Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en computación

Programación Orientada a Objetos

Semestre I, 2023

Proyecto 1

Juego Microorganismos - Herencia y Polimorfismo

Realizado por:

Alejandra Delgado Pérez 2022104899

Jeremmy Aguilar villanueva 2022191791

Profesor: Yuen Law

Cartago, abril 2023

Descripción

La programación orientada a objetos (POO) es un paradigma de programación que se basa en la creación de objetos que interactúan entre sí para resolver un problema o realizar una tarea.

En POO, los objetos son entidades que tienen un estado y un comportamiento definidos por su clase. Una clase es una plantilla o molde que define las propiedades y métodos que tendrán los objetos que se creen a partir de ella.

A continuación, el juego que presentamos es el primer proyecto de la asignatura Programación Orientada a Objetos el cual combina diversión y aprendizaje. Se trata de un juego sencillo pero divertido que permitirá a el usuario familiarizarse con los conceptos de la programación orientada a objetos mientras se divierte empleando los principios de la programación orientada a objetos, la herencia y el polimorfismo con clases que representan a los organismos involucrados (alimentos, microorganismos npc y el jugador), objetos en el juego y métodos que manejan la lógica del juego.

La herencia y el polimorfismo son dos conceptos fundamentales de POO. La herencia es un mecanismo que permite crear nuevas clases a partir de clases existentes, aprovechando las propiedades y métodos ya definidos en la clase base o padre. La nueva clase que se crea se denomina subclase/ clase derivada o clase hija, y ésta hereda todas las características de la clase base o superclase/ padre.

El polimorfismo, se refiere a la capacidad de los objetos de una misma clase o de clases relacionadas por herencia de responder de forma diferente al mismo mensaje o llamado a un método. Es decir, un objeto puede tener diferentes formas o comportamientos según el contexto en el que se utilice.

En este juego el jugador (el usuario) podrá controlar un personaje el cual se encuentra en una casilla de un tablero de 50x50, alrededor de alimentos y microorganismos npc, el mismo podrá moverlo a su conveniencia, por supuesto considerando las reglas del juego y los atributos que este tenga (energía, velocidad, visión y edad), el jugador tendrá como prioridad alimentarse ya sea de organismos de tipo alimento u otros microorganismos, para incrementar su energía, velocidad y visión, para ello el jugador tendrá que esperar su turno para jugar.

Los organismos de tipo microorganismos NPC se comportan igual que el jugador y se moverán de forma independiente por código y este jugará en su respectivo turno, estos

tienen como prioridad aumentar su velocidad o visión pero su objetivo principal siempre será aumentar su energía para ganar. Es importante considerar las siguientes características para que el juego se lleve a cabo:

1. La energía permite al organismo moverse

2. La energía incrementa cuando los organismos se alimentan

3. La energía disminuye con cada movimiento.

4. La visión permite al organismo ver en un radio a su alrededor y detectar otros organismos y alimentos.

5. La visión se incrementa con ciertos alimentos.

6. La visión decrementa con la edad.

7. La velocidad determina cuántos pasos puede moverse el organismo en un solo turno.

8. La velocidad incrementa con ciertos alimentos.

9. La velocidad decrementa conforme aumenta la energía.

10. La edad aumenta con cada turno.

Cuando los organismos se encuentran, la decisión de quién come a quién se determina de la siguiente manera:

1. El organismo con mayor energía gana.

2. Si los 2 tienen igual energía, gana el de mayor velocidad.

3. Si los 2 tienen igual velocidad, gana el de mayor edad.

4. Si los 2 tienen igual edad, se selecciona un ganador aleatoriamente.

Además cuando un organismo consume a otro obtiene la mitad de sus características: energía, visión y velocidad. Cada organismo se moverá y tomará decisiones independientemente, pero basado en información de su entorno, y según su personalidad.

