Punto 2-A.

Newtons-Lab-1.

Clase Space.

En la clase space, que hereda las características de la superclase World, tenemos los siguientes métodos:

1. Space(): genera el mundo según los métodos que llamemos.
2. sunAndPlanet(): primero llama al método removeAllObjects(), que lo que hace es remover todos los objetos que estén en el espacio anteriormente, y luego, añade dos objetos al mundo con el método addObject, dándole como parámetros un nuevo objeto tipo Body(), indicando los parámetros de cada uno de estos objetos, y además le da las posiciones donde quiere que se genere el objeto.
3. sunAndTwoPlanets(): método parecido al sunAndPlanet(), con la diferencia es que genera el sol y dos planetas, en vez de solamente un planeta.
4. sunPlanetMoon(): método que genera el sol, un planeta y una luna.
5. removeAllObjects(): método que remueve todos los objetos existentes en el mundo.

Nota: sunAndPlanet(), sunAndTwoPlanets(), sunPlanetMoon() son lo mismo con la diferencia que cambian los parámetros que se ingresan al crear un objeto tipo Body.

Clase Body.

La clase Body hereda las características de la superclase SmoothMover

Esta clase tiene dos constantes privadas y estáticas, una tipo double que indica la gravedad que tendrá el cuerpo, y la otra es tipo Color, que indica el color default de los cuerpos.

También tiene una variable tipo double que indica la masa del cuerpo y no tiene un valor inicial.

Esta clase tiene dos constructores, una sin parámetros que al crear un objeto tipo Body, se creará con las características indicadas en ese constructor, y un constructor con parámetros en donde, al llamar a este método se creará un objeto tipo Body con las características que indiquemos.

Un método act() no tiene implementación alguna.

El método getMass(), de tipo double que retorna la masa del cuerpo.

Clase SmoothMover.

La clase SmoothMover, es una clase que hereda las características de la superclase Actor

Tiene tres variables privadas, exactX y exactY de tipo double y una variable velocity de tipo Vector.

Métodos:

1. SmoothMover(): constuctor que genera un objeto de la clase Vector con los valores predefinidos en la clase Vector.
2. SmoothMover(Vector velocity): constructor que genera un objeto donde nosotros le damos como parámetro el vector que queremos como velocidad.
3. Move(): a las variables exactX y exactY le suma los valores que retornan getX() y getY() respectivamente. Luego, según el objeto tipo Body que llama a este método, lo posiciona en las coordenadas exactX y exactY con el método setLocation().
4. setLocation(double x, double y): posiciona un objeto en las coordenadas x e y, tiene como parámetros variables de tipo double.
5. setLocation(int x, int y): posiciona un objeto en las coordenadas x e y, tiene como parámetros variables de tipo int.
6. getExactX(): retorna los valores de exactX.
7. getExactY(): retorna los valores de exactY.
8. addToVelocity(): añade un nuevo vector para aumentar la velocidad del objeto.
9. accelerate(): aumenta la velocidad del objeto según la escala que le demos, llamando al método scale().
10. getSpeed(): devuelve la velocidad, llamado a un método que retorna la variable length del vector.
11. invertHorizontalVelocity(): llama al método revertHorizontal() de la clase vector, complementando la variable dx.
12. invertVerticalVelocity(): llama al método revertVertical() de la clase vector, complementando la variable dy.

Clase Vector.

La clase vector tiene las variables dx, dy y length de tipo double, y direction de tipo int.

Métodos:

1. Vector(): constructor que tiene como valores (0,0).
2. Vector(int direction, double length): constructor que le damos como parámetro la dirección y el tamaño del vector. Luego llama al método updateCartesian().
3. Vector(double dx, double dy): constructor que le damos como parámetro la posición dx y dy, y luego llamada al método updatePolar()
4. setDirection(int direction): método que cambia la dirección de vector, dándole como parámetro la dirección que deseamos. Llama al método updateCartesian().
5. add(): añade otro vector, cambiando la dirección y magnitud del vector mediante el método updatePolar().
6. setLength(): método que cambia la magnitud del vector, dándole como parámetro la longitud que deseamos.
7. scale(): método que aumenta la longitud del vector por un factor que nosotros le damos como parámetro.
8. setNeutral(): cambia la dirección, magnitud y posiciones del vector a 0.
9. revertHorizontal(): invierte la variable dx.
10. revertVertical(): invierte la variable dy.
11. getX(): retorna el valor dx del vector.
12. getY(): retorna el valor dy del vector.
13. getLength(): retorna el valor length del vector.
14. updatePolar(): cambia las direcciones y la longitud del vector mediante una fórmula matemática.
15. updateCartesian(): actualiza los valores dx y dy mediante una fórmula matemática.

