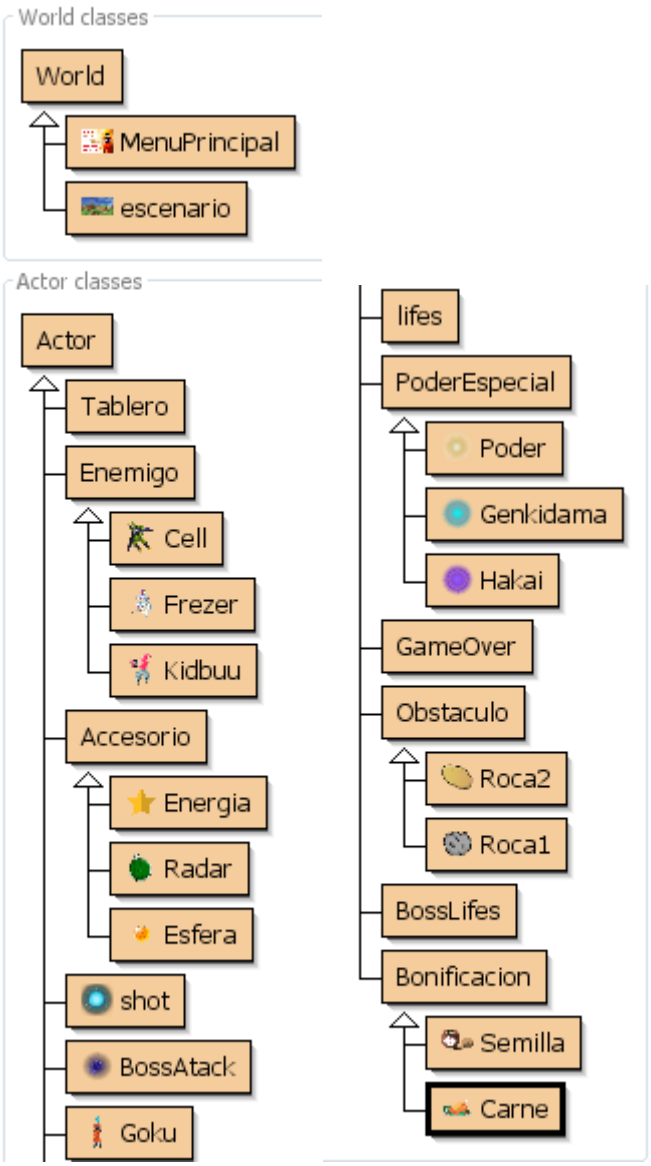
Criterios de proyecto

En el siguiente documento se redactará como se cumplió con los requisitos del proyecto establecidos.

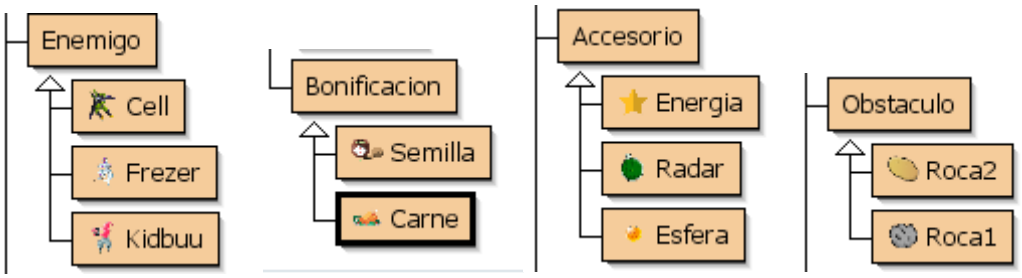
Clases.

- Un mínimo de 5 clases principales: jugador, enemigo, bonificación, accesorio y obstáculo