Los organismos vendrán representados de la siguiente forma:

- Amarillo: Microorganismos
- Azul: Alimentos
- Rojo: Jugador

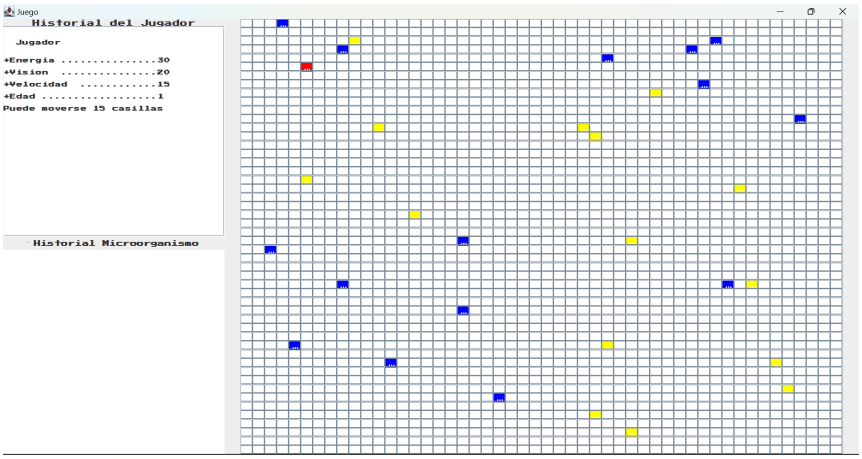


Imagen 1: Representación gráfica del juego (organismos)

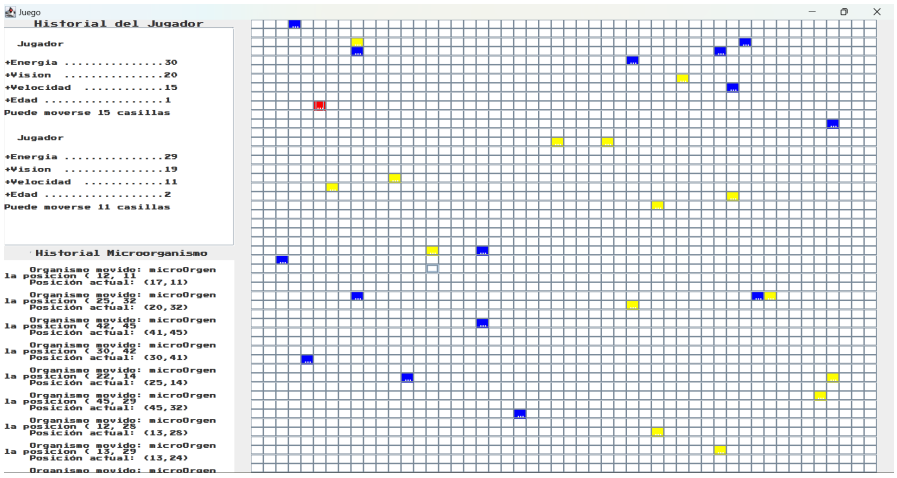
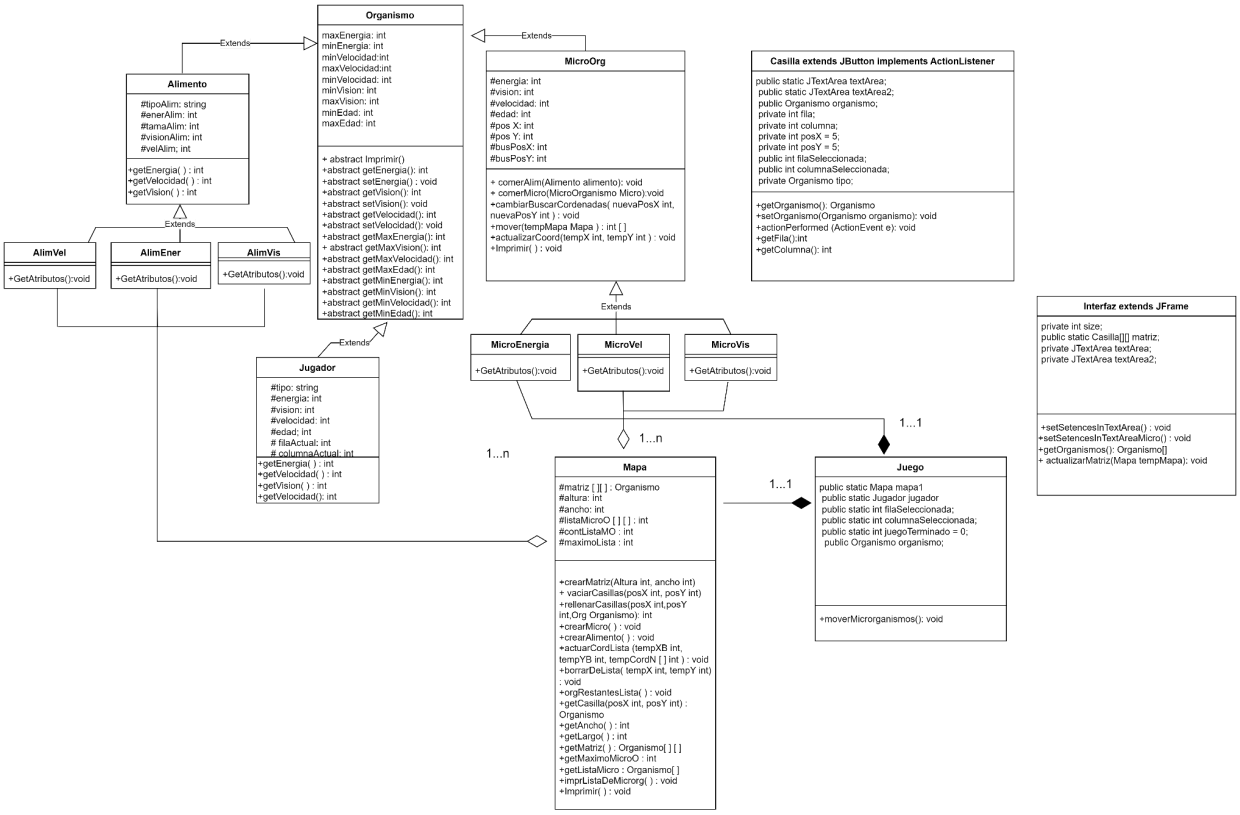


Imagen 2: Representación gráfica del juego (organismos)- Actualización del historial del juego

Diagrama de clases



Explicación de cada clase

Clase Organismo: esta clase es la clase base- padre de la clase Alimento y Microorganismo. En esta clase se puede obtener el tipo de organismo, las posiciones actuales de los distintos organismos, sus atributos (energía, visión, velocidad y edad). Es la clase abstracta que representa un organismo en el juego y contiene los métodos abstractos para moverse y comer.

Clase Alimento: esta clase es la clase hija/ subclase de la clase Organismo. Esta hereda todos los métodos abstractos y variables estáticas definidas en la clase padre, en esta se obtiene el tipo de organismo y se guardan las características o atributos del alimento(energía, visión, velocidad). Es la clase que representa el alimento en el juego.

Clase AlimGrande: esta clase es la clase hija/ subclase de la clase Alimento. Esta genera de forma aleatoria el alimento de tamaño grande con sus respectivas características.

Clase AlimPeque: esta clase es la clase hija/ subclase de la clase Alimento. Esta genera de forma aleatoria el alimento de tamaño pequeño con sus respectivas características.

Clase AlimMediano: esta clase es la clase hija/ subclase de la clase Alimento. Esta genera de forma aleatoria el alimento de tamaño mediano con sus respectivas características.

Clase Jugador: Esta es la clase hija de la clase Organismo, esta clase hereda todos los métodos abstractos de la clase padre, en esta clase se guarda los atributos/ características del jugador (energía, velocidad, visión y edad) para luego desde cualquier clase obtenerlas, obtener su posición, generar movimientos.

Clase MicroOrg: La subclase de Organismo que representa a los microorganismos en el juego.

Clase MicroEnergia: esta clase es la clase hija/ subclase de la clase MicroOrg. Esta genera de forma aleatoria un microorganismo con más energía.

Clase MicroVision: esta clase es la clase hija/ subclase de la clase MicroOrg. Esta genera de forma aleatoria un microorganismo con más visión.

*Clase MicroVelocidad:*esta clase es la clase hija/ subclase de la clase MicroOrg. Esta genera de forma aleatoria un microorganismo con más velocidad.

Clase Casilla: Se crea la clase casilla que extiende de JButton e implementa ActionListener. La clase representa las casillas del juego del tamaño que se solicita y contiene atributos como su posición, el tipo de organismo en la casilla y si se ha hecho clic o no. El método actionPerformed es el método principal de esta clase ya que se llama cada vez que se hace clic en un botón. El método primero verifica si el juego aún se está ejecutando y luego calcula la distancia entre la casilla en el que se hizo clic y la posición actual del jugador. Si la distancia es menor o igual a la velocidad del jugador, el jugador puede moverse a esa casilla. Si la casilla está vacía, el jugador se mueve hacia él y el mosaico se marca con el icono del jugador. Si la ficha contiene comida, el jugador gana energía y demás atributos que contenga dicho alimento.

Clase Interfaz: La clase que representa la interfaz gráfica de usuario (GUI) del juego. Interfaz utiliza la instancia de Mapa para mostrar la representación gráfica del mapa y actualizar su estado. El constructor toma dos parámetros, un tamaño entero y un objeto Mapa. El parámetro de tamaño determina el tamaño de la matriz de botones a crear. El objeto Mapa se usa para llenar la matriz de botones con información sobre el juego. El

constructor configura la GUI creando dos paneles, uno a la izquierda y otro a la derecha. El panel de la izquierda contiene un componente JTextArea que muestra el historial del juego, movimientos realizados, información general del estado del juego y las características actualizadas del jugador con cada movimiento. El panel de la derecha contiene la matriz de botones que representa el tablero de juego. La clase interfaz contiene un método moverMicroorganismos que mueve los microorganismos en el juego. Este método utiliza el objeto Mapa para determinar la ubicación de cada microorganismo y lo mueve a una nueva ubicación.

Clase Mapa: Clase que representa el mapa del juego y contiene una matriz de casillas que contienen organismos y alimentos. Mapa contiene una matriz de casillas que puede contener organismos y alimentos.

Clase Juego: Es la clase principal que contiene el método main y actúa como controlador del juego. Juego tiene una instancia de Mapa, Jugador e Interfaz, y llama a los métodos de estas clases para controlar el flujo del juego.

Clase Jugador: La clase que representa al jugador y contiene sus atributos como la posición actual, la cantidad de alimentos y la energía restante. Jugador es un tipo de Organismo y puede estar contenido en una casilla del mapa.

Herencia y polimorfismo

Se aplica la herencia en la clase de Organismo quien es padre de las clases Alimento y MicroOrg, estas adquieren los métodos de la clase Organismo, estos métodos y variables estáticas son los siguientes:

```
// Variable global (tipo de organismo: alimento o microorganismo)
String tipo;
// Variables estáticas [MAXIMOS Y MINIMOS]
```

Estas variables son las encargadas de delimitar los atributos de los organismos con un máximo y un mínimo.

```
static final int maxEnergia = 100;
static final int maxVision = 10;
static final int maxVelocidad = 10;
```

```
static final int  maxEdad = 30;

static final int minEnergia = 0;

static final int  minVision = 1;

static final int  minVelocidad = 1;

static final int  minEdad = 0;

//Métodos abstractos
```

Estos métodos abstractos permiten la obtención de los atributos de los organismos dependiendo de su tipo se podrá conocer sus atributos energía, vision, velocidad o edad para los organismos de tipo Jugador y Microorganismos NPC y también los atributos energía, visión y velocidad que aporta los organismos de tipo Alimento (dependiendo de su tamaño), también se podrá obtener la posición actual del organismo.

Se encuentra también como método abstracto el de consumir alimento, este es abstracto ya que el jugador como él microorganismos pueden consumir alimentos, es por esta razón que se hereda, el de mover ya que los organismo se pueden mover de casilla dependiendo de sus necesidades.

```
public abstract void setEnergia(int energía);

public abstract void setVision(int vision);

public abstract void setEdad(int edad);

public abstract int getMaxEnergia();

public abstract int getMaxVision();

public abstract int getMaxVelocidad();

public abstract int getMaxEdad();

public abstract int getMinEnergia();

public abstract int getMinVision();

public abstract int getMinVelocidad();

public abstract int getMinEdad();
```

```
public abstract String getTipo();

public abstract int getEnergia();

public abstract int getVelocidad();

public abstract int getVision() ;

public abstract int getEdad();

public abstract int getPosX();

public abstract int getPosY();

public abstract int getBuscPosX();

public abstract int getBuscPosY();

public abstract void vaciarCordBuscar(Mapa tempMapa);

public abstract void consAlim(int enA, int viA, int veA);

public abstract void cambiarBuscarCorden(int nuevoBuscPosX, int nuevoBuscPosY);

public abstract int[] mover(Mapa temMapa);

public abstract void actualizarCoord(int tempX, int tempY);

public abstract void Imprimir();

public abstract int valEsJug();
```

