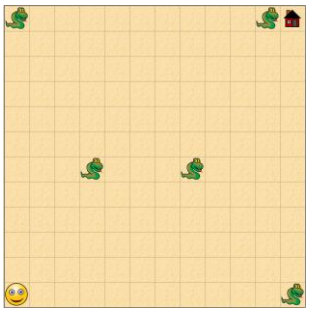
OOP: SMILEY MAN

Unser Ziel ist es, ein Spiel zu erstellen, das so ablaufen soll: Wir steuern eine Spielfigur (einen Smiley), dessen Ziel es ist von links unten nach rechts oben (zum Haus) zu gelangen. Auf dem Weg dorthin lauern allerdings einige Schlangen, die den Smiley fressen wollen. Wenn der Smiley und eine Schlange auf demselben Feld sind, wird der Smiley gefressen und das Spiel endet. Nur wenn der Smiley unbeschadet zum Haus gelangt, haben wir das Spiel gewonnen.

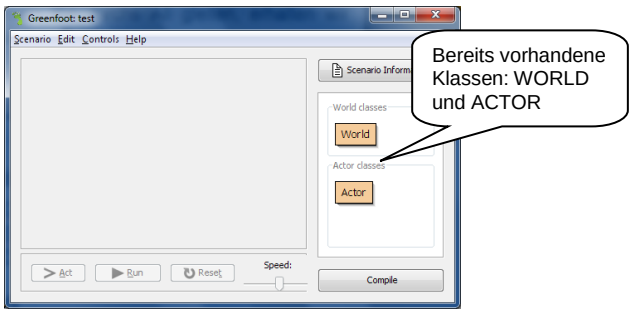


Zu Erstellung des Spiels verwenden wir das Programm Greenfoot (www.greenfoot.org). Greenfoot ist eine JAVA-Entwicklungsumgebung, die hauptsächlich dafür verwendet wird, 2D-Animationen und Spiele zu erstellen.

Ein Projekt erstellen

- Greenfoot starten
- Scenario / New…
- Dateiname vergeben und speichern

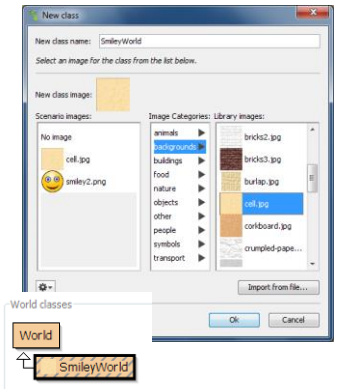
Im rechten Teil des Fensters sehen Sie, dass bereits zwei Klassen vorhanden sind: WORLD und ACTOR.



Wir erinnern uns: Eine KLASSE ist eine Art Schablone, die beschreibt, was alle Objekte „wissen“ und „können“. In der Klasse WORLD ist z.B. festgelegt, was alle Welten gemeinsam haben (z.B. eine Größe, ein bestimmtes Aussehen, etc.).

Ein Welt erstellen

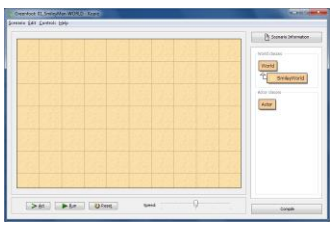
- RMT (=Rechte Maustaste) auf WORLD
- New Subclass…
- Image: cell aus backgrounds,
  New Class Name: SmileyWorld



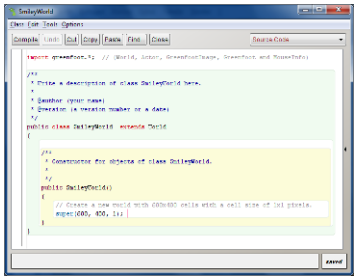
Im Hauptfenster sehen Sie nun, dass eine neue Klasse erschienen ist (SmileyWorld). Die Klasse ist schraffiert, was bedeutet, dass die Klasse erst noch in eine Sprache übersetzt werden muss, die der Computer versteht. Diesen Vorgang nennt man kompilieren. Klicken Sie deshalb auf:

- Compile

Die Schraffierung verschwindet und Sie sehen, wie die Welt aussieht!



