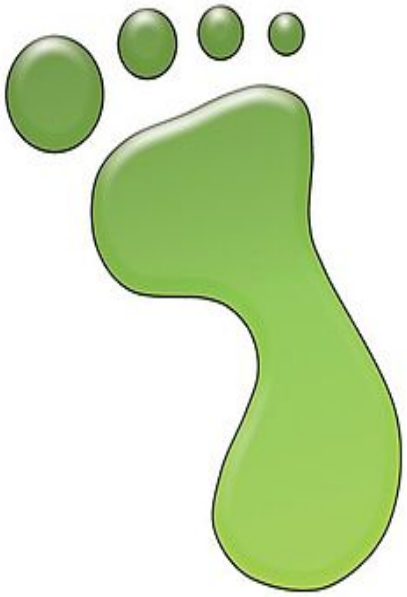
Greenfoot Workshop

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

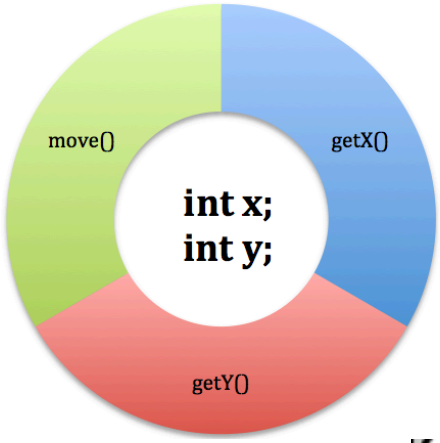
Classes et Objets

- Pour décrire un **objet**, on va définir une **classe**.

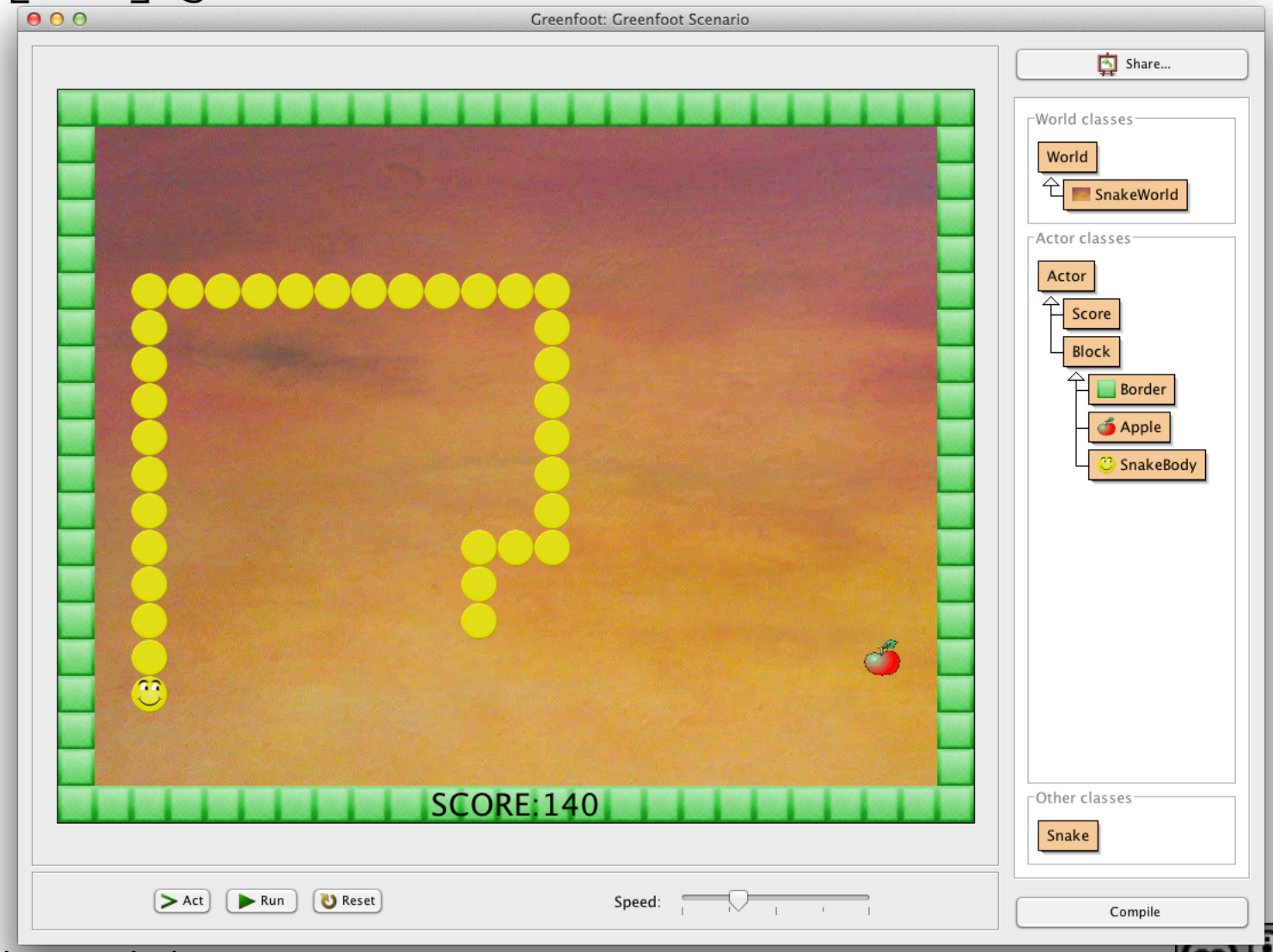
- Une **classe** est une sorte de « **modèle** » qui permet de construire des objets
  - On dit aussi que ces objets sont des « instances » de la classe

Une classe Java

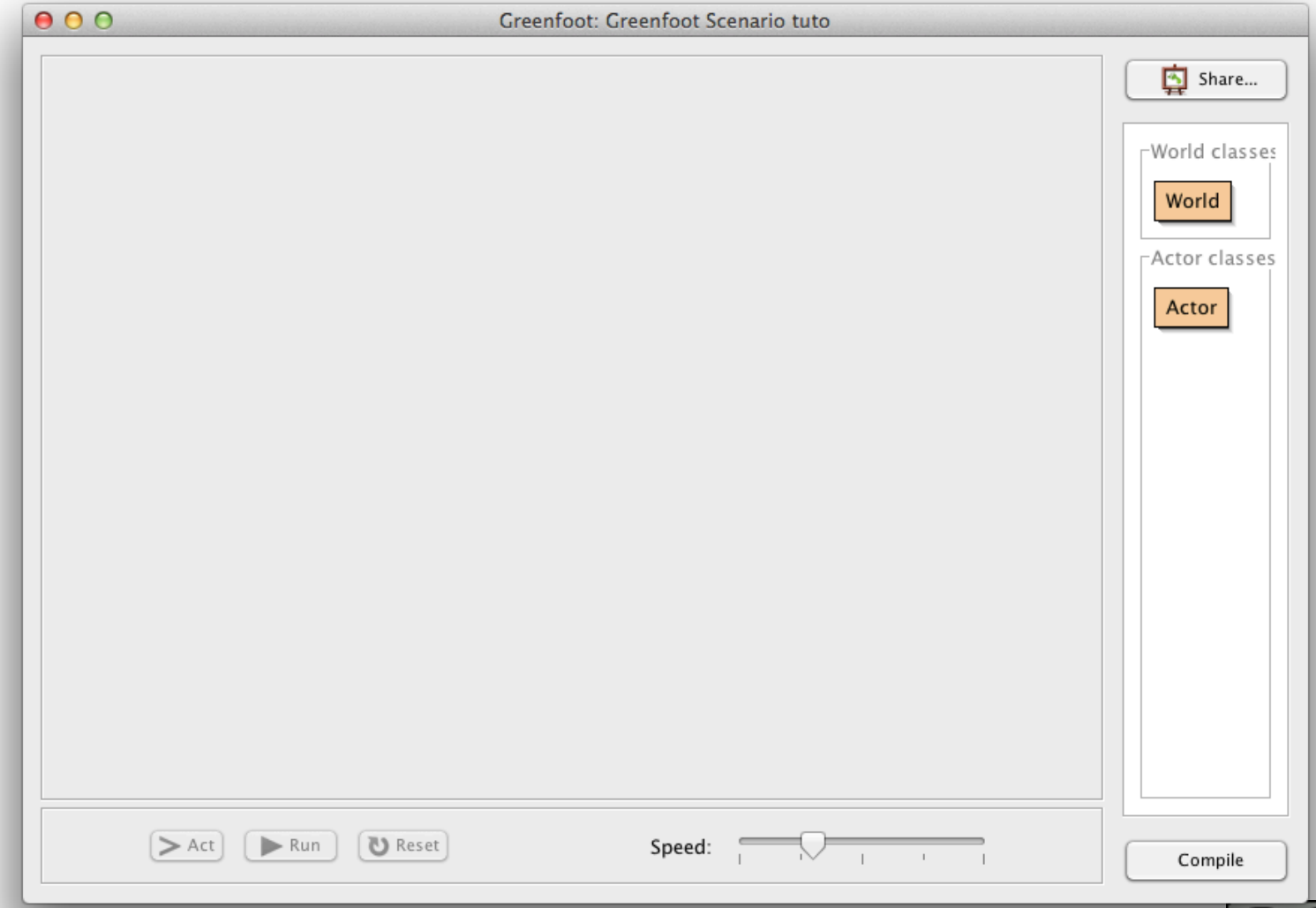
```
public class GameElement {

	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;
	}
}
```



