2022

PAC-MAN

PRACTICA#1

MANUAL TECNICO

ANDREA CHAVEZ

202109641

INTRODUCCIÓN

El manual contiene las explicaciones sobre el funcionamiento técnico del programa así como los aspectos y requerimientos con los que se deben contar para poder crear una interacción con el programa ejecutable o ya sea tomarlo como una guía para los demás desarrolladores.

OBJETIVOS

- Crear un juego ejecutable en consola sobre el famoso juego "Pac-Man" utilizando arreglos matriciales, ciclos, bucles, clases, entre muchas otras funciones.
- El fácil manejo del programa para el usuario que lo desee ejecutar.
- Asi mismo, presentar un código de fácil comprensión para demás desarrolladores.

SOFTWARE

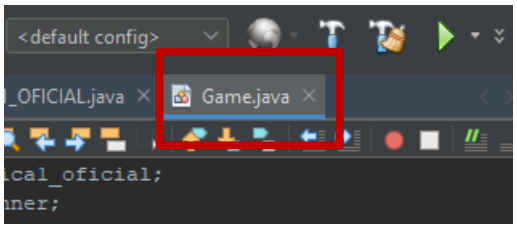
- Navegador
- Memoria mínima 4gb
- Desarrollado en lenguaje de programación JAVA.
- NetBeands Aplication, entorno de desarrollo libre.

LIBRERÍAS

- **Scanner**
  Se importa para obtener la entrada de los tipos de datos primitivos.
- **Random**
  Genera datos aleatorios para varias distribuciones.

CLASES

Clases utilizada en el proyecto: GAME.JAVA



Este genera el tablero, posiciona los personajes y les da movimiento, como también definen los atributos públicos, el tablero de juego e items a posicionar.

MENU

Utilizando la sentencia switch en función de la variable OPCION. Dependiendo la opción que el usuario ingrese, esta se guardara en la variable antes mencionada y ejecutará lo indicado en el case que corresponda al valor.

```
switch(opcion){
    case 1:
        int TABLERO;
        int JUEGO;
        break;
    case 2:
        int HISTORIAL;
        break;
    case 3:
        int SALIR;
        break;}
```

JUGAR

Ejecutar case 1

Información principal

Este pedirá información principal del usuario como su nombre y edad.

Tablero

Al pedir las dimensiones del tablero al usuario, con ayuda de un ciclo while se restringe al rango de entrada de filas y columnas (mayor o igual a 8x8).

Juego

Con ayuda de la clase GAME se definen los atributos de filas/columnas, ítems, movimientos, punteo y coordenadas del jugador.

MÉTODOS

GAME:

```
public Game(int rowsY, int columnsX){
```

Llamado método constructor, el cual asigna el numero de filas y columnas.

GENERATEBOARD:

```
public String[][] generateBoard() {
```

Generador del tablero, con if indicara la fila en donde iniciara el túnel. Declarando la matriz y guardando la información de las variables ROWS y COLUMS, indicar dentro de un for que las filas/columnas deben iniciar en la posición 0, menor a filas/columnas e indicarle que debe aumentar (++), esto para recorrer y asignar strings a la coordenada de la matriz.

Con una condición if, se ha de indicar que la variable de filas antes asignada debe ser igual a 0 o igual al numero de columnas -1, si esto se cumple deberá imprimir "*", sino imprimirá un espacio en blanco. De esta forma se generará los márgenes del tablero a utilizar.

Con un nuevo objeto Random, se generarán las paredes internas del tablero, utilizando una condición for, un ciclo para que se ejecute 5 veces, esto lo hará 3 veces para las paredes verticales y 2 para las parades horizontales, retornando la matriz.

VALIDATECOORDANDRANDOMAADDSYMBOL:

```
public void validateCoordAndRandomAAddSymbol(String[][] array, int rowY, int columnX, String symbol) {
```

Función que validará la posición y si la coordenada esta libre, insertará el símbolo indicado. Si no cumple esta función, con ayuda de un while, seguirá recorriendo la matriz y obtendrá una nueva coordenada aleatoria.

INITBOARD:

```
public void initBoard(){
```

Al pedir las coordenadas aleatorias del jugador con un nuevo objeto de random. Al obtenerlos, se llama al método validateCoordAndRandomAAddSymbol y se indica las coordenadas de la posición aleatoria del jugador y se asigna