Die Größe der Welt verändern

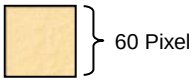


Wenn Sie im rechten Teil des Fensters einen Doppelklick auf die Klasse SmileyWorld setzen, können Sie sich anschauen, wie die Klasse programmiert wurde. Uns interessiert momentan nur dieser Teil:

```
public SmileyWorld()
{
   // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
   super(600, 400, 1);
}
```

Sie können erkennen, dass die Welt aus 600x400 Zellen besteht, wobei eine Zelle eine Größe von einem Pixel besitzt. Um unser Spiel übersichtlicher zu halten, werden wir die Zellen vergrößern, so dass sich ein Spielfeld von 12x12 Feldern ergibt, deren Größe 60 Pixel beträgt. Dies erreichen wir, indem wir den oben fett gedruckten Befehl ersetzen durch:

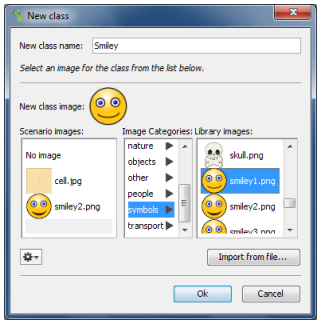
```
// Eine Welt mit 12x12 Feldern. Ein Feld hat die Größe 60 Pixel.
super(12, 12, 60);
```



Der Computer beginnt übrigens beim Feld 0 mit der Zählung. Beim Positionieren von Objekten werden also später die Felder 0 bis 11 angesprochen.

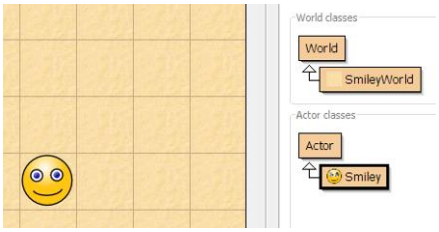
Eine ACTOR-Klasse erstellen

ACTORs sind die Klassen der Objekte, die in der Welt agieren, also etwas tun können (z.B. Smiley und Spinnen). Wir erstellen zuerst den Smiley:



- RMT auf ACTOR
- New Subclass…
- Image: smiley1.png aus *symbols*, Name: Smiley
- Vergessen sie das kompilieren nicht!

Den Smiley können Sie jetzt so in die Welt einfügen:

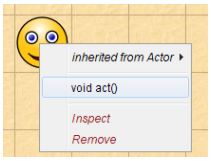


- RMT auf Smiley
- new Smiley()
- Klick auf die Welt: Dort wird der Smiley eingefügt

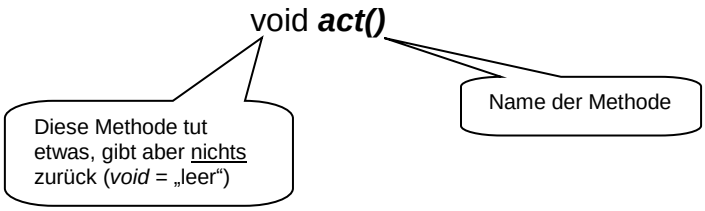
Machen Sie sich noch mal klar, was gerade eben passiert ist: Sie haben ein **Objekt** *der* **Klasse Smiley** *erzeugt.*

Genauer gesagt: Mit new Smiley() *wird der sogenannte* **Konstruktor** *der Klasse Smiley aufgerufen. Der Konstruktor ist eine besondere Methode, die dafür sorgt, dass man Objekte erzeugen kann.*