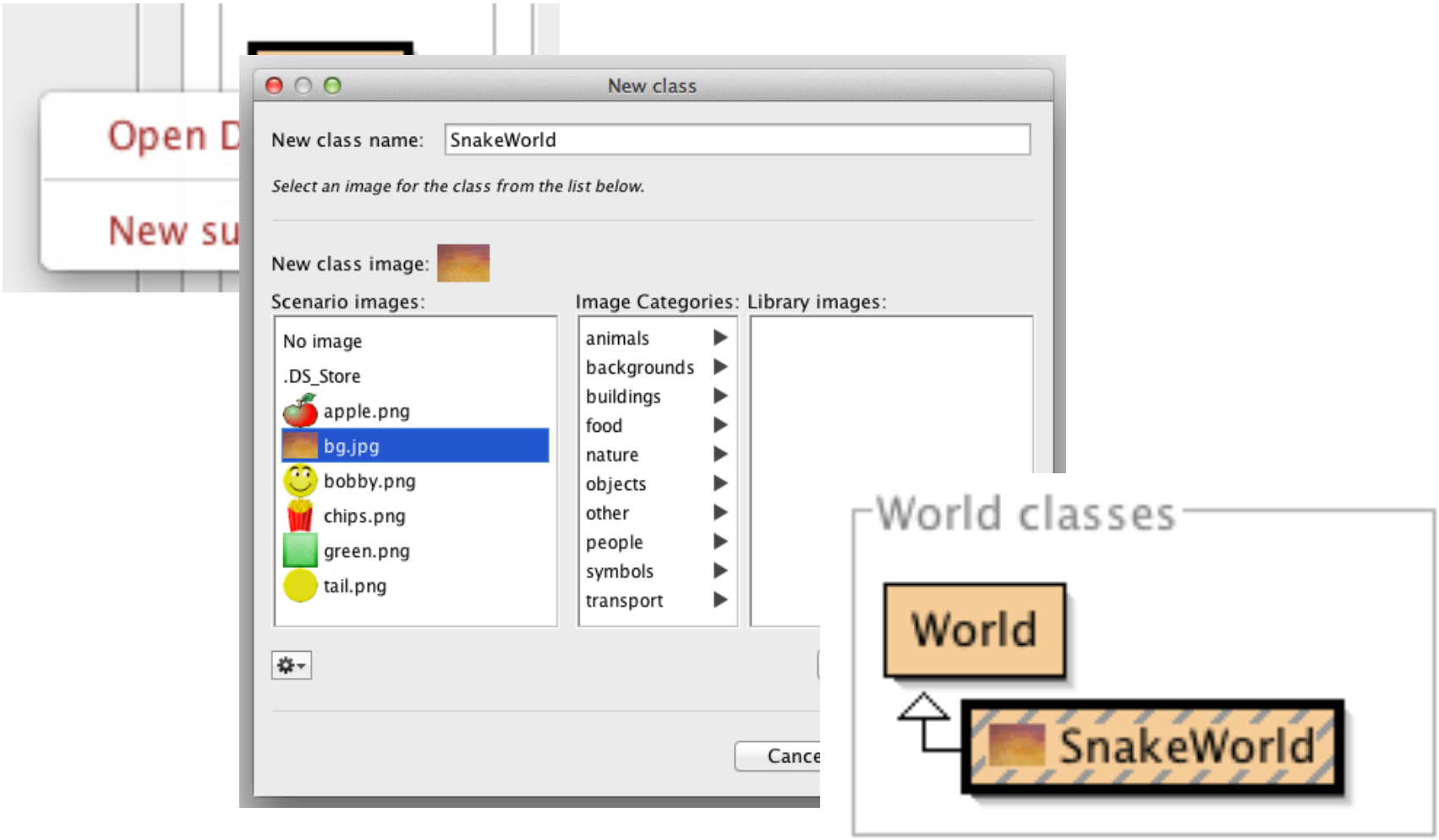
Bobby-Snake


Greenfoot: Greenfoot Scenario
Share...
World classes
World
SnakeWorld
Actor classes
Actor
Score
Block
Border
Apple
SnakeBody
Other classes
Snake
SCORE:140
Act
Run
Reset
Speed:
Compile

