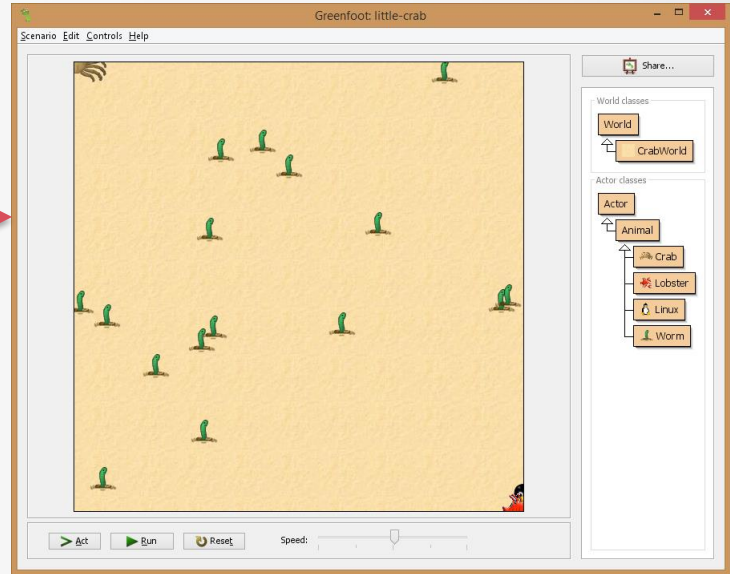
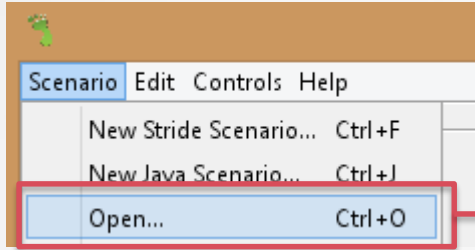


# **Erstes Szenario - LittleCrab**

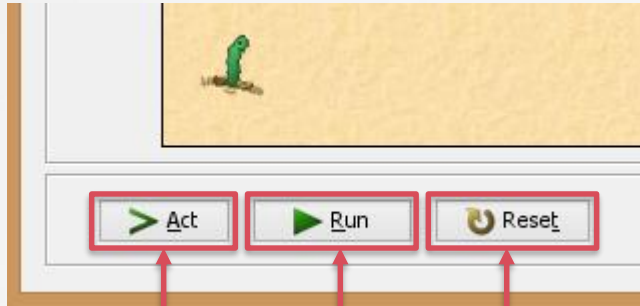


# Öffnen des ersten Szenarios "LittleCrab"



# Basisfunktionen von Greenfoot I

---



Rücksetzen

Starten des Spieles

Einen Handlungsschritt im Spiel weiter

# Basisfunktionen von Greenfoot II

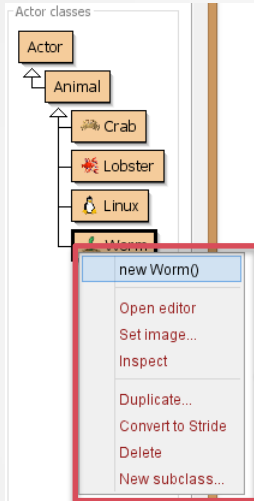
The screenshot shows the 'inherited from World' section of the IDE. It lists several methods: `void act()`, `void addObject(Actor, int, int)`, `GreenfootImage getBackground()`, `int getCellSize()`, `Color getColorAt(int, int)`, `int getHeight()`, `<A> List<A> getObjects(Class<A>)`, `<A> List<A> getObjectsAt(int, int, Class<A>)`, `int getWidth()`, `int numberOfObjects()`, `void removeObject(Actor)`, `void removeObjects(Collection<? extends Actor>)`, `void repaint()`, `void setActOrder(Class ...)`, and `void setBackground(GreenfootImage)`. Below the list is a preview of the World class, which is a grid of yellow cells with green crabs on it. A red box highlights the 'Inspect' button in the IDE interface.

Methoden des  
Objektes der Klasse  
Welt

The screenshot shows the 'crabworld : CrabWorld' object inspector. It displays the following fields and their values: `int cellSize` (1), `int width` (560), `int height` (560), and `private GreenfootImage backgroundIm...` (represented by a circular arrow icon). There are buttons for 'Inspect', 'Get', 'Show static fields', and 'Close'.

