

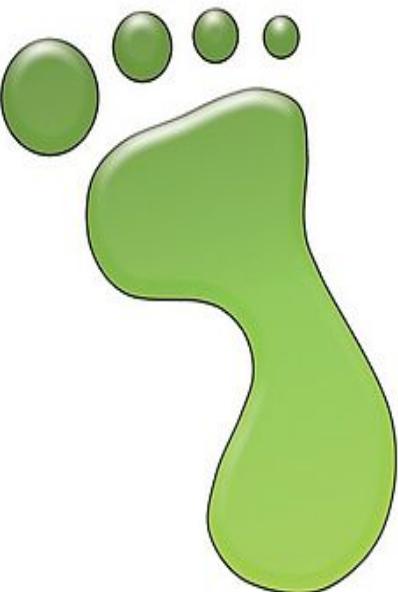


Greenfoot Workshop

Bobby - Snake

Greenfoot

- Development Environment based on the Java programming language.
- Mainly designed to ease the Java language learning
- Allow to easily create 2D games
- Greenfoot is freely available on Microsoft Windows, Mac OS X, and Linux.
- <http://www.greenfoot.org/>





The Java language



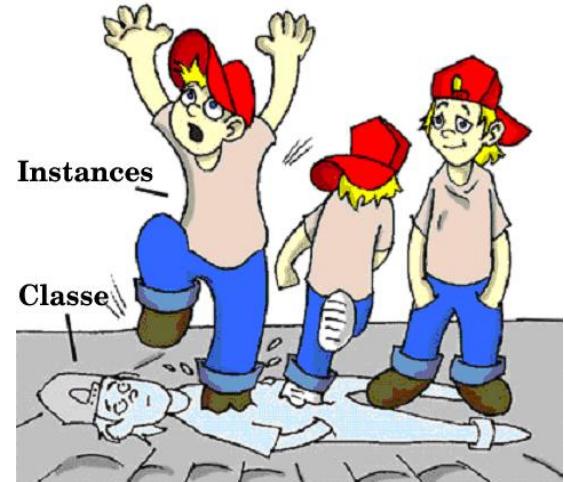
- « Object Oriented » Language
- One of the World's most used programming language
- Allow to develop applications running on multiple systems
 - Windows, Mac, Linux, Android, ...

« Object Oriented »

- Java applications are made of
« Objects »
- Each Object is composed of two main parts:
 - A set of internal properties
 - its « behavior », the actions it can do

Classes and Objects

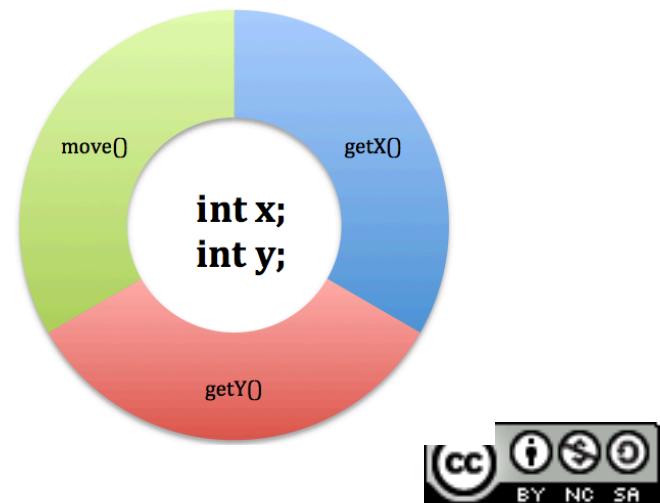
- To describe an **object**, we will define a **class**.



- A **class** is a kind of « **model** » allowing to build the **objects**
 - We also say that the objects are « **instances** » of the class

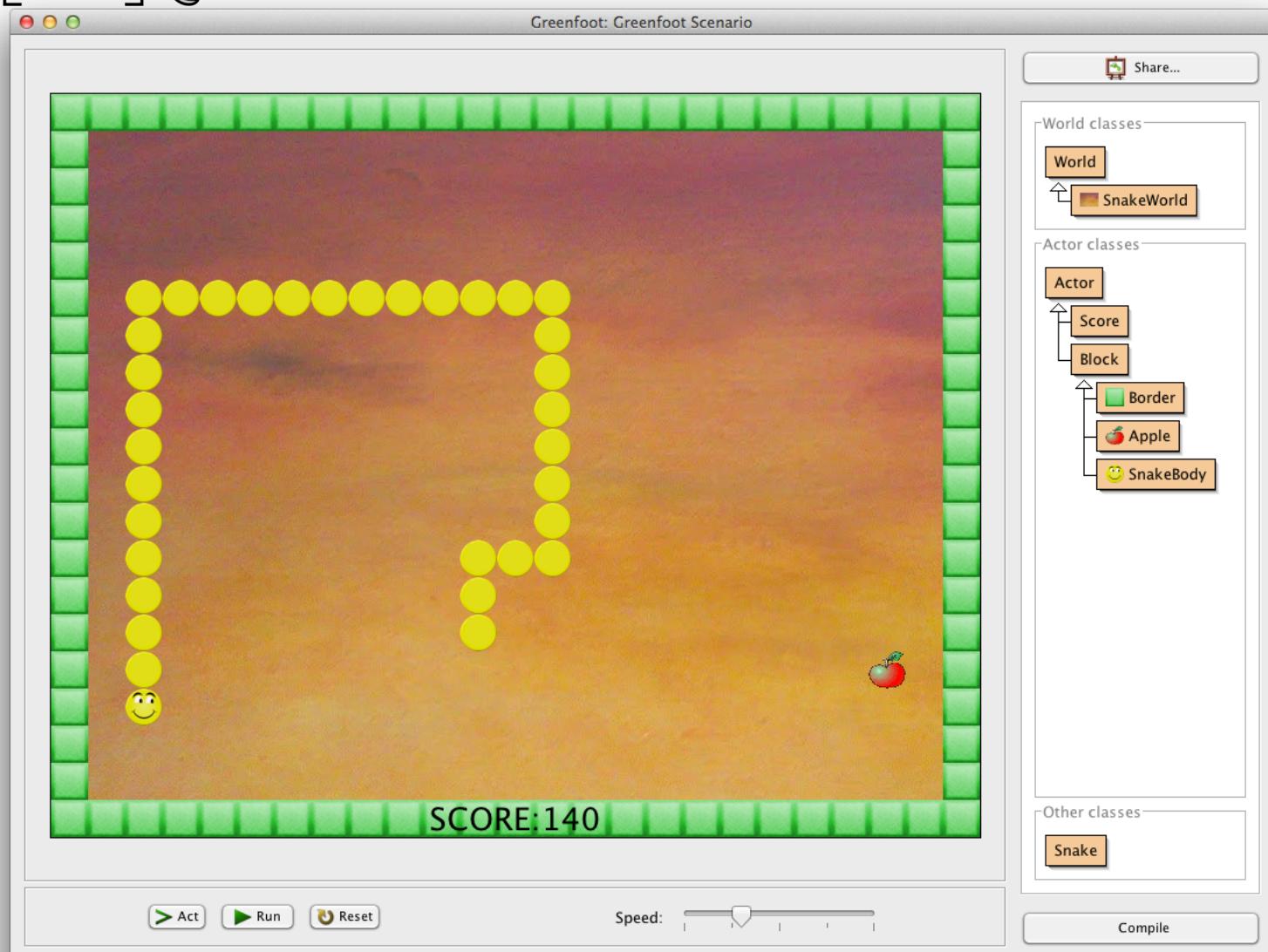
A Java Class sample

```
public class GameElement {  
  
    private int x;  
    private int y;  
  
    public GameElement(int initX, int initY) {  
        x = initX;  
        y = initY;  
    }  
  
    public int getX() {  
        return x;  
    }  
  
    public int getY() {  
        return y;  
    }  
  
    public void move(int xMove, int yMove) {  
        x = x + xMove;  
        y = y + yMove;  
    }  
}
```



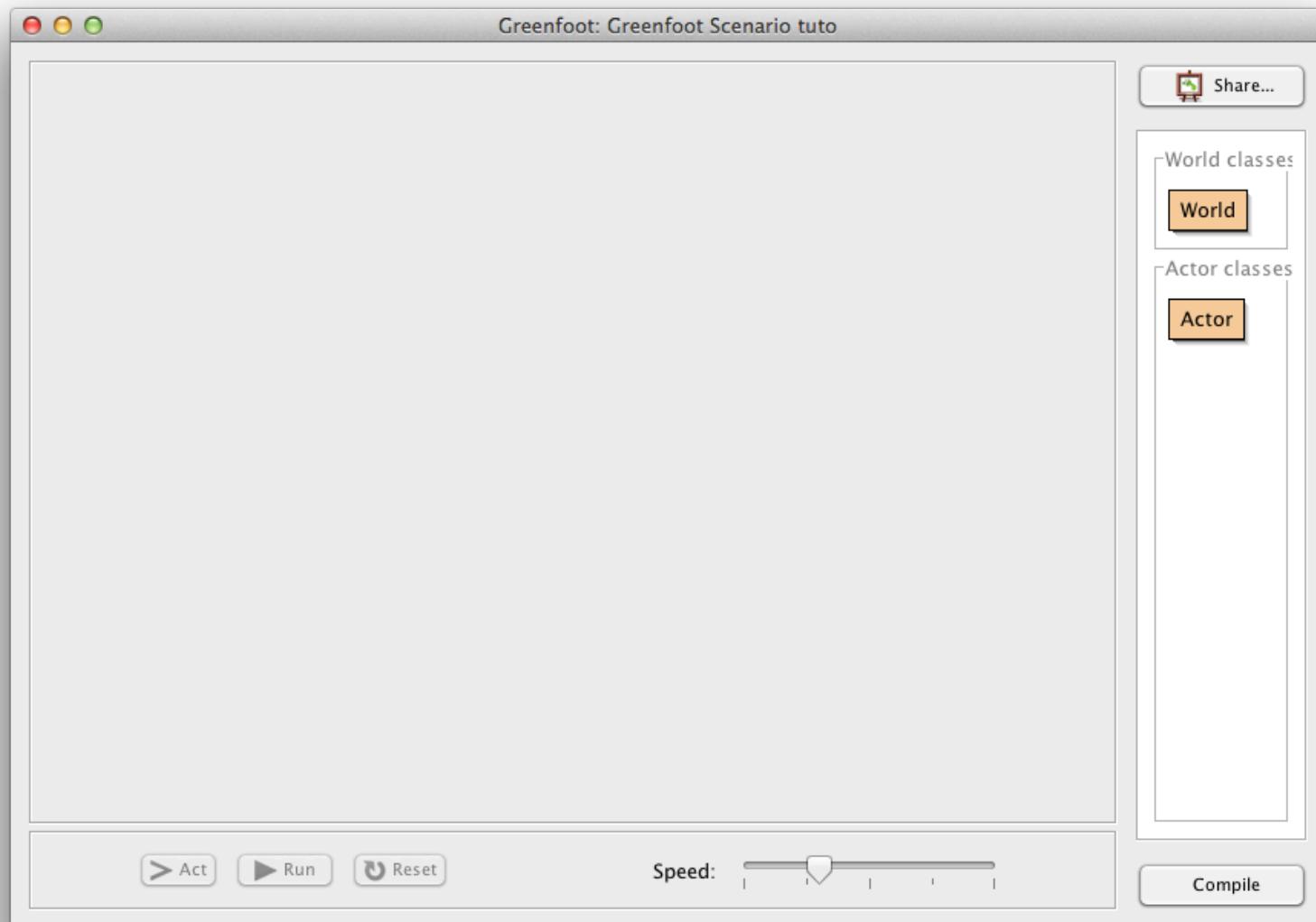


Bobby-Snake

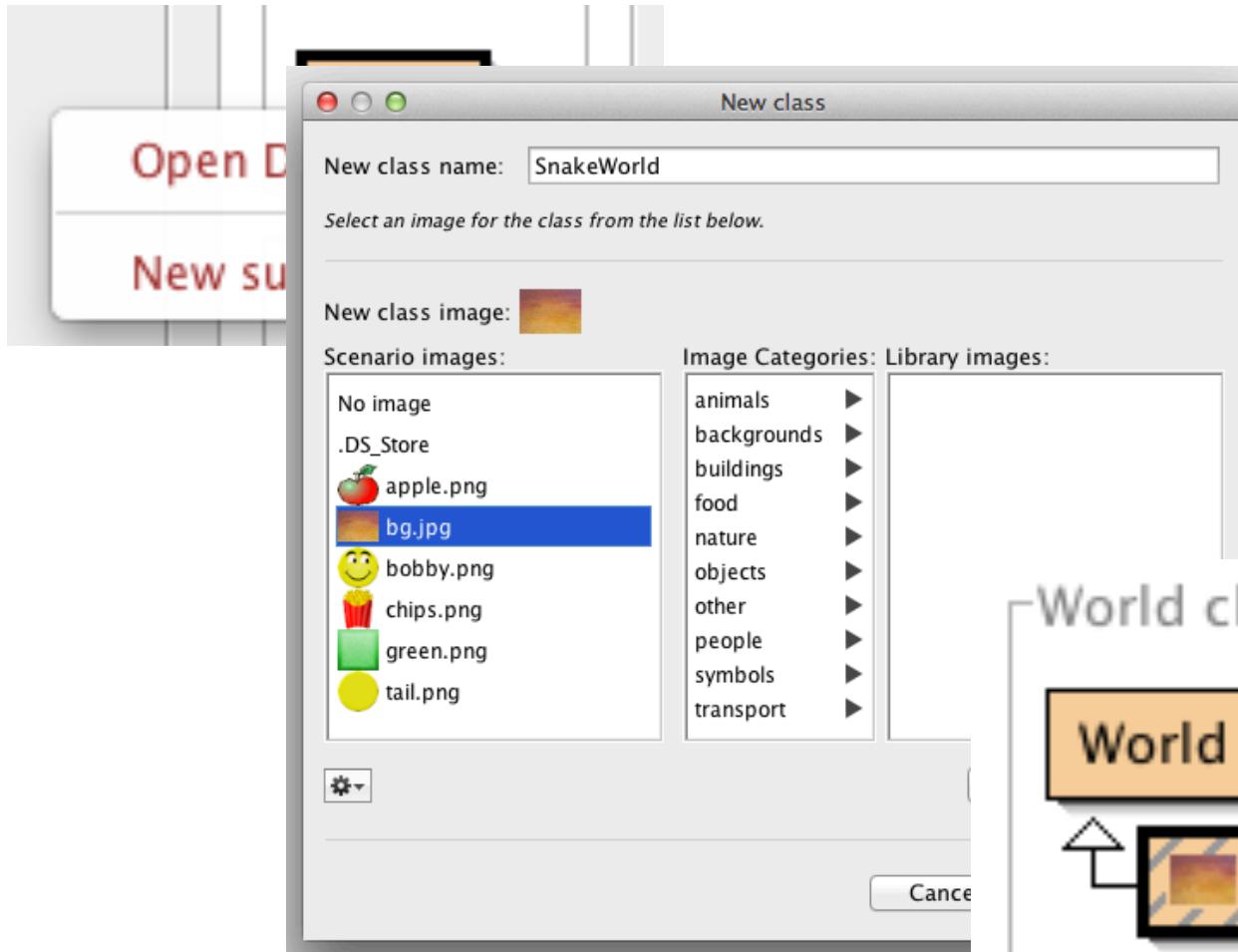




New scenario

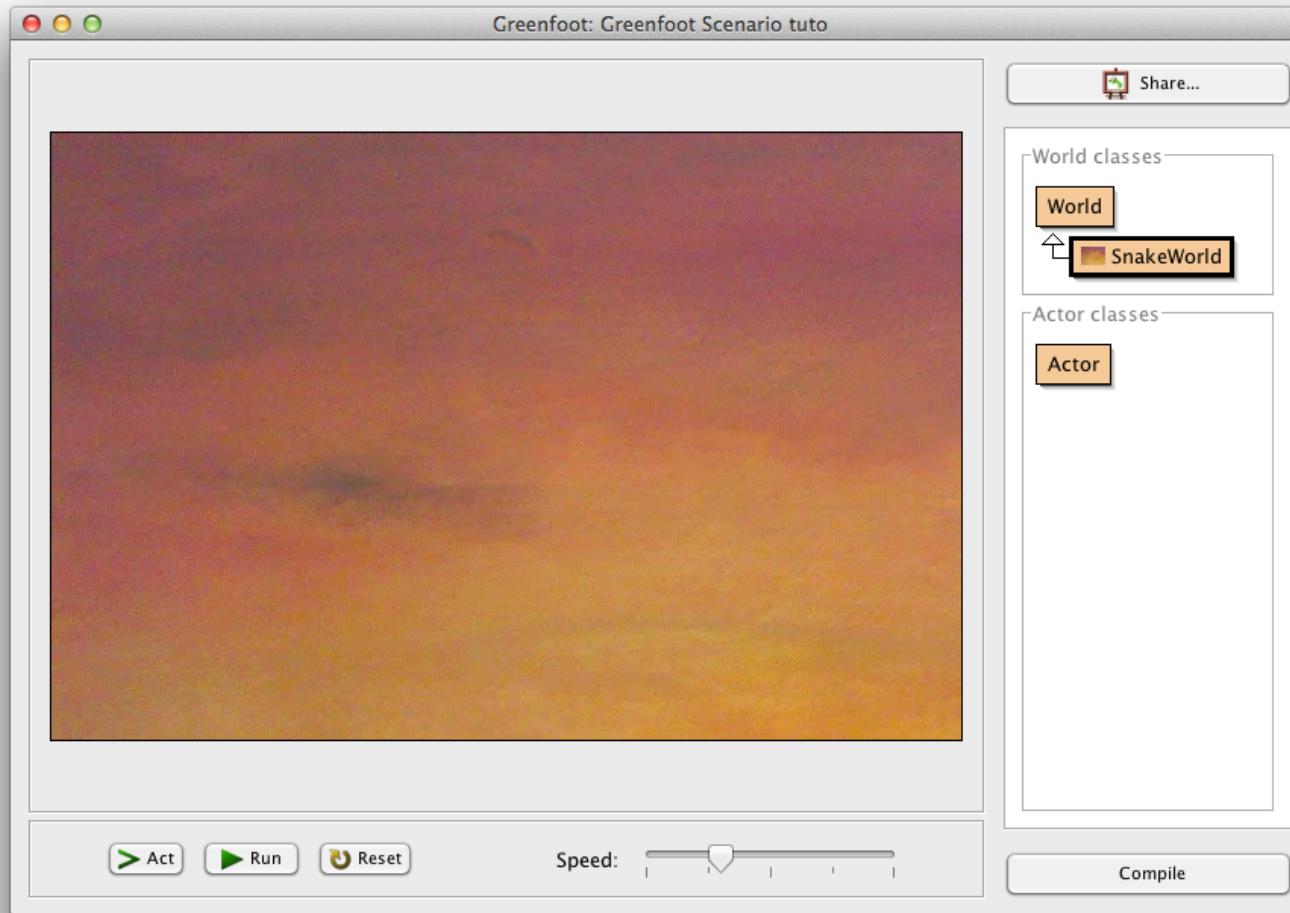


World creation

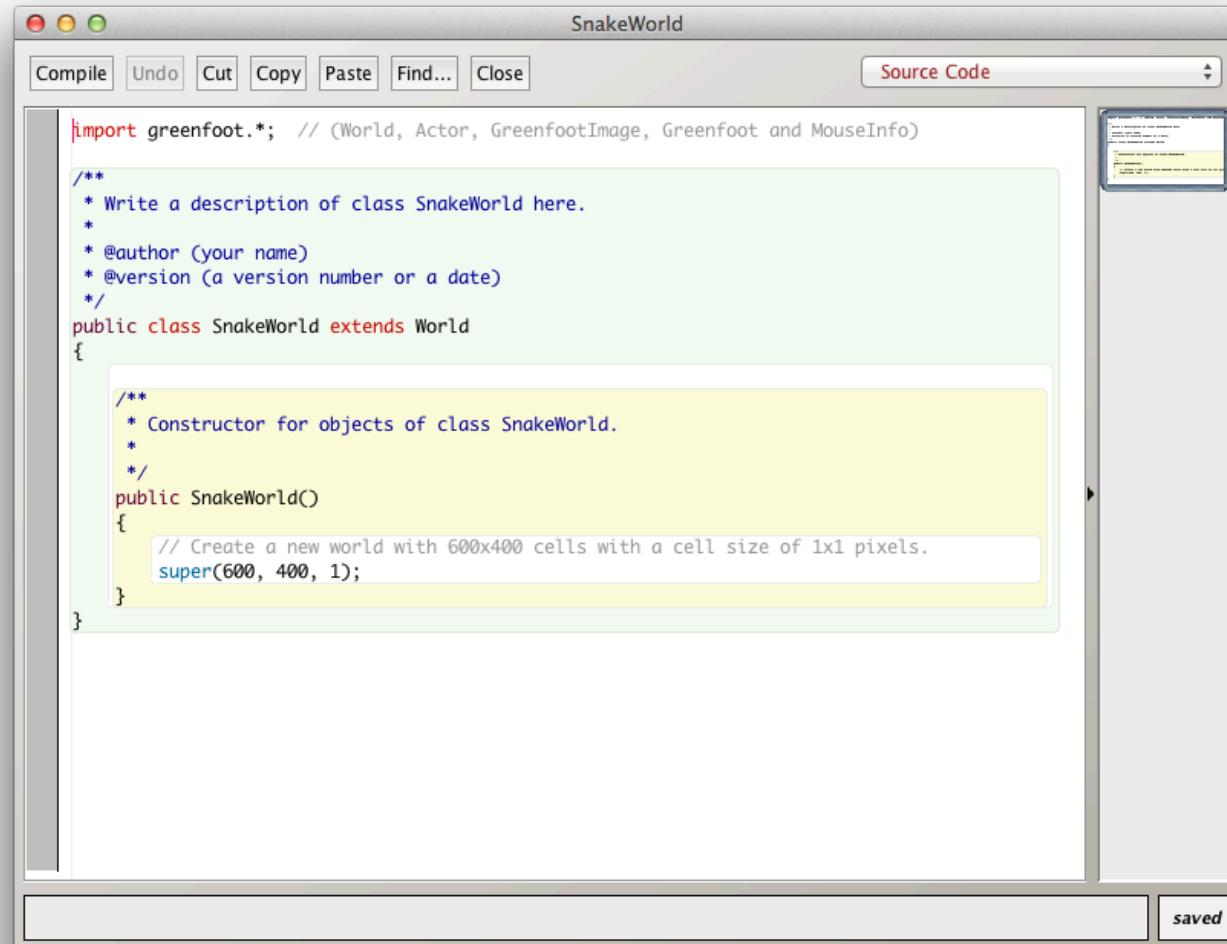




Snake World



SnakeWorld code

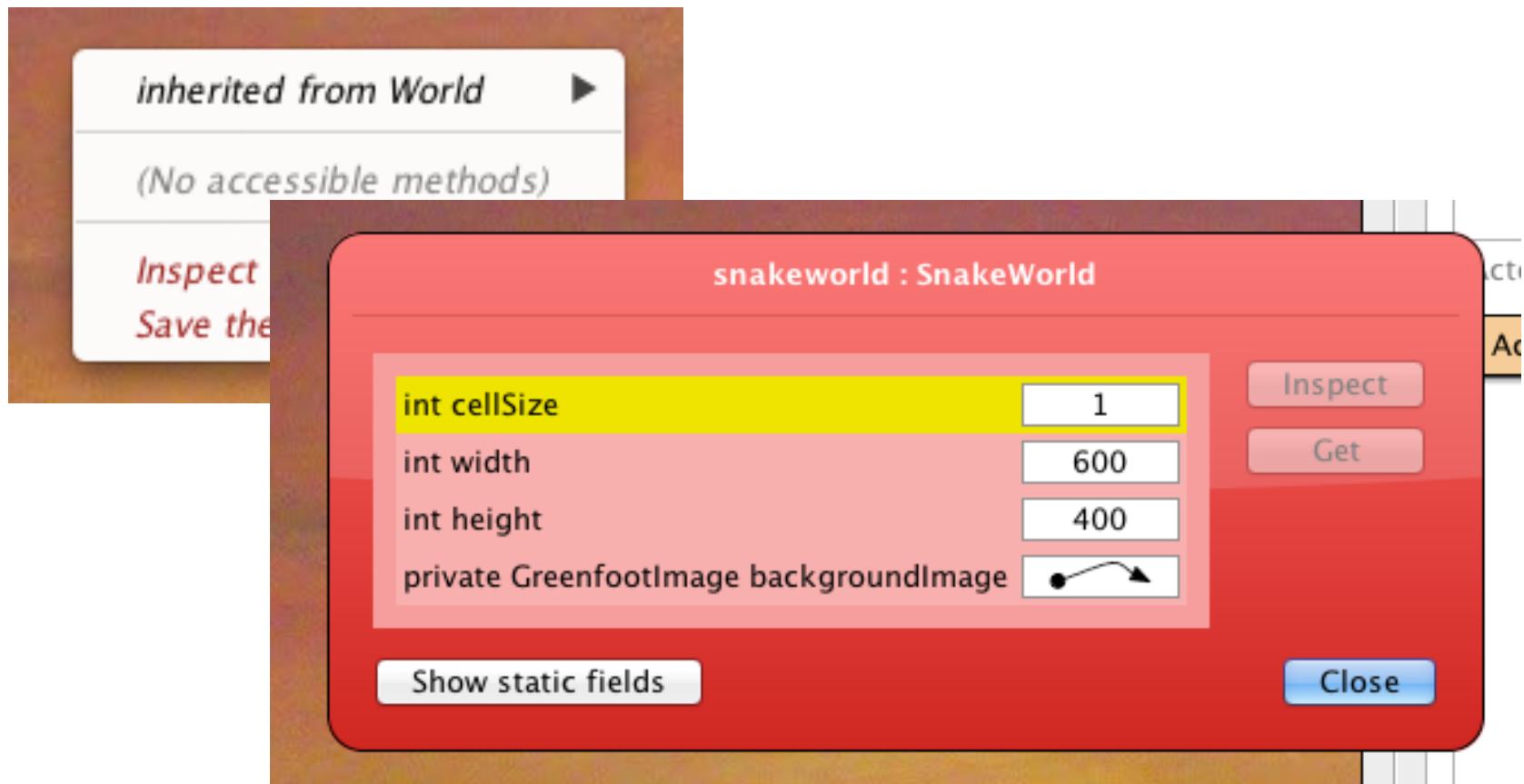
A screenshot of a Java code editor window titled "SnakeWorld". The window has a menu bar with "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". A toolbar below the menu bar contains buttons for "Source Code" and other options. The main area displays the following Java code:

```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class SnakeWorld here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class SnakeWorld extends World
{
    /**
     * Constructor for objects of class SnakeWorld.
     *