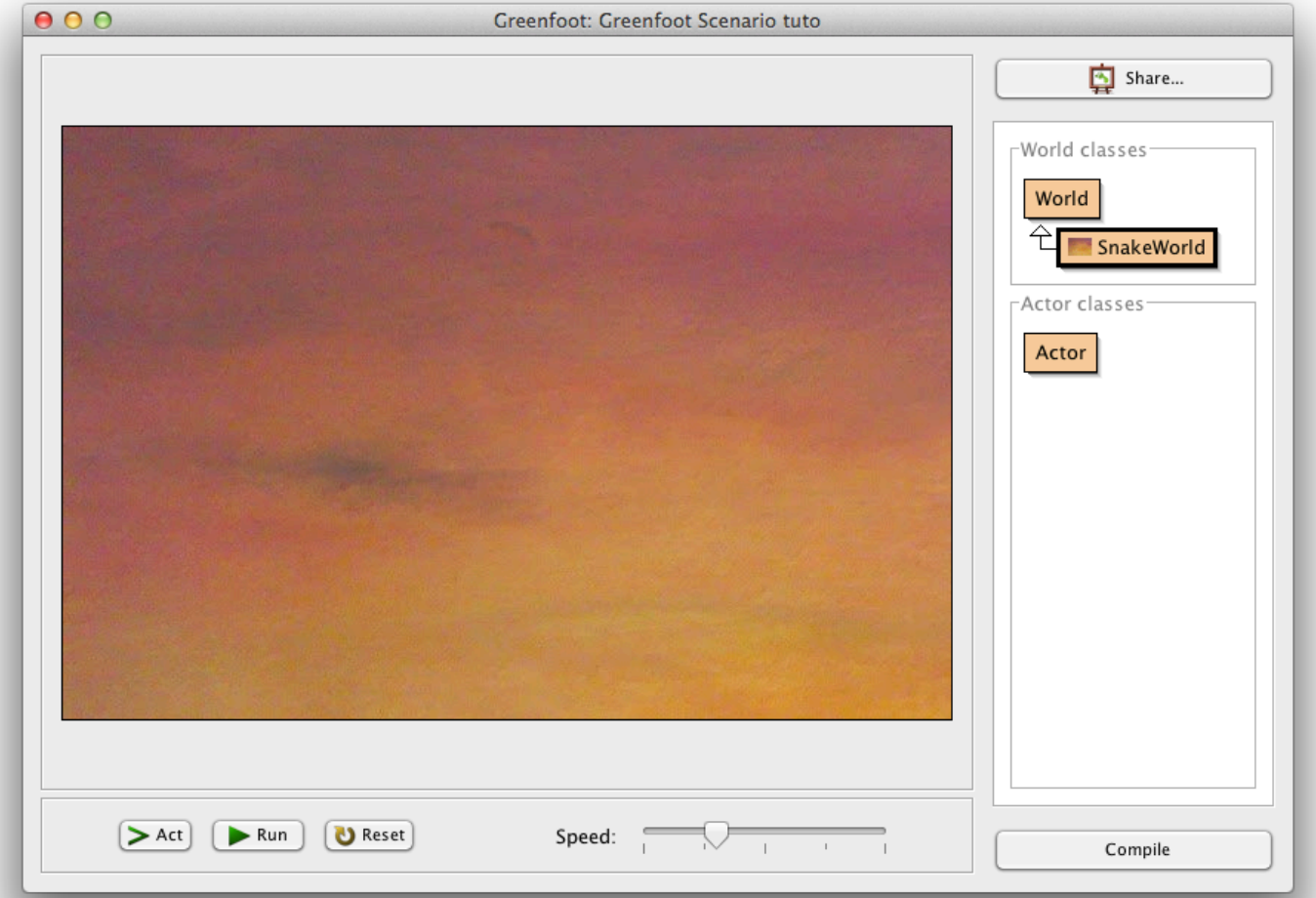
Nouveau scénario



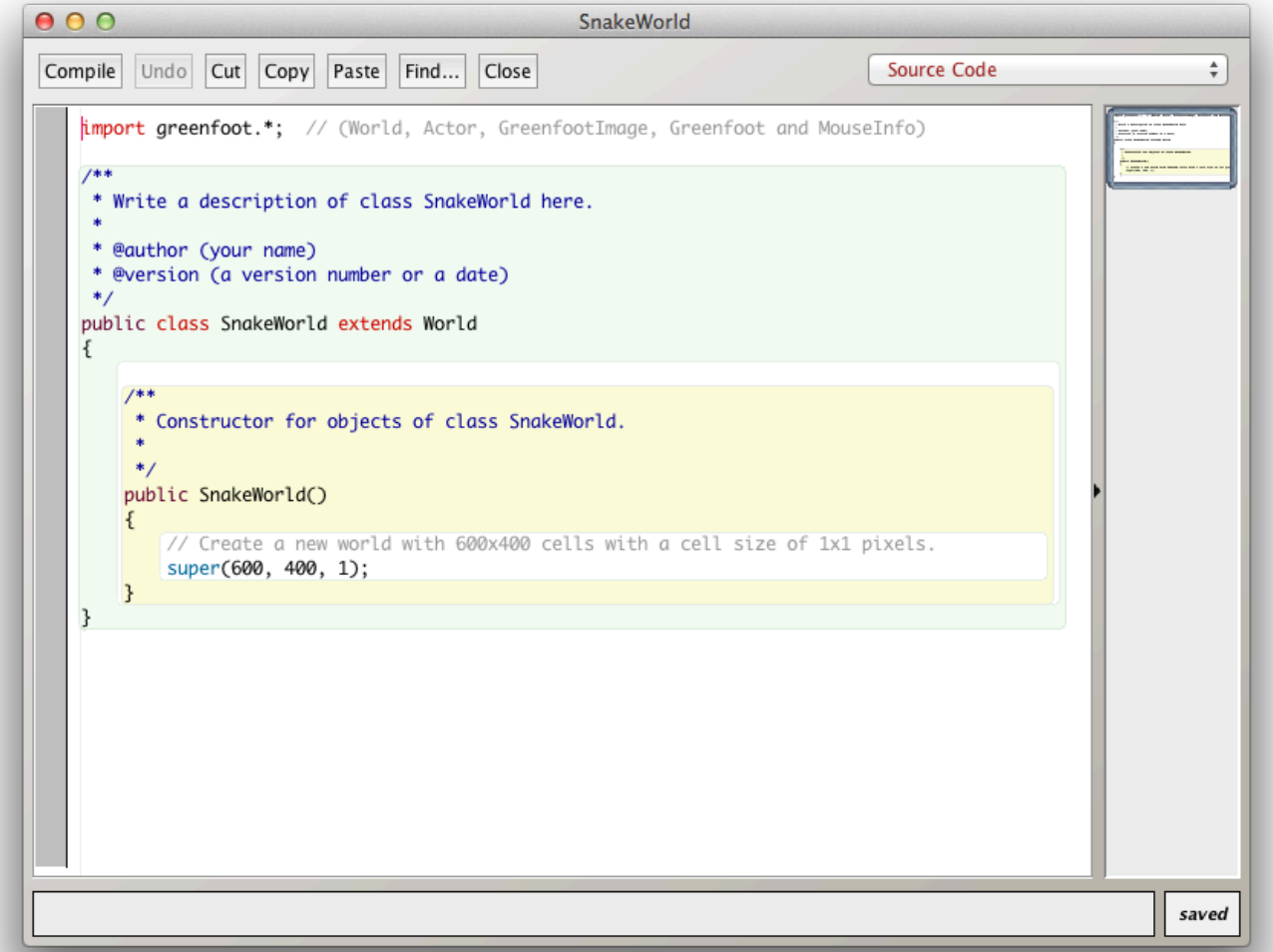
Création du World



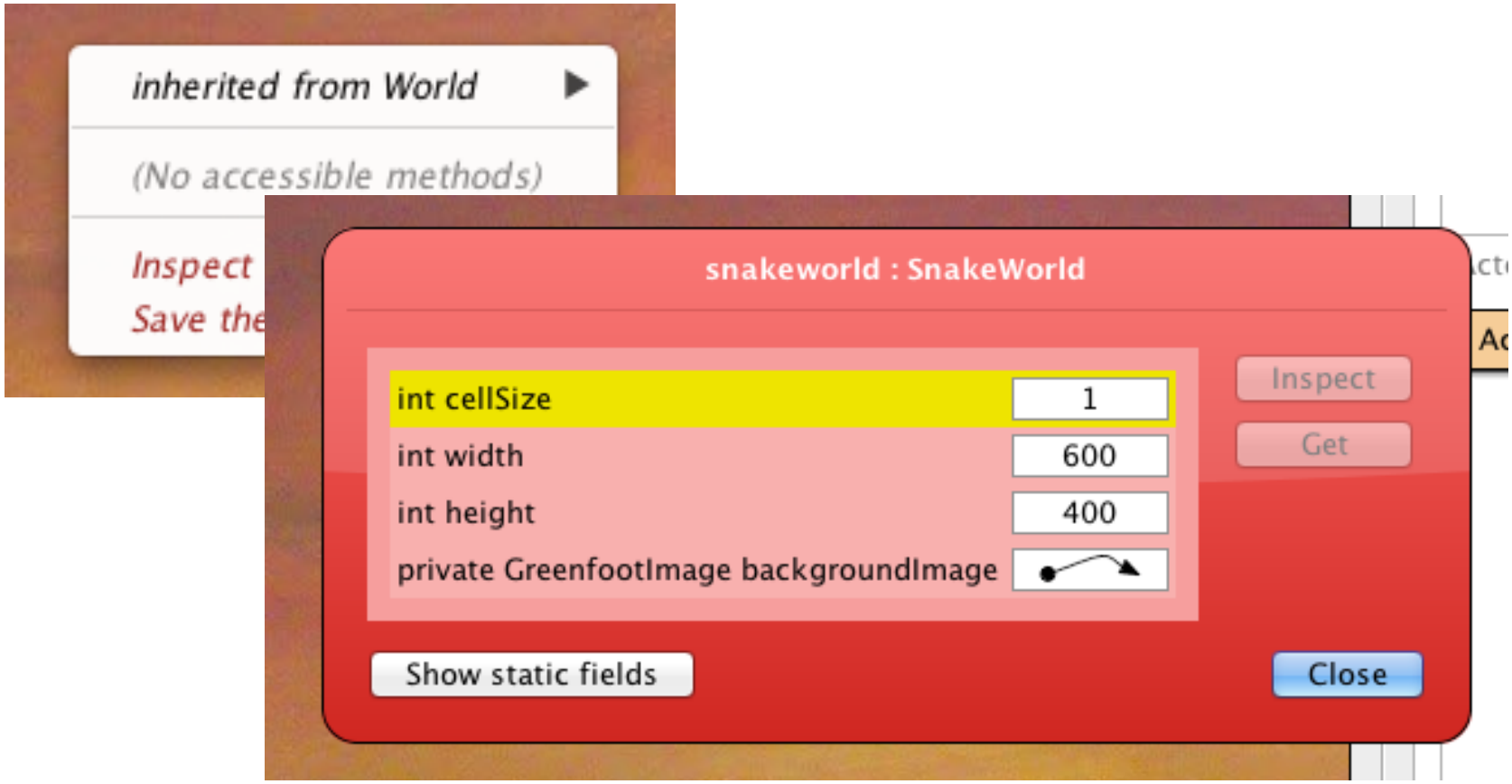
Snake World


Greenfoot: Greenfoot Scenario tuto
Share...
World classes
World
SnakeWorld
Actor classes
Actor
Act
Run
Reset
Speed:
Compile

Le code de SnakeWorld



Inspect World

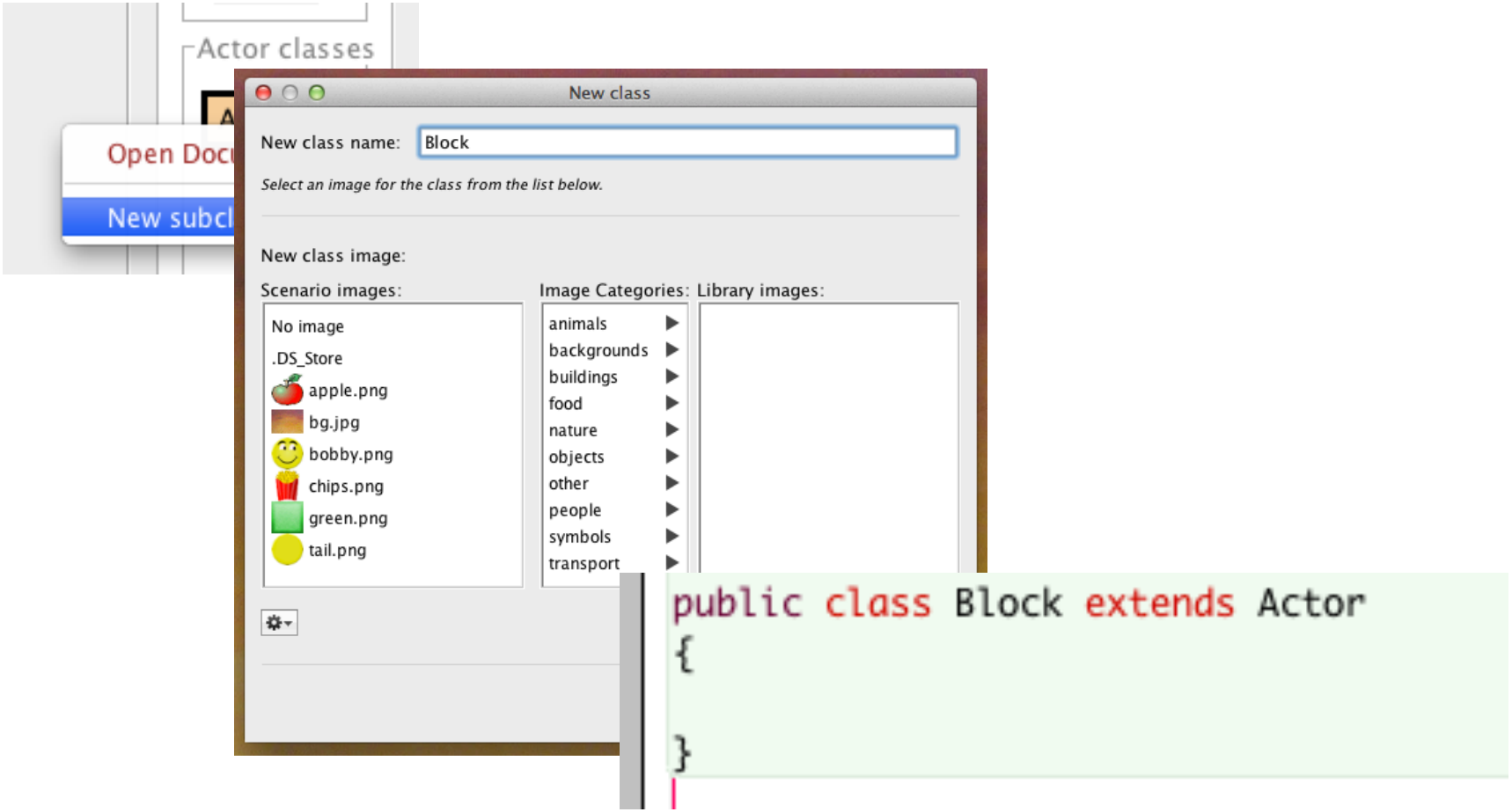


Changer le code

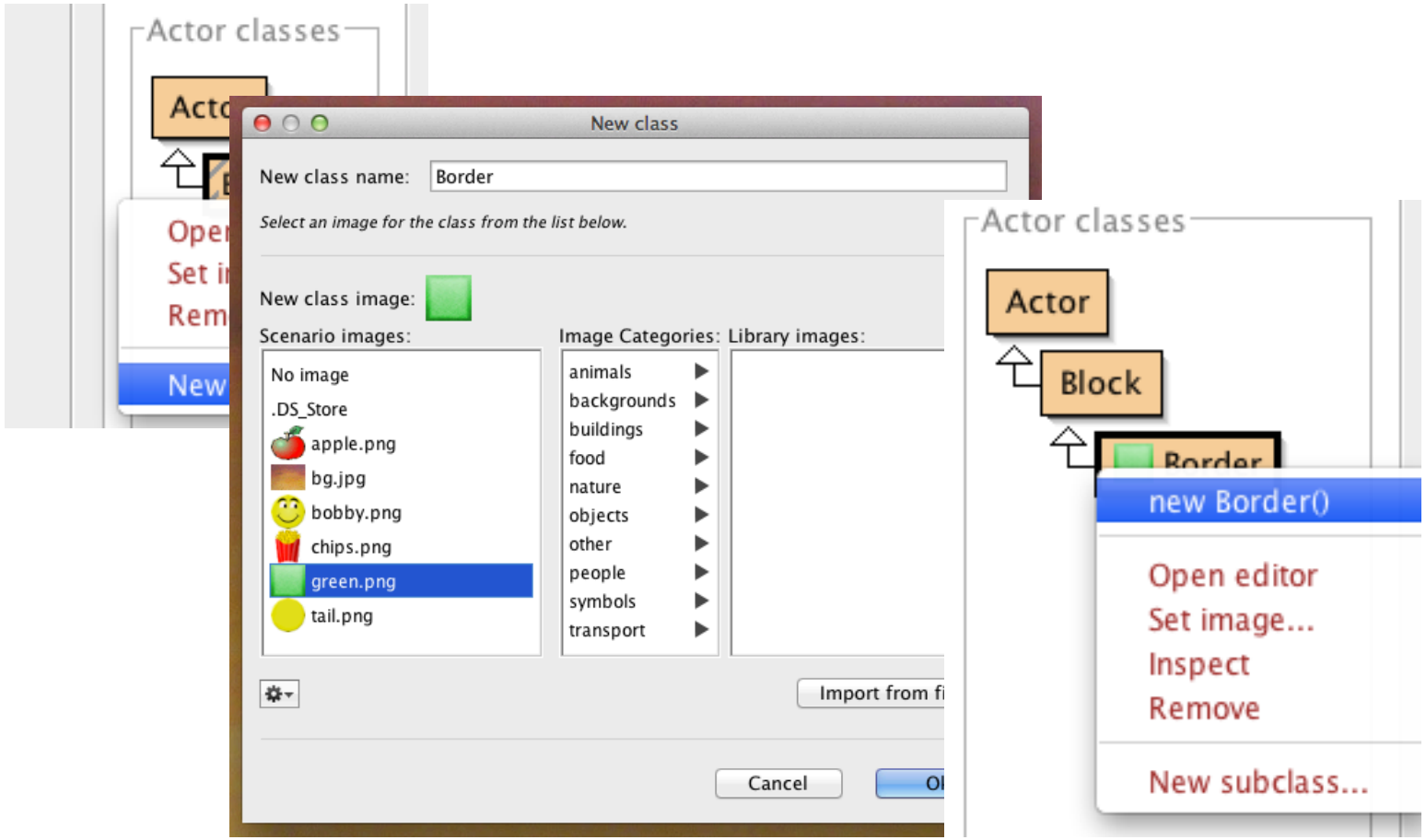
- Nous allons utiliser des blocs de 32 pixels dans notre jeu
- La taille du jeu sera de 25 x 20 blocs

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

Créer un bloc



Création du Border



Coordonées

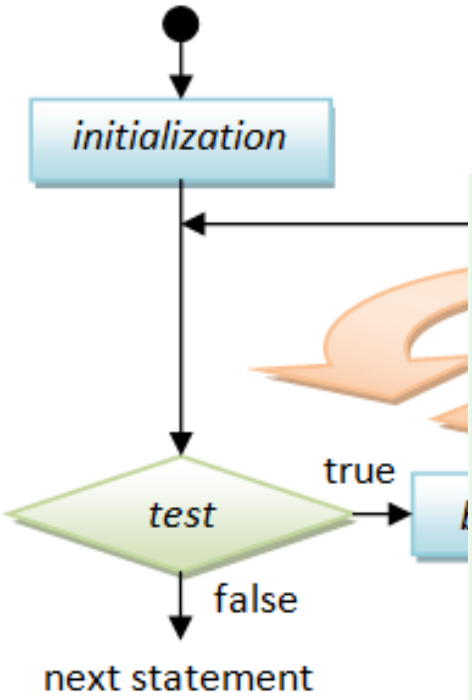
(0,0)	(1,0)	(2,0)	…	(24,0)
(0,1)	(1,1)	(2,1)	…	(24,1)
(0,2)	(1,2)	(2,2)	…	(24,2)
…	…	…	…	…
(0,19)	(1,19)	(2,19)	…	(24,19)

