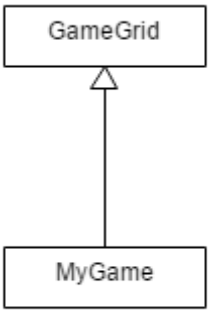
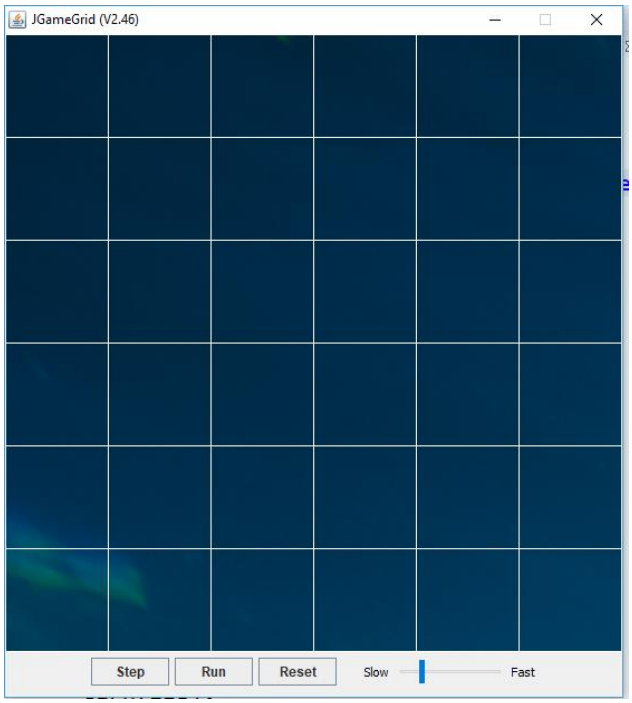
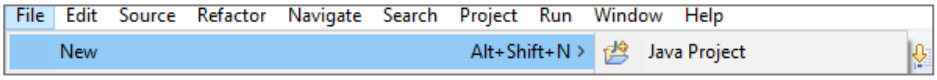
1 - EIN NEUES GAMEGRID ANLEGEN





EIN NEUES GAMEGRID ANLEGEN

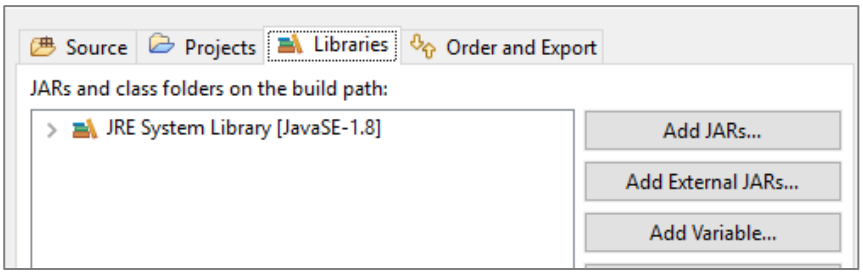
1. Schritt: Ein neues Java-Projekt anlegen:



2. Schritt: Namen des Projekts angeben und auf Next klicken (Achtung: Nicht auf Finish klicken)

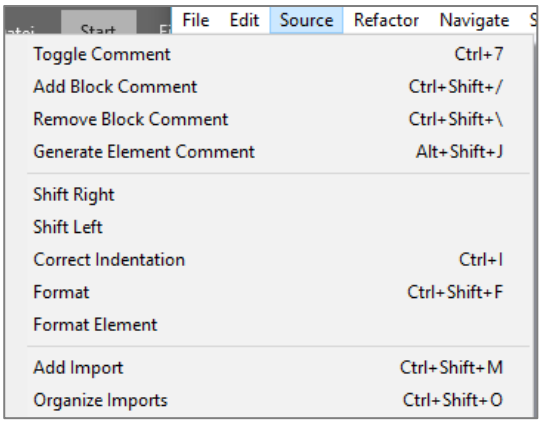
3. Schritt: Im Reiter Libraries auf Add External Jars klicken:

4. Schritt: Auf diesem Wege die Datei applu5.jar und die Date JGameGrid hinzufügen



5. Folgendes Rahmenprogramm ergibt das erste Programm

```
1  import java.awt.Color;
2  import ch.aplu.jgamegrid.GameGrid;
3
4  public class MeinProjekt extends GameGrid {
5      public MeinProjekt() {
6          super(3, 3, 60, Color.WHITE);
7          show();
8      }
9
10     public static void main(String[] args) {
11         new MeinProjekt();
12     }
13 }
```



Tipp: Die ersten zwei Zeilen müssen sie nicht tippen. Diese kann Eclipse über die Funktion Source->Organize Imports oder dem Tastenkürzel CTRL+SHIFT+O automatisch hinzufügen, sobald sie Zeile 4 geschrieben haben.

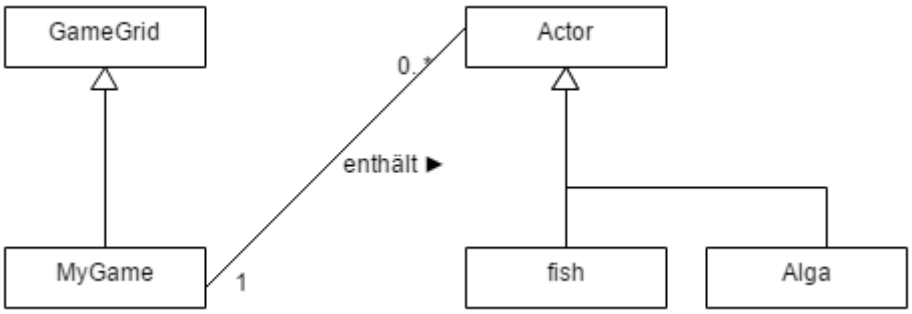
WICHTIGE CODEZEILEN

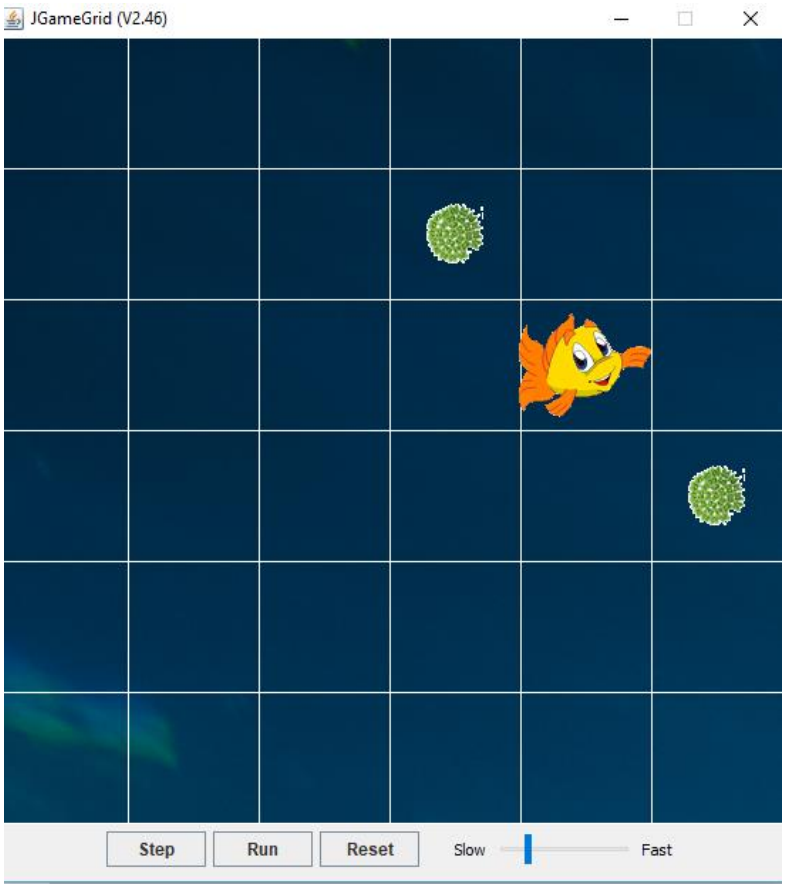
`public class MeinProjekt extends GameGrid {`	Erbt die Eigenschaften der Gamegrid-Klasse
`super(3, 3, 60, Color.WHITE);`	Erstellt ein 3x3 Feld mit weißen Rändern
`show();`	Das Fenster zeigt sich

MÖGLICHE ERWEITERUNGEN

`super(3, 3, 60, Color.BLACK, "sprites/purple.png");`	Lädt die Datei purple als Hintergrundbild. Die Datei muss im Unterordner sprites liegen, der ein Unterordner des src-Ordners ihres Projektes ist. Sie finden den Ordner, indem sie in Eclipse mit Rechts auf src klicken und dort auf Show in->System Explorer
`super(3, 3, 60, Color.BLACK, "sprites/purple.png",false);`	Der vierte Parameter FALSE blendet die Fußzeile aus.