     */
    public SnakeWorld()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(600, 400, 1);
    }
}
```

The code editor highlights certain parts of the code in different colors: red for keywords like `import`, `public`, `class`, etc.; blue for comments; and green for strings. A tooltip-like callout box is shown over the constructor's first line, containing the text: "Constructor for objects of class SnakeWorld." In the bottom right corner of the editor window, there is a "saved" indicator.

Inspect World



Change the code

- We will use blocks of 32x32 pixels
- The game size will be 25 x 20 blocks

```
/**  
 * Constructor for objects of class SnakeWorld.  
 *  
 */  
public SnakeWorld()  
{  
    super(25, 20, 32);
```

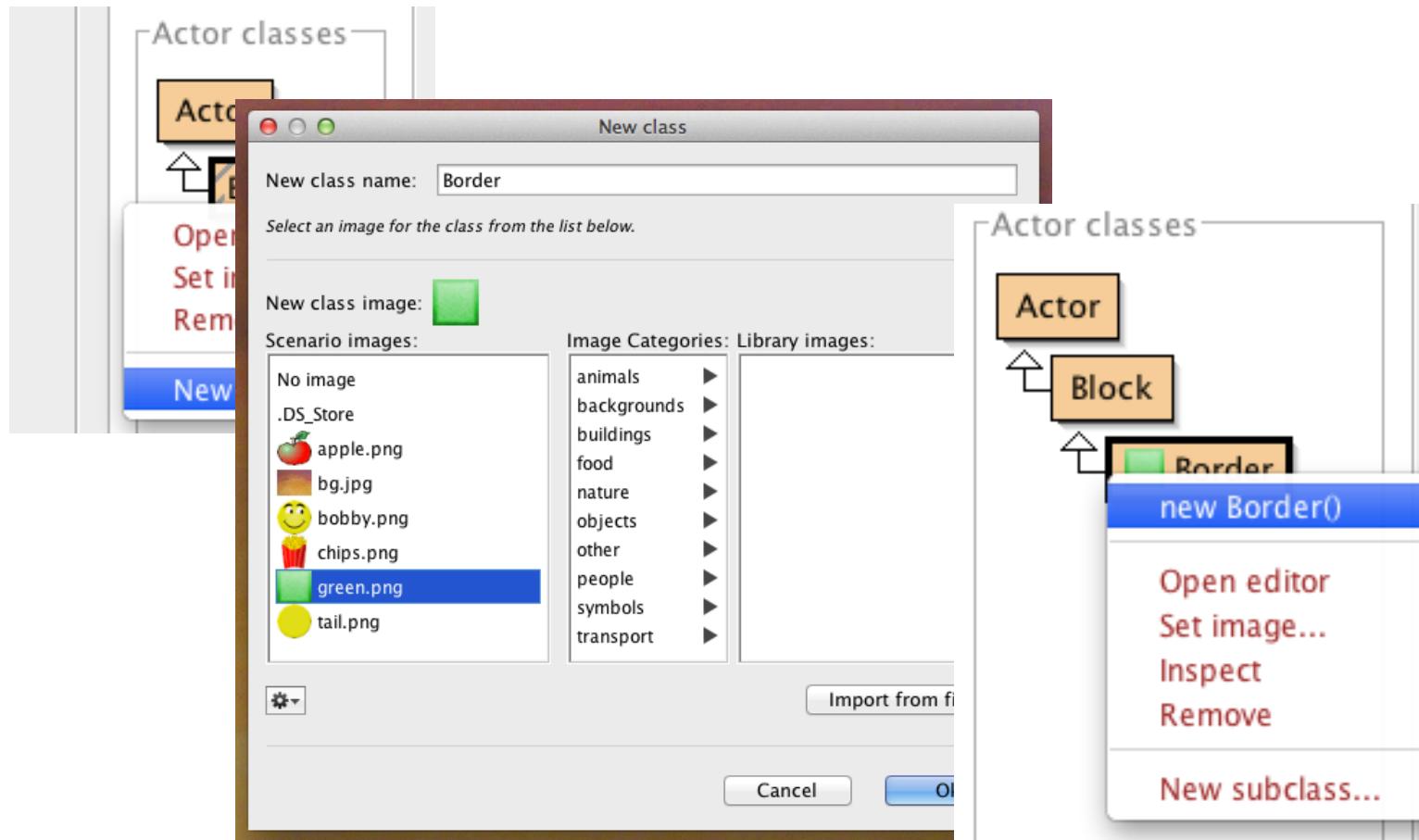


Create a Block

A screenshot of a software interface for creating a new class. The main window title is "New class". The "New class name:" field contains "Block". Below it, a note says "Select an image for the class from the list below." On the left, under "Scenario images:", there is a list of file names with corresponding icons: ".DS_Store", "apple.png" (an apple), "bg.jpg" (a brown background), "bobby.png" (a smiling face), "chips.png" (a red cup with chips), "green.png" (a green square), and "tail.png" (a yellow circle). On the right, under "Image Categories: Library images:", there is a list of categories: animals, backgrounds, buildings, food, nature, objects, other, people, symbols, and transport. At the bottom of the interface, a code editor shows the following Java code:

```
public class Block extends Actor
{
}
```

