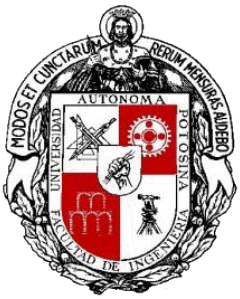
Universidad Autónoma De San Luis Potosí



Facultad De Ingeniería

Ciencias De La Computación

Ingeniería En Computación

Programación Orientada a Objetos

Proyecto Avatar

Manual del Programador

Profesor: Cesar Augusto Puente Montejano

Elaboro: Hernández Cuevas Jeneffer Stephane

La clase AvatarIntro representa a la pantalla principal y hereda de la clase World

```
public class AvatarIntro extends World
{
```

Variables para poder cambiar de mundo y para un botón que nos lo permitiera

```
  ButtonHow b;
    Nivel1 n1;
```

Utilizamos su constructor

```
    public AvatarIntro();
```

Método actuar para poder hacer el cambio de mundo

```
   public void act();
}
```

Clase del mundo para usar como ayuda que hereda de World

```
public class HowToPlayWorld extends World
{
```

Se utiliza el constructor

```
    public HowToPlayWorld();
```

Método actuar ya que requiere hacer cambio de mundo

```
    public void act();
}
```

Clase que representa el nivel 1 y hereda de World

```
public class Nivel1 extends World
{
```

Representan los contadores y el personaje principal que está en el nivel

```
    Avatar aang;
    Counter water;
    Counter earth;
    Counter air;
    Counter fire;
    Counter life;
```

Constructor del nivel 1

```
    public Nivel1();
```

Método actuar que servirá para que los actores actúen dentro del nivel

```
    public void act();
```

Método para poner contadores

```
    public void counters();
```

Método para agregar a los elementos

```
    public void addElements();
```

Método para agregar a los Bonus

```
    public void addBonus();
```

Método para agregar a los Enemigos

```
    public void addEnemy();
```

Método que decrementa a un contador Agua

```
    public void decreaseWater();
```

Método que decrementa a un contador Fuego

```
    public void decreaseFire();
```

Método que decrementa a un contador Tierra

```
    public void decreaseEarth();
```

Método que decrementa a un contador Aire

```
    public void decreaseAir();
```

Método que incrementa un contador de Vida

```
  public void increaseLife();
```

Método que decrementa a los contadores de Elementos

```
  public void decreaseElements();
```

```
Método que incrementa a los contadores de Elementos
  public void increaseElements();
  Método que cambia el contador de vida a cero
  public void removeLife();
  Método que verifica si el contador de vida ha llegado a cero para poder cambiar de mundo
     public void checkLife();
  Método que incrementa el contador de vida;
  public void decreaseLife();
  Método que verifica si los contadores de elementos han llegado a cero para poder cambiar al mundo
  public void checkElements();
  Método que llena el contador de vida
  public void fullLife();
}
Clase  que representa el nivel 2 y hereda de World
public class Nivel2 extends World
{
Representan los contadores y el personaje principal que está en el nivel
    Avatar aang;
    Counter water;
    Counter earth;
    Counter air;
    Counter fire;
    Counter life;
Constructor del nivel 2
    public Nivel2();
    Método actuar que servirá para que los actores actúen dentro del nivel
    public void act();
    Método para poner contadores
    public void counters();
       Método para agregar a los elementos
    public void addElements();
       Método para agregar a los Bonus
    public void addBonus();
    Método para agregar a los Enemigos
    public void addEnemy();
    Método que decrementa a un contador Agua
    public void decreaseWater();
    Método que decrementa a un contador Fuego
    public void decreaseFire();
    Método que decrementa a un contador Tierra
    public void decreaseEarth();
    Método que decrementa a un contador Aire
    public void decreaseAir();
  Método que incrementa un contador de Vida
  public void increaseLife();
  Método que decrementa a los contadores de Elementos
  public void decreaseElements();
  Método que incrementa a los contadores de Elementos
  public void increaseElements();
  }
  Método que cambia el contador de vida a cero
  public void removeLife();
  Método que verifica si el contador de vida a llegado a cero para poder cambiar de mundo
     public void checkLife();
```

```
  Método que incrementa el contador de vida;
  public void decreaseLife();
  Método que verifica si los contadores de elementos han llegado a cero para poder cambiar al mundo
  public void checkElements();
  Método que llena el contador de vida
  public void fullLife();
}
```

