2- (60 pts) Sobre el escenario del Laboratorio de Newton, se deben realizar las siguientes tareas (**grupal - colaborativo**):

A: Describir las principales características de los 3 Newton Labs (funcionamiento, clases, objetos, métodos, etc)

B: A partir del Newton Lab3, incorporar una modificación **significativa** a la elección.

En la presentación de la solución de este punto deberá incluirse:

- una memoria descriptiva del cambio a realizar.

- imágenes, figuras, fotos, etc que documenten los cambios realizados.

- un pequeño video comentando el resultado obtenido

Consigna A:

Newton Labs 1:

Clases:

-Space: Esta clase de hereda de World y es donde se crea el mundo con sus dimensiones y tenemos 3 métodos comentados que van a crear diferentes objetos:
Métodos de tipo void:

- -sunAndPlanet(): Crea el objeto de la clase body con las características del sol y otro de un planeta
- -sunAndTwoPlanets(): Crea el objeto de la clase body con las características del sol y dos planeta
- -sunPlanetMoon(): Crea el objeto de la clase body con las características del sol, la luna y un planeta.

También está el método removeAllObjets(), el cual remueve todos los objetos que hereden de la clase “Actor”

-SmoothMover: Hereda de la superclase Actor, aquí se inician las variables de instancia:
-velocity: Es de tipo Vector.
-exactX: Es de tipo Double.
-exactY: Es de tipo Double.

Se crea el constructor y este crea un objeto Vector.
Existe otro constructor que permite especificar al Vector con sus parámetros.

Métodos:
Métodos tipo Void:

-move(): Este método va a realizar el movimiento de los objetos de la clase body y funciona con la velocidad del vector

-setLocation(): A este método se le van a dar dos parámetros “X” y “Y” y en base a estos parámetros se va a cambiar la posición del objeto body en cuestión.
En particular existen dos de estos métodos, uno que trabaja con variables de tipo double para ser más exacto y otro con variables tipo int.

-addToVelocity(): Este recibe como parámetro un Vector boost que le va a aumentar velocidad al propio vector añadiendo otro vector.

-accelerate(): Este va a recibir como parámetro una variable de tipo vector y va a llamar al método scale() de la clase vector.

-invertHorizontalVelocity(): Este método va a revertir la velocidad horizontal.
-invertVerticalVelocity(): Este método va a revertir la velocidad vertical.

Métodos tipo double:

-getExactX(): Va a retornar la variable exactX.

-getExactY(): Va a retornar la variable exactY.

-getSpeed(): Este va a llamar al método getLength() de la clase Vector.

-Body:Se hereda de la clase SmoothMover y se crean las variables de instancia Gravity de tipo double con el valor 5.8 y el color defaultColor que va ser de la clase Color del propio GreenFoot. También se crea la variable mass de tipo double.

Constructor:
Se le puede dar los parámetros de Tamaño(size, de tipo int), Masa(mass, de tipo double), Velocidad (velocity, de tipo Vector) y color (color, de tipo Color).

Método Void act: Va a ser el método main de la clase donde se van a hacer todos los movimientos de los objetos etc.

Método double getMass: Va a retornar la masa (mass) del objeto.

Clase Vector: Es la clase que se encarga de especificar los objetos de tipo vector con su dirección y magnitud. Posee varios métodos entre ellos:
De tipo void:

-setDirection(): Sirve para cambiar la dirección.
-add(): Es para añadir otro vector al mismo vector.
-setLength(): Sirve para cambiar la magnitud
-scale(): Es para aumentar la magnitud de velocidad con un factor
-setNeutral(): Convierte el vector a 0 en velocidad y magnitud.
-revertHorizontal(): Invierte la componente del movimiento horizontalmente.
-revertVertical(): Invierte la componente del movimiento verticalmente.
-updatePolar(): Actualiza la dirección y la magnitud del vector.
-updateCartesian(): Actualiza la dirección y la magnitud del vector.

De tipo double:
-getDirection(): Devuelve dirección.
-getLength: Devuelve magnitud.
-getX(): Devuelve posición en X.
-getY(): Devuelve posición en Y.

Newton-Lab-2:

Posee todo lo mencionado en el Newton-Lab-1 y se le añade las siguientes funciones:
En la clase Body se agregan los siguientes métodos:

De tipo Void:
-applyForces(): Se crea una lista con todos los objetos Body y a todos estos objetos se le aplica la gravedad con el método applyGravity().

-applyGravity(): La cual recibe como parámetro un objeto de tipo body.
Se crean las variables locales de tipo double: dx, dy, distance, force y acceleration.
Y la variable de tipo Vector: dv.
La distancia (distance) se calcula desde la raíz cuadrada de la posición en “x” y la posición de “y” al cuadrado.
La fuerza (force) se va a calcular con la ecuación de gravedad por masa (del objeto que se va a aplicar la fuerza) por masa (del objeto que se usa como parámetro) sobre distancia elevada al cuadrado.
La aceleración (aceleration) se calcula con la fuerza divido masa. Esa aceleración va a ser la nueva magnitud que se le va a agregar al Vector dv con el método setLength() y este vector va a llamar el método addToVelocity() para agregarse al Vector velocidad (velocity).

