

Paradigmas de Programación

Punto 2 A

Escenario Newton's-Lab 1

Todos los escenarios están compuestos por dos superclases, una llamada Actor y otra llamada World de las que se pueden heredar más clases.

En cuanto a este escenario de la clase Actor se especializa la clase SmoothMover (Movimiento Suave) que a su vez de esta se especializa la clase Body (Cuerpo). Con respecto a la clase World se especializa la clase Space.

Además tenemos una clase llamada Vector que se utiliza para el atributo de la velocidad de cada Body.

El programa nos permite crear una instancia de Body con sus valores por defecto y también nos permite crear una instancia de Body pero con valores que nosotros ingresemos. Pero estos cuerpos no realizan ningún comportamiento ya que el método Act() dentro de estas clase está vacío, para que estos cuerpos realicen algún movimiento se debería escribir el código de la operación dentro del método Act().

Además, el programa cuenta con otros métodos que realizan operaciones simples como retornar valores de masa, posición, etc

En la clase Space hay métodos para la instanciación de Body's por defecto cuando se ejecuta el programa. Pero las llamadas a estos métodos dentro del constructor de Space se encuentran comentados por lo tanto al ejecutar el juego no se crea ningún Body por sí solo.

Si descomentamos estas llamadas a los métodos , se van a crear Body's definidos en él, en una posición determinada, con un determinado color, masa, etc.

Escenario Newton's-Lab 2

Con respecto al Newtons-Lab 1, las modificaciones que presenta este escenario se realizan en la clase Body:

En este caso el método Act() contiene un código, en el cual se llama a otros métodos (move(), applyForces()).

El método move() hace que el cuerpo se mueva con respecto al vector velocidad que tiene en dicho momento.

El método applyForces() crea una lista con todos los Body's que se encuentran dentro de World, luego mediante un ciclo for y con un if compara los body's entre sí, en caso de que sean distintos se llama al método applyGravity(body other), lo que hace este método es aplicar de la fuerza de gravedad entre los body's.

Escenario Newton's-Lab 3

En este escenario se agrega una nueva clase que es una especialización de Actor llamada Obstacle. Cada vez que se crea un objeto del tipo Obstacle el mismo se asocia a un archivo con un sonido. A su vez esta clase contiene un método Act() que hace que cada vez que un objeto del tipo Body se choca con

Paradigmas de Programación

un objeto del tipo Obstacle se llama al método playSound() y este produce un sonido.

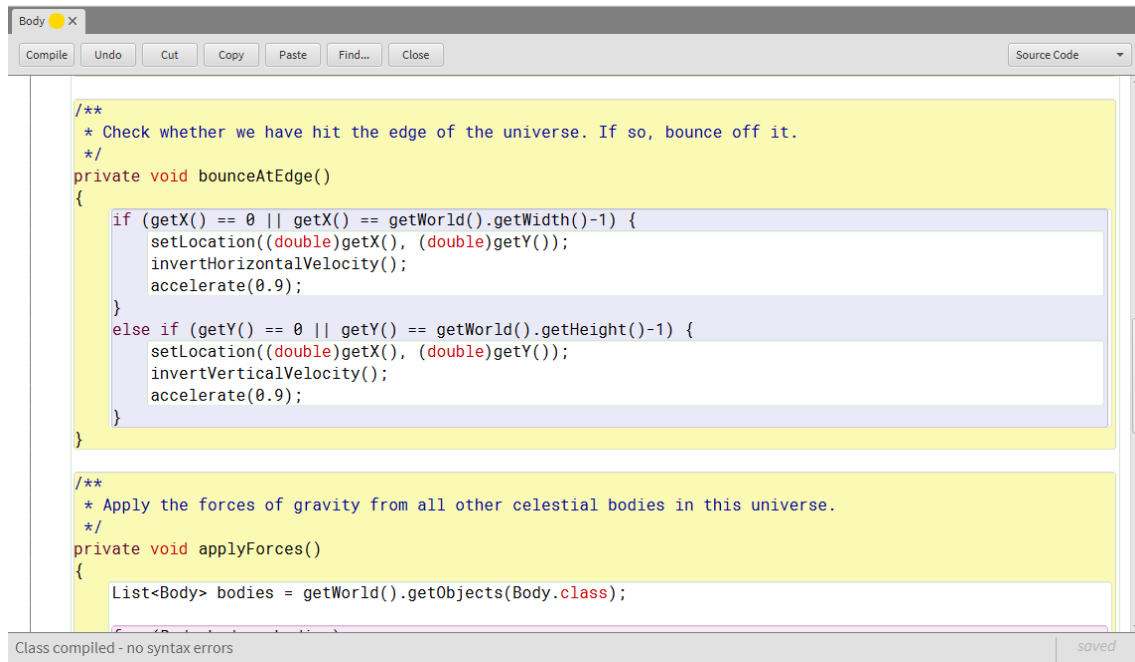
En cuanto a las demás clases existen diferencias con respecto a los demás escenarios:

En la clase Space el constructor llama a los métodos createObstacles(), randomBodies(int number). Donde el método createObstacles() instancia objetos del tipo Obstacle, con un sonido y ubicación determinado en el mismo. Por otro lado, el método randomBodies(int number) instancia 5 Body's (la cantidad de body's que se crean se especifica cuando se llama al método) donde cada uno de sus atributos se crea de manera aleatoria.

Con respecto a la clase Body, es igual al del escenario Newtons-Lab 2, solo que dentro del método Act() se llama a un nuevo método bounceAtEdge(), este lo que hace es comprobar si el objeto del tipo Body llegó al borde del universo, y de ser así rebota.

