

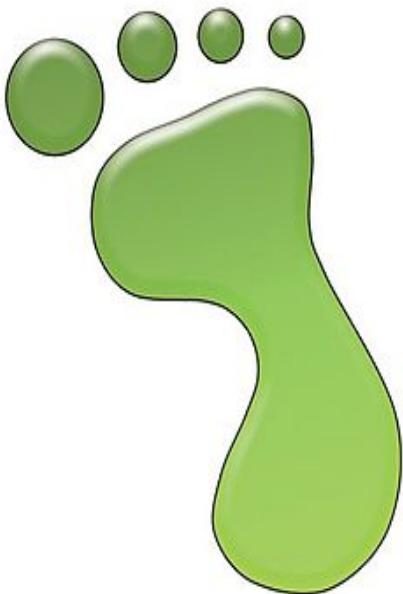


Greenfoot Workshop

Bobby - Snake

Greenfoot

- Environnement de développement basé sur le langage de programmation Java.
- Principalement conçu pour faciliter l'apprentissage du langage Java
- Permet de facilement créer des jeux en 2D.
- Greenfoot est un logiciel « libre » (gratuit) sous Microsoft Windows, Mac OS X, ou Linux.
- <http://www.greenfoot.org/>





Le langage Java



- Langage « orienté objet »
- Un des langages les plus utilisés dans le monde
- Permet d'écrire des programmes qui fonctionnent sur différents systèmes
 - Windows, Mac, Linux, Android...

« Orienté Objet »

- Une application Java est constituée « d'objets »
- Chaque objet est constitué de deux parties
 - un ensemble de propriétés internes
 - son « comportement », les actions qu'il peut faire

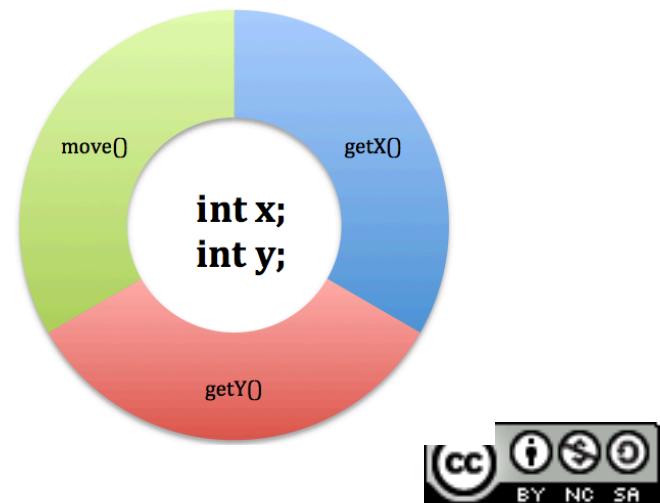
Classes et Objets

- Pour décrire un **objet**, on va définir une **classe**.
- Une **classe** est une sorte de « **modèle** » qui permet de construire des objets
 - On dit aussi que ces objets sont des « **instances** » de la classe



