**Documentación: Proyecto 1**

Andrey Ureña Bermúdez

2022017442

Jorge Luis Rodríguez Cruz

2020010773

Escuela de Computación, Tecnológico de Costa Rica

**Programación orientada a objetos**

**Grupo #1**

Yuen Cheong Law Wan

Primer semestre, 2023

Fecha de entrega: 14 de abril

* **Descripción general de la solución implementada y el diagrama de clases y una breve explicación de cada clase.**

\*Insertar el diagrama de clases actualizado, yo lo puedo actualizar si gusta.

**Controller:**

* Controlador:
* Main:

**Model:**

* Gestor de partida:
* Organismo: Se crean métodos importantes para los organismos; como el de seguir un alimento u organismo, huir de otro organismo, elegir cierta dirección según las condiciones del entorno, así como obtener las coordenadas e información de los atributos de los organismos en juego.
* Alimento: Se crean en esta clase, métodos para obtener las coordenadas de un alimento, además, el método “serComido” el cual va a tomar un papel muy importante en la lógica del juego.
* Constantes: Se utiliza para crear de manera predeterminada una cierta cantidad de los atributos, al momento de iniciar el juego.
* Direccion**: \*\*\*NO se usa\*\*\* creo**
* Energizante: Se utiliza para el alimento que brinda energía a un organismo al ser comido. Tiene métodos que permiten saber si este ya fue comido, o si, al contrario, ninguno ha logrado llegar hasta sus coordenadas.
* Visión: Se utiliza para el alimento que brinda visión a un organismo al ser comido. Tiene métodos que permiten saber si este ya fue comido, o si, al contrario, ninguno ha logrado llegar hasta sus coordenadas.
* Velocidad: Se utiliza para el alimento que brinda velocidad a un organismo al ser comido. Tiene métodos que permiten saber si este ya fue comido, o si, al contrario, ninguno ha logrado llegar hasta sus coordenadas.
* Velocista: Esta clase, es para que un organismo *random*, tome el tipo velocista y con esto, además de la energía, siempre se enfoque en conseguir aumentar su atributo de velocidad.
* Visionador: Esta clase, es para que un organismo *random,* tome el tipo visionador y con esto, además de la energía, siempre se enfoque en conseguir aumentar su atributo de visión.

**View:**

* Información: Es una ventana secundaria que aparecerá con los datos del organismo presionado, en el mapa de juego. Menciona atributos como la edad, visión, energía y velocidad del organismo al momento de ser presionado.
* Mapa: Se crea el área de juego; el cual será una matriz de 50x50. En este lugar es donde también se agregan los componentes necesarios para el juego, como los organismos y alimentos
* MicroGameGUI: Se encarga de iniciar los componentes y con esto, crear la ventana principal del juego, en donde se solicitará al jugador ingresar los valores máximos y mínimos de los atributos, así como el aumento y decremento. Valida que los valores ingresados sean números, de lo contrario, solicitará de nuevo los datos y no podrá pasar al mapa de juego
* **Aplicación de la herencia y polimorfismo, así como ventajas y desventajas de la implementación de estas.**

La herencia y el polimorfismo son dos conceptos fundamentales de la programación orientada a objetos que se utilizan para diseñar y desarrollar sistemas complejos y flexibles. En el contexto de un juego donde se pasan atributos de organismos a organismos e igual con alimentos, se pueden utilizar estos conceptos para crear una jerarquía de clases que representen los diferentes tipos de organismos y alimentos en el juego.

En este juego, los organismos pueden tener diferentes atributos, como tamaño, velocidad, nivel de energía y resistencia, y también pueden consumir diferentes tipos de alimentos que tienen diferentes propiedades nutricionales. Para modelar esto en un sistema de programación orientado a objetos, se puede utilizar la herencia y el polimorfismo.

La herencia se puede utilizar para crear una clase base que represente un organismo genérico y, a partir de ella, crear clases que representen diferentes tipos de organismos. Por ejemplo, se puede crear una clase "Organismo" que tenga los atributos y métodos comunes a todos los organismos, y luego crear subclases como "Planta", "Herbívoro" y "Carnívoro", que tienen atributos y métodos específicos para cada tipo de organismo.

Por otro lado, el polimorfismo permite que diferentes objetos de diferentes clases se comporten de la misma manera. En el contexto de un juego donde se pasan atributos de organismos a organismos e igual con alimentos, esto significa que se pueden crear diferentes tipos de alimentos que tengan diferentes propiedades nutricionales y que puedan ser consumidos por diferentes tipos de organismos, sin importar la clase a la que pertenecen.

Por ejemplo, se puede crear una clase "Alimento" que tenga los atributos y métodos comunes a todos los alimentos, y luego crear subclases como "Planta", "Carne" y "Fruta", que tienen diferentes propiedades nutricionales y pueden ser consumidos por diferentes tipos de organismos. Los organismos también pueden tener métodos de consumo que tomen como parámetro un objeto de tipo "Alimento", lo que permite que diferentes tipos de organismos consuman diferentes tipos de alimentos.

En resumen, la herencia y el polimorfismo se pueden utilizar en un juego donde se pasan atributos de organismos a organismos e igual con alimentos para crear una jerarquía de clases que representen los diferentes tipos de organismos y alimentos en el juego, lo que permite crear sistemas complejos y flexibles que se adapten a diferentes situaciones y escenarios en el juego.

Por otra parte, con el tema de las ventajas y desventajas de la implementación de la herencia y polimorfismo, se lograron notar ventajas tales como:

* Reutilización de código: permite crear nuevas clases a partir de clases existentes, lo que permite reutilizar el código ya escrito y reduce el tiempo de desarrollo.
* Facilidad de mantenimiento: al utilizar herencia, cualquier cambio en la clase base se refleja automáticamente en las clases derivadas, lo que facilita el mantenimiento del código.
* Facilidad de lectura: la jerarquía de clases puede hacer que el código sea más fácil de leer y entender.
* Flexibilidad: permite que diferentes objetos de diferentes clases se comporten de la misma manera, lo que aumenta la flexibilidad del código y lo hace mas adaptable a diversas situaciones.

Además, de desventajas como:

* Acoplamiento: las clases derivadas están acopladas a la clase base, lo que puede dificultar los cambios en la clase base y aumentar la complejidad del código.
* Problemas de diseño: el mal diseño de la jerarquía de clases puede resultar en una estructura compleja y difícil de entender.
* Complejidad: el uso excesivo de la herencia puede aumentar la complejidad del código y hacer que sea difícil de entender.

En general, la herencia y el polimorfismo son herramientas muy útiles en la programación orientada a objetos, pero es importante usarlos con cuidado y comprender sus ventajas y desventajas para crear un código efectivo y mantenible.

**Referencias bibliográficas**

Álvarez, C. (2022). Controller: comunicando vista y modelo. ArquitecturaJava.

<https://www.arquitecturajava.com/spring-controller-comunicando-vista-y-modelo/>

González, M. (2015). Programación en Lenguaje Java: Herencia y polimorfismo. Universidad de Cantabria.

<https://ocw.unican.es/pluginfile.php/2330/course/section/2281/cap8-herencia.pdf>

Martínez, J. (s.f.) Fundamentos de programación en Java. Universidad Complutense de Madrid.

<https://www.tesuva.edu.co/phocadownloadpap/Fundamentos%20de%20programcion%20en%20Java.pdf>