SpaceInvaders Arbeitsauftrag1.pdf**

Das Projekt SpaceInvadersVorlage2 enthält eine mögliche Lösung für den Arbeitsauftrag 1

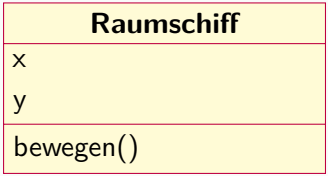
Von der Klassenkarte zum Code




```
public class Raumschiff // Klassendefinition
{ // Attribute
  int x;
  int y;

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

Von der Klasse zum Objekt



$\Rightarrow$

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.

Raumschiff
x y
bewegen()

```
public class Raumschiff {
int x; // Attribute
int y;
public Raumschiff () { // Konstruktor
  x = 300;
  y = 565;
}
```

Konstruktor

Was wenn die Attributwerte nicht immer gleich sein sollen?

Konstruktor mit Parametern

Was wenn die Attributwerte nicht immer gleich sein sollen?

$\Rightarrow$ Wir können dem Konstruktor Parameter übergeben.

Konstruktor mit Parametern

Was wenn die Attributwerte nicht immer gleich sein sollen?

$\Rightarrow$ Wir können dem Konstruktor Parameter übergeben.

```
new Raumschiff(13, 12);
```

$\Rightarrow$

Konstruktor mit Parametern

Was wenn die Attributwerte nicht immer gleich sein sollen?

⇒ Wir können dem Konstruktor Parameter übergeben.

```
new Raumschiff(13, 12);
```

⇒

rs1: Raumschiff
x = 13
y = 12

Arbeitsauftrag - Erzeuge neue Aliens 20min

Bearbeite den Arbeitsauftrag **SpaceInvaders Arbeitsauftrag2.pdf**

Methoden aufrufen

Was wir bereits können:

Methoden des eigenen Objekts aufrufen.

```
bewegen(); // Ruft die Methode bewegen auf dem Objekt selbst auf
```

Was wir neues brauchen:

Methoden anderer Objekte aufrufen.

```
 // Ruft die Methode bewegen auf anderesObject auf
anderesObject.bewegen();
```

Arbeitsauftrag - Schiessen Schiessen Schiessen 15min

Bearbeite den Arbeitsauftrag **SpaceInvaders AA 2.pdf**

Arbeitsauftrag - Schiessen Schiessen Schiessen 15min

Bearbeite den Arbeitsauftrag **SpaceInvaders AA 2.pdf**

Eine eigene Klasse für die Schüsse der Aliens

Die Klasse `Lazer`, soll die Schüsse der Aliens implementieren.

Warum?

- Ein anderes Bild um die Schüsse zu unterscheiden.
- Einfacher um später zu erkennen wer von was getroffen wurde.

Eine eigene Klasse für die Schüsse der Aliens

Die Klasse `Lazer`, soll die Schüsse der Aliens implementieren.

Warum?

- Ein anderes Bild um die Schüsse zu unterscheiden.
- Einfacher um später zu erkennen wer von was getroffen wurde.

💡 Überlegt euch wie ihr die Klasse `Lazer` implementieren würdet.

Zweimal der gleiche Code

```
pubic class Laser {
  int geschwindigkeit = 6;
  public void bewegen() {
    setLocation(getX(),
      getY() - geschwindigkeit);
  }
}
```

```
pubic class Lazer {
  int geschwindigkeit = 5;
  public void bewegen() {
    setLocation(getX(),
      getY() + geschwindigkeit);
  }
}
```

Zweimal der gleiche Code

```
pubic class Laser {
  int geschwindigkeit = 6;
  public void bewegen() {
    setLocation(getX(),
      getY() - geschwindigkeit);
  }
}
```

```
pubic class Lazer {
  int geschwindigkeit = 5;
  public void bewegen() {
    setLocation(getX(),
      getY() + geschwindigkeit);
  }
}
```

Geht dass 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

Laser
geschwindigkeit
act()

Lazer
geschwindigkeit
act()

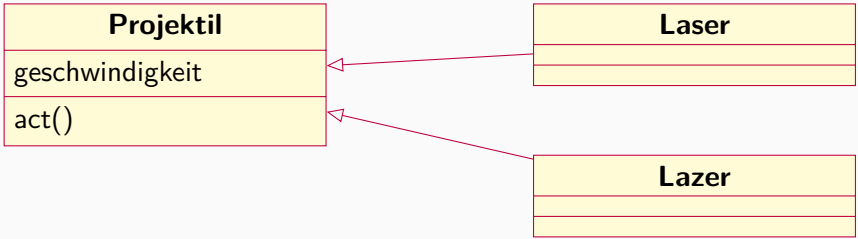
Gemeinsamkeiten mehreren Klassen in eine Oberklasse *auszulagern* nennt man **Generalisierung**.

Vererbung - Generalisierung

Laser
geschwindigkeit
act()

Lazer
geschwindigkeit
act()

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



Arbeitsauftrag - Rest der Stunde

Bearbeite den Arbeitsauftrag **SpaceInvaders Arbeitsauftrag3.pdf**