2 - EINEN ACTOR ZUM GRID HINZUFÜGEN





EINEN ACTOR HINZUFÜGEN

1. SCHRITT: EINE NEUE KLASSE FÜR DEN ACTOR ANLEGEN:

```
21 class Fish extends Actor {
22     public Fish() {
23         super(true, "sprites/nemo.gif");
24     }
```

Die Klasse dient als ‚Bauplan' zum anlegen mehrerer Objekte dieser Klasse.

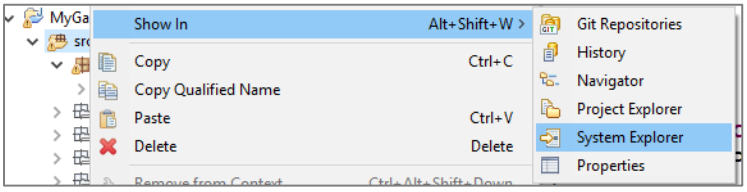
WICHTIGE CODEZEILEN

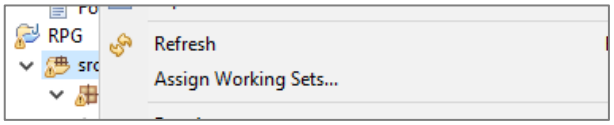
`class Fish extends Actor {`	Die Klasse MyActor erbt alle Eigenschaften der Klasse Actor.
`public Fish() {`	Der Konstruktor der Klasse Fish. Diese Methode wird beim Erzeugen eines neuen Fish-Objektes aufgerufen
`super(true, "sprites/nemo.gif");`	Der Konstruktor der Basisklasse Actor wird aufgerufen und diesem wird der Pfad zu einem Bild mitgegeben. Dieses dient als Hintergrundbild für alle Objekte dieser Klasse.

HINTERGRUNDBILD EINES ACTORS

Das Hintergrundbild eines Actors muss in einem Unterordner des src-Ordners des Projektes liegen.

Am einfachsten erreichen sie den Ordner, wenn sie mit rechter Maustaste auf den Ordner src in Eclipse klicken und dann auf Show In->System Explorer auswählen. Hier legen sie am besten einen neuen Unterordner sprites an.





Wichtig: Wenn das Programm eine Fehlermeldung wirft, obwohl sie sich sicher sind, die Bilddatei hinzugefügt werden, dann klicken sie einmal mit rechter Maustaste auf src und anschließend auf Refresh.

2. SCHRITT: DEN ACTOR ZUM SPIEL HINZUFÜGEN

Jetzt muss der Actor noch zum Spiel hinzugefügt werden. Dies geht, indem sie folgenden Code in den Konstruktor der Spielklasse ergänzen:

```
7  public class MeinProjekt extends GameGrid {
8      public MeinProjekt() {
9          super(6, 6, 92, Color.WHITE);
10         Actor myFish = new Fish();
11         this.addActor(myFish, new Location(4, 2));
12         show();
13     }
```

WICHTIGE CODE-ZEILEN

`Actor myFish = new Fish();`	Erzeugt ein neues Objekt vom Typ Fish
`this.addActor(myFish, new Location(4, 2));`	Fügt das neu erzeugte Objekt zum Spiel (an der Location (4|2) hinzu.

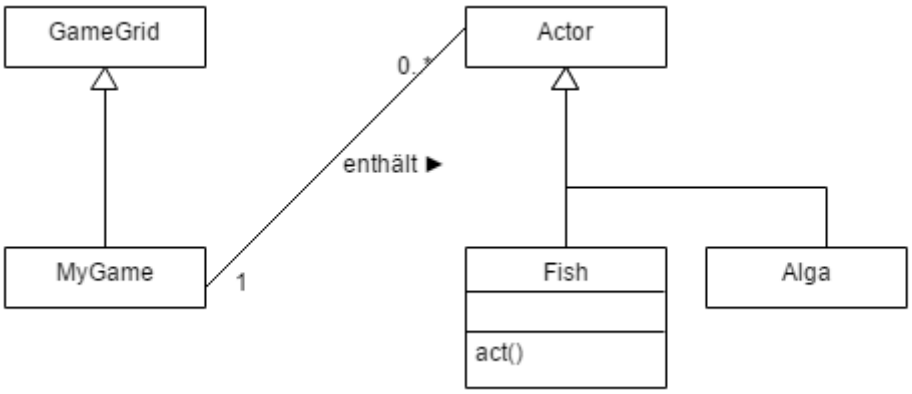
ERWEITERUNGEN

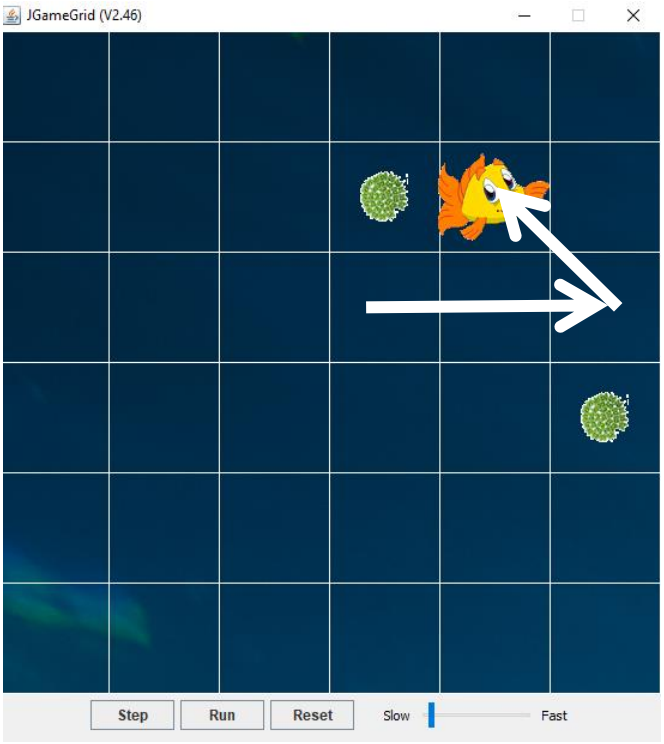
```
for (int i=1;i<7;i=i+1) {
    for (int j=1;j<7;j=j+1) {
        if (Math.random()<0.1 && isEmpty(new Location(i,j))) {
            Actor alga=new Alga();
            this.addActor(alga, new Location(i,j));
        }

    }
}
```

Zufällige weitere Objekte positionieren
Die beiden Schleifen durchlaufen das komplette Grid. Mittels der Methode Math.random(), die einen Zufallswert zwischen 0 und 1 erzeugt wird mit 10% Wahrscheinlichkeit eine Alge auf ein (leeres) Feld gestellt.

3 - EINEN ACTOR BEWEGEN





EINEN ACTOR BEWEGEN

1. SCHRITT: DIE ACT-FUNKTION ÜBERSCHREIBEN

Die Actor-Klasse verfügt über eine Methode act(). Diese Methode wird nach Start des Programmes mit Run in einem bestimmten (einstellbaren) Intervall immer wieder aufgerufen.

```
class MyActor extends Actor {
    public MyActor() {
        super("sprites/nemo.gif");
    }

    public void act() {
        move();
    }
```

BEISPIELE

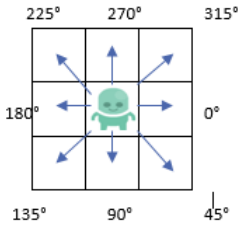
`public void act() {`
`  move();`
`}`
	Geradeaus bewegen
	Die Figur bewegt sich nach Start des Programms mit Run immer geradeaus

`public void act() {`
`  if (isNearBorder())`
`  turn(180);`
`  move();`
`}`
	Am Rand abprallen
	Das Objekt bewegt sich in einer horizontalen Ebene hin- und her. Es dreht sich mit turn(180) um 180° immer wenn es den Rand erreicht.