Une classe Java

```
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

Greenfoot: Greenfoot Scenario

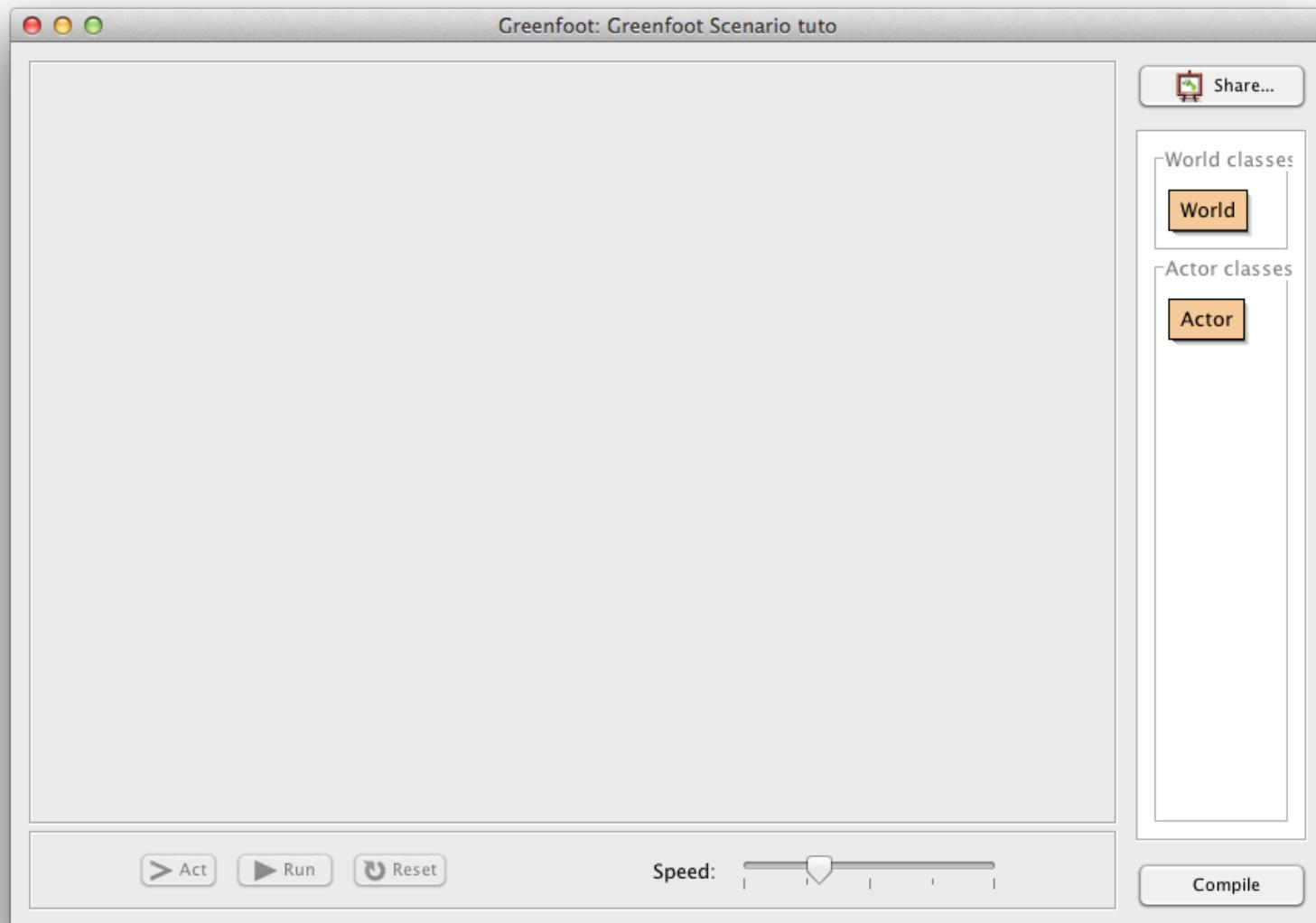
The screenshot shows a Greenfoot scenario titled "Greenfoot: Greenfoot Scenario". The main area is a 2D game board with a green border made of blocks. Inside, there's a brown floor with a red gradient. A yellow snake is on the board, consisting of 14 segments. One segment has a smiley face. A single red apple is located in the bottom right corner. In the bottom left corner, there's a green bar displaying the text "SCORE: 140". At the bottom of the screen are three buttons: "Act" (with a play icon), "Run" (with a play icon), and "Reset" (with a shield icon). To the right of these buttons is a "Speed:" slider. On the far right is a sidebar with class hierarchies:

- World classes**: World (orange box) which contains SnakeWorld (brown box).
- Actor classes**: Actor (orange box) which contains Score (brown box), Block (brown box), Border (green box), Apple (red box), and SnakeBody (yellow box).
- Other classes**: Snake (orange box).