Methoden erstellen: Den Smiley bewegen



Wenn Sie mit der rechten Maustaste auf den Smiley klicken, sehen Sie, dass dieser bereits über eine Methode verfügt, nämlich:



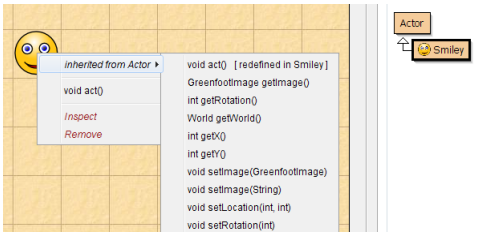
Ein Doppelklick auf die Klasse *Smiley* bringt uns in die Programmier-Umgebung. Hier können wir festlegen, was passieren soll, wenn jemand die act()-Methode aufruft.

```
public void act()
   {
      // Add your action code here.
   }
```

Der Smiley soll sich nach oben, unten, links und rechts bewegen können. Dies kann man über die *setLocation()*-Methode erreichen.

Woher kennt unser Smiley eigentlich die setLocation-*Methode? Eben haben wir doch noch behauptet, dass es nur die act()-Methode gibt!*

Des Rätsels Lösung ist relativ einfach: Alle ACTORs kennen eine bestimmte Anzahl an Methoden. Welche Methoden das sind, kann man sich ganz einfach anzeigen lassen, indem man auf der Rakete auf inherited from Actor *klickt.*



Inherited from Actor *heißt übrigens „geerbt vom Actor“. Hiermit haben Sie eines der wichtigsten Konzepte der objektorientierten Programmmierung kennen gelernt: Eine Subclass (der Smiley) erbt alles von der Super Class (Actor).*

Wir experimentieren mit der setLocation-Methode, indem wir diese so ändern:

```
public void act()
{
   this.setLocation(1, 1);
}
```

Klicken Sie anschließend auf die Schaltfläche **Act**.

→ Der Smiley bewegt sich auf die Koordinaten: x = 1 y = 1

(Achtung: Der Ursprung (0 | 0) liegt beim Computer immer links oben, weil das in einem Fenster der einzige fixe Punkt ist. Das Koordinatensystem ist also gekippt. Nach unten werden die y-Koordinaten größer!)

Eine konstante Bewegung

In der obigen Lösung bewegt sich unser Smiley zum angegebenen Punkt. Ab dann verändert er seine Position nicht mehr. Damit der Smiley jedes Mal die Position ändert, wenn die act()-Methode aufgerufen wird, verändern wir diese so:

```
public void act()
{
   this.setLocation(this.getX() + 1, this.getY());
}
```

Tipp:
Code-Ergänzung mit
STRG + LEERTASTE

Wenn Sie immer wieder auf die Act-Schalftläche klicken, bewegt sich der Smiley immer weiter nach rechts. Alternativ können Sie auch auf die Run-Schaltfläche klicken, die nichts weiter tut, als die act()-Methode immer wieder aufzurufen.

Bei der Run-Schaltfläche werden Sie wahrscheinlich feststellen, dass die Bewegung für ein Spiel eher zu schnell ist. Wir können unserem Smiley aber ganz einfach sagen, dass er nach jedem Schritt kurz warten soll, indem wir die Act-Methode um diese Anweisung ergänzen:

```
Greenfoot.delay(5);
```

Tastatur-Steuerung

In der obigen Lösung bewegt sich unsere Spielfigur zum angegebenen Punkt. Ab dann verändert sie ihre Position nicht mehr. Damit diese jedes Mal die Position ändert, wenn die act()-Methode aufgerufen wird, verändern wir diese so:

```
public void act()
   {
      if(Greenfoot.isKeyDown("right"))
      {
         setLocation(getX() + 1, getY());
         Greenfoot.delay(10);
      }
   }
```

Arbeitsauftrag

Erweitern Sie die Methode ***act()*** so, dass der Smiley über die Pfeiltasten auch nach links, oben und unten gesteuert werden kann!