Newtons-Lab-2.

Los cambios realizados con respecto a Newtons-Lab-1 son:

Clase Body.

1. En el método act() se llama a los métodos applyForces() y move().
2. Se agregan dos métodos nuevos.
    a. applyFoces(): crea una variable de tipo lista en donde todos los elementos de esta lista serán los cuerpos que existan en el mundo. Luego, crea un ciclo que recorre la lista de objetos Body, y si el body que se analiza es distinto del objeto que está llamado al método, se llama al método applyGravity(), dándole como parámetro el Body.
    b. applyGravity(): teniendo en cuenta que se trabaja con dos objetos tipo cuerpo, uno que llama al método y otro que se da como parámetro, el método realiza lo siguiente: Crea un vector que se le da como parámetro dx y dy que se consiguen con la diferencia de exactX y exactY entre los dos cuerpos. Luego se calcula la distancia entre ambos cuerpos, la fuerza y la aceleración que se realiza entre ambos cuerpos. Teniendo la aceleración de los cuerpos, se llama al método setLength(), dándole como parámetro la aceleración.

Ya teniendo la gravedad que se realizan entre ambos cuerpos, éstos luego se mueven con el método move() que está en la línea siguiente al applyForces(), moviéndose de forma dependiente a los valores a los que se llegan en applyForces().

Newtons-Lab-3.

Los cambios con respecto a los anteriores Newtons-Lab, son los siguientes:

Se agrega una nueva clase llamada **Obstacles.**

Esta nueva clase hereda las características de la superclase Actor. En el método act() de esta clase, lo que se realiza es que si un objeto de esta clase es tocado, genera un sonido correspondiente según se haya especificado al crear estos objetos.

Clase Space.

1. Se crea una variable donde se guardan las notas musicales.
2. En el método Space() se llama a los métodos createObstacles() y randomBodies.

Los métodos son:

a. createObstacles(): añade al mundo un objeto tipo Obstacles dándole como parámetros el sonido que debe emitir este objeto al ser tocado, y las posiciones de cada uno.
b. randomBodies(): genera objetos de tipo Body según la cantidad que le damos como parámetro. Estos objetos creados tienen como parámetro todos los valores aleatorios.

Clase Body.

1. Se realiza un cambio en el método applyForces(), se asegura de que, si la velocidad del objeto es mayor que 7, se acelera a un factor de 0,9 el cual reduce la velocidad del objeto.
2. Se añade un método llamado bounceAtEdge(): el cual hace que el cuerpo invierta su dirección, si este está al borde del mundo.

Punto 2-B.

Cambios realizados.

1. Se cambiaron la cantidad de cuerpos generados en el mundo de forma aleatoria, con un mínimo de dos, para que el juego tenga sentido y un máximo de 10, para no generar muchos cuerpos.

```
public Space()
{
    super(960, 620, 1);

    createObstacles();
    randomBodies(2 + Greenfoot.getRandomNumber(8));
}
```

2. Se añadió una nueva fila de obstáculos que los sonidos están invertidos a la fila de obstáculos anterior

```
public void createObstacles()
{
    int i = 0;
    int j = soundFiles.length-1;
    while (i < soundFiles.length)
    {
        addObject (new Obstacle (soundFiles[i] + ".wav"), 80 + i*60, 450);
        addObject (new Obstacle (soundFiles[j] + ".wav"), 80 + i*60, 150);
        i++;
        j--;
    }
}
```

3. Se añadió un método que emite un sonido que se realiza al chocar entre los objetos.

```
private void choqueCuerpos(){
    Actor body = getOneIntersectingObject(Body.class);
    if (touched && body == null)
    {
        touched = false;
    }
    else if (!touched && body != null)
    {
        touched = true;
        Greenfoot.playSound("blup.wav");
    }
}
```

```
public void act()
{
    applyForces();
    move();
    bounceAtEdge();
    choqueCuerpos();
}
```