Create the Border

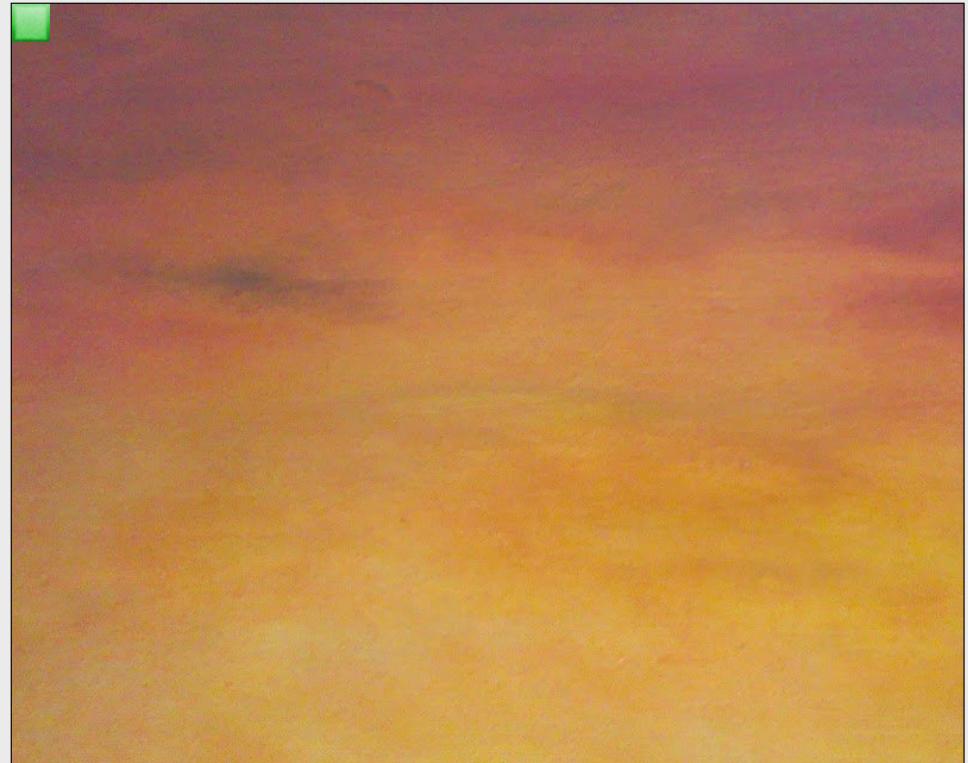


Coordinates

| | | | | |
|--------|--------|--------|-----|---------|
| (0,0) | (1,0) | (2,0) | ... | (24,0) |
| (0,1) | (1,1) | (2,1) | ... | (24,1) |
| (0,2) | (1,2) | (2,2) | ... | (24,2) |
| ... | ... | ... | ... | ... |
| (0,19) | (1,19) | (2,19) | ... | (24,19) |

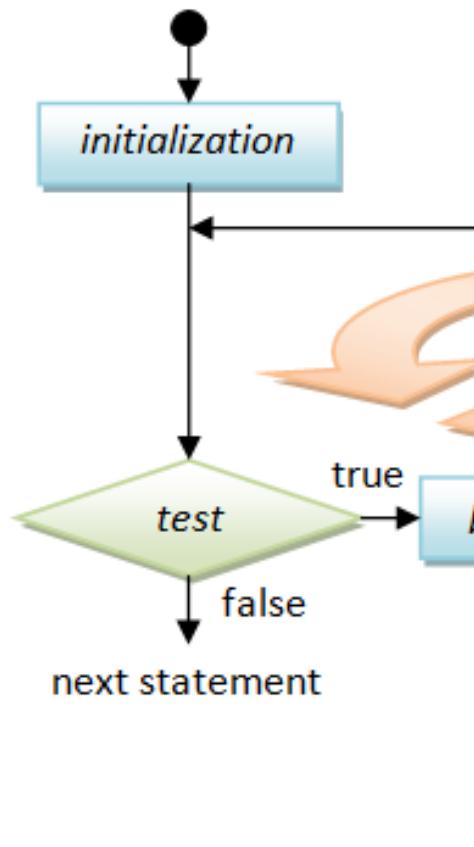
Display the borders

```
public SnakeWorld()  
{  
    super(25, 20, 32);  
    addObject(new Border(), 0, 0);  
}
```



for loops

```
for ( initialization ; test ; post-processing ) {
    body ;
}
```

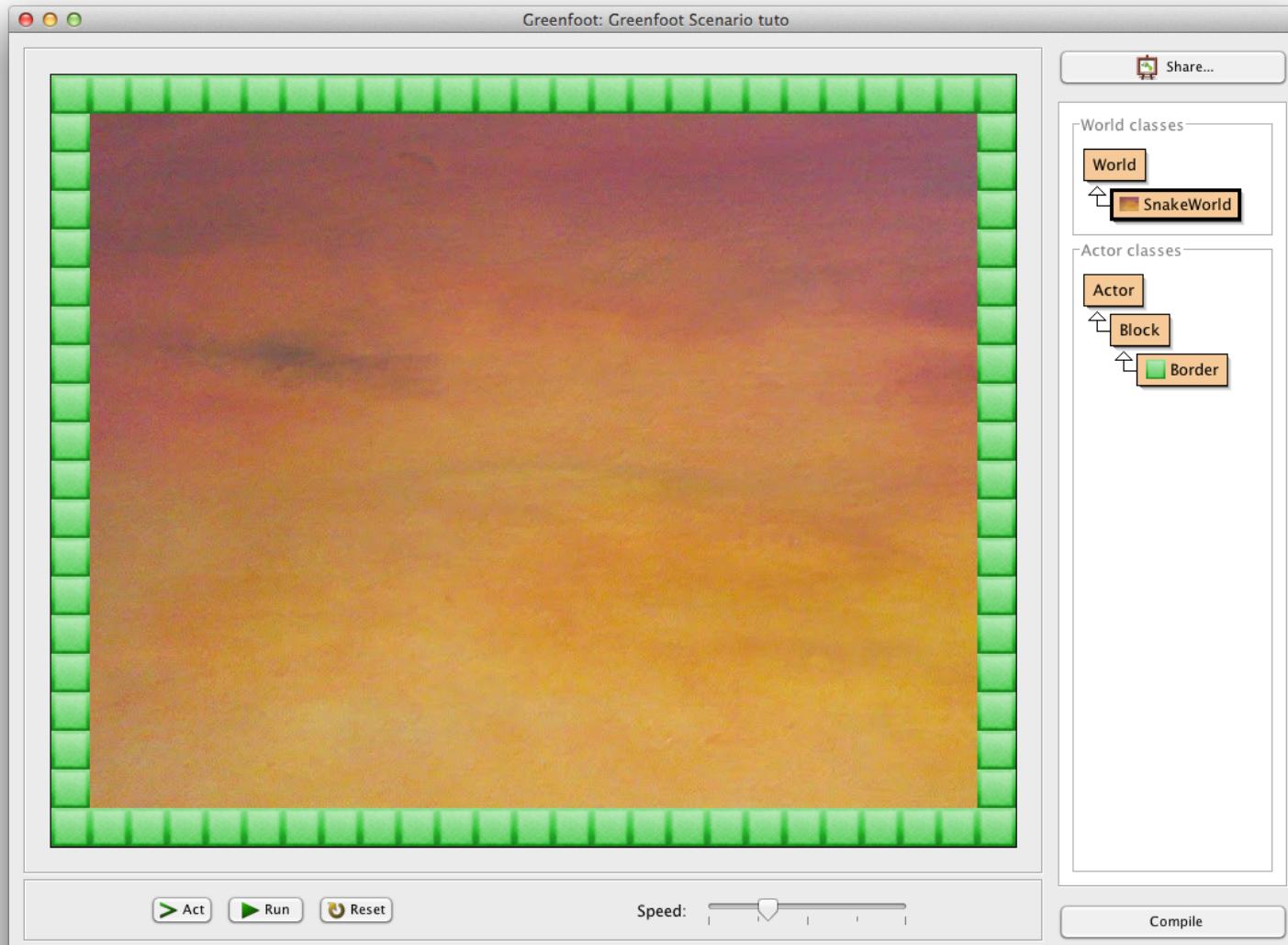


```
public SnakeWorld()
{
    super(25, 20, 32);

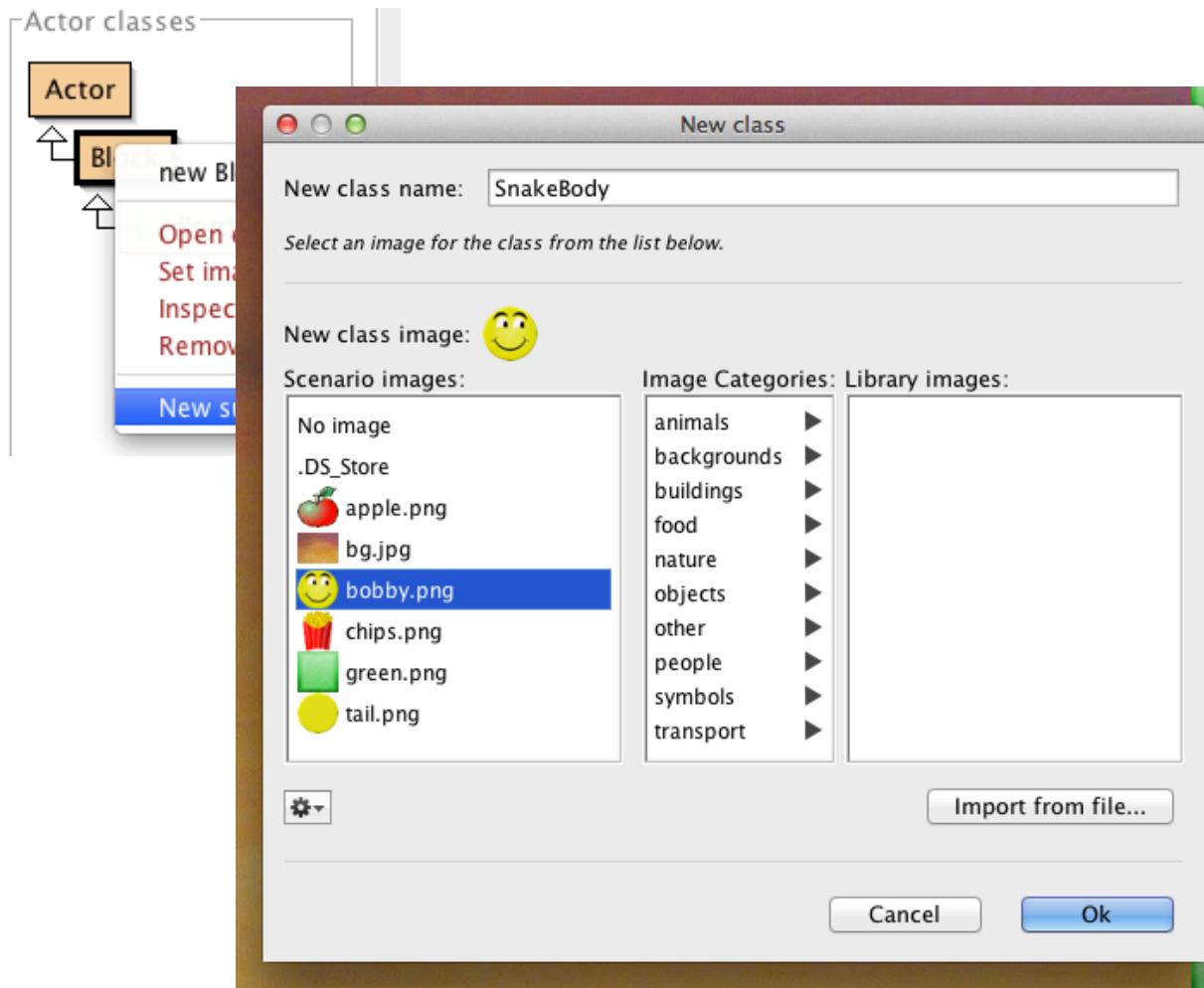
    for (int x = 0; x < getWidth(); x++) {
        addObject(new Border(), x, 0);
        addObject(new Border(), x, getHeight() - 1);
    }

    for (int y = 0; y < getHeight(); y++) {
        addObject(new Border(), 0, y);
        addObject(new Border(), getWidth() - 1, y);
    }
}
```

The game borders



Class SnakeBody



SnakeWorld changes

```
public class SnakeWorld extends World
{
    private LinkedList<SnakeBody> snake = new LinkedList<SnakeBody>();
```

```
import greenfoot.*;
import java.util.*;
```

```
public SnakeWorld()
{
    super(25, 20, 32);

    SnakeBody body = new SnakeBody();
    snake.add(body);
    addObject(body, 2, 2);
```



Movement

```

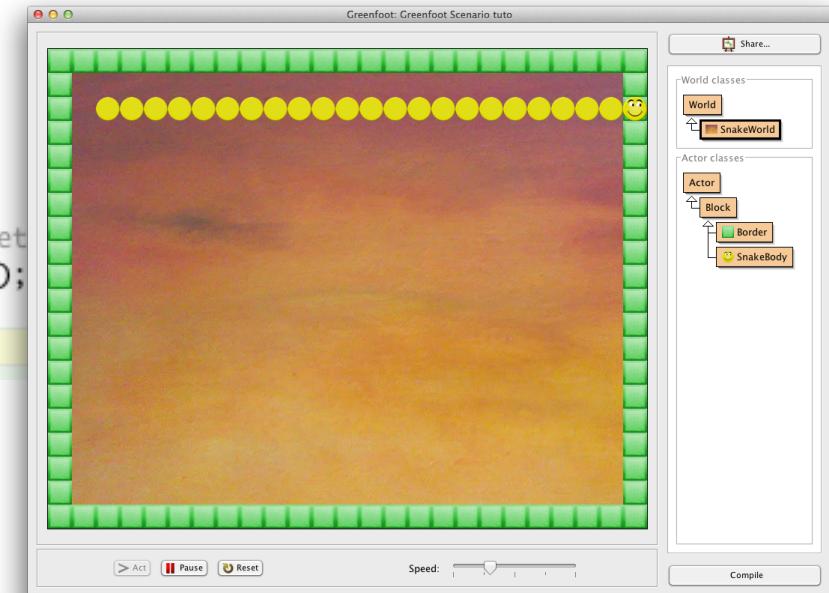
public class SnakeWorld extends World
{
    private LinkedList<SnakeBody> snake = new LinkedList<SnakeBody>();
    private int dx = 1;
    private int dy = 0;

    public void act()
    {
        //on remplace l'image de la tête
        SnakeBody head = snake.getLast();
        head.setImage("tail.png");

        //crée une nouvelle tête
        SnakeBody newHead = new SnakeBody();
        int newHeadX = head.getX() + dx;
        int newHeadY = head.getY() + dy;

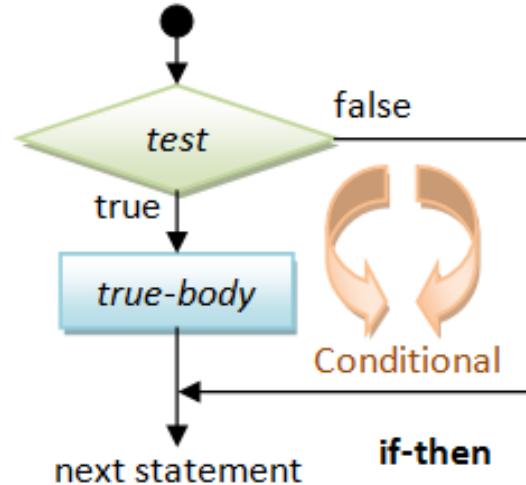
        //ajoute la nouvelle tête à la liste et
        addObject(newHead, newHeadX, newHeadY);
        snake.add(newHead);
    }
}

```

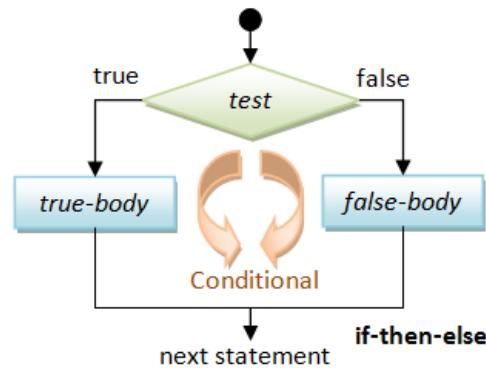


If Statement

```
if ( condition ) {
    true-body ;
}
```



```
if ( condition ) {
    true-body ;
} else {
    false-body ;
}
```