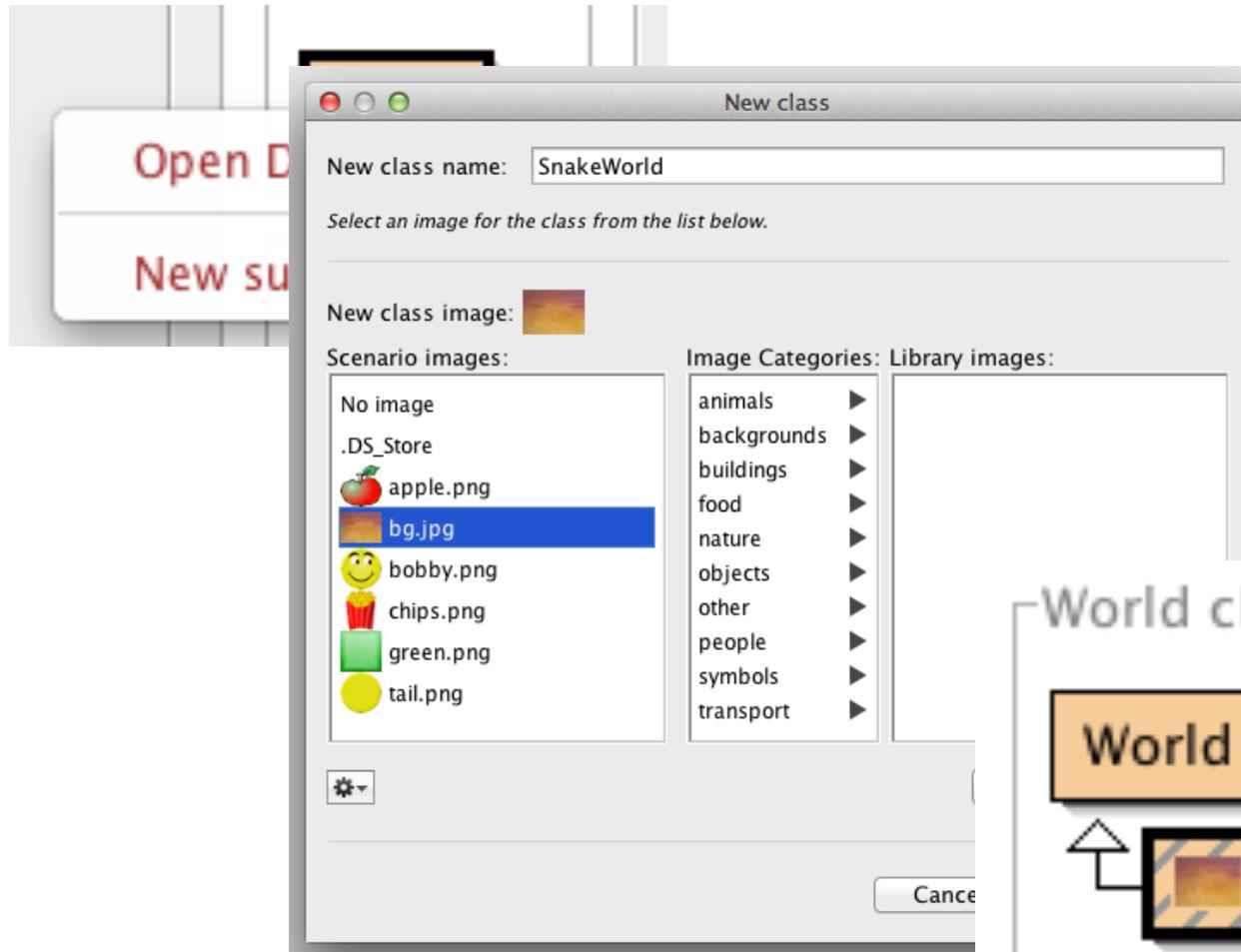
At the very bottom right of the slide is a Creative Commons license logo with the text "BY NC SA".



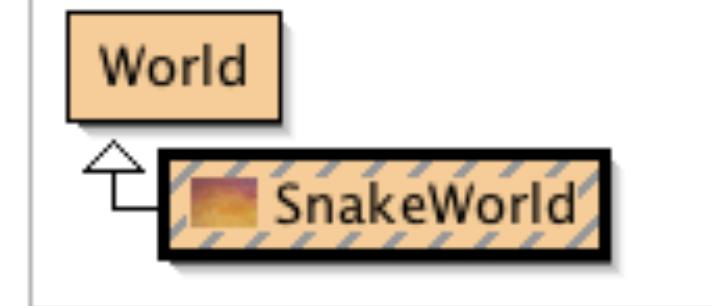
Nouveau scénario



Création du World

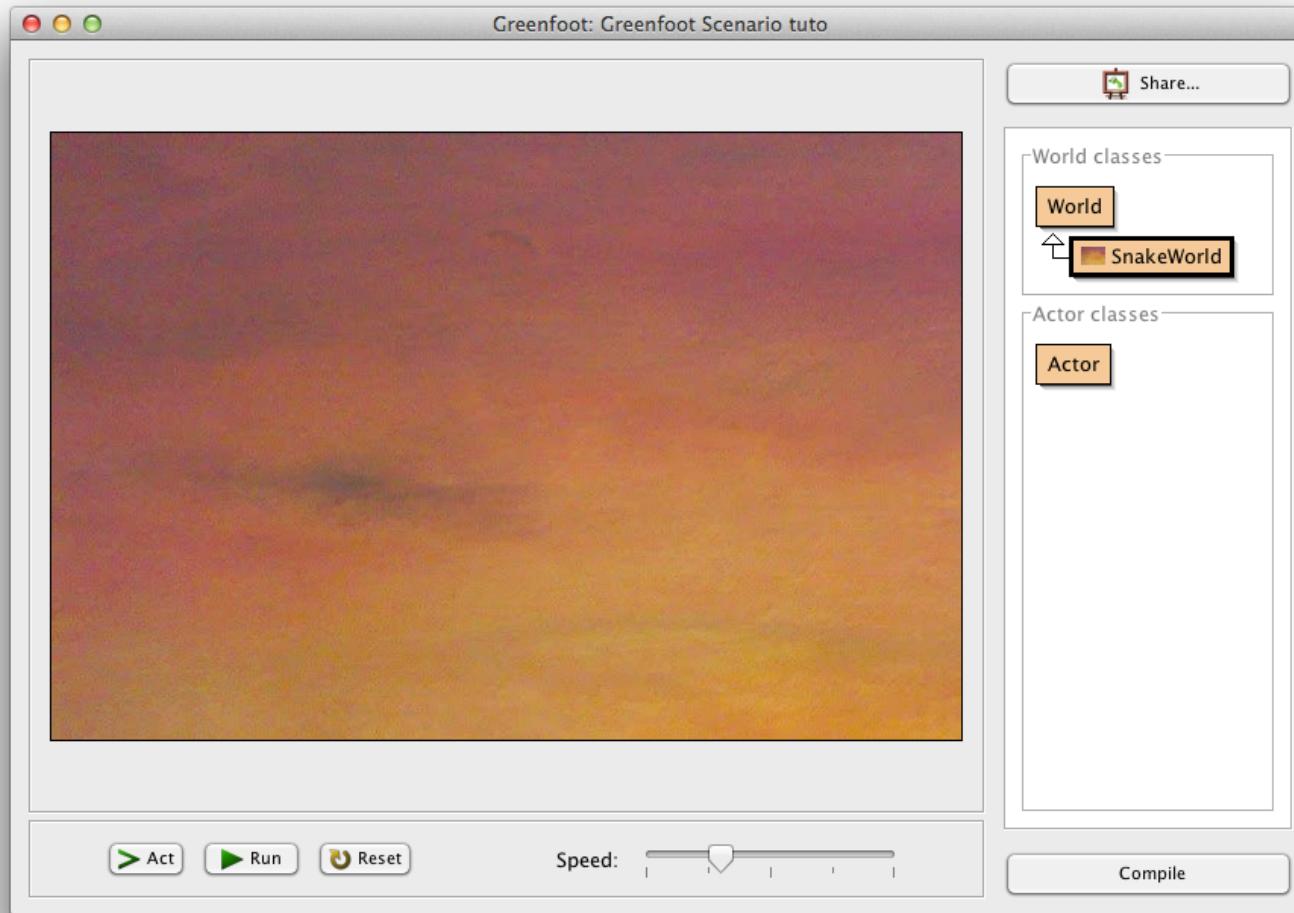


World classes





Snake World





Le code de SnakeWorld

SnakeWorld

Compile Undo Cut Copy Paste Find... Close Source Code

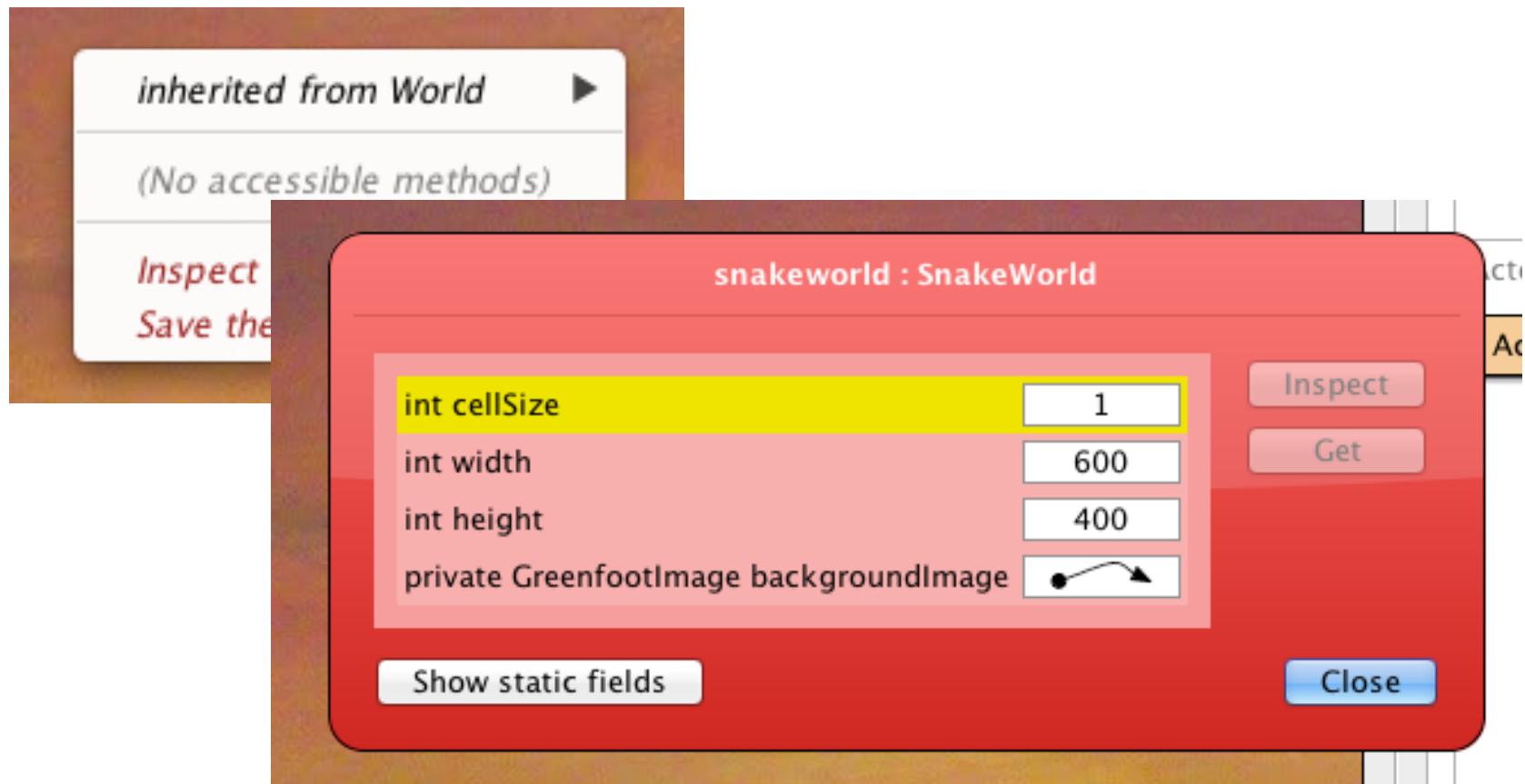
```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * 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);
    }
}
```

saved

Inspect World

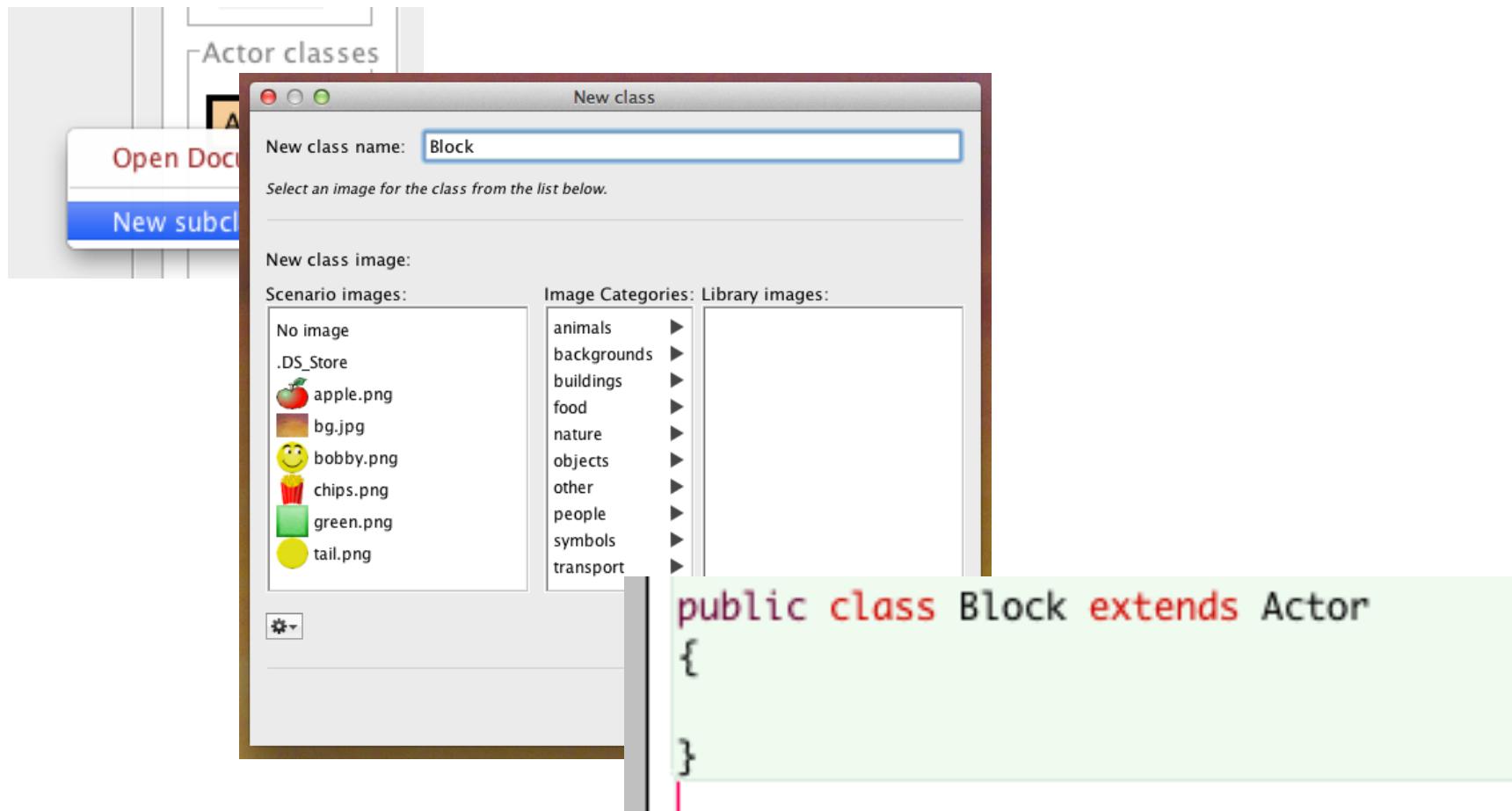


Changer le code

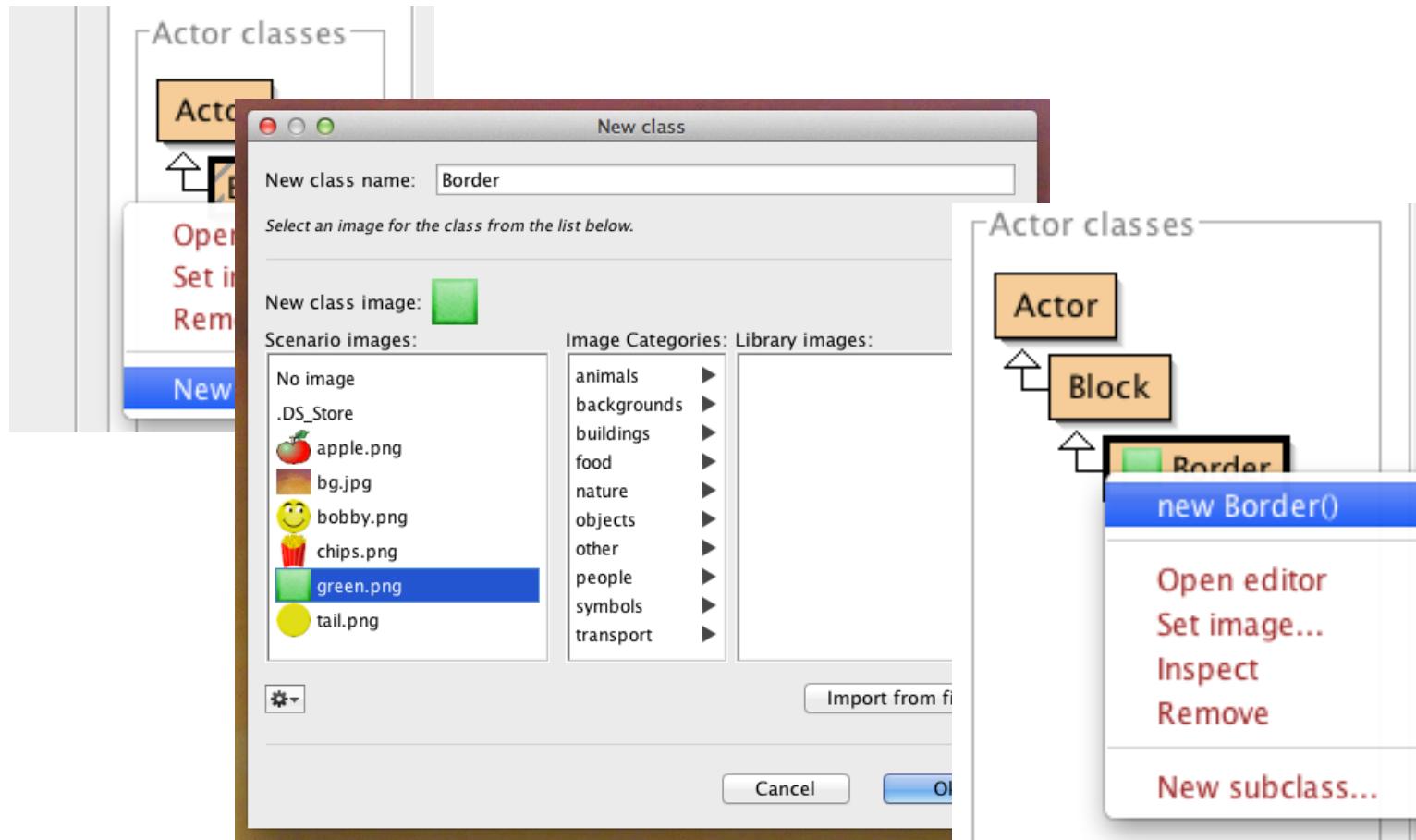
- Nous allons utiliser des blocs de 32 pixels dans notre jeu
- La taille du jeu sera de 25×20 blocs

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

Créer un bloc



Création du Border

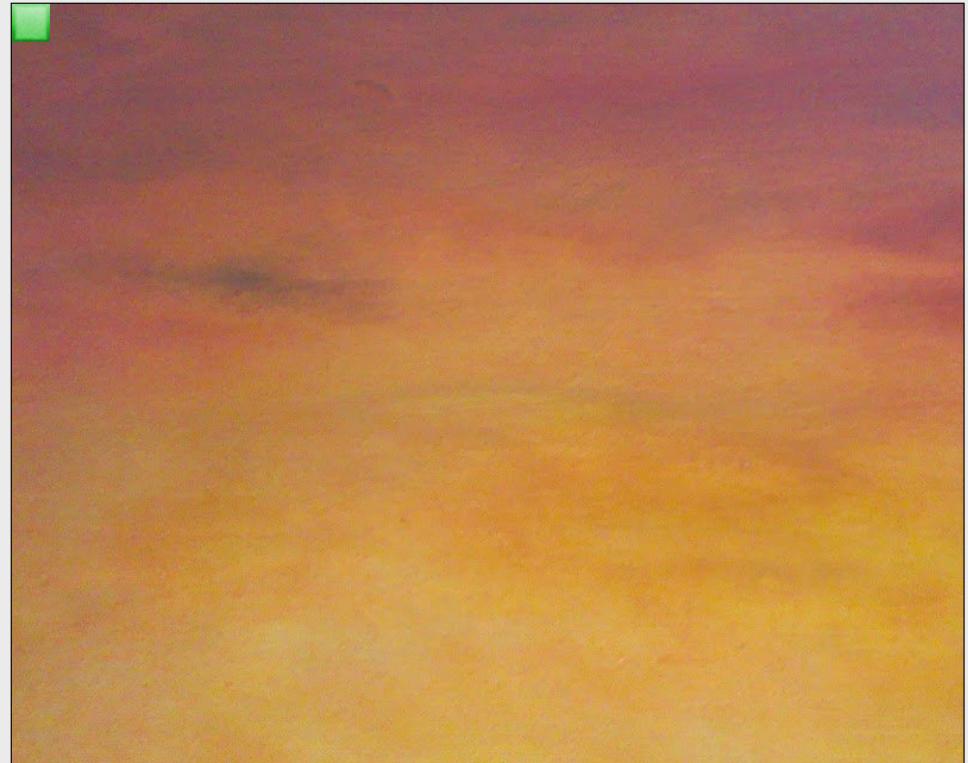


Coordonées

(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)

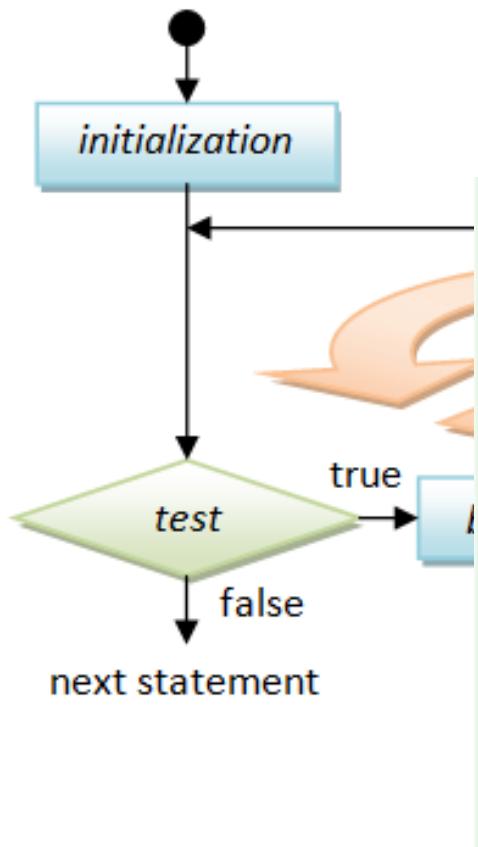
Afficher les bords

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



Les boucles for

