TECNICATURA UNIVERSITARIA EN DISEÑO
INTEGRAL DE VIDEOJUEGOS

FACULTAD DE INGENIERÍA
Universidad Nacional de Jujuy

FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS

Trabajo Práctico/Actividad

N° 2

Aban Selene Marisol – TUV000557

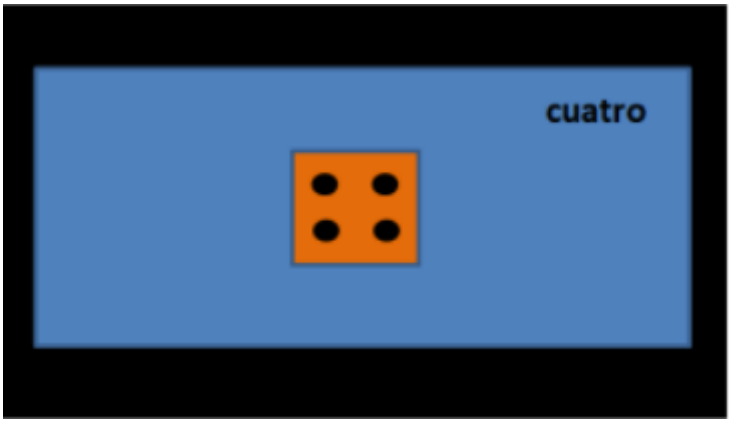
Profesores:

Mg. Ing. Ariel Alejandro Vega

- Punto 1: Desarrolle una historia de usuario, en la cual defina la visualización y movimientode una clase GameObject, de la que heredan Shooter y Asteroide. GameObjects es abstracta, y posee atributos protegidos: posición, imagen; además del método abstracto display() y mover(). Además debe poseer un HUD que visualice la cantidad de vidas delShooter. Utilce un JoyPad para generar los movimientos.

- Punto 2: Desarrolle un videojuego que cumpla con las siguientes especificaciones:

  Realice un diagrama de clases



Como se observa se trata de un dado. El cual al presionar un botón debe generar un número aleatorio entre 1 y 6 y dibujarlo. Además, debe mostrar el número en la parte superior derecha. Repetir esto cuantas veces lo desee y al finalizar (con otro botón) debe dibujar por consola y agrupado en filas de 4 columnas los dados obtenidos.

Al momento de programar utilice constructores sobrecargados. Considere que el dado se muestra en un tablero, este tablero contiene al dado, y al texto.

Además, almacene cada dado obtenido en un arreglo. Considere aplicar la herencia respecto de que existe una clase abstracta padre GameObject, de la que hereda la posición y el método abstracto display(). Luego recrear otra versión donde use imágenes en lugar de dibujar con las primitivas.

- Punto 3: Realice el modelado de las clases que intervienen en el juego frogger a partir de laFig. 1. Realice la construcción de las clases en processing. El juego debe llegar a podermostrar en pantalla la visualización de los diferentes objetos modelados. Utilice herencia y encapsulamiento para los vehículos. Además, los vehículos deben guardarse en una listade objetos que es atributo de la clase SpawnerVehiculos.

- Punto 4

Considere programar un juego de naves. Debe usar imágenes para las naves, los asteroides y los enemigos. Aplique herencia. Use una interface denominada IDisplayable que tenga el método display(). Defina dos interfaces más: IMoveable que tenga el método mover() y Otra IControler que tenga el método readCommand();

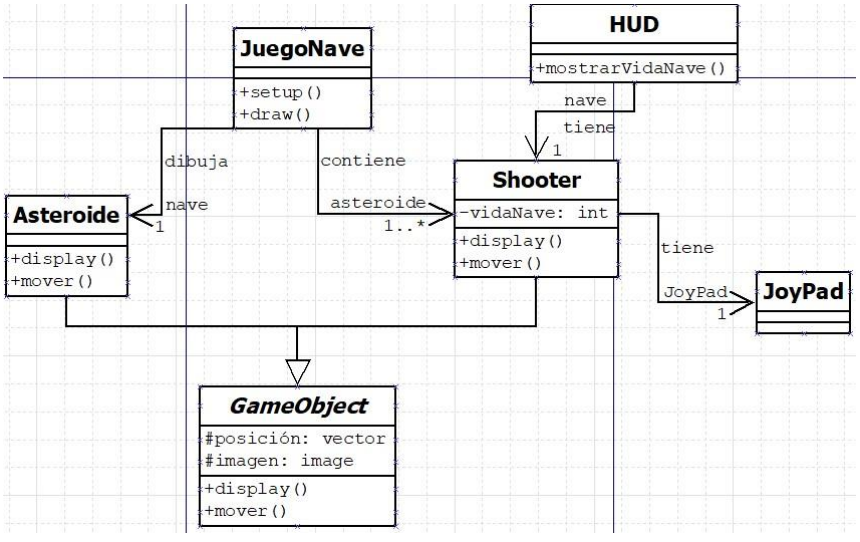
Usando el sentido común haga que las clases Nave, Asteroid y Enemy implementen las interfaces correspondientes. Finalmente use la dependencia para que la nave dispare

balas que serán almacenadas en una lista de balas. Las balas se deben destruir cuandosalen de pantalla.

Figura 1. Modelo juego Fogger

Desarrollo

- Punto 1:



HISTORIA DEL USUARIO	
Código: HU001	Usuario: Desarrollador
Nombre de Historia de usuario: Construcción de escenario y ubicación de game objects	
Prioridad: Alta	Riesgo de desarrollo: Alta
Estimación: 1hora	Iteración asignada: 1
Responsable: Selene	

Descripción:

Como desarrollador de un juego espacial Quiero definir la visualización y movimiento de una clase GameObject, Para crear una experiencia de juego interactiva y emocionante.

Criterios de aceptación:

Ver la representación visual de los GameObjects en la pantalla para poder interactuar con ellos.

Los GameObjects se muevan de acuerdo con sus atributos de velocidad y dirección para que el juego sea desafiante y divertido.

Un indicador visual en la pantalla que muestre la cantidad de vidas restantes del Shooter para poder monitorear mi progreso en el juego.

Que el usuario controle el movimiento del Shooter utilizando un JoyPad para tener una experiencia de juego más inmersiva y cómoda.

He de asegurar de que la clase GameObject sea abstracta y proporcione métodos y atributos necesarios para su correcto funcionamiento.

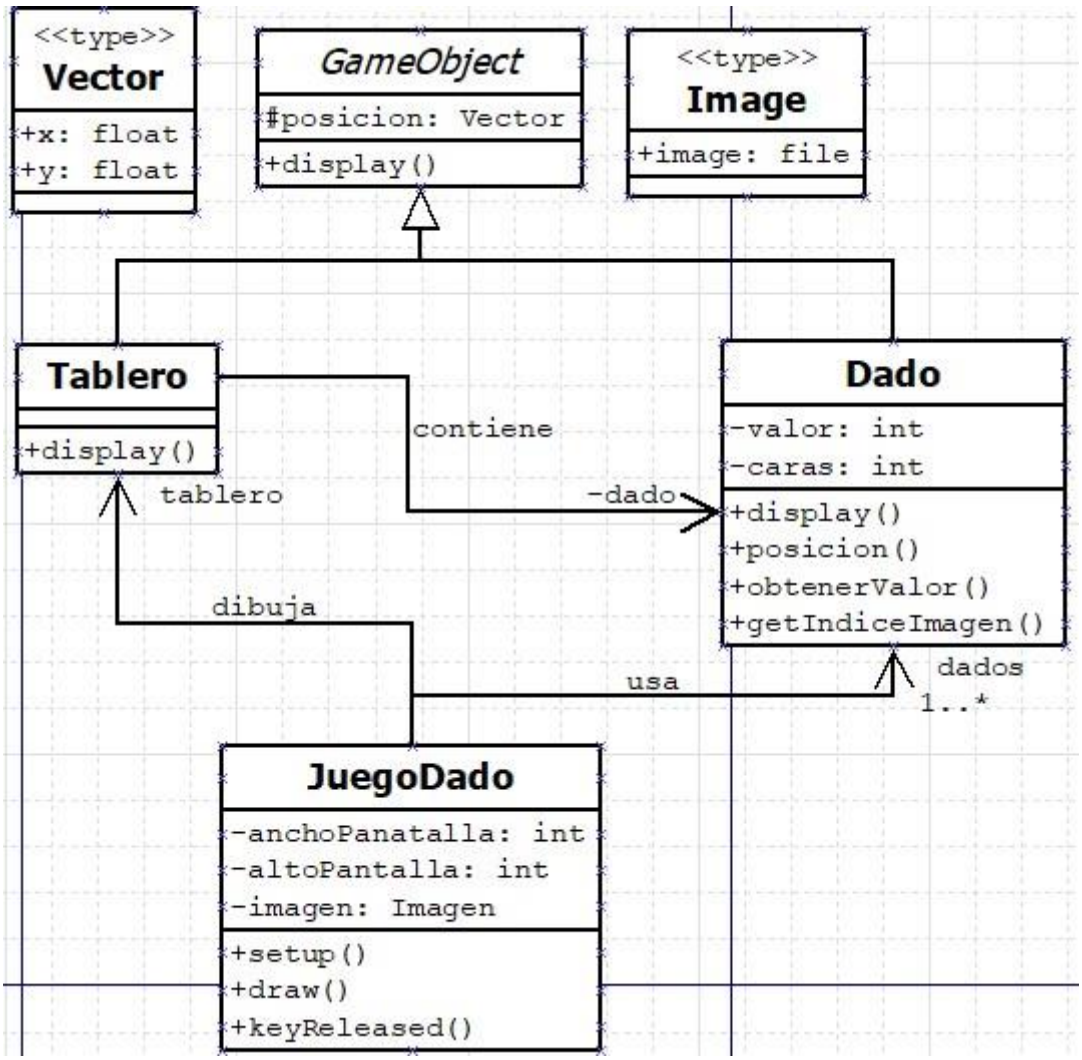
Implementar las clases Shooter y Asteroide que hereden de GameObject y sobrescriban los métodos abstractos según sea necesario.

Diseñar e implementar un HUD que muestre la cantidad de vidas del Shooter de manera clara y legible.

Observaciones: En este modelo

En este modelo se debe desarrollar una historia de usuario que defina la visualización y movimiento de una clase GameObject, de la que heredan Shooter y Asteroide. GameObjects es abstracta, y posee atributos protegidos como posición e imagen, además del método abstracto display() y mover(). Además, debe poseer un HUD que visualice la cantidad de vidas del Shooter. Utilice un JoyPad para generar los movimientos.

- Punto 2:


<<type>>
Vector
+x: float
+y: float
GameObject
#posicion: Vector
+display()
<<type>>
Image
+image: file
Tablero
+display()
contiene
tablero
-dado
Dado
-valor: int
-caras: int
+display()
+posicion()
+obtenerValor()
+getIndiceImagen()
dibuja
usa
dados
1..*
JuegoDado
-anchoPanatalla: int
-altoPantalla: int
-imagen: Imagen
+setup()
+draw()
+keyReleased()

JuegoDado | Dado | GameObject | HUD | Tablero

```
1  private Tablero tablero;
2  private Dado dado;
3  private Hud hud;
4  PImage[] imagenes;
5  int imagen = 0;
6  int numeroDado;
7
8  public void setup(){
9    size(600,400);
10   tablero = new Tablero();
11   tablero.Tablero(new PVector(50,50));
12   dado = new Dado();
13   imagenes = new PImage[6];
14   hud = new Hud(dado);
15
16   int img = 0;
17   do {
18     imagenes[img] = loadImage("cara" + img + ".png");
19     img++;
20    } while (img < imagenes.length);
21 }
22
23 public void draw(){
24   background(0);
25   tablero.display();
26   hud.display();
27   image(imagenes[dado.getIndiceImagen()], width/2, height/2,200,200);
28   imageMode(CENTER);
29 }
30
```

```
37
38 void keyReleased() {
39     if (key == ' ') {
40         dado.display();
41         hud.display();
42         int indiceImg = dado.getIndiceImagen();
43         numeroDado = indiceImg + 1;
44         println("El valor del dado es: " + numeroDado);
45     }
46 }
```

JuegoDado | Dado | GameObject | HUD | Tablero

```
class Dado extends GameObject {
  // Atributos
  private int imagen;
  private int[] valor;

  // Constructor por defecto
  public Dado() {
    // Inicializamos los atributos
    valor = new int[6];
    posicion = new PVector(0, 0);
  }
  //
    public void Posicion(int x, int y) {
    posicion.set(x,y);
  }

  // Método para mostrar la representación visual del dado
  @Override
  public void display(){
   imagen = (int) random(imagenes.length);
   int dado = 0;
   while (dado < valor.length) {
    valor[dado] = (int) random(1, 7);
    dado++;
  }
}

  // Métodos getter y setter para el atributo imagen
  public int getIndiceImagen(){
    return imagen;
  }
  }

  public void setImagen(int imagen) {
    this.imagen = imagen;
  }

  // Métodos getter y setter para el atributo imagen
    public int[] getValor(){
    return valor;
  }
}
```

Java

JuegoDado | Dado | GameObject | HUD | Tablero

```
class Hud {
  Dado dado;

  public Hud(Dado dado) {
    this.dado = dado;
  }

  public void display() {
    int imagenDado = dado.getIndiceImagen();
    textSize(50);
    text((imagenDado+1), 500, 100);

    // Centrar texto horizontalmente
    textAlign(CENTER, CENTER);
    textSize(30);
    text("Presione espacio", width/2, height - 320);
    fill(255);
  }
}
```

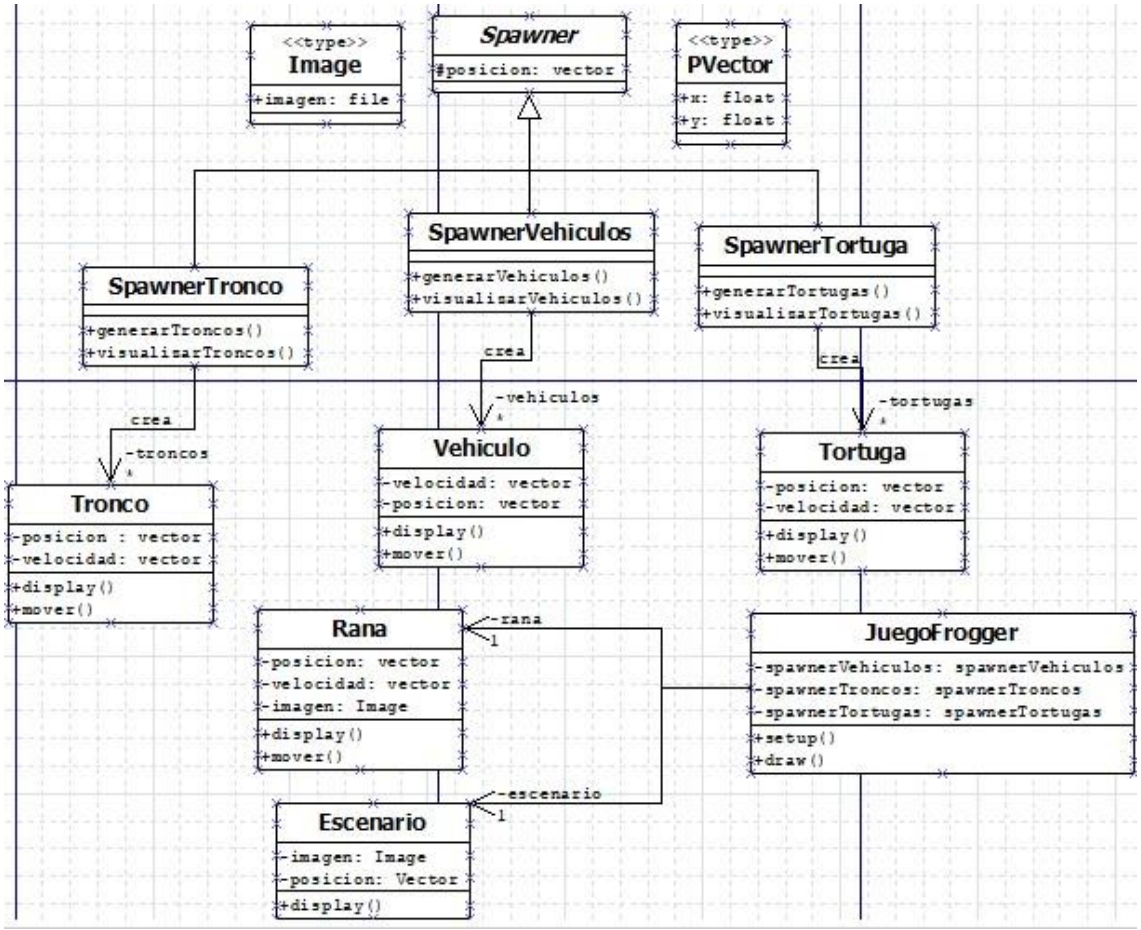
JuegoDado | Dado | GameObject | HUD | Tablero

```
1  class Tablero extends GameObject{
2    //atributos
3
4    //constructor por defecto
5    public Tablero() {
6    }
7
8    //constructor parametrizado
9    public void Tablero(PVector posicion){
10     this.posicion=posicion;
11   }
12
13   //metodos
14   public void display(){
15    rect(this.posicion.x, this.posicion.y, ((width *5)/6), ((height * 4)/6));
16    fill(0);
17   }
18 }
```

JuegoDado | Dado | GameObject

```
1  class GameObject{
2
3    //atributos
4    protected PVector posicion;
5
6    //constructor por defecto
7    public GameObject() {
8
9    }
10   //constructor parametrizado
11
12   //metodos
13   public void display(){
14   }
15 }
16
```

- Punto 3:


<<type>>
Image
+imagen: file
Spawner
#posicion: vector
<<type>>
PVector
+x: float
+y: float
SpawnerVehiculos
+generarVehiculos()
+visualizarVehiculos()
SpawnerTortuga
+generarTortugas()
+visualizarTortugas()
SpawnerTronco
+generarTroncos()
+visualizarTroncos()
crea
crea
crea
-vehiculos
*
-tortugas
*
-troncos
*
Vehiculo
-velocidad: vector
-posicion: vector
+display()
+mover()
Tortuga
-posicion: vector
-velocidad: vector
+display()
+mover()
Tronco
-posicion : vector
-velocidad: vector
+display()
+mover()
Rana
-posicion: vector
-velocidad: vector
-imagen: Image
+display()
+mover()
-rana
1
JuegoFrogger
-spawnerVehiculos: spawnerVehiculos
-spawnerTroncos: spawnerTroncos
-spawnerTortugas: spawnerTortugas
+setup()
+draw()
-escenario
1
Escenario
-imagen: Image
-posicion: Vector
+display()

JuegoFrogger | Escenario | Rana | SpawnTroncos | Spawner | Sp

```
private Escenario escenario;
private Rana rana;
private SpawnerVehiculos spawnerVehiculos;
private SpawnerTortugas spawnerTortugas;
private SpawnerTroncos spawnerTroncos;

void setup() {
  size(600, 700);
  //Escenario
  PVector pos1 = new PVector(0, height - 100);
  PVector pos2 = new PVector(0, height - 400);
  PVector posCesped = new PVector(0, 0);
  escenario = new Escenario(pos1, pos2, posCesped, width);
  escenario.setPosicion1(new PVector(0,height-400 ));
  escenario.setPosicion2(new PVector(0,height-110));
  escenario.setPosicionCesped(new PVector(0, 0));

  //Rana
  rana = new Rana();
  rana.setPosicion(new PVector(width/2, height - 50));
  rana.setVelocidad(new PVector(0, 0));

  //SpawnVehiculos
  spawnerVehiculos = new SpawnerVehiculos(new PVector(0,0));


  //SpawnTortugas
  spawnerTortugas = new SpawnerTortugas(new PVector(0,0));

  //SpawnTroncos
    spawnerTroncos = new SpawnerTroncos(new PVector(0,0));
}

void draw() {
  background(0);
  escenario.display();
  rana.display();
  rana.mover();
  spawnerVehiculos.visualizarVehiculos();
  spawnerVehiculos.mover(width);
  spawnerTortugas.visualizarTortugas();
  spawnerTortugas.mover(width);
  spawnerTroncos.visualizarTroncos();
  spawnerTroncos.mover(width);
}
```

JuegoFrogger | Escenario | Rana | SpawnTroncos | Spawner | SpawnerTortugas | Sp

```
1  class Escenario {
2    private PVector posicion1;
3    private PVector posicion2;
4    private PVector posicionCesped;
5    private PImage imagenSuelo;
6    private PImage imagenCesped;
7
8    // Constructor parametrizado
9    public Escenario(PVector pos1, PVector pos2, PVector posCesped, int ancho) {
10     this.posicion1 = pos1;
11     this.posicion2 = pos2;
12     this.posicionCesped = posCesped;
13     cargarImagenes();
14   }
15
16   // Método para cargar las imágenes del suelo y el césped
17   void cargarImagenes() {
18     imagenSuelo = loadImage("suelo.png");
19     imagenSuelo.resize(50, 0);
20     imagenCesped = loadImage("cesped.png");
21     imagenCesped.resize(150, 0);
22   }
23
24   // Método para dibujar el escenario con las imágenes
25   void display() {
26     float x1 = posicion1.x;
27     while (x1 < width) {
28       image(imagenSuelo, x1, posicion1.y);
29       x1 += imagenSuelo.width;
30     }
31
```

JuegoFrogger | Escenario | Rana | SpawnTroncos | Spawn

```
34       image(imagenSuelo, x2, posicion2.y);
35       x2 += imagenSuelo.width;
36     }
37
38     float xCesped = posicionCesped.x;
39     while (xCesped < width) {
40       image(imagenCesped, xCesped, posicionCesped.y);
41       xCesped += imagenCesped.width;
42     }
43   }
44
45   // Métodos get y set para posicion1
46   public PVector getPosicion1() {
47     return this.posicion1;
48   }
49
50   public void setPosicion1(PVector posicion1) {
51     this.posicion1 = posicion1;
52   }
53
54   // Métodos get y set para posicion2
55   public PVector getPosicion2() {
56     return this.posicion2;
57   }
58
59   public void setPosicion2(PVector posicion2) {
60     this.posicion2 = posicion2;
61   }
62
63   // Métodos get y set para posicionCesped
64   public PVector getPosicionCesped() {
```

```
  public PVector getPosicionCesped() {
    return this.posicionCesped;
  }

  public void setPosicionCesped(PVector posicionCesped) {
    this.posicionCesped = posicionCesped;
  }
}
```

JuegoFrogger | Escenario | Rana | SpawnTroncos | Spawne

```
class Rana {
  private PVector posicion;
  private PVector velocidad;
  private PImage imagen;

  // Constructor por defecto
  public Rana() {
    cargarImagen();
    posicion = new PVector(0, 0);
    velocidad = new PVector(0, 0);
  }

  // Método para cargar la imagen de la rana
  private void cargarImagen() {
    imagen = loadImage("rana.png");
    imagen.resize(30, 0);
  }

  // Método para mostrar la rana en su posición actual
  public void display() {
    image(imagen, posicion.x, posicion.y);
  }

  // Método para mover la rana y controlarla
  public void mover() {
    if (keyPressed) {
      if (key == 'w') {
        velocidad.y = -2;
      } else if (key == 's') {
        velocidad.y = 2;
      } else if (key == 'a') {
```

```
        velocidad.x = -2;
      } else if (key == 'd') {
        velocidad.x = 2;
      }
    } else {
      velocidad.x = 0;
      velocidad.y = 0;
    }
    posicion.add(velocidad);
  }

   // Métodos get
  public PVector getPosicion() {
    return this.posicion;
  }

  public PVector getVelocidad() {
    return this.velocidad;
  }

  // Métodos set
  public void setPosicion(PVector posicion){
    this.posicion=posicion;
  }

  public void setVelocidad(PVector velocidad){
    this.velocidad=velocidad;
  }
}
```

JuegoFrogger | Rana | SpawnTroncos

```
class SpawnerTroncos extends Spawner {
    private ArrayList<Tronco> troncos;

    public SpawnerTroncos(PVector posicion) {
        super(posicion);
        troncos = new ArrayList<Tronco>();
        generarTroncos();
    }

    private void generarTroncos() {
        for (int i = 0; i < 4; i++) {
            PVector pos = new PVector(100 * i, 120);
            PVector vel = new PVector(1.7, 0);
            PImage img = loadImage("tronco.png");
            Tronco tronco = new Tronco(pos, vel, img);
            troncos.add(tronco);
        }

        for (int i = 0; i < 3; i++) {
            PVector pos = new PVector(150 * i, 175);
            PVector vel = new PVector(4, 0);
            PImage img = loadImage("tronco.png");
            Tronco tronco = new Tronco(pos, vel, img);
            troncos.add(tronco);
        }

        for (int i = 0; i < 2; i++) {
            PVector pos = new PVector(200 * i, 240);
            PVector vel = new PVector(2.5, 0);
            PImage img = loadImage("tronco.png");
            Tronco tronco = new Tronco(pos, vel, img);
```

```
            troncos.add(tronco);
        }
    }

    public void visualizarTroncos() {
        for (Tronco tronco : troncos) {
            tronco.display();
        }
    }

    public void mover(int width) {
        for (Tronco tronco : troncos) {
            tronco.mover();

            if (tronco.getPosicion().x > width) {
                tronco.setPosicion(new PVector(0, tronco.getPosicion().y));
            }
        }
    }
}
```

JuegoFrogger | SpawnTroncos | Spawner

```
abstract class Spawner {
  protected PVector posicion;

  public Spawner() {
  }

  public Spawner(PVector posicion) {
    this.posicion = posicion;
  }
}
```

JuegoFrogger | SpawnerTortugas

```
class SpawnerTortugas extends Spawner {
    private ArrayList<Tortuga> tortugas;

    public SpawnerTortugas(PVector posicion) {
        super(posicion);
        tortugas = new ArrayList<Tortuga>();
        generarTortugas();
    }

    private void generarTortugas() {
      for (int i = 0; i < 4; i++) {
            PVector pos = new PVector(30 * i, 200);
            PVector vel = new PVector(2.5, 0);
            PImage img = loadImage("tortuga.png");
            Tortuga tortuga = new Tortuga(pos, vel, img);
            tortugas.add(tortuga);
        }

        for (int i = 0; i < 3; i++) {
            PVector pos = new PVector(30 * i, 270);
            PVector vel = new PVector(3, 0);
            PImage img = loadImage("tortuga.png");
            Tortuga tortuga = new Tortuga(pos, vel, img);
            tortugas.add(tortuga);
        }

         for (int i = 0; i < 2; i++) {
            PVector pos = new PVector(30 * i, 150);
            PVector vel = new PVector(3, 0);
            PImage img = loadImage("tortuga.png");
            Tortuga tortuga = new Tortuga(pos, vel, img);
            tortugas.add(tortuga);
        }

    }

    public void visualizarTortugas() {
        for (Tortuga tortuga : tortugas) {
            tortuga.display();
        }
    }

    public void mover(int width) {
        for (Tortuga tortuga : tortugas) {
            tortuga.mover();

            if (tortuga.getPosicion().x > width) {
                tortuga.setPosicion(new PVector(0, tortuga.getPosicion().y));
            }
        }
    }
}
```

JuegoFrogger | SpawnerVehiculos

```
1  class SpawnerVehiculos extends Spawner {
2    private ArrayList<Vehiculo> vehiculos;
3
4    public SpawnerVehiculos(PVector posicion) {
5      super(posicion);
6      vehiculos = new ArrayList<Vehiculo>();
7      generarVehiculos();
8    }
9
10   private void generarVehiculos() {
11     for (int i = 0; i < 100; i++) {
12       PVector pos = new PVector(100 * i, 350);
13       PVector vel = new PVector(2, 0);
14       PImage img = loadImage("auto" + 1 + ".png");
15       Vehiculo vehiculo = new Vehiculo(pos, vel, img);
16       vehiculos.add(vehiculo);
17     }
18     for (int i = 0; i < 100; i++) {
19       PVector pos = new PVector(100 * i, 550);
20       PVector vel = new PVector(2.5, 0);
21       PImage img = loadImage("auto" + 3 + ".png");
22       Vehiculo vehiculo = new Vehiculo(pos, vel, img);
23       vehiculos.add(vehiculo);
24     }
25     for (int i = 0; i < 100; i++) {
26       PVector pos = new PVector(100 * i, 420);
27       PVector vel = new PVector(2.5, 0);
28       PImage img = loadImage("auto" + 2 + ".png");
29       Vehiculo vehiculo = new Vehiculo(pos, vel, img);
30       vehiculos.add(vehiculo);
31     }
```

JuegoFrogger | SpawnerVehiculos

```
29       Vehiculo vehiculo = new Vehiculo(pos, vel, img);
30       vehiculos.add(vehiculo);
31     }
32
33     for (int i = 0; i < 100; i++) {
34       PVector pos = new PVector(100 * i, 480);
35       PVector vel = new PVector(1.4, 0);
36       PImage img = loadImage("auto" + 0 + ".png");
37       Vehiculo vehiculo = new Vehiculo(pos, vel, img);
38       vehiculos.add(vehiculo);
39     }
40   }
41
42   public void visualizarVehiculos() {
43     for (Vehiculo vehiculo : vehiculos) {
44       vehiculo.display();
45     }
46   }
47
48  public void mover(int width) {
49     for (Vehiculo vehiculo : vehiculos) {
50         vehiculo.mover();
51
52         if (vehiculo.getPosicion().x > width) {
53             vehiculo.setPosicion(new PVector(0, vehiculo.getPosicion().y));
54         }
55     }
56 }
57
58 }
```

JuegoFrogger | Tortuga

```
class Tortuga {
    private PVector posicion;
    private PVector velocidad;
    private PImage imagen;

    public Tortuga(PVector posicion, PVector velocidad, PImage imagen) {
        this.posicion = posicion;
        this.velocidad = velocidad;
        this.imagen = imagen;
    }

    public void display() {
        image(imagen, posicion.x, posicion.y);
    }

    public void mover() {
        posicion.add(velocidad);
    }

    public PVector getPosicion() {
        return posicion;
    }

    public void setPosicion(PVector posicion) {
        this.posicion = posicion;
    }

    public PVector getVelocidad() {
        return velocidad;
    }
```

```
    public void setVelocidad(PVector velocidad) {
        this.velocidad = velocidad;
    }
}
```