# Basisfunktionen von Greenfoot III

---

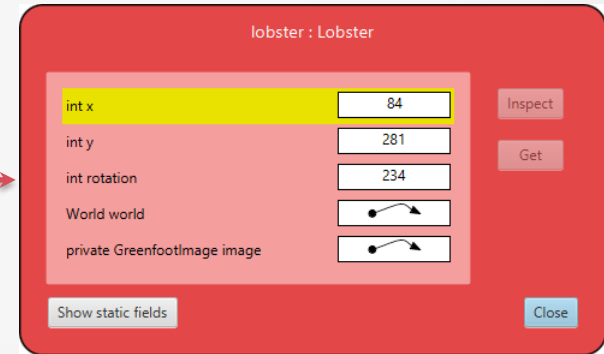
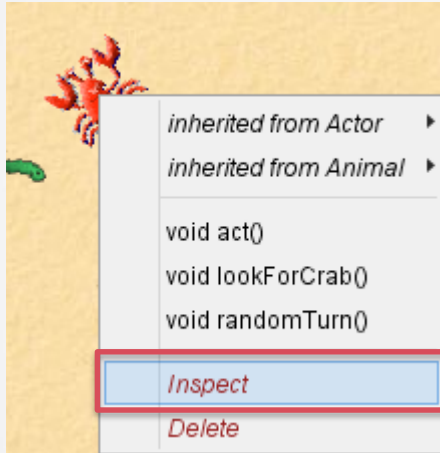


Klick mit der rechten Maustaste bringt ein BlueJ-ähnliches Menü

# Inspizieren der Eigenschaften des Krabbenobjektes I

1.

## Rechte Maustaste



# Inspizieren der Eigenschaften des Krabbenobjektes II

2.

## Mehrmaliges Betätigen der Act-Schaltfläche

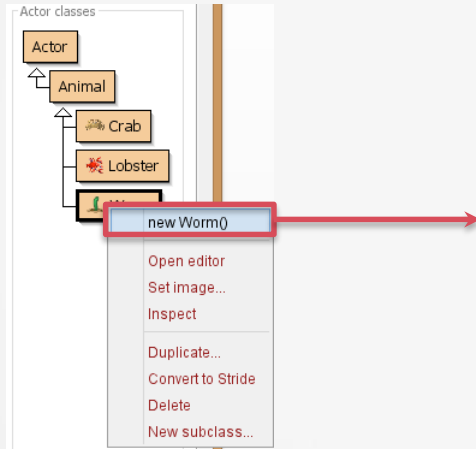


# Zufügen von Objekten

---

1.

Rechte Maustaste auf das Objekt



2.