Afficher les bords

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

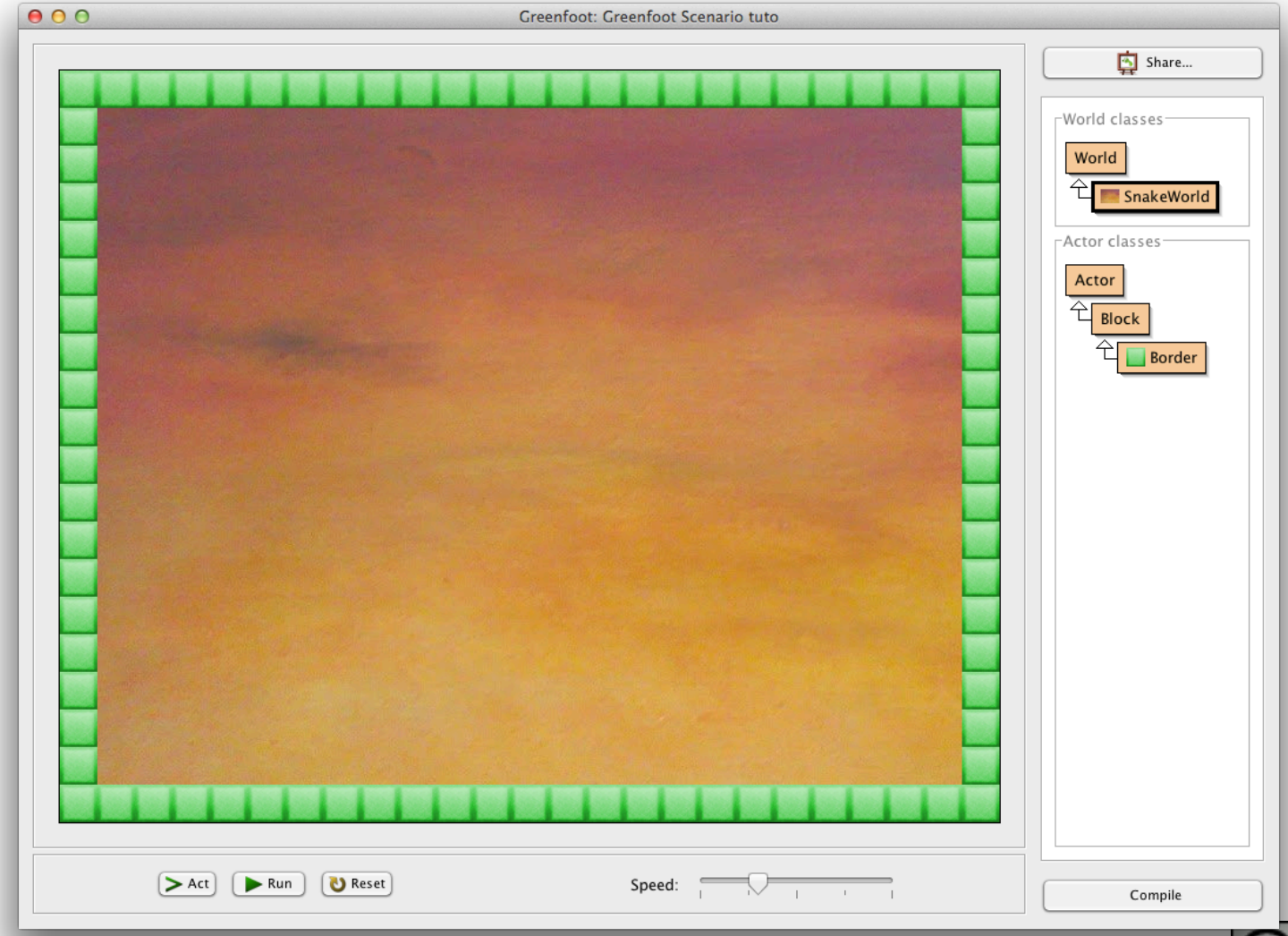
Les boucles for

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

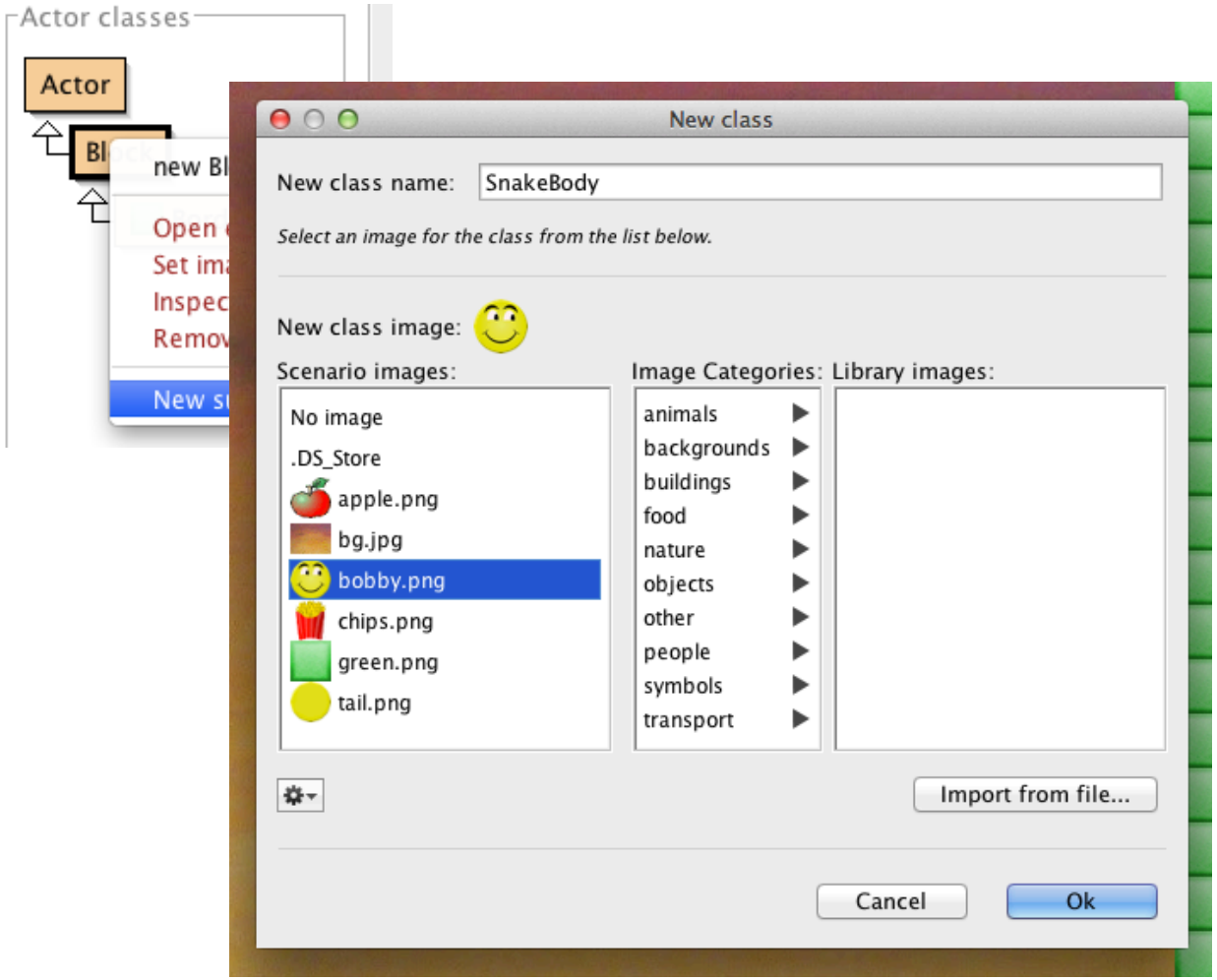



```
public SnakeWorld()
{
    super(25, 20, 32);

    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



Class SnakeBody

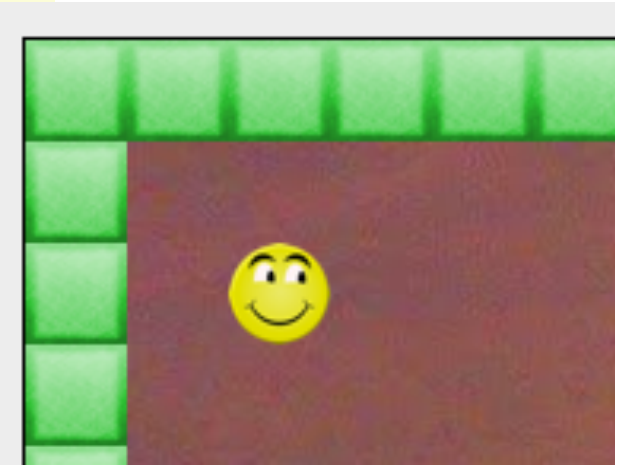


Modif SnakeWorld

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

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

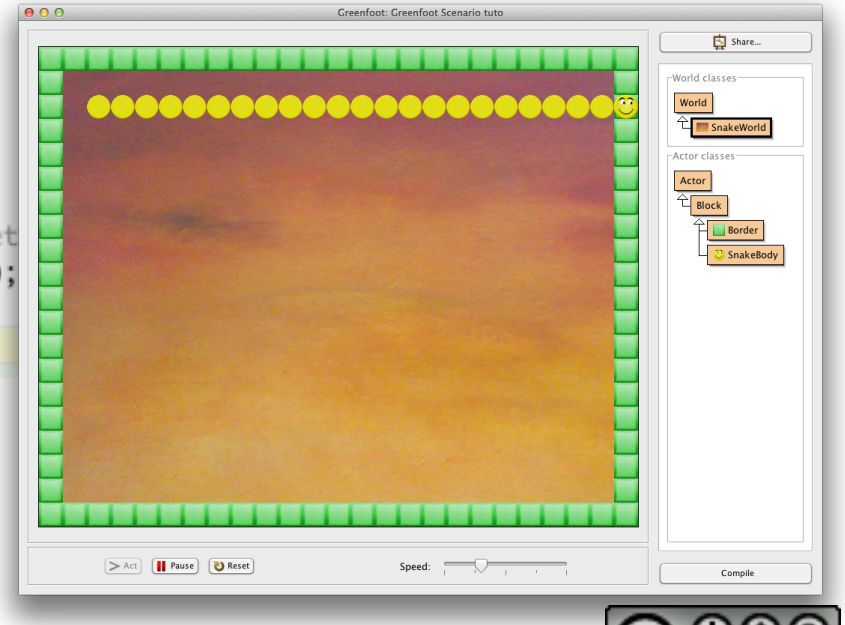
```
public SnakeWorld()
{
    super(25, 20, 32);