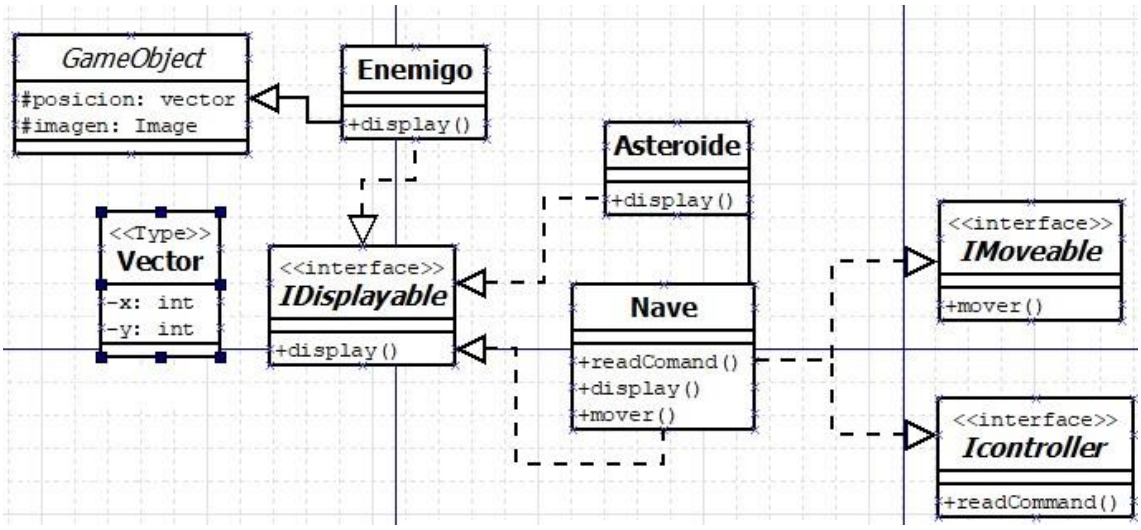
JuegoFrogger | Tortuga | Tronco

```
class Tronco extends Spawner {
    private PVector velocidad;
    private PImage imagen;

    public Tronco(PVector posicion, PVector velocidad, PImage imagen) {
        super(posicion);
        this.velocidad = velocidad;
        this.imagen = imagen;
    }

    public void display() {
        image(imagen, posicion.x, posicion.y);
    }

    public void mover() {
        posicion.add(velocidad);
    }

    public PVector getPosicion() {
        return posicion;
    }

    public void setPosicion(PVector posicion) {
        this.posicion = posicion;
    }
}
```

JuegoFrogger | Tortuga | Tronco | Vehiculo

```
1  class Vehiculo {
2    private PVector posicion;
3    private PVector velocidad;
4    private PImage imagen;
5
6    public Vehiculo(PVector posicion, PVector velocidad, PImage imagen) {
7      this.posicion = posicion;
8      this.velocidad = velocidad;
9      this.imagen = imagen;
10   }
11
12   public void display() {
13     image(imagen, posicion.x, posicion.y);
14   }
15
16   public void mover() {
17     posicion.add(velocidad);
18   }
19
20   // Métodos getter y setter para la posición
21   public PVector getPosicion() {
22     return posicion;
23   }
24
25   public void setPosicion(PVector posicion) {
26     this.posicion = posicion;
27   }
28
29   // Métodos getter y setter para la velocidad
30   public PVector getVelocidad() {
31     return velocidad;
32   }
33
34   public void setVelocidad(PVector velocidad) {
35     this.velocidad = velocidad;
36   }
37
38 }
39
```

- Punto 4:



JuegoNave | Asteroide | Enemigo | GameObject | IDisplayable | Icontroller

```
1  void setup(){}
2
3  void draw(){}
4
5
```

JuegoNave | Asteroide | Enemigo | GameObject | IDisplayable | Icontroller

```
1  class Asteroide extends GameObject implements IDisplayable, IController, IMoveable{
2
3  public Asteroide(){
4    super();
5  }
6
7  public void display(){
8    }
9  public void readCommand(){
10 }
11  public void mover(){
12  }
13 }
14
```

JuegoNave | Asteroide | Enemigo | GameObject | IDisplayable | Icontroller

```
1  class Enemigo extends GameObject implements IDisplayable, IController, IMoveable{
2
3    public Enemigo(){
4    super();
5  }
6
7  public void display(){
8    }
9  public void readCommand(){
10 }
11  public void mover(){
12  }
13 }
14
```

JuegoNave | Asteroide | Enemigo | GameObject | IDisplayable | Icontroller

```
1  abstract class GameObject{
2
3    protected PVector posicion;
4    protected PImage imagen;
5    protected int velocidad;
6
7    public GameObject(){
8    }
9
10 }
11
```

JuegoNave | Asteroide | Enemigo | GameObject | **IDisplayable** | Icontroller

```
1  interface IDisplayable{
2
3  abstract public void display();
4
5  }
6
```

JuegoNave | Asteroide | Enemigo | GameObject | IDisplayable | **IMoveable**

```
1  interface IMoveable{
2
3    abstract public void mover();
4
5  }
```

JuegoNave | Enemigo | GameObject | IDisplayable | IMoveable | **Icontroller**

```
1  interface IController{
2
3    abstract public void readCommand();
4
5  }
```

JuegoNave | Enemigo | GameObject | IDisplayable | IMoveable | Icontroller

```
1  class Nave extends GameObject implements IDisplayable, IController, IMoveable{
2
3  public Nave(){
4    }
5
6  public void readCommand(){
7  }
8
9  public void display(){
10 }
11
12 public void mover(){
13   }
14 }
15
```