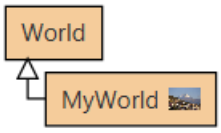
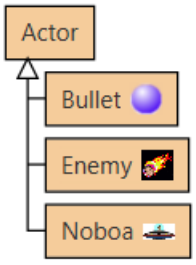

Game Over





World

```
import greenfoot.*;


public class MyWorld extends World {
    public MyWorld() {
        super(640, 425, 1); // Crea un mundo de 640x425 píxeles con una celda de tamaño 1x1
        addObject(new Noboa(), getWidth() / 2, getHeight() - 50);
```

```
    }

    public void act() {
        // Agrega enemigos periódicamente
        if (Greenfoot.getRandomNumber(100) < 5) {
            addObject(new Enemy(), getWidth(), Greenfoot.getRandomNumber(getHeight()));
        }
    }

     public void gameOver() {
        showText("Game Over", getWidth() / 2, getHeight() / 2);
        Greenfoot.stop(); // Detener el juego
    }
}
```

Actor

Bullet

```
import greenfoot.*;

public class Bullet extends Actor {
    public void act() {
        move(5); // Mover hacia arriba

        // Verificar colisión con enemigos
        checkCollision();

        // Eliminar la bala si sale del mundo por arriba
        if (getY() <= 0) {
            getWorld().removeObject(this);
        }
    }
```

```
    // Verificar colisión con enemigos
    public void checkCollision() {
        Actor enemy = getOneIntersectingObject(Enemy.class);
        if (enemy != null) {
            getWorld().removeObject(enemy); // Eliminar el enemigo
            // No eliminar la bala, dejarla continuar su movimiento
        }
    }
}
```

Enemy

```
import greenfoot.*;

public class Enemy extends Actor {
    public Enemy() {
        // Obtener la imagen actual del actor
        GreenfootImage image = getImage();

        // Establecer el nuevo tamaño de la imagen (ancho y alto)
        int newWidth = image.getWidth() / 20;  // La mitad del ancho original
        int newHeight = image.getHeight() / 20;  // La mitad del alto original

        // Escalar la imagen al nuevo tamaño
        image.scale(newWidth, newHeight);

        // Establecer la nueva imagen al actor
        setImage(image);
    }

    public void act() {
```

```
        move(-2); // Mover hacia la izquierda

        // Eliminar el enemigo si alcanza el borde izquierdo del mundo
        if (getX() <= 0) {
            getWorld().removeObject(this);
        } else {
            checkCollision(); // Verificar colisión con balas
        }
    }

    // Verificar colisión con balas
    public void checkCollision() {
        Actor bullet = getOneIntersectingObject(Bullet.class);
        if (bullet != null) {
            getWorld().removeObject(this); // Eliminar el enemigo
            // No eliminar la bala, dejarla continuar su movimiento
        }
    }
}
```

Noboa

```
import greenfoot.*;

public class Noboa extends Actor {
    public void act() {
        // Verificar colisión con Enemy y terminar el juego
        Actor enemy = getOneIntersectingObject(Enemy.class);
        if (enemy != null) {
            MyWorld world = (MyWorld) getWorld();
            world.gameOver();
        }
```

```
    // Mover hacia la izquierda
    if (Greenfoot.isKeyDown("left")) {
        move(-5);
    }

    // Mover hacia la derecha
    if (Greenfoot.isKeyDown("right")) {
        move(5);
    }

    // Mover hacia arriba
    if (Greenfoot.isKeyDown("up")) {
        setLocation(getX(), getY() - 5);
    }

    // Mover hacia abajo
    if (Greenfoot.isKeyDown("down")) {
        setLocation(getX(), getY() + 5);
    }

    // Disparar balas al presionar la barra espaciadora
    if (Greenfoot.isKeyDown("space")) {
        shoot();
    }

}

public void shoot() {
    Bullet bullet = new Bullet();
    getWorld().addObject(bullet, getX(), getY() - 30);
```

```
    }
}
```