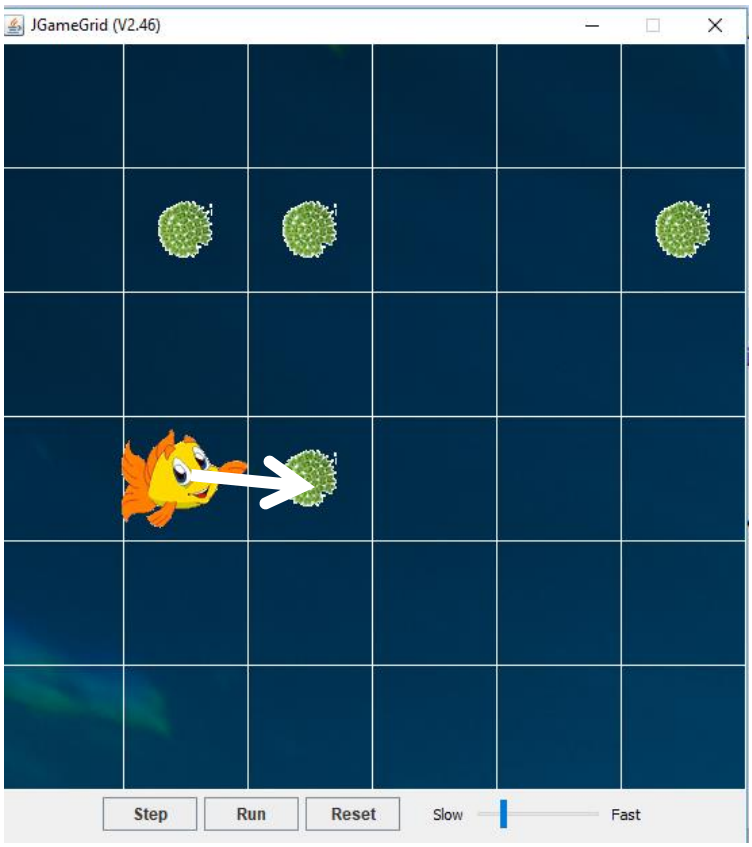
`public void act() {`
`  if (isNearBorder()) {`
`  turn(180);`
`  setHorzMirror(!isHorzMirror());`
`  }`
`  move();`
	In andere Richtung „schauen“
	Mit der Zeile setHorzMirror(!isHorzMirror()) wird das Bild gespiegelt, sofern es nicht bereits gespiegelt ist. Damit schaut das Objekt nach dem Abprallen vom Rand in eine andere Richtung.

`if (Math.random()>0.5) {`
`  turn(45);`
`}`
	Zufällige Bewegung
	Math.random() liefert eine zufällige Zahl zwischen 0 und 1. Wenn die Zahl größer 0,5 ist (also mit 50% Wahrscheinlichkeit), dreht sich die Figur um 45°.

`public MyActor() {`
`  super(true, "sprites/nemo.gif");`
`}`
	In Bewegungsrichtung drehen
	Wenn im Konstruktor ein true als erster Parameter übergeben wird, wird die Figur in Bewegungsrichtung gedreht.

`setDirection(45);`	**Die Richtung absolut festlegen** Während turn(45) die Richtung relativ zur eigenen Person ändert, ändert setDirection(45) die Position absolut nach folgendem Schema: 

4 - KOLLISIONEN IM GRID



KOLLISIONEN IM GRID

Kollisionen sind wichtig um zu erkennen, ob ein Feld bereits besetzt ist. Es gibt prinzipiell zwei Funktionen mit der man Kollisionen überprüfen kann: getActorsAt() gibt alle Figuren in einer Zelle als Liste zurück, getOneActorAt() gibt einen einzelnen Actor in einer Zelle zurück bzw. null, wenn sich dort kein Actor befindet.

SCHRITT 1: METHODE GETONEACTORAT ERGÄNZEN

Die Methode getOneActorAt() kann in der act() Methode ergänzt werden:

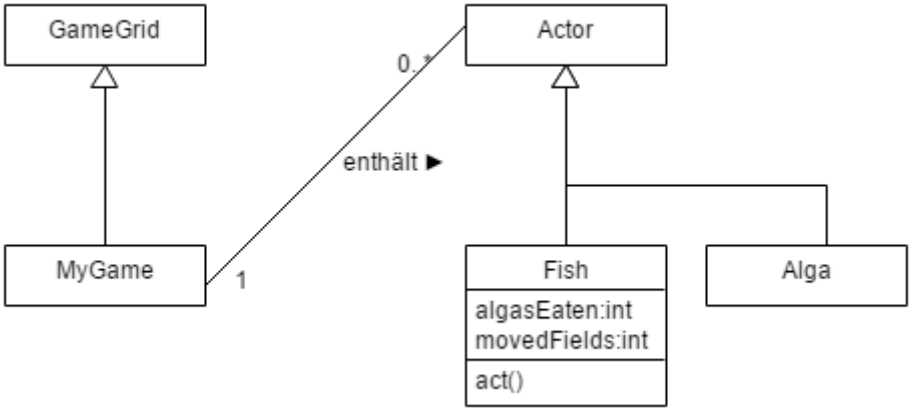
```
37      Actor actor = gameGrid.getOneActorAt(this.getLocation(), Alga.class);
38      if (actor != null) {
39          actor.removeSelf();
40      }
```

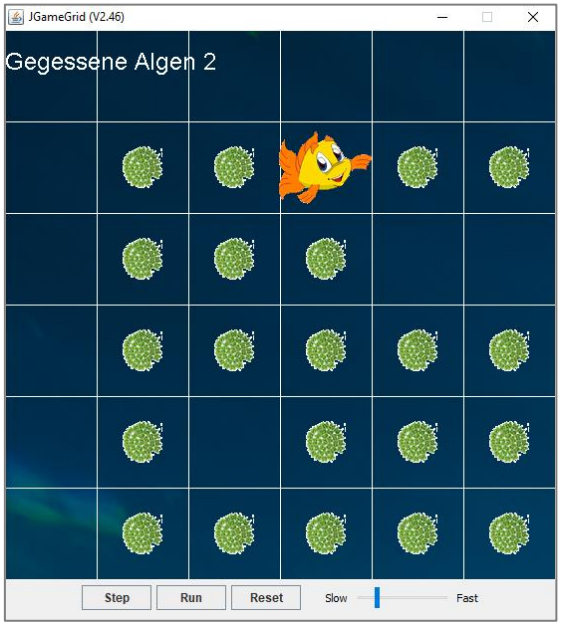
In Zeile 37 wird in die Variable actor die Figur geladen, die sich an derselben Position wie die aktuelle Figur befindet. Der Parameter Alga.class sagt aus, dass nur nach Objekten vom Typ Alge gesucht werden soll, andere Objekte werden ignoriert. Die Methode actor.removeSelf(); entfernt die Alge aus dem Gitter. Mittels dieser Funktion kann der Fisch also auf Algenjagd gehen.

ERWEITERUNGEN

41 Actor neighbour = 42 gameGrid.getOneActorAt(this.getNextMoveLocation(), Fish.class);	Mit Hilfe von this.getNextMoveLocation() kann überprüft werden, ob sich eine Figur an der Position befindet, auf die sich die aktuelle Figur als nächstes zubewegen wird.

5 - ATTRIBUTE


GameGrid
Actor
0..*
enthält ▶
MyGame
1
Fish
algasEaten:int
movedFields:int
act()
Alga


JGameGrid (V2.46)
Gegessene Algen 2
Step
Run
Reset
Slow
Fast

ATTRIBUTE

Mit Hilfe von Attributen erhalten Spielfiguren ein Gedächtnis.

ATTRIBUTE ANLEGEN

Attribute fügt man vor dem Konstruktor und direkt nach dem Klassennamen hinzu:

```
int algaeEaten=0;
int movedFields=0;
```

In diesem Fall werden die Attribute algaeEaten und movedFields erstellt, um die Anzahl der gefressenen Algen und die Anzahl der bewegten Felder zu messen.

ATTRIBUTE VERÄNDERN

```
if (actor != null) {
    algaeEaten=algaeEaten+1;
    actor.removeSelf();
}
```

Den Code kann man folgendermaßen lesen:
Wenn die Spielfigur auf eine Alge stößt, dann zähle die Anzahl der gefressenen Algen um 1 hoch. Entferne anschließend die Alge.

Die Zeile algaeEaten=algaeEaten+1;
Kann man auch abkürzen als algaeEaten++;

ATTRIBUTE AUSGEBEN

Bisher wird die Anzahl gegessener Algen nur angezeigt. Damit diese auch ausgegeben werden können, muss ein Text auf den Hintergrund geschrieben werden:

```
52 gameGrid.getBg().clear();
53 gameGrid.getBg().drawText("Gegessene Algen "+algaeEaten, new Point(0, 40));
```

Zeile 52: Der Hintergrund wird zurückgesetzt. Damit werden alle Texte, die auf den Hintergrund gemalt wurden, gelöscht.

Zeile 53: Die Anzahl der Gegessenen Algen wird notiert und am Punkt (0|40) (obere linke Ecke) ausgegeben.