Limit the Snake size

```

public class SnakeWorld extends World
{
    private LinkedList<SnakeBody> snake = new LinkedList<SnakeBody>();
    private int dx = 1;
    private int dy = 0;
    private int tailCounter = 5;

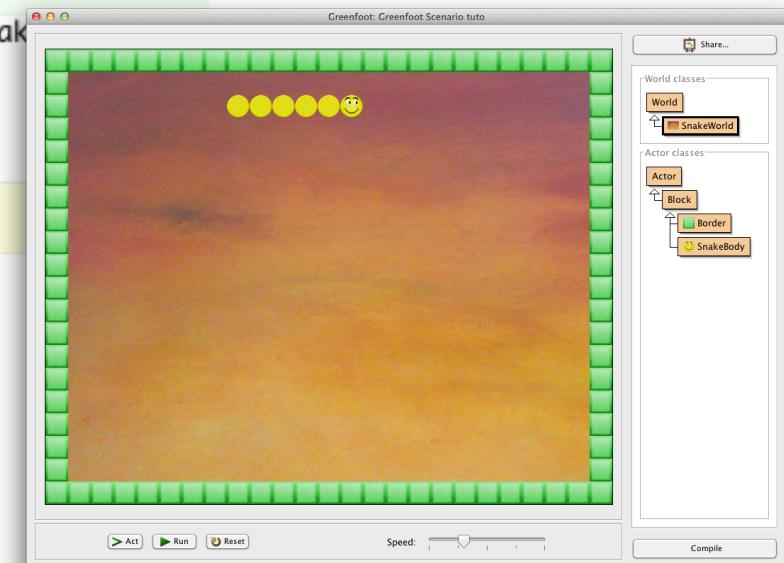
    public void act()
    {
        //on remplace l'image de la tête
        SnakeBody head = snake.getLast();
        head.setImage("tail.png");

        //crée une nouvelle tête
        SnakeBody newHead = new SnakeBody();
        int newHeadX = head.getX() + dx;
        int newHeadY = head.getY() + dy;

        //ajoute la nouvelle tête à la liste et au world
        addObject(newHead, newHeadX, newHeadY);
        snake.add(newHead);

        if (tailCounter == 0) {
            SnakeBody tail = snake.removeFirst();
            removeObject(tail);
        } else {
            tailCounter--;
        }
    }
}

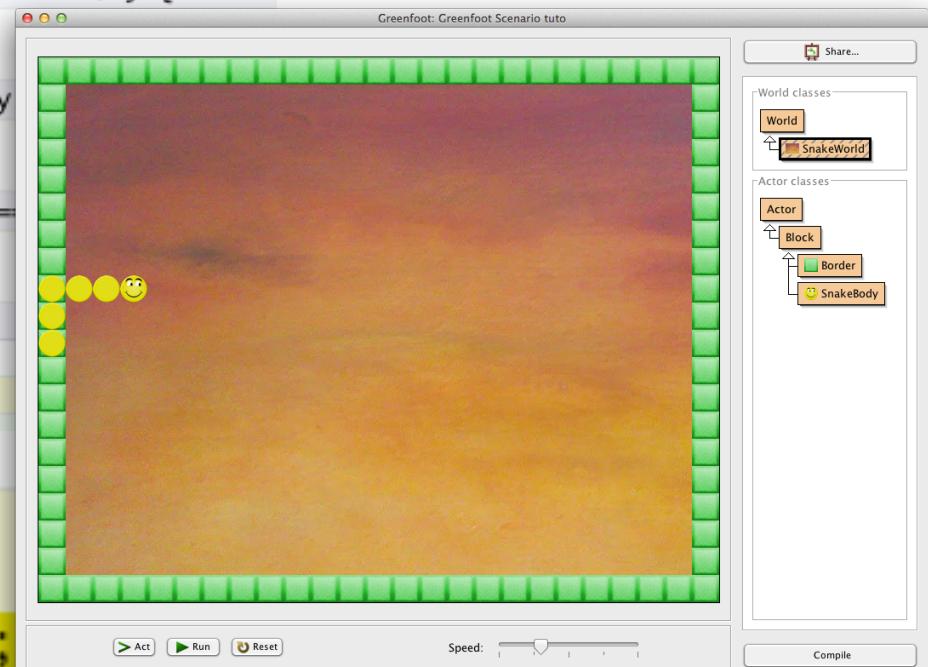
```



Change of direction

```
private void changeDirection() {
    if (Greenfoot.isKeyDown("left") && dx == 0 ) {
        dx = -1;
        dy = 0;
    } else if (Greenfoot.isKeyDown("right") && dx == 0 ) {
        dx = 1;
        dy = 0;
    } else if (Greenfoot.isKeyDown("down") && dy == 0) {
        dx = 0;
        dy = 1;
    } else if (Greenfoot.isKeyDown("up") && dy == 0) {
        dx = 0;
        dy = -1;
    }
}
```

```
public void act()
{
    changeDirection();
    //on remplace l'image de l'
```



Manage collisions (1)

```
public class SnakeWorld extends World
{
    private LinkedList<SnakeBody> snake = new LinkedList<SnakeBody>();
    private int dx = 1;
    private int dy = 0;
    private int tailCounter = 5;
    private boolean dead = false;
```

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

```
            if (dead) {
                return;
            }
```

```
            changeDirection();
```

```
        public void dead() {
```

```
            dead = true;
        }
```

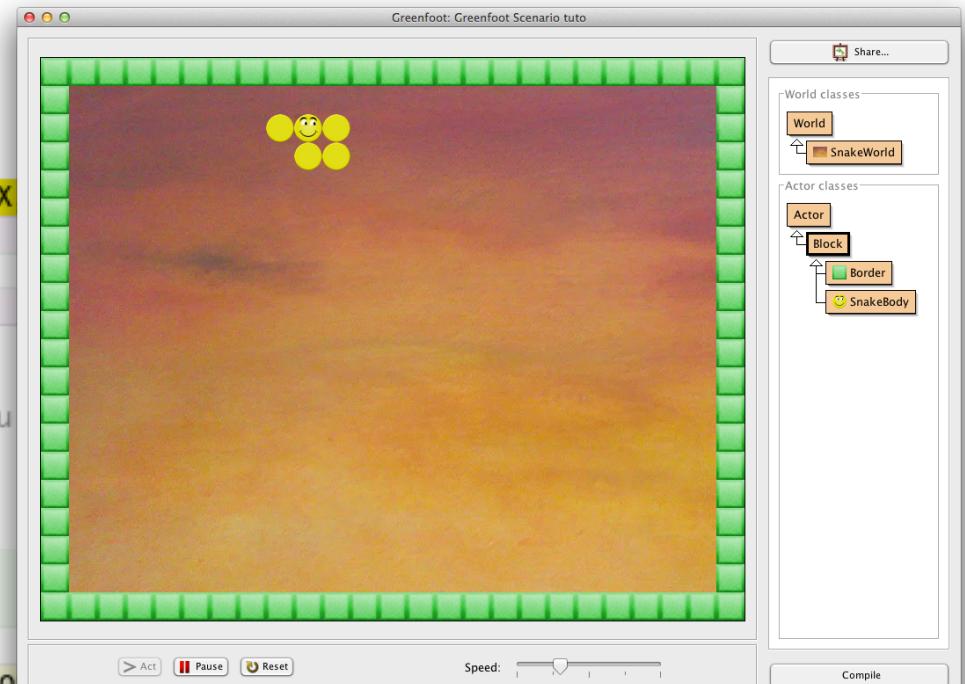
Manage collisions (2)

```
//crée une nouvelle tête
SnakeBody newHead = new SnakeBody();
int newHeadX = head.getX() + dx;
int newHeadY = head.getY() + dy;

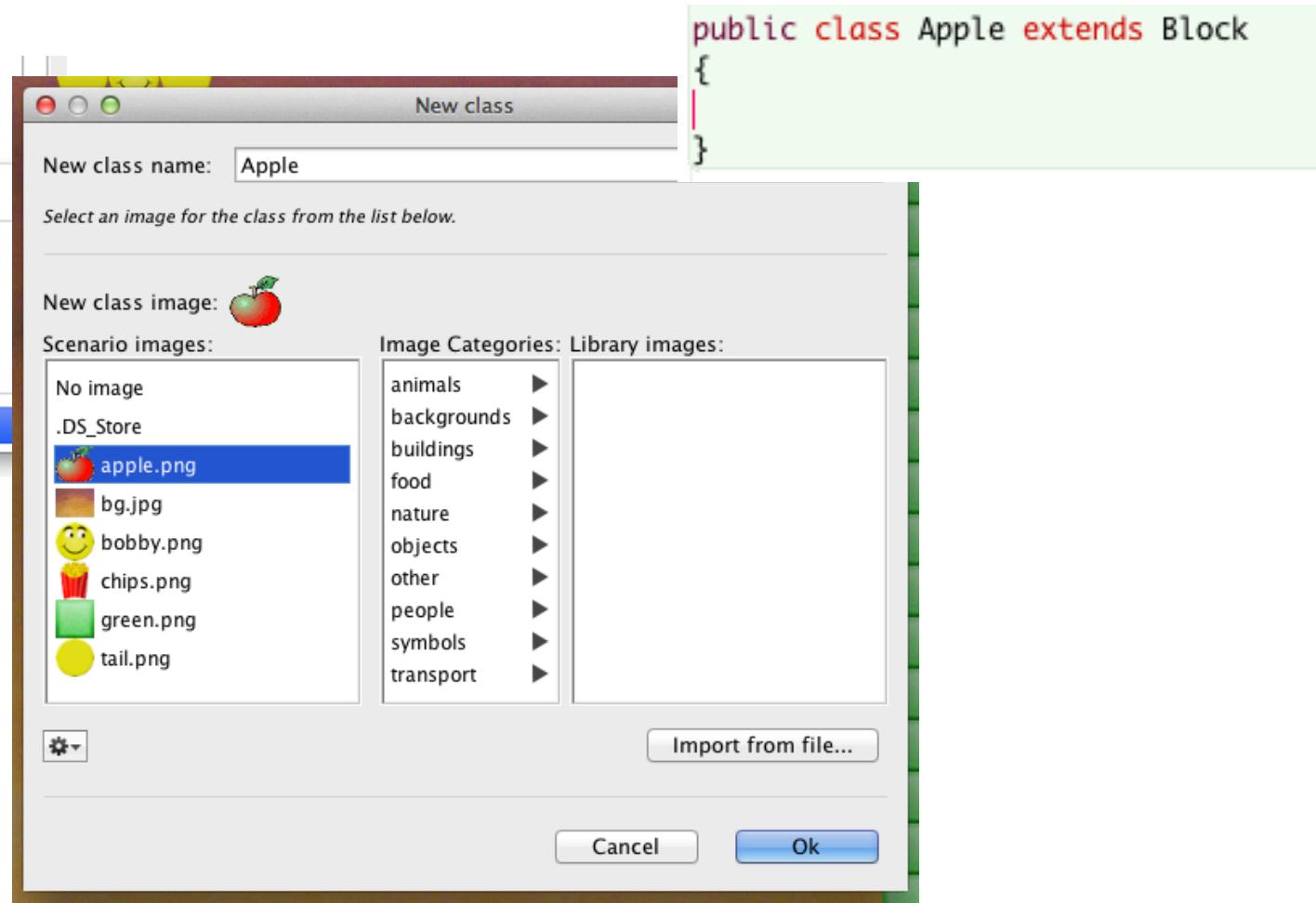
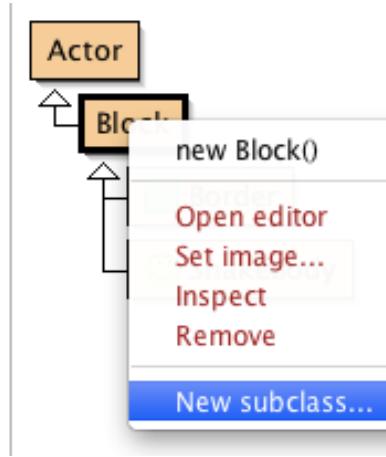
List<Block> blocks = getObjectsAt(newHeadX);
for(Block block : blocks) {
    block.collision(this);
}
```

```
//ajoute la nouvelle tête à la liste et au
addObject(newHead, newHeadX, newHeadY);
snake.add(newHead);
```

```
public class Block extends Actor
{
    public void collision(SnakeWorld world)
        world.dead();
}
```



Add Apples (1)



Add Apples (2)

```
public SnakeWorld()
{
    super(25, 20, 32);

    SnakeBody body = new SnakeBody();
    snake.add(body);
    addObject(body, 2, 2);

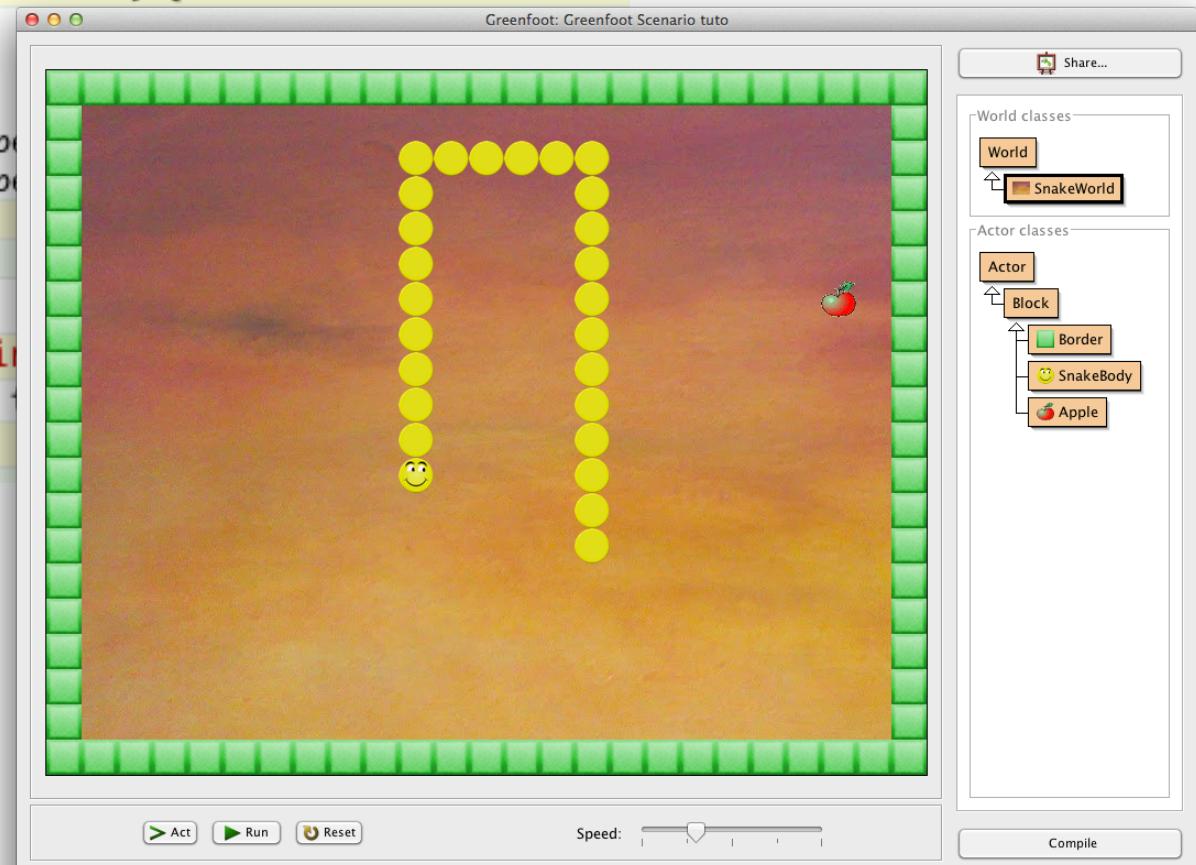
    Apple apple = new Apple();
    addObject(apple, Greenfoot.getRandomNumber(25),
              Greenfoot.getRandomNumber(20));
}
```



Collision with an apple

```
public class Apple extends Block
{
    public void collision(SnakeWorld world) {
        world.grow(2);
        setLocation(
            Greenfoot.getRandomNumber(
                Greenfoot.getRandomNumber(
                    ...
    }

    public void grow(int i)
    {
        tailCounter = ...
    }
}
```



Add sounds

```
public void collision(SnakeWorld world) {  
    Greenfoot.playSound("slurp.mp3");  
    world.grow(2);  
    setLocation(  
        Greenfoot.getRandomNumber(getWorld().getWidth()-2)+1,  
        Greenfoot.getRandomNumber(getWorld().getHeight()-2)+1);  
}
```

```
public void collision(SnakeWorld world) {  
    Greenfoot.playSound("dead.mp3");  
    world.dead();  
}
```

Congratulations!

- You have completed your first Greenfoot game!
- You can now start to add new features in your game or start creating new games !