Newton-Lab-3:

Posee las características de los Newton-Labs anteriores y se les agregan las siguientes funcionalidades:

•El método applyForce() de la clase body se le añade la funcionalidad de reducir la velocidad cuando esta misma es mayor que 7 y se logra haciendo que accelerate sea menor que 1.

•Se crea la clase Obstacle que se hereda de Actor, va a tener las variables de instancia sound de tipo String, touched de tipo boolean que se inicia en falso.

•En el constructor se crea el parámetro soundFile de tipo String el cual fue creado en la clase Space y es un vector de tipo String que posee las notas.

•En el método Void act de Obstacle se crea un objeto de tipo Body donde existe un If que verifica si se está tocando el objeto Body con el objeto Obstacle se prende la luz cambiando la imagen del objeto Obstacle y se ejecuta el sonido llamado al método playSound(). Cuando se dejan de tocar se vuelve a la imagen anterior.

•En el constructor de Space se crean cinco objetos aleatorios y se crean los obstáculos con los métodos:
-createObstacle(): De tipo void. Donde se inicia un ciclo while por la cantidad de soundFiles que existen en el vector creando objetos del tipo Obstacle, asignándole un archivo .wav en el soundFile y su posición que va a aumentar en 60 en “x” por cada objeto, para que se sitúen uno al lado de otro.
-randomBodies(): Este método va a recibir como parámetro un número entero el cual es 5 especificado en el constructor. En este método va a haber un while que se cumple la cantidad veces asignadas por el parámetro.
Los objetos se generarán con todos sus parámetros aleatorios: tamaño, masa, dirección y velocidad del vector, posición “x” “y” del espacio y color. La masa varía en función del tamaño.

•En la clase Body se agregó un nuevo método llamado bounceAtEdge() que cuando un objeto llega a la posición de x=0 o a la posición máxima en “x” o en “y” se invierte la velocidad horizontalmente o verticalmente respectivamente, además se reduce la aceleración.

Consigna B:

Los cambios realizados en el Newton-Lab-3 son los siguientes:

- Implementamos la función que cuando un objeto de la clase body (círculos) impacte contra un objeto de la clase obstacle (rectángulos), el mismo rebota. Además, si el tamaño de los círculos no supera el valor de 40, modifica su color y su tamaño de forma random en el momento que se topa con un obstáculo. Si el tamaño de los círculos es igual a 50, tienen la habilidad de destruir a los obstáculos, los cuales se crearán nuevamente, pero en una posición diferente y aleatoria.

```
private double mass;
public int size; //cambios

/**
 * Construct a Body with default size, mass, velocity and color.
 */
public Body()
{
    this (20, 300, new Vector(0, 0.0), defaultColor);
}

/**
 * Construct a Body with a specified size, mass, velocity and color.
 */
public Body(int size, double mass, Vector velocity, Color color)
{
    this.mass = mass;
    this.size = size; //cambios
    addToVelocity(velocity);
    GreenfootImage image = new GreenfootImage (this.size, this.size); //cambios
    image.setColor (color);
    image.fillOval (0, 0, this.size-1, this.size-1); //cambios
    setImage (image);
}

/**
 * Act. That is: apply  the gravitation forces from
 * all other bodies around, and then move.
 */
public void act()
{
    applyForces();
    move();
    bounceAtEdge();
    bounceAtObstacle(); //cambios
    destroyObstacles(); //cambios
}
```

```
private void bounceAtObstacle() //cambios
{
    if (isTouching(Obstacle.class)){
        setLocation((double)getX(), (double)getY());
        invertVerticalVelocity();
        accelerate(0.9);
    }
}
private void destroyObstacles(){ //cambios
    if (isTouching(Obstacle.class)){
        if (this.size<50){
            this.size=60;
             GreenfootImage image = new GreenfootImage (this.size, this.size);
             int r = Greenfoot.getRandomNumber(255);
             int g = Greenfoot.getRandomNumber(255);
             int b = Greenfoot.getRandomNumber(255);
             Color col = new Color(r, g, b);
             image.setColor (col);
             image.fillOval (0, 0, this.size-1, this.size-1);
             setImage (image);
        }else{
            Actor o = getOneIntersectingObject(Obstacle.class);
            int x = Greenfoot.getRandomNumber(960);
            int y = Greenfoot.getRandomNumber(620);
            o.setLocation(x,y);
        }
    }
}
```

La partes modificadas o agregadas del código están especificadas con un comentario a su derecha