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



Déplacement

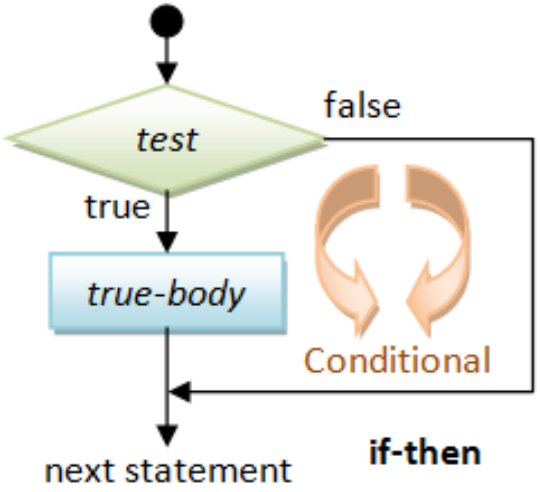
```
public class SnakeWorld extends World
{
    private LinkedList<SnakeBody> snake = new LinkedList<SnakeBody>();
    private int dx = 1;
    private int dy = 0;
```

```
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);
}
```

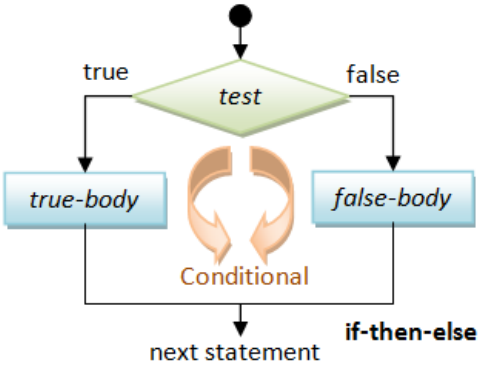


L'instruction « if »

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



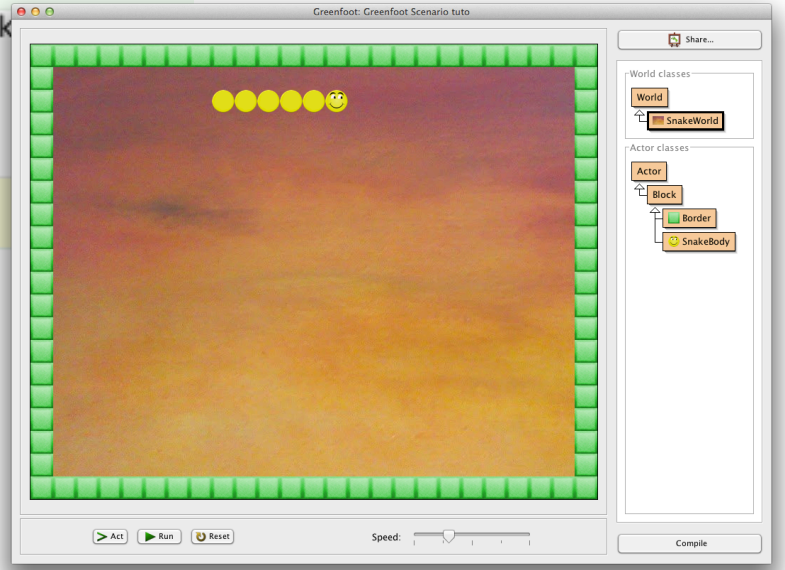

```
if ( condition ) {
   true-body ;
} else {
   false-body ;
}
```



Taille du serpent

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

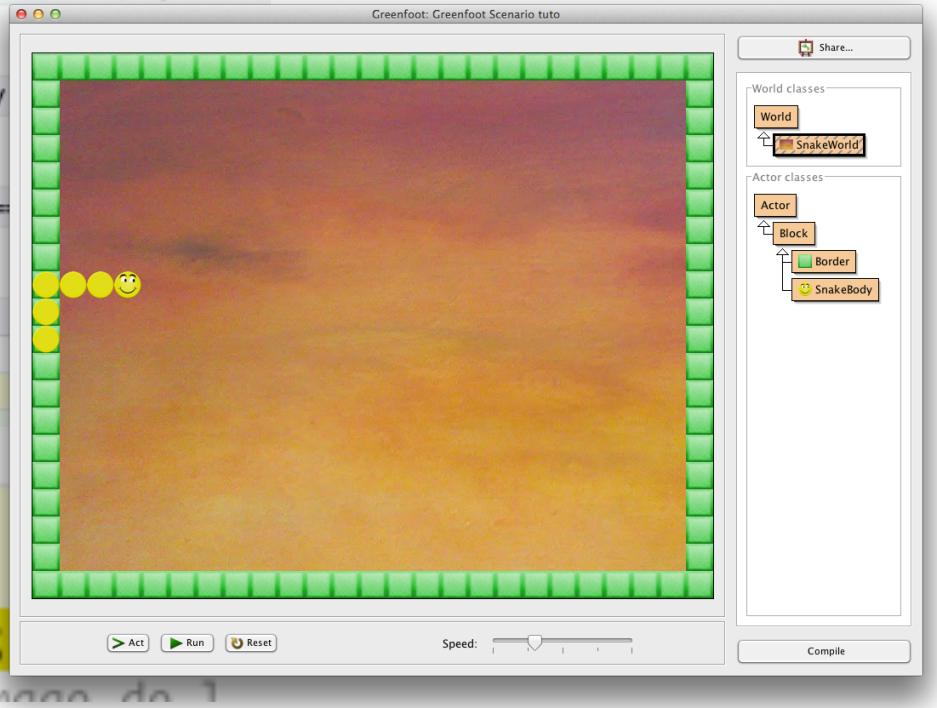
```
    public void act()
    {
      //on remplace l'image de la tête
      SnakeBody head = snake.getLast();
      head.setImage("tail.png");

      //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
        dx = 0;
        dy = 1;
    } else if (Greenfoot.isKeyDown("up") && dy =
        dx = 0;
        dy = -1;
    }

}
```

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



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;
```

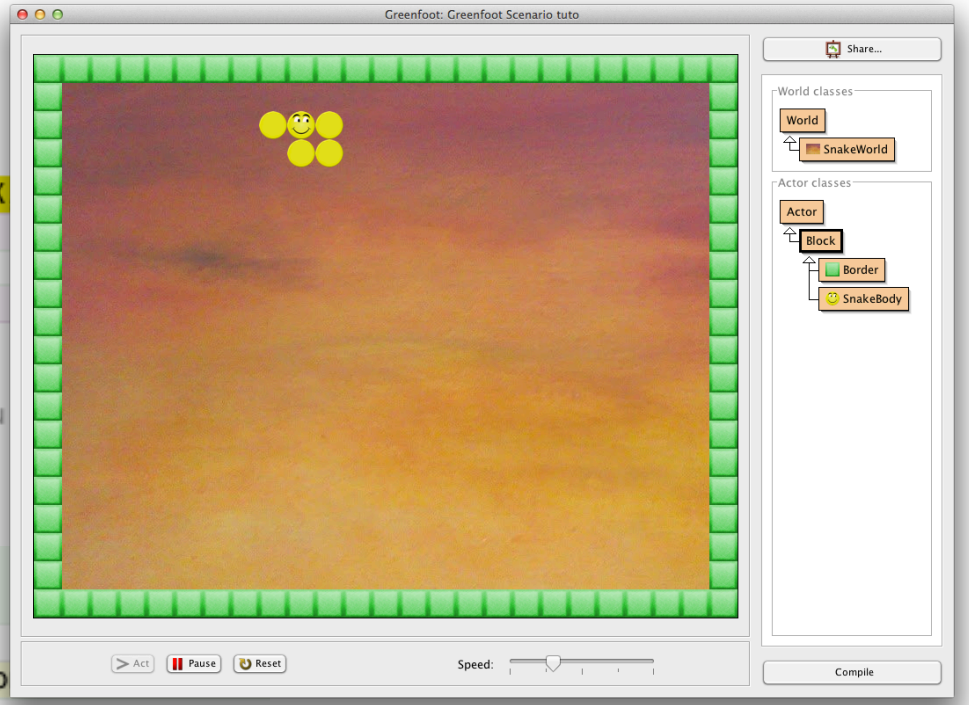
```
    public void act()
    {
        if (dead) {
            return;
        }
        changeDirection();
```

```
    public void dead() {
        dead = true;
    }
```

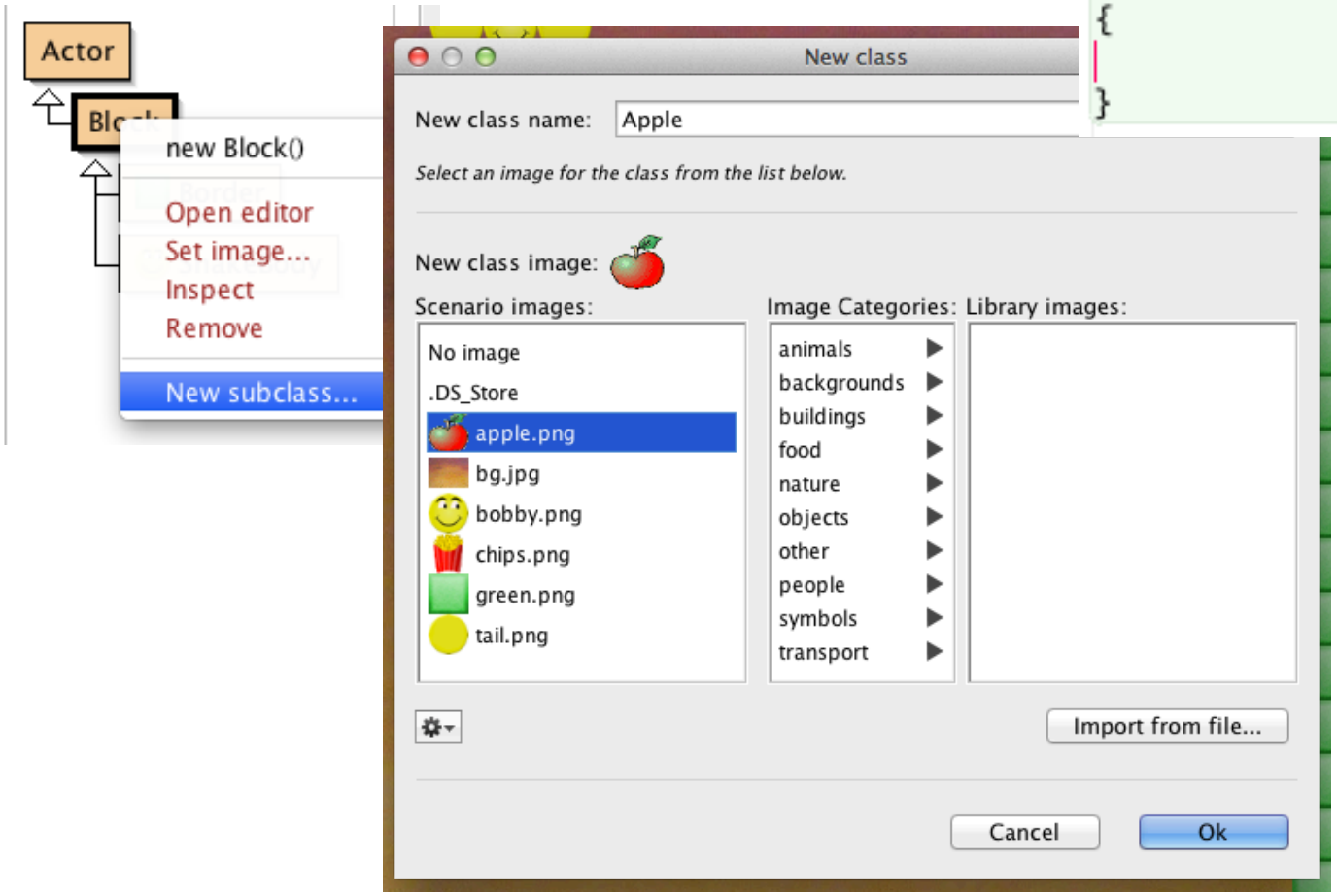
Les collisions (2)

```
//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(SnakeWo
        world.dead();
    }
}
```



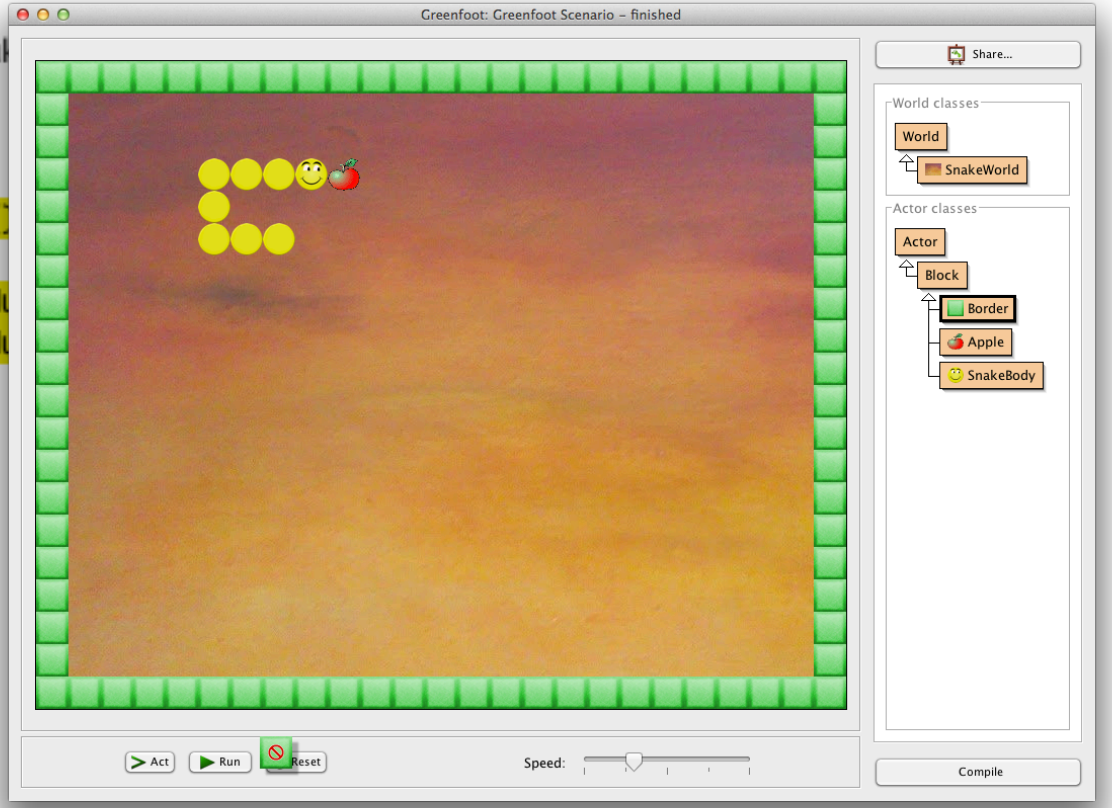
Ajout d'une pomme (1)




