Manual Técnico

* Proyecto: Maven 4.0.0
* Version: 1.0 – SnapShot
* Packaging: Jar
* Aplicación: Apache Netbeans 12.6
* Codigo: Java
* JDK 15

Algoritmo para declarar las variables que tendrá nuestras casillas

1. Saber que variables vamos a tener en nuestro tablero
2. Las casillas cambian conforme el juego vaya avanzando
3. Tiene que haber una casilla que tenga al jugador, que tenga oro, que tenga al bot, que sea salida y la misma casilla de oro tendrá una cantidad de oro
4. Declarar como booleanas las casillas inicializadas con false para que cuando sea el momento se vuelvan true

Algoritmo para saber cuantas casillas vamos a tener en nuestro tablero

1. Hacer una matriz con columnas y filas
2. Declarar las variables de filas y columnas del bot y jugador
3. Declarar variables enteras donde esté el jugador y donde esté el bot para poder moverlas restándoles o sumándoles
4. Hacer un ciclo for con variables enteras para construir el tablero

Algoritmo para darle valores al tablero

1. Hacer un ciclo for con 2 variables que serian columnas y filas para darle valores al tablero
2. Iniciamos el tablero con todas las casillas vacias
3. Vamos llenando cada fila y columna con paredes, oro y salidas

Algoritmo para colocar al jugador y bot en una casilla vacia aleatoria

1. Declaramos variables para poner al jugador y al bot
2. Estas variables las igualamos a un random
3. Declaramos variables de tipo booleanas para saber si la casilla estará vacia o no
4. Si la casilla random no es vacia, hacer un ciclo en donde vamos a salir hasta que la casilla esté vacia
5. Igualamos la casilla para poner al jugador, con la casilla donde está el jugador

Algoritmo para imprimir el tablero

1. Retornamos los valores de la matriz tablero

Algoritmo para imprimir los caracteres de la matriz

1. Con los valores del tablero iniciamos un ciclo for
2. Si el tablero donde antes declaramos si era pared, salida, oro, tiene al jugador, tiene al bot o era vacia es verdadera, le colocamos el caracter

Algoritmo para el comando mirar

1. Declaramos variables iniciales y finales de columnas y filas para imprimir la vista
2. A las variables iniciales para imprimir las igualamos con las casillas donde esta el jugador y le restamos 2
3. A las variables finales para imprimir las igualamos con las casilla donde esta el jugador y le sumamos 2
4. Verificamos que no se imprima si el valor es mayor o menos al numero de filas o columnas
5. Si es mayor o menor el valor a los valores de filas y columnas, los igualamos a 0 a 29.
6. Finalmente usamos el ciclo for del tablero para imprimir la vista

Algoritmo para colocar el oro que nos quitan en una casilla vacía

1. Declaramos una variable que tendrá el oro a colocar
2. Declaramos variables de filas y columnas para colocar el oro
3. Igualamos las variables a filas o columnas random del tablero
4. Si la casilla es vacia colocamos la variable del oro allí
5. Si no, entramos a un ciclo hasta que la casilla sea vacia
6. Definimos que la casilla c y la casilla f en el tablero ahora tendrán oro

Algoritmo para el menú del juego

1. Declaramos una variable para que aunque una partida termine, el menú se vuelva a repetir
2. Entramos a un ciclo con esa variable true porque siempre se va a repetir el menú
3. Declaramos variables que vamos a necesitar en el juego y en el reporte final
4. Imprimimos nuestro menú
5. Iniciamos sentencias para cada opción

Algoritmo para jugar y movernos

1. Declaramos una variable mientras el juego este en proceso que solo será falsa si la partida acaba
2. Para hacer una jugada, declaramos una variable que almacenará los comandos
3. Para movernos verificamos si el jugador a la casilla que se quiere mover es mayor o menos al numero de columnas y filas
4. Verificamos que no sea una pared
5. Si no es pared y es menor o mayor al numero de columnas, nos podemos mover
6. Nos movemos restándole o sumándole 1 a la fila o columna donde está el jugador
7. Declaramos que ahora la casilla donde estaba el jugador es vacia y que ahora la casilla donde va a estar es verdadera

Algoritmo para levantar el oro

1. Verificamos que el jugador este en una casilla de oro
2. Declaramos una variable que almacenará el oro del jugador
3. Esta variable la igualamos a la cantidad de oro de la casilla para saber cuanto oro levantamos
4. Imprimimos cuanto oro obtuvimos y con la variable que declaramos al principio en el menú le sumamos esa cantidad para saber cuanto oro tenemos en total
5. Si no había oro en esa casilla pues imprimimos que no había oro

Algoritmo para ver cuanto oro necesitamos

1. Imprimimos el oro que se necesita que fue establecido en el menú

Algoritmo para comando oro

1. Imprimimos el oro que tenemos en total con las variables anteriormente descritas

Algoritmo para comando salir

1. Verificamos si está en una casilla de salida
2. Si el oro obtenido es igual al oro requerido puede salir
3. Si sale, se le mostrarán mensajes indicando que salió y los reportes finales
4. Si no tiene el oro requerido, se le indicará con un mensaje que no tiene el oro suficiente para salir

Algoritmo por si supera el limite de comandos inexistentes ingresados

1. Declaramos una variable que se igualará a 0 cada vez que ingresen un comando existente
2. Si ingresan un comando inexistente, entonces se incrementará a 1
3. Si el contador de comando inexistente llega a 3, entonces la partida acabará
4. Se le mostrará un mensaje indicando que perdió por los comandos inexistentes superados
5. Se le mostrarán los reportes finales

Algoritmo para quitar oro por comando inexistente

1. Si el juego detecta un comando inexistente y el oro obtenido es mayor a 0, el juego le restará 1 al oro obtenido
2. Llamaremos al método que colocará ese oro en la casilla vacia aleatoria que anteriormente definimos

Algoritmo para el bot

1. Declaramos mover el bot como un entero
2. Este entero lo igualamos a random.nextInt(4), para que hayan 4 movimientos en posibles
3. Si el movimiento es igual a 0,1,2,3, entonces indicaremos un movimientos
4. Por ejemplo si el movimiento es igual a 0, entonces lo moveremos al norte
5. Definimos que las columnas y filas donde está el bot es mayor o menor al numero de filas y columnas del tablero
6. Si es mayor o menor se moverá hacia donde el numero del movimiento le indique
7. Definimos que la fila o columna donde estaba el bot ahora es vacia
8. Definimos que la fila o columna donde esta el bot ahora está ocupada por el
9. Si el bot esta en la misma casilla y fila que el jugador, entonces el jugador pierde y se le muestra un mensaje indicándolo y mostrándole los reportes

Algoritmo para los reportes Generales

1. Declaramos en el public main de nuestro programa que habran variables que acumularan los datos de todo el juego
2. A estas variables se le sumarán los movimientos o datos que estén sucediendo en el juego
3. Para cada variable que el juego tenga que imprimir, habrá otra variable adentro del juego que estará contando las estadísticas
4. Estas variables dentro del juego se le sumaran a las variables acumuladoras de todo el juego
5. Se imprimen los mensajes con los reportes y estadísticas del juego
6. Si no se ha jugado al menos 1 partida, entonces el apartado de reportes mostrará un mensaje que no se pueden mostrar los reportes