Punto 2 B

Antes del cambio:



```

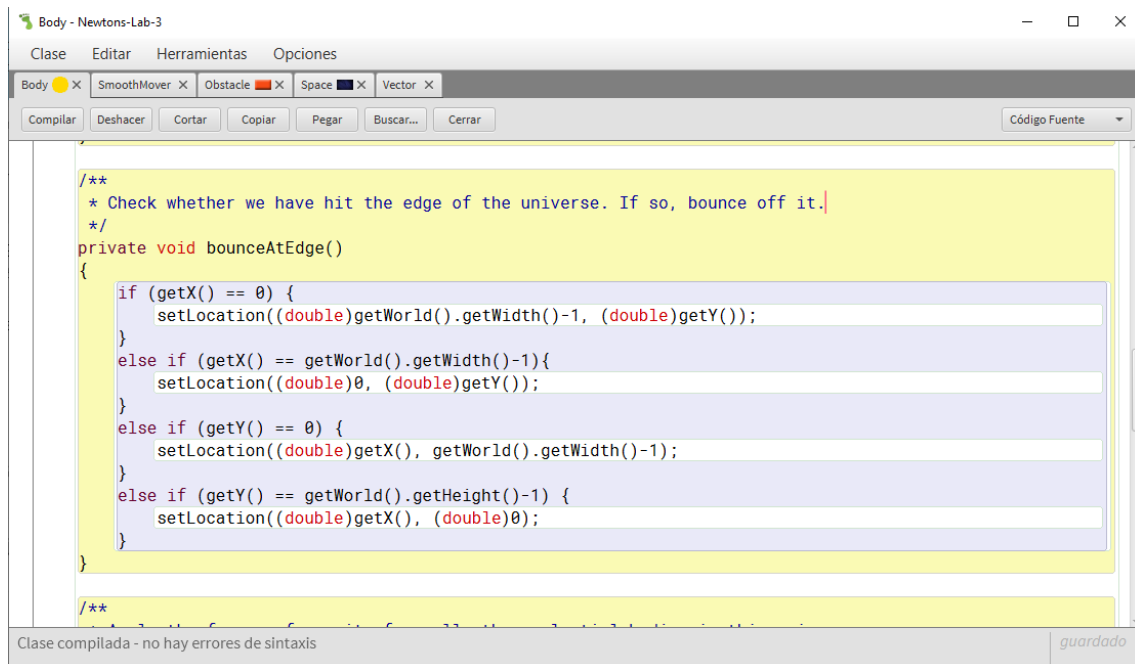
/**
 * Check whether we have hit the edge of the universe. If so, bounce off it.
 */
private void bounceAtEdge()
{
    if (getX() == 0 || getX() == getWorld().getWidth()-1) {
        setLocation((double)getX(), (double)getY());
        invertHorizontalVelocity();
        accelerate(0.9);
    }
    else if (getY() == 0 || getY() == getWorld().getHeight()-1) {
        setLocation((double)getX(), (double)getY());
        invertVerticalVelocity();
        accelerate(0.9);
    }
}

/**
 * Apply the forces of gravity from all other celestial bodies in this universe.
 */
private void applyForces()
{
    List<Body> bodies = getWorld().getObjects(Body.class);

```

Class compiled - no syntax errors

Después del cambio:



```

/**
 * Check whether we have hit the edge of the universe. If so, bounce off it.
 */
private void bounceAtEdge()
{
    if (getX() == 0) {
        setLocation((double)getWorld().getWidth()-1, (double)getY());
    }
    else if (getX() == getWorld().getWidth()-1){
        setLocation((double)0, (double)getY());
    }
    else if (getY() == 0) {
        setLocation((double)getX(), getWorld().getHeight()-1);
    }
    else if (getY() == getWorld().getHeight()-1) {
        setLocation((double)getX(), (double)0);
    }
}

/**

```

Clase compilada - no hay errores de sintaxis

La modificación que realizamos fue en el método `bounceAtEdge()`, anteriormente lo que hacía era comprobar si el objeto del tipo `Body` llegó al borde del universo, y de ser así rebota. Ahora con la modificación cuando un objeto del tipo `body` se encuentra en el borde del universo, este no rebota, sino que aparece por el lado contrario del universo (simulando un universo infinito).

Video: <https://drive.google.com/file/d/1SX9dE7w1He4lyH1sTwJ7kxIOoCsWfuQF/view?usp=sharing>

Integrantes: Bolcato María Julieta, Goia Julián, Meloni Gregorio Tomás y Pairetti Franca.

Punto 3 A

```
1  matrizAux=[["0","1","2","3","4","5","6","7","8","9"],["6","2","5","5","4","5","6","3","7","6"]]
2  cantidadNumeros=int(input())
3
4  for i in range(cantidadNumeros):
5      numero=str(input())
6      cantidadLed = 0
7      for j in range(len(numero)):
8          for k in range(len(matrizAux[0])):
9              if numero[j]==matrizAux[0][k]:
10                 cantidadLed = cantidadLed + int(matrizAux[1][k])
11  print(cantidadLed, "leds")
```

Punto 3 B

```
13  s, t, f = input().split()
14  s = int(s)
15  t = int(t)
16  f = int(f)
17
18  horaLlegada= s+t+f
19  if horaLlegada>=0 and horaLlegada<24:
20      print(horaLlegada)
21  elif horaLlegada>=24:
22      print(horaLlegada-24)
23  elif horaLlegada < 0:
24      print(24+horaLlegada)
```