```
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);
    }
}
```



Les bords du jeu

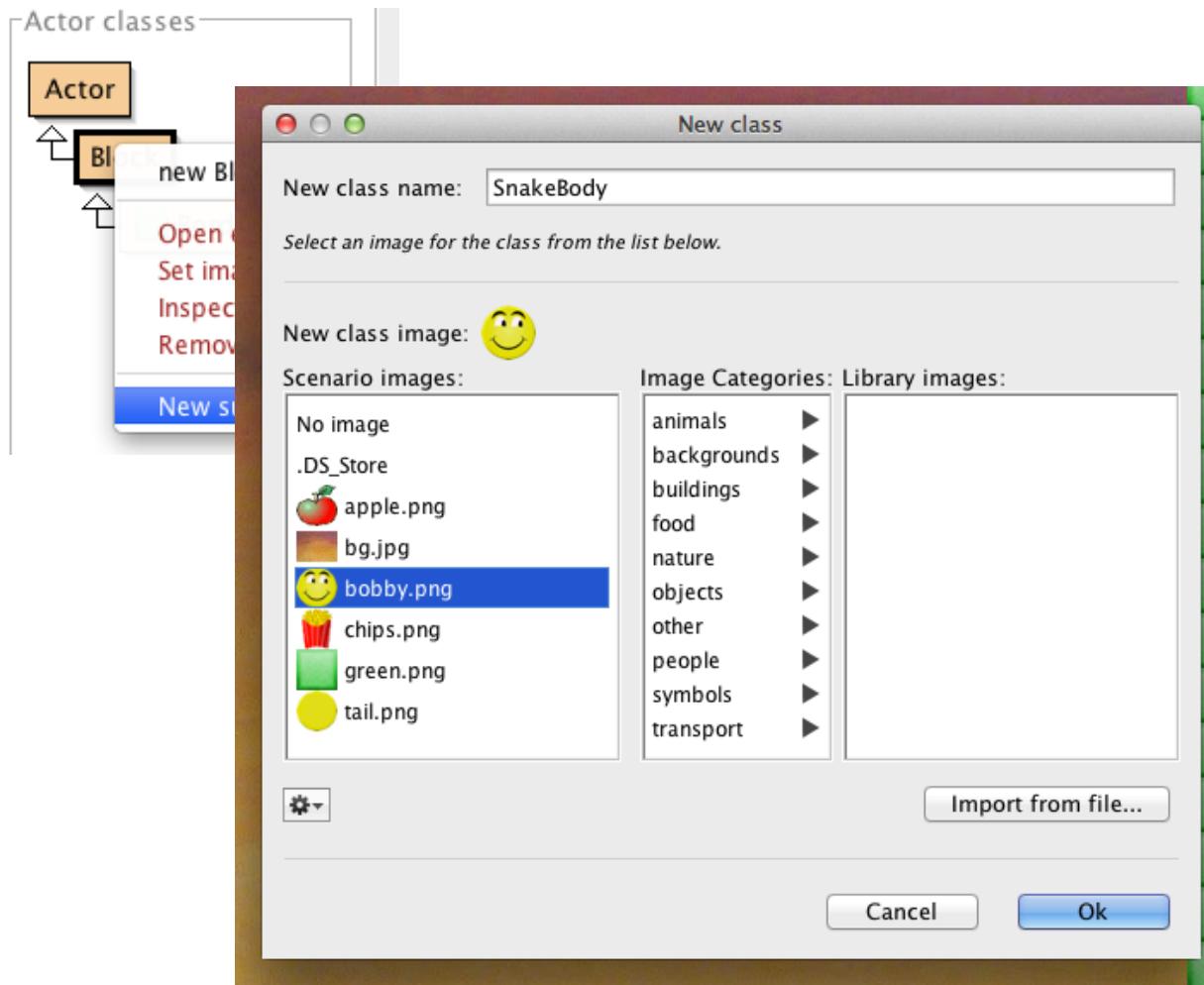
Greenfoot: Greenfoot Scenario tuto

The screenshot shows the Greenfoot application window. On the left is a simulation area with a brown gradient background and a green border made of green rectangular blocks. At the bottom of the simulation area, there is a faint watermark of a person's face. On the right side of the window, there is a sidebar with the following sections:

- World classes**: A tree view showing "World" at the top, which has "SnakeWorld" as a child node.
- Actor classes**: A tree view showing "Actor" at the top, which has "Block" as a child node, and "Block" has "Border" as a child node.

At the bottom of the window, there are several buttons: "Act", "Run", "Reset", "Speed" (with a slider), "Compile", and "Share...".

Class SnakeBody



Modif SnakeWorld

```
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);
```





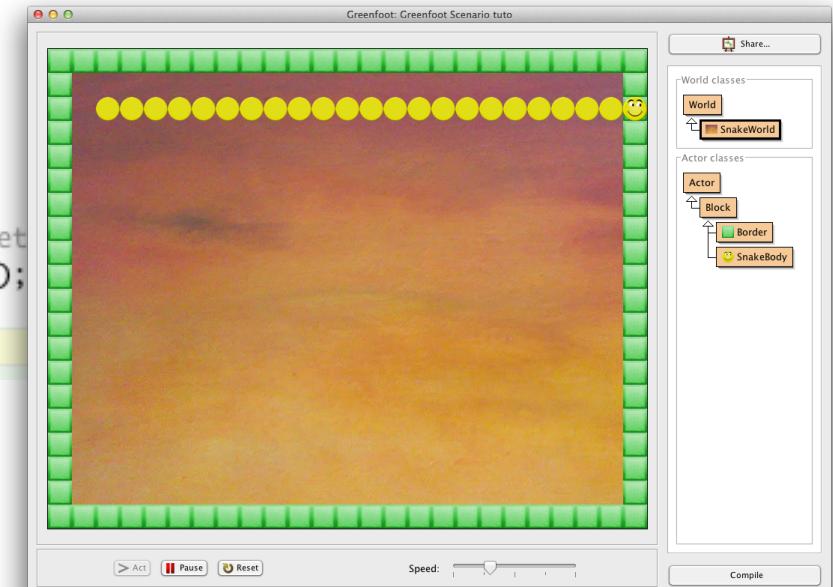
Déplacement

```
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");

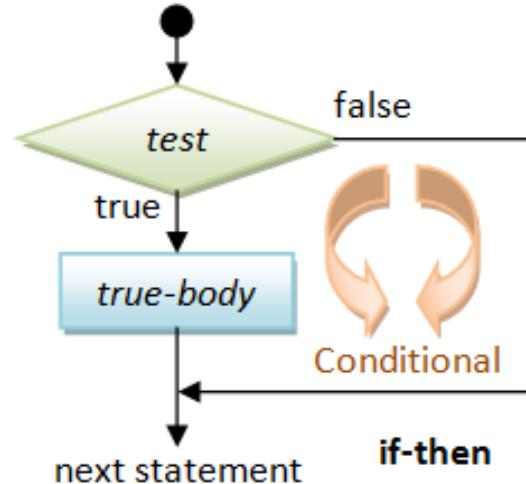
        //crée une nouvelle tête
        SnakeBody newHead = new SnakeBody();
        int newHeadX = head.getX() + dx;
        int newHeadY = head.getY() + dy;

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

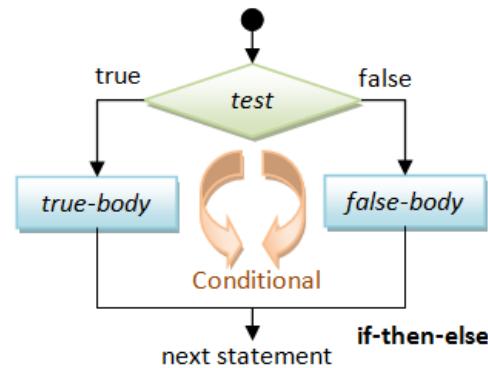


L'instruction « if »

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



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



Taille du serpent

```

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

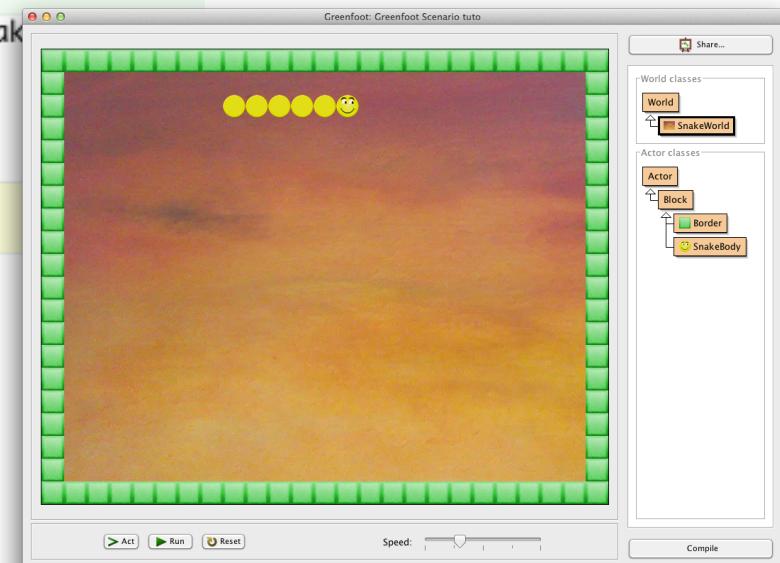
    public void act()
    {
        //on remplace l'image de la tête
        SnakeBody head = snake.getLast();
        head.setImage("tail.png");

        //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--;
        }
    }
}

```



Changement de direction

```

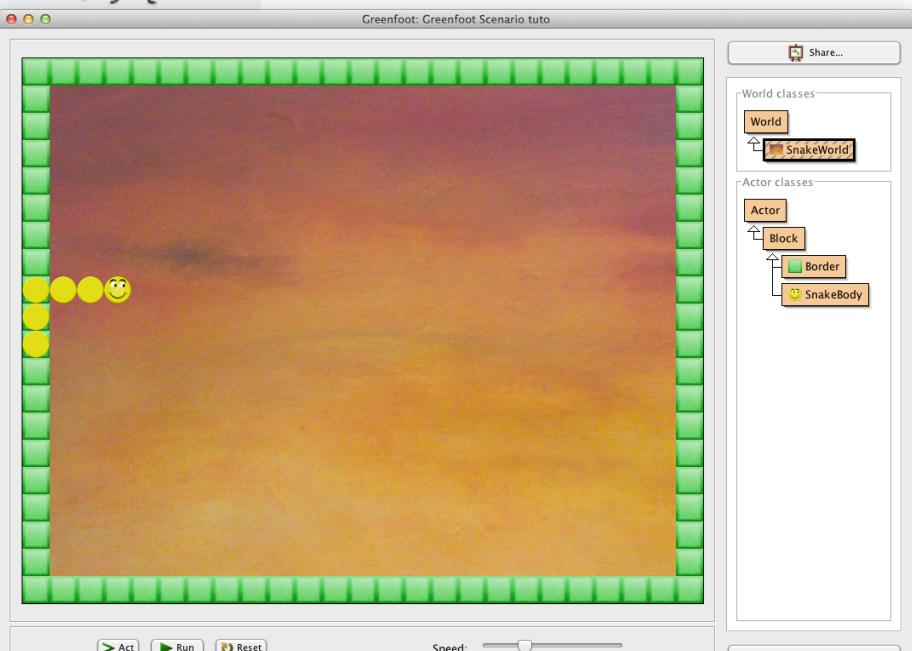
private void changeDirection() {
    if (Greenfoot.isKeyDown("left") && dx == 0 ) {
        dx = -1;
        dy = 0;
    } else if (Greenfoot.isKeyDown("right") && dx == 0 ) {
        dx = 1;
        dy = 0;
    } else if (Greenfoot.isKeyDown("down") && dy == 0) {
        dx = 0;
        dy = 1;
   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

```

```

public void act()
{
    changeDirection();
}

```



Les 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;
        }
```

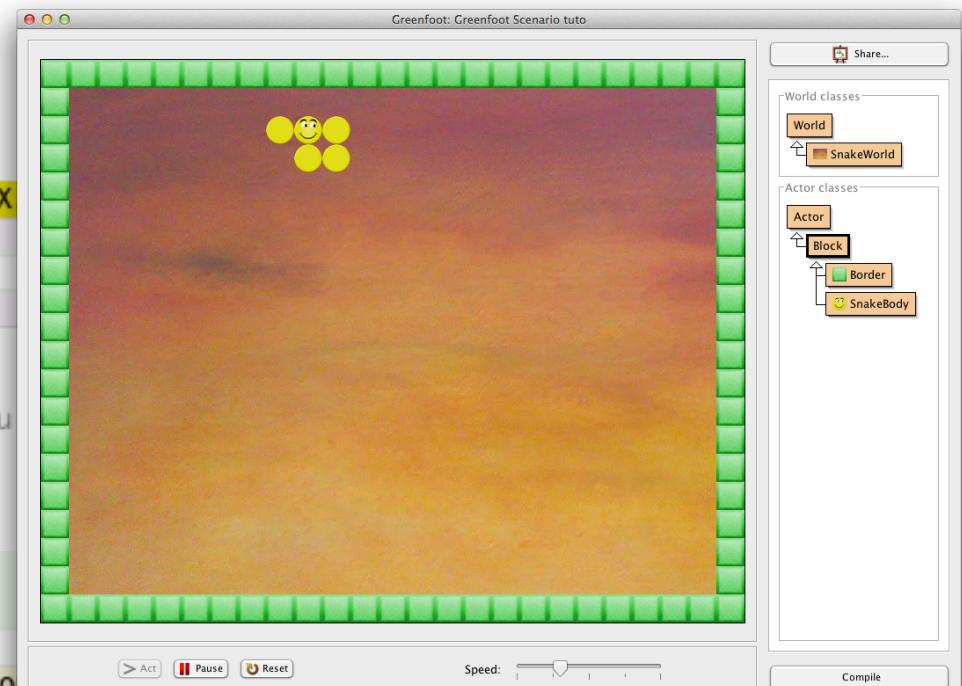
Les collisions (2)

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

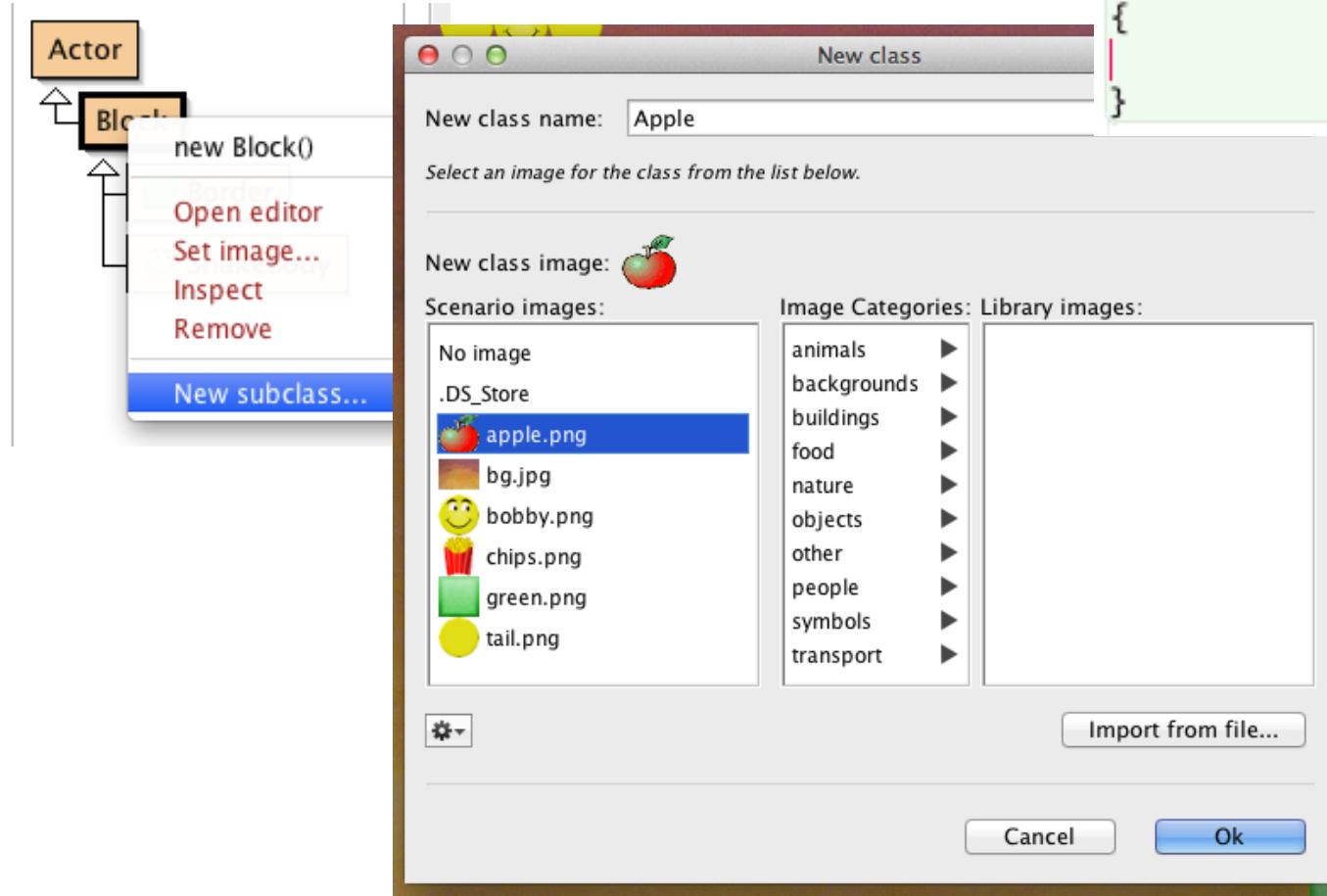
List<Block> blocks = getObjectsAt(newHeadX);
for(Block block : blocks) {
    block.collision(this);
}

//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();
}
```



Ajout d'une pomme (1)

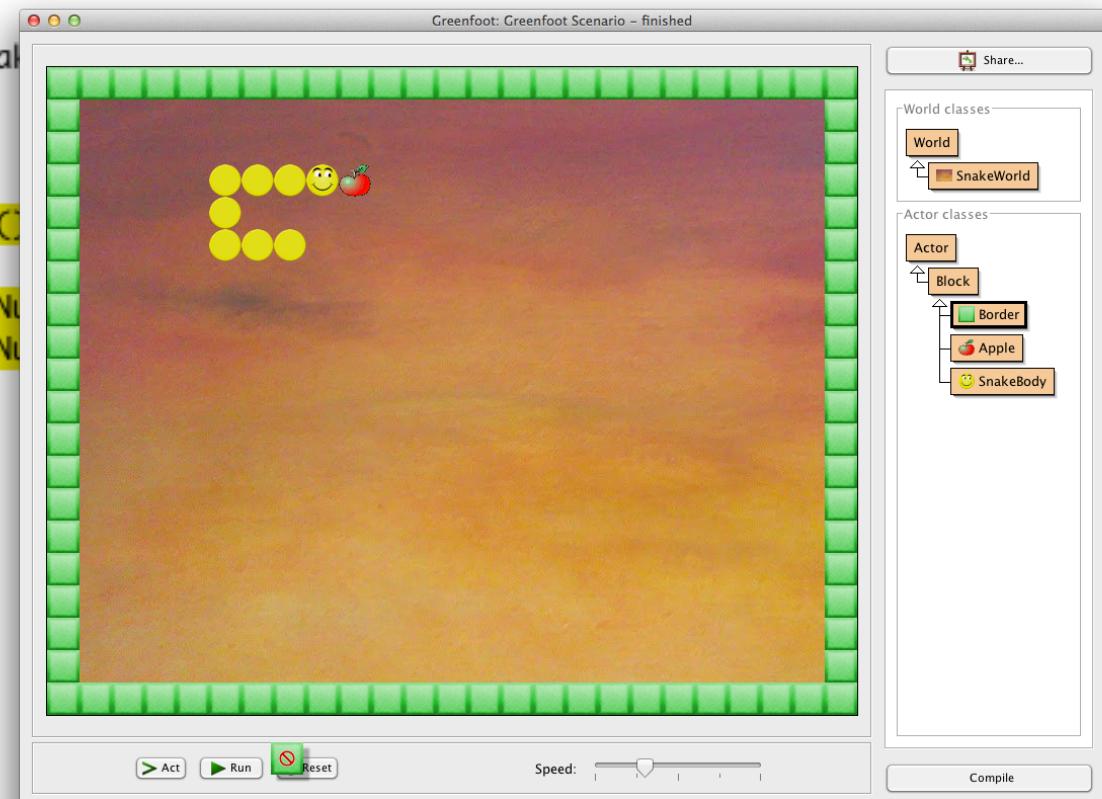


Ajout d'une pomme (2)

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

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

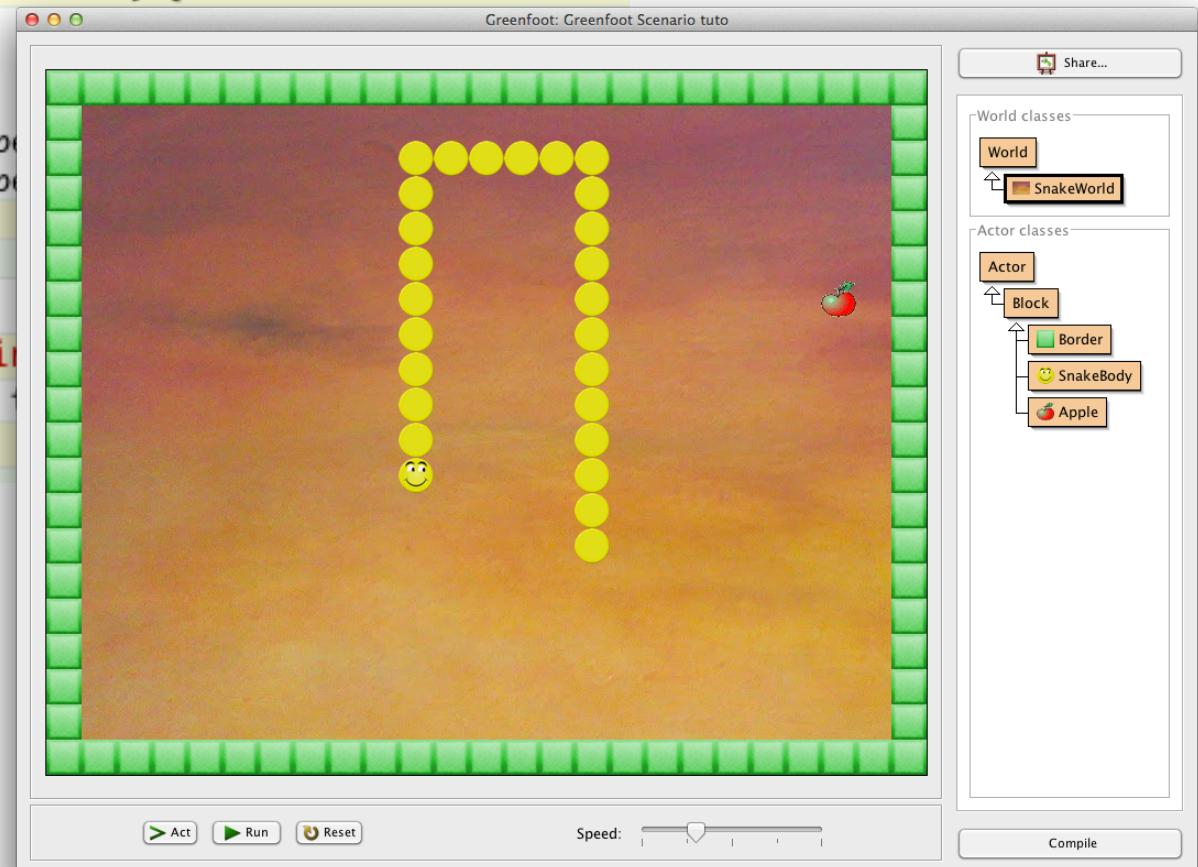
    Apple apple = new Apple();
    addObject(apple, Greenfoot.getRandomNumber(25),
              Greenfoot.getRandomNumber(20));
}
```



Collision d'une pomme

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

public void grow(int i)
 tailCounter = 1;
}



Ajouter du son

```
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();  
}
```



Félicitations !

- Vous avez terminé votre premier jeu Greenfoot !
- Vous pouvez maintenant continuer à apporter de nouvelles fonctionnalités dans votre jeu, ou en créer d'autres !

```
public class GameElement
```

```
private int x;  
private int y;
```

Définition des prop

```
public int getX()  
{  
    return x;  
}  
  
public int getY()  
{  
    return y;  
}  
  
public void setXY(int x, int y)  
{  
    this.x = x;  
    this.y = y;  
}
```

Définition

Les noms des classes commencent par une **majuscule**

Les propriétés sont généralement « privée » (private)
(Les autres objets n'y ont pas accès)

Les noms des propriétés commencent par une **minuscule**

La définition d'une propriété est composée du type de la donnée et du nom de la propriété

- Int : valeur entière (0, 1, 2, ...)

Un constructeur est comme une méthode, mais

- son nom est le même que celui de la classe,
- Il n'a pas de type de retour

Il est utilisé pour construire de nouvelles instances en utilisant une instruction new comme suit :
new GameElement(2,3);

Certaines méthodes ne renvoient rien. Dans ce cas, chaque argument est défini avec son type.

Dans ce cas, le type de retour est **void**