Herencia

- Al menos 2 o 3 subclases para cada una de las clases principales (excepto tal vez para el jugador). Por ejemplo, 3 tipos de enemigos, 2 tipos de bonificaciones, 3 tipos de accesorios y 2 tipos de obstáculos diferentes.
- Todas las subclases deberán tener algún atributo o comportamiento en común que están heredando de la superclase.
- Cada subclase deberá tener sus propias características y comportamiento particulares que extiendan a la superclase.




```
public abstract class Enemigo extends Actor
{

    public void act()
    {

    }

    /**
     * Metodo que se encarga de comprobar las vidas de goku.
     *
     */
    public void reglas()
    {
        Goku n = (Goku)getOneIntersectingObject(Goku.class);
        if(n!=null)
        {
            escenario esc = (escenario)getWorld();
            esc.removeObject(n);
            esc.addObject(new Goku(),50,300);
            esc.vidas.decrementar();
            if(esc.vidas.obtenerValor() == 0)
            {
                GameOver t =new GameOver();
                getWorld().addObject(t,((getWorld().getWidth()/2)+30),((getWorld().getHeight
            }
        }

    }
```

```
public class Cell extends Enemigo
{
    int  velocidad = 6;
    int direccion = 1;
    boolean cambia = false;

    /**
     * Metodo que se encarga de ejecutar este metodo mientras este el enemigo
     * escenario
     */
    public void act()
    {
        if(getY()>=30 && !cambia)
        {
            direccion = -1;
        }
        else
        {
            cambia = true;
        }
        if(getY() <= getWorld().getHeight()-30 && cambia)
        {
            direccion = 1;
        }
        else
        {
            cambia = false;
        }
        setLocation(getX(),getY()+(velocidad*direccion));
        reglas();
        ataque();
```

```
public class Frezer extends Enemigo
{
    int  velocidad = 4;
    int direccion = 1;
    boolean cambia = false;

    /**
     * Metodo que se encarga de las acciones que realizara frezer cuando aparesca
     *
     */
    public void act()
    {
        if(getY()>=30 && !cambia)
        {
            direccion = -1;
        }
        else
        {
            cambia = true;
        }
        if(getY() <= getWorld().getHeight()-30 && cambia)
        {
            direccion = 1;
        }
        else
        {
            cambia = false;
        }
        setLocation(getX(),getY()+(velocidad*direccion));
        reglas();
        ataque();
```

```java
public class Kidbuu extends Enemigo
{
    int  velocidad = 8;
    int direccion = 1;
    boolean cambia = false;

    /**
     * Metodo que se encarga de ejecutar las acciones de Kidbuu cuando aparece
     */
    public void act()
    {
        if(getY()>=30 && !cambia)
        {
            direccion = -1;
        }
        else
        {
            cambia = true;
        }
        if(getY() <= getWorld().getHeight()-30 && cambia)
        {
            direccion = 1;
        }
        else
        {
            cambia = false;
        }

        setLocation(getX(),getY()+(velocidad*direccion));
        reglas();
        ataque();
```

Polimorfismo

- Un mínimo de 2 métodos polimórficos en alguna de las subclases.
- Uno de los métodos polimórficos deberá sobrescribir a un método abstracto y el otro a un método concreto.

```java
public abstract class Enemigo extends Actor
{
    /**
     * Metodo que se encarga de hacer los ataques del enemigo cada que se de un numero rando
     *
     */
    public abstract void ataque();
```

```java
public class Cell extends Enemigo
{
    int  velocidad = 6;
    int direccion = 1;
    boolean cambia = false;

    @Override
    public void ataque()
    {
        int x = Greenfoot.getRandomNumber(30);

        if(x == 10)
        {
            BossAtack attack = new BossAtack(2);
            getWorld().addObject(attack,getX()-10,getY());
        }
    }
```

```
public class Frezer extends Enemigo
{
    int  velocidad = 4;
    int direccion = 1;
    boolean cambia = false;

    @Override
    public void ataque()
    {
       int x = Greenfoot.getRandomNumber(40);

       if(x == 10)
          {
           BossAtack attack = new BossAtack(2);
             getWorld().addObject(attack,getX()-10,getY());
          }
    }
```

```
public class Kidbuu extends Enemigo
{
    int  velocidad = 8;
    int direccion = 1;
    boolean cambia = false;

    @Override
    public void ataque()
    {
       int x = Greenfoot.getRandomNumber(20);

       if(x == 10)
        {
          BossAtack attack = new BossAtack(2);
          getWorld().addObject(attack,getX()-10,getY());
        }
    }
```

Clases abstractas

- Al menos 1 de las clases principales deberá ser clase abstracta.

```
public abstract class Accesorio extends Actor
{
    public void act()
    {
    }
}
```

Niveles de dificultad

- Para el caso de juegos, se deberá contar con un mínimo de 3 niveles de dificultad: básico, intermedio y difícil.

  Básico:

  Intermedio:

  Difícil: