Instituto Tecnológico de Costa Rica
Escuela De Ingenieria En Computación

Programacion Orientada A Objetos

Herencia y polimorfismo

Alumno: Manuel Alejandro Rodríguez Murillo, (2021028686)
Profesor: Yuen Cheong Law Wan

22 de abril de 2022

Índice

1. Solución

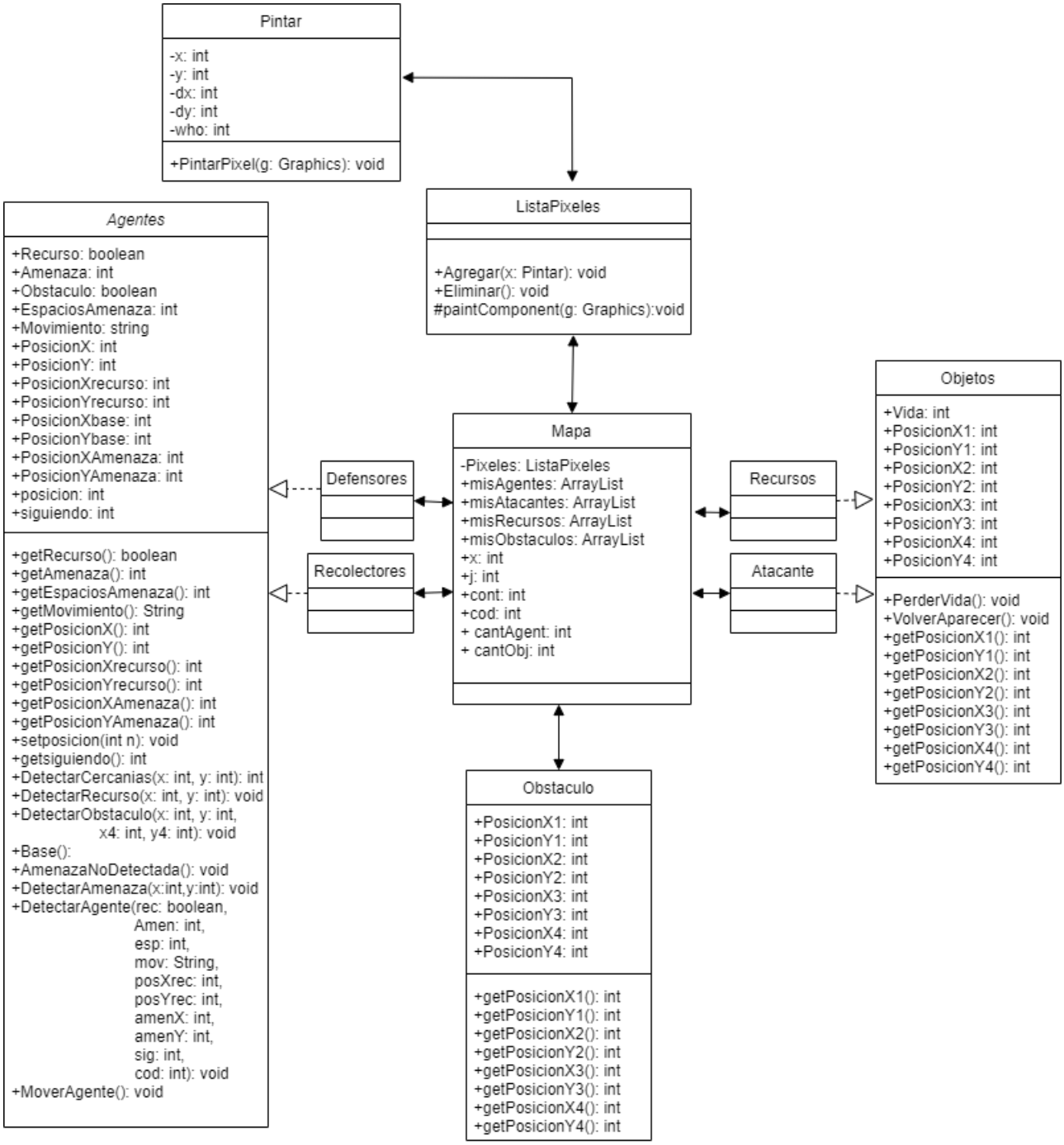
El proyecto se puede dividir en tres bloques, los objetos principales, la interacción entre ellos y la interfaz utilizada. En primer lugar, existen dos súper clases que son la Agentes y Objetos, de la clase Agentes heredan Defensores y Recolectores, en clase Objetos heredan los Atacantes y los Recursos. Aunque poseen la misma cantidad de atributos y métodos cada uno actúa según la manera especificada, por ejemplo, la forma en que actúa el método DetectarAmenaza es diferente para los Defensores que para los Recolectores. En el caso de los Obstáculos estos poseen su propia clase porque estos no necesitan los atributos y métodos relacionados a la vida.

Para el funcionamiento general de los objetos, es por medio de coordenadas. Estos verifican si posen un objeto (Agente, Recurso, Atacante, Obstáculo) en el siguiente espacio a ellos, restando sus coordenadas con las de el por lo que si en el eje x o y da como resultado 1 o -1 quiere decir que esta junto a él (el otro eje debe dar 1,-1 o 0 obligatoriamente, sino quiere decir que está en la fila o columna a la par del objeto mas no está a la par del mismo), una vez se verifica que están a la par se procede con los métodos necesarios dependiendo de los objetos que sean.

Para el funcionamiento general se crearon listas de objetos, una por cada tipo de objeto, excepción de los Defensores y Recolectores que comparten lista. Se utilizan cinco ciclos principales para las interacciones, uno para el movimiento de los agentes y los otros cuatro, es uno por tipo de objeto (Agentes-Agentes, Agentes-Atacante, Agentes-Recurso, Agentes-Obstáculo) de esta manera se recorren todas las listas y se verifica las posiciones de cada objeto.

El apartado de la interfaz funciona mediante Graphics. Hay dos funciones principales en este apartado ListaPixeles y pintar, ListaPixeles es crea una lista de objetos Pintar, puede eliminar a los elementos y una vez se agrega un elemento esta llama al método para imprimir en pantalla, por otro lado, Pintar almacena la posición del objeto mediante coordenadas, el tamaño deseado y dependiendo del tipo de objeto el color. Entonces para el funcionamiento de esta interfaz conforme se crean los elementos se agregan a la lista empezando a mostrarla en pantalla y una vez se vuelve a presionar el botón se eliminan todos los objetos creados anteriormente y se vuelven a crear con la nueva posición. Además, para la generación del tablero se utiliza la misma lógica solo que se dejan los cuadros con solo los bordes pintados por lo que deja la imagen se tablero.

2. Diagrama de clases



3. Clases

Todos los métodos get... regresan el valor del mismo nombre que el método, en el caso del set asigna el valor recibido al atributo del mismo nombre que el método.

3.1. Agentes

Recurso: Significa si el lleva o no lleva recurso.
Amenaza: Int de tres estados en caso de amenaza, 1 no hay amenaza, 2 huyendo de amenaza y 3 atacando amenaza.
Obstaculo: Indica si se encontró con obstáculo (orden de movimiento).
EspaciosAmenaza: Indica cuantos espacios debe alejarse un recolector de una Amenaza.
Movimiento: Próxima dirección a desplazarse.
PosicionX: Posición en el eje x.
PosicionY: Posición en el eje y.
PosicionXrecurso y PosicionYrecurso: En caso de encontrar recurso registra su posición para parsársela al resto de agentes.
PosicionXbase y PosicionYbase: Posición de la base
PosicionXAmenaza y PosicionYAmenaza: En caso de encontrar amenaza registra su posición para parsársela algún defensor si se lo encuentra.
posicion: Su posicón en la lista de Agentes.
siguiendo: En caso de seguir un agente se guarda su posición.

3.1.1. Defensores

DetectarCercanias(): Resta la posición x,y recibida con la del agente si da 1 o -1 y la resta del otro eje da 1, -1 o 0 quiere decir que están a la par por lo que se regresa la posición en la lista del agente, en caso de no ser así se regresa -1.

DetectarRecurso(): Coloca Recurso como true y registra la posición del recurso.

DetectarObstaculo(): Una vez verificado que están a la par, se verifica en qué lado del obstáculo esta y dependiendo de la posición se la asigna un movimiento a realizar.

Base(): Restablece los atributos a cómo eran originalmente.

AmenazaNoDetectada(): Coloca el atributo amenaza en 1 y reinicia los EspaciosAmenaza.

DetectarAmenaza(): Establece Amenaza en 3 (Atacando) y registra la posición de la Amenaza.

DetectarAgente(): Primero se verifica si el otro agente trae recurso si es el caso se verifica si él también trae recurso si los dos trae lo empezara a seguir sino registrara la ubicación de donde tomó el recurso. Luego verifica Amenazas si es 2 coloca la amenaza en 3 copia la dirección de la amenaza y se mueve en dirección contraria al agente. Si es 3 copia este estado, la posición de la amenaza y la dirección

MoverAgente(): En caso de obstáculo este tiene prioridad absoluta, luego se verifica amenaza y se mueve en dirección a las coordenadas de la amenaza, en caso de tener un recurso se van a empezar a restar las coordenadas para llegar a la base (0,0), en caso de saber la ubicación de un recurso se van a empezar a restar o sumar las coordenadas hasta llegar a este y por último se efectúa el movimiento registrado y se registra el próximo movimiento (esto es especialmente útil para cuando nos sigue un agente).

3.1.2. Recolectores

DetectarCercanias(): Resta la posición x,y recibida con la del agente si da 1 o -1 y la resta del otro eje da 1, -1 o 0 quiere decir que están a la par por lo que se regresa la posición en la lista del agente, en caso de no ser así se regresa -1.

DetectarRecurso(): Coloca Recurso como true y registra la posición del recurso.
DetectarObstaculo(): Una vez verificado que están a la par, se verifica en qué lado del obstáculo esta y dependiendo de la posición se la asigna un movimiento a realizar.

Base(): Restablece los atributos a cómo eran originalmente.

AmenazaNoDetectada(): Coloca el atributo amenaza en 1 y reinicia los EspaciosAmenaza.

DetectarAmenaza(): Establece Amenaza en 2 (Huyendo) y registra la posición de la Amenaza.

DetectarAgente(): Primero se verifica si el otro agente trae recurso si es el caso se verifica si él también trae recurso si los dos trae lo empezara a seguir sino registrara la ubicación de donde tomó el recurso. Luego verifica Amenazas si es 2 copia este estado los espacios que ya se alejó la posición de la amenaza y la dirección, en caso de ser 3 coloca la amenaza en 2 copia la dirección de la amenaza y se mueve en dirección contraria al agente.

MoverAgente(): En caso de obstáculo este tiene prioridad absoluta, luego se verifica amenaza y se mueve en dirección contraria a las coordenadas de la amenaza, en caso de tener un recurso se van a empezar a restar las coordenadas para llegar a la base (0,0), en caso de saber la ubicación de un recurso se van a empezar a restar o sumar las coordenadas hasta llegar a este y por último se efectúa el movimiento registrado y se registra el próximo movimiento (esto es especialmente útil para cuando nos sigue un agente).

3.2. Objetos

Vida: Indica la cantidad de veces en las que se puede interactuar con él.

Como los Objetos miden 4x4 debe tener cuatro coordenadas distintas para su correcta interacción.

PosicionX1 y PosicionY1: Registran el cuadro de la izquierda superior.
PosicionX2 y PosicionY2: Registran el cuadro de la derecha superior.
PosicionX3 y PosicionY3: Registran el cuadro de la izquierda inferior.
PosicionX4 y PosicionY4: Registran el cuadro de la derecha inferior.

3.2.1. Atacante y Recursos

PerderVida(): Reduce en 1 el atributo vida y llama al método VolverAparecer().

VolverAparecer(): Verifica si el atributo vida es igual a cero, en caso de que lo sea lo restablece en 10 y genera una nueva posición para los objetos

3.3. Obstaculos

PosicionX1 y PosicionY1: Registran el cuadro de la izquierda superior.
PosicionX2 y PosicionY2: Registran el cuadro de la derecha superior.
PosicionX3 y PosicionY3: Registran el cuadro de la izquierda inferior.
PosicionX4 y PosicionY4: Registran el cuadro de la derecha inferior.

3.4. Mapa

Pixeles: Objeto ListaPixeles
misAgentes, misAtacantes, misRecursos y misObstaculos: Son listas de los objetos.
x,j: Son variables utilizadas en ciclos
cont: Verifica que una amenaza no sea Atacada dos o más veces por el mismo agente en el mismo turno.
cod: Registra la posición del agente o objeto en caso de que esten a la par de otro objeto o agente.
cantAgent: Cantidad de agentes en simulación (cuando se define es la mitad de los agentes a mostrar).
cantObj: Cantidad de objetos en simulación:

El Mapa es una sola función que primero inicializa todas las listas de objetos y queda expectante a que el usuario presione el botón una vez presionado se inicializan un conjunto de ciclos que nos sirven para comparar la posición de los Agentes y los Objetos en caso de que se encuentren, con la posición registrada en cod se solicitan los datos necesarios para que el agente u objeto ponga en marcha sus métodos.

Al final se llama a ListaPixeles para que actualice las posiciones en pantalla (la eliminación de los objetos pintados se hace antes de la ejecución de los for).

3.5. ListaPixeles

Agregar(): Agrega un nuevo elemento Pintar a la lista.
Eliminar(): Elimina el último elemento agregado.
paintComponent(): Llama a PintarPixel para imprimir el cubo

3.5.1. Pintar

x: Coordenada x.
y: Coordenada y.
dx y dy: Registran las dimensione de los cuadros.
who: Para saber si se imprime solo bordes o bloque entero y que color.
PintarPixel(): Imprime el cuadro en pantalla.

4. Herencia y polimorfismo

4.1. Herencia

La herencia se aplica en dos ocasiones, de la clase Agentes a las clases Defensores y Recolectores, y de la clase Objetos a las clases Atacante y Recursos.

4.1.1. Ventajas

Rápida creación de más clases.
Permitió llevar un mejor control de los atributos y métodos a utilizar evitan problemas a la hora de interactuar entre ellos. Se pueden encapsular todos los objetos heredados en una misma lista.

4.1.2. Desventaja

Conceptos complicados de entender y aplicar en un inicio.

4.2. Polimorfismo

El polimorfismo esta aplicado en varios métodos de los agentes, por ejemplo, DetectarAgente mientras el Recolector coloca sus atributos para huir del Atacante, el Defensor los coloca para buscarlo. Lo mismo ocurre en MoverAgente, mientras el Recolector se aleja de la amenaza, el Defensor se acerca a esta.

4.2.1. Ventaja

Facilidad a la hora de trabajar con varias clases heredadas.

4.2.2. Desventaja

Puede llegar a ser más complicado saber de dónde provienen errores por los múltiples métodos modificados.

5. Github

https://github.com/Manuel2710/Proyecto-POO-Enjambre.git.

6. Referencias

Carvajal, J. [Josue Carvajal]. (2016, 19 de julio). Lista enlazada simple con objetos - Insertar y Mostrar JAVA. [Archivo de video]. YouTube. https://www.youtube.com/watch?v=I8NBaLtD95kt=1210sab$_c$*hanne* *JosueCarvajal*

Ballesteros, C. [cristian ballesteros]. (2017, 06 de noviembre). Como List y ArrayList de Objetos en Java PARTE 1/2. [Archivo de video]. YouTube. https://www.youtube.com/watch?v=D4$_s$5$VEdGOUt =$ $294sab_channel = cristianballesteros$

Ballesteros, C. [cristian ballesteros]. (2017, 06 de noviembre). Como List y ArrayList de Objetos en Java PARTE 2/2. [Archivo de video]. YouTube. https://www.youtube.com/watch?v=0$_K$$oNJLgFJEt =$ $244sab_channel = cristianballesteros$

Hamerski, E. [Eduardo Hamerski]. Java - Swing/Repaint. [Archivo de video]. YouTube. https://www.youtube.co *EHamerski*