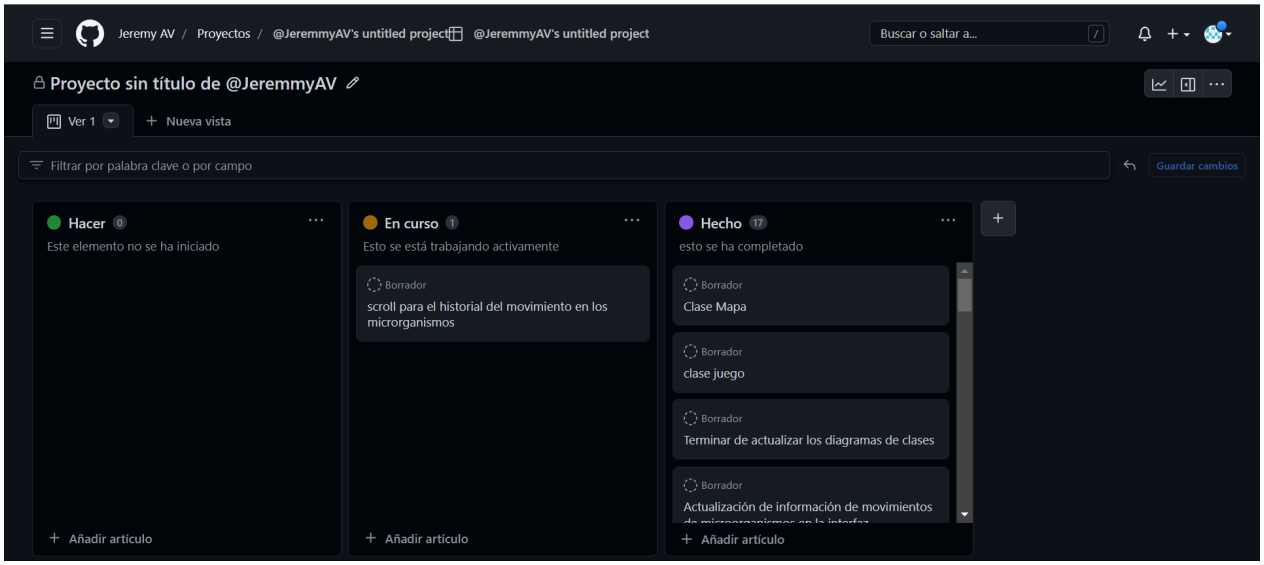
La clase Jugador, Alimento y MicroOrg extiende la clase Organismo, lo que significa que hereda todas las propiedades y métodos de Organismo. Al hacerlo, estas clases pueden acceder a los métodos y propiedades de Organismo. Además, la clase Organismo es abstracta, lo cual significa que no se puede crear instancias directamente de ella. En cambio, se deben crear instancias de una subclase concreta como Jugador, Alimento o MicroOrg depende sea el caso. Al hacerlo, se utiliza la misma interfaz que el Organismo proporciona, pero se utiliza la implementación específica de la clase Jugador, Alimento o MicroOrg, por lo que se aplica la herencia y polimorfismo.

Ventajas y desventajas de la implementación del juego

Como ventaja, se puede simplificar mucho el código para cuando una clase es muy similar a la otra, y así nos ahorra muchas líneas que no necesitamos ver, ya que se repiten en otra clase en la cual la diferencia es casi nula. Además de que si fuera necesario editarlo, solo se tendría que hacer una vez, y así recortamos horas de trabajo inutil.

La reutilización del código es la gran ventaja al utilizar la herencia, podemos crear nuevas clases que heredan las propiedades y comportamientos de las clases existentes, lo que nos permite reutilizar el código. La flexibilidad en el polimorfismo nos permite escribir código que funciona con una interfaz común, en lugar de tener que escribir código específico para cada objeto. Esto nos da más flexibilidad en el diseño de nuestro juego. Lo que observamos desventajoso en la implementación de la herencia no es solo la complejidad del código sino la sobrecarga, es decir, el uso excesivo de la herencia y el polimorfismo puede resultar en una sobrecarga de clases y objetos.

Uso del board de GitHub



Referencias

XxThe Genius XD.(2011, enero). Herencia y polimorfismo en java NetBeans 6.9 [Video].Youtube.*https://www.youtube.com/watch?v=CoOgbO3JZwM*

Santiago Vanegas. (2014). juego de damas [Repositorio de software]. GitHub. *https://github.com/svanegas/checkers.git*