Das Objekt auf das Spielfeld ziehen





# Speichern von Welten

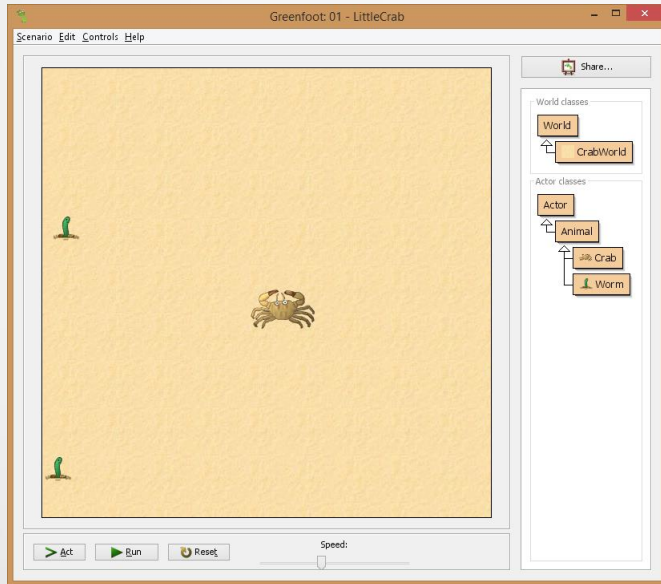
---

## Rechte Maustaste auf das Objekt

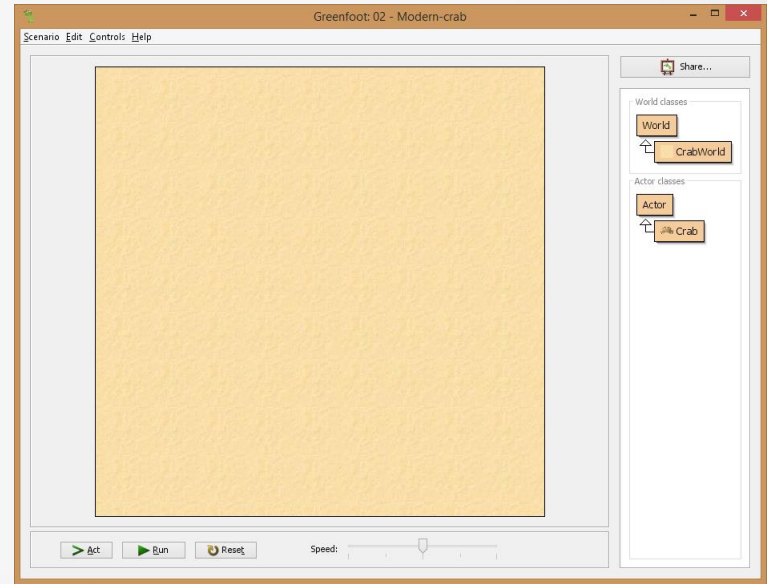


# Beispiele von Welten

Bis jetzt: „Fertige“ Welt: Little Crab



Bis jetzt: „Leere“ Welt: Modern Crab

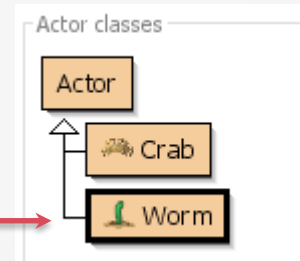
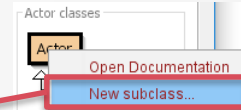
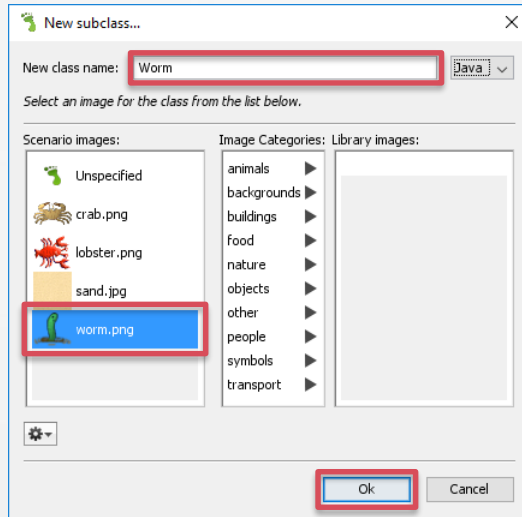


# Neue Kindklasse erstellen

1.

Rechte Maustaste auf das Actor als Elternklasse

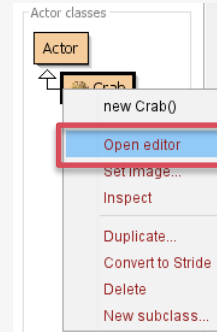
2.



# Bewegen von Objekten

1.

Rechte Maustaste auf das Objekt



2.

```
public void act()
{
  if (Greenfoot.isKeyDown("left"))
    this.move(-4);
}
```

Maustaste „links“ ist gedrückt worden

# Objekte "fressen"

```
public void act() {
```

```
    if (Greenfoot.isKeyDown("left"))  
        this.move(-4);
```

Maustaste „links“

```
    if (Greenfoot.isKeyDown("right"))  
        this.move(4);
```

Maustaste „rechts“

```
    Actor worm;  
    worm=getOneObjectAtOffset(0,0,Worm.class);
```

Neues Wurmobjekt erzeugen & zuweisen wenn es an der Stelle von der Krabbe eines gibt

```
    if (worm!=null)
```

null=leeres Objekt

```
    {
```

```
        World world;  
        world=getWorld();
```

Neues Weltobjekt und aktuelle Welt hineinspeichern

```
        world.removeObject(worm);
```

Wurm aus aktuellem Weltobjekt löschen

```
    }
```

```
}
```



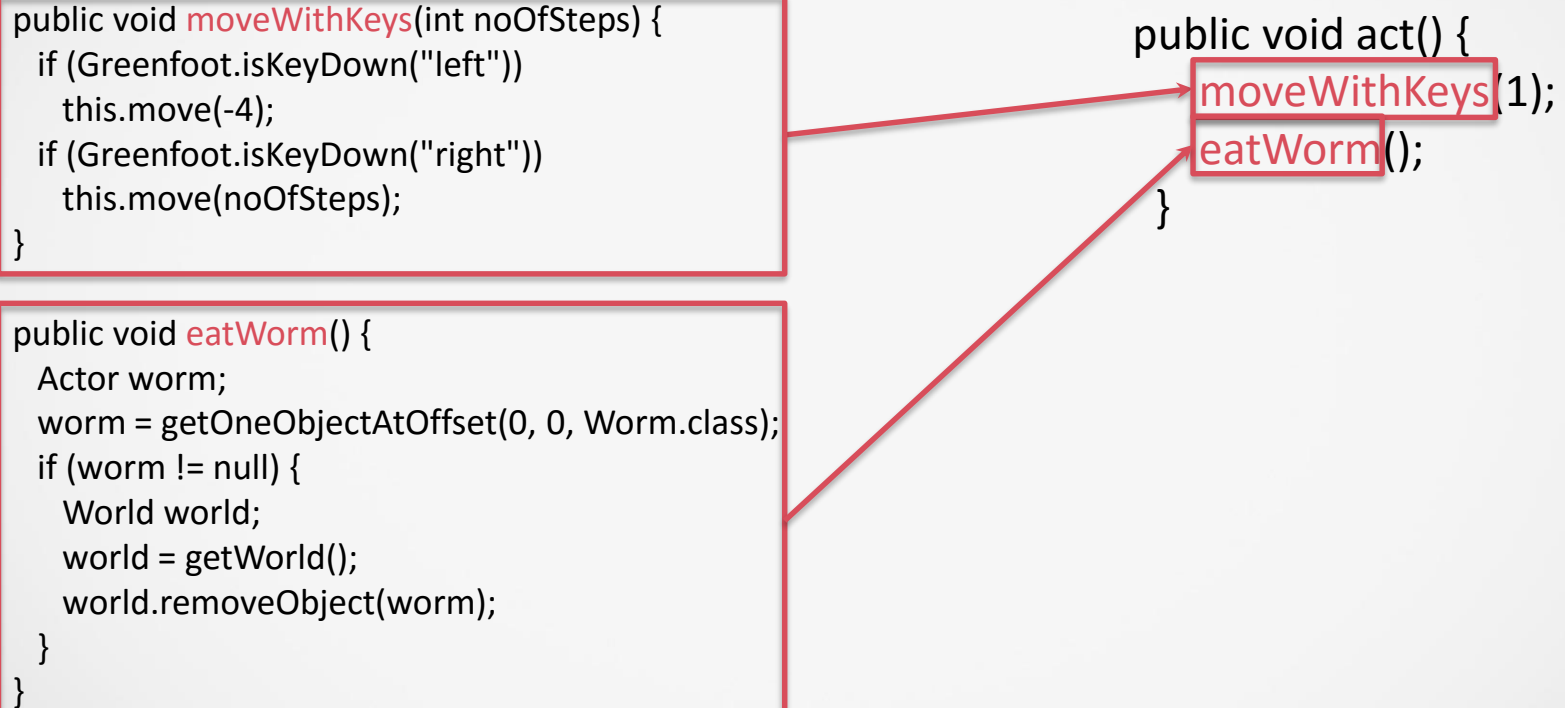
# Quellcode strukturieren

---

```
public void moveWithKeys(int noOfSteps) {  
    if (Greenfoot.isKeyDown("left"))  
        this.move(-4);  
    if (Greenfoot.isKeyDown("right"))  
        this.move(noOfSteps);  
}
```

```
public void eatWorm() {  
    Actor worm;  
    worm = getObjectAtOffset(0, 0, Worm.class);  
    if (worm != null) {  
        World world;  
        world = getWorld();  
        world.removeObject(worm);  
    }  
}
```

```
public void act() {  
    moveWithKeys(1);  
    eatWorm();  
}
```



# Eigenschaften und Methoden von Objekten auflisten – Variante A

A.

## Kontextsensitives Menü der rechten Maustaste



Methoden vom Elternobjekt  
Actor

Eigene(selbst kodierte)  
Methoden

# Eigenschaften und Methoden von Objekten auflisten – Variante B

B.

```
public void test() {  
    Actor worm;  
    worm.  
}
```

**void act ()**

boolean equals(Object)

Class<?> getClass()

Greenfoot... getImage ()

int getRotation()

World getWorld()

W getWorldOfType (Class<W>)

int getX()

int getY()

int hashCode()

boolean isAtEdge ()

**greenfoot.Actor**

void act ()

The act method is called by the greenfoot framework to give actors a chance to perform some action. At each action step in the environment, each object's act method is invoked, in unspecified order.

The default implementation does nothing. This method should be overridden in subclasses to

Objekt deklarieren und nach Punkt „Strg+Leertaste“



## Sounds bei Aktionen

---

Greenfoot.playSound("eating.wav");



# Objekt per Zufall durch Welt laufen lassen

---

```
public class Lobster2 extends Actor {  
    public void act() {  
        moveAround();  
    }  
    public void moveAround() {  
        move(3);  
        if (Greenfoot.getRandomNumber(100) < 10)  
            turn(Greenfoot.getRandomNumber(90) - 45);  
        if (getX() <= 5 || getX() >= getWorld().getWidth() - 5)  
            turn(180);  
        if (getY() <= 5 || getY() >= getWorld().getHeight() - 5)  
            turn(180);  
    }  
}
```



3 Einheiten in die Richtung

Zufallszahl <10: Drehung=10% aller Fälle

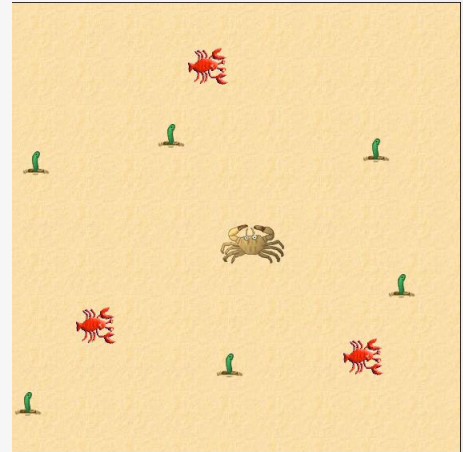
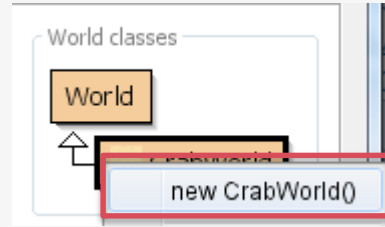
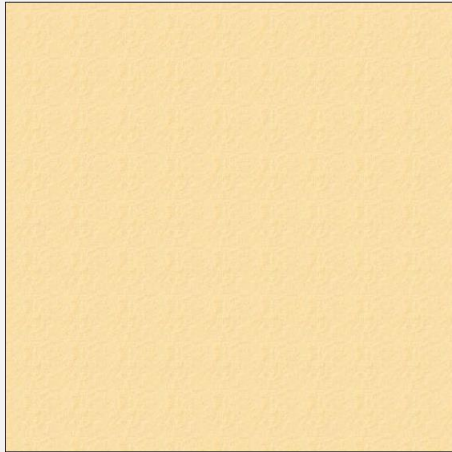
Horizontal „angestossen“

Vertikal „angestossen“



# Erstellen einer neuen Welt I

---



# Erstellen einer neuen Welt II

---

```
public CrabWorld() {  
    super(560, 560, 1);  
    populateWorld();  
    prepare(),  
}  
  
private void populateWorld()  
{  
    addObject(new Crab(), 300, 300);  
    addObject(new Worm(), 20, 500);  
    addObject(new Worm(), 30, 200);  
}
```