Clase que representa el nivel 3 y hereda de World

```
public class Nivel3 extends World
{
Representan los contadores y el personaje principal que está en el nivel
    Avatar aang;
    Counter water;
    Counter earth;
    Counter air;
    Counter fire;
    Counter life;
Constructor del nivel 3
    public Nivel3();
    Método actuar que servirá para que los actores actúen dentro del nivel
    public void act();
    Método para poner contadores
    public void counters();
       Método para agregar a los elementos
    public void addElements();
       Método para agregar a los Bonus
    public void addBonus();
    Método para agregar a los Enemigos
    public void addEnemy();
    Método que decrementa a un contador Agua
    public void decreaseWater();
    Método que decrementa a un contador Fuego
    public void decreaseFire();
    Método que decrementa a un contador Tierra
    public void decreaseEarth();
    Método que decrementa a un contador Aire
    public void decreaseAir();
  Método que incrementa un contador de Vida
    public void increaseLife();
  Método que decrementa a los contadores de Elementos
   public void decreaseElements();
  Método que incrementa a los contadores de Elementos
  public void increaseElements();
  Método que cambia el contador de vida a cero
   public void removeLife();
  Método que verifica si el contador de vida ha llegado a cero para poder cambiar de mundo
  public void checkLife();
  Método que incrementa el contador de vida;
  public void decreaseLife();
  Método que verifica si los contadores de elementos han llegado a cero para poder cambiar al mundo
  public void checkElements();
  Método que llena el contador de vida
  public void fullLife();
}
```

Clase que representa el mundo si es que se pierde, hereda de la clase World

```
public class LostWorld extends World
{
Constructor de la clase
    public LostWorld();
    Método para que el mundo actué para cambiar de mundo
    public void act();
}
```

Clase que representa el mundo cuando se gana, hereda de la clase World

```
public class FinalWorld extends World
{
Constructor de la clase
    public FinalWorld();
}
```

Clase que representa el botón, hereda de la clase Actor

```
public class ButtonHow extends Actor
{
    Método actúa para poder cambiar de mundo
    public void act();
}
```

Clase que representa el personaje principal, hereda de la clase Actor

```
public class Avatar extends Actor
{
    Para poder indicar el mundo en que esta
    private int mundo;
    Constructor de la clase que recibe como parámetro un entero para checar el mundo en el que está el personaje principal
    public Avatar(int m);
    Método actúa para que el personaje  interactúe con más objetos
    public void act();
    Método para el  movimiento del personaje principal
    public void keys();
    Método para indicar que se hace si se atrapan los elementos
    public void catchElements();
    Método para indicar que se hace si se atrapan los bonus
    public void catchBonus();
    Método para indicar que se hace si se tocan los enemigos
    public void catchEnemy();
}
```

Clase abstracta que representa el enemigo, hereda de la clase Actor y no se pueden crear objetos de esta clase

```
abstract public class Enemy extends Actor
{
    Método actúa abstracto
    abstract public void act();
    Método para el  movimiento del enemigo
    public void move();
}
```

Clase magia que representa un tipo de enemigo, hereda la clase Enemy

```
public class Magic extends Enemy
{
    Método actúa para que la magia se pueda mover
    public void act();
}
```

Clase rayo que representa un tipo de enemigo, hereda la clase Enemy

```
public class Ray extends Enemy
{
    Método actúa para que el rayo se pueda mover
    public void act();
}
```

Clase bomba que representa un tipo de enemigo, hereda la clase Enemy

```
public class Bomb extends Enemy
{
    Método actúa para que la bomba se pueda mover
    public void act();
}
```

Clase abstracta que representa el bonus, hereda de la clase Actor y no se pueden crear objetos de esta clase

```
abstract public class Bonus extends Actor
{
    Método actúa del Bonus abstracto
    abstract public void act();
    Método para el movimiento de los bonus
    public void move();
}
```

Clase de vida que representa un tipo de bonus, hereda la clase Bonus

```
public class Quick extends Bonus
{
    Método actúa del Quick para que se mueva
    public void act();
}
```

Clase de Reducción que representa un tipo de bonus, hereda la clase Bonus

```
public class Reduction extends Bonus
{
    Método actúa del Reduction para que se mueva
    public void act();
}
```

Clase que incrementa Vida que representa un tipo de bonus, hereda la clase Bonus

```
public class LifePlus extends Bonus
{
    Método actúa del LifePLus para que se mueva
    public void act();
}
```

Clase abstracta que representa los Elementos, hereda de la clase Actor y no se pueden crear objetos de esta clase

```
abstract public class  Element  extends Actor
{
    Método actúa abstracto
    abstract public void act();
}
```

Clase agua que representa un tipo de elemento, hereda la clase Element

```
public class Water extends Element
{
    Método actúa para el agua
    public void act();
}
```

Clase aire que representa un tipo de elemento, hereda la clase Element

```
public class Air extends Element
{
    Método actúa para el aire
    public void act();
}
```

Clase fuego que representa un tipo de elemento, hereda la clase Element

```
public class Fire extends Element
{
    Método actúa para el fuego
    public void act();
}
```

Clase tierra que representa un tipo de elemento, hereda la clase Element

```
public class Earth extends Element
{
    Método actúa para la tierra
    public void act();
}
```