```
public class Apple extends Block
{

}
```

Ajout d'une pomme (2)

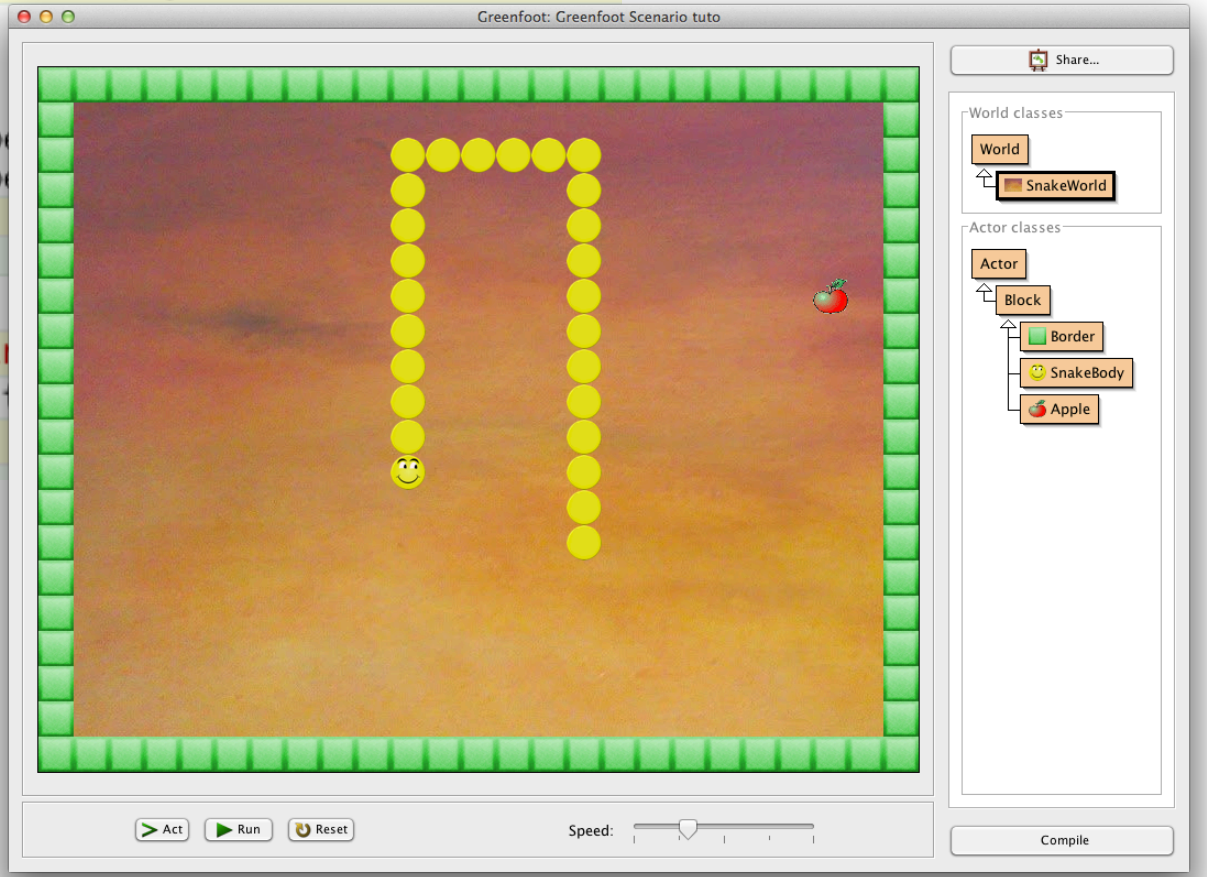
```
public SnakeWorld()
{
    super(25, 20, 32);

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

    Apple apple = new Apple()
    addObject(apple,
        Greenfoot.getRandomNu
        Greenfoot.getRandomNu
```



Collision d'une pomme

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

        setLocation(
            Greenfoot.getRandomNumbe
            Greenfoot.getRandomNumbe
    }
}
```

```
    public void grow(i
        tailCounter =
    }
```



www.devoxx4kids.com

Ajouter du son

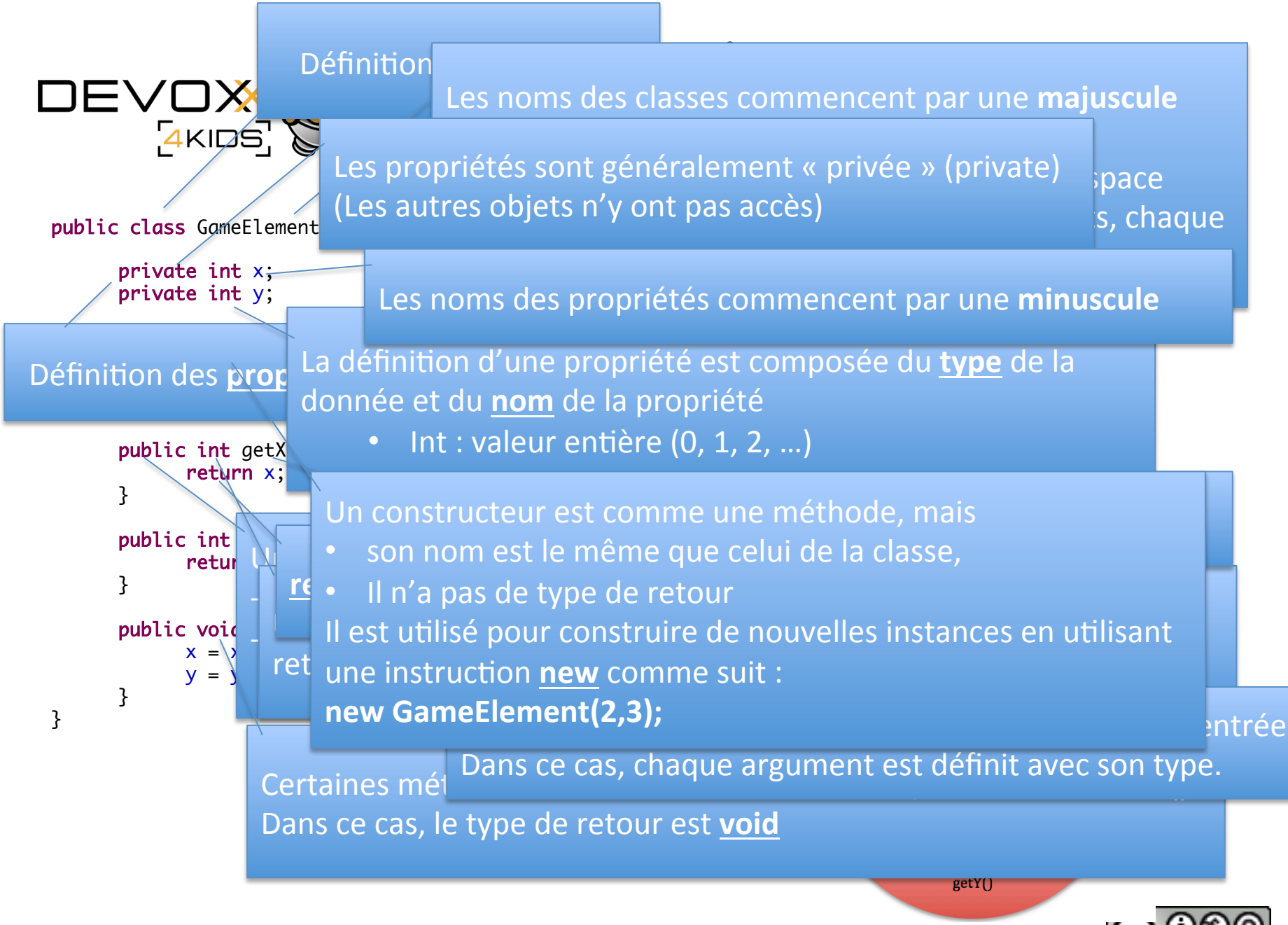
```
public void collision(SnakeWorld world) {
    Greenfoot.playSound("slurp.mp3");
    world.grow(2);
    setLocation(
        Greenfoot.getRandomNumber(getWorld().getWidth()-2)+1,
        Greenfoot.getRandomNumber(getWorld().getHeight()-2)+1);
}
```

```
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 !


DEVOXX
4KIDS
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);
Dans ce cas, chaque argument est définit avec son type.
Dans ce cas, le type de retour est **void**
public class GameElement
private int x;
private int y;
public int getX
return x;
public int
public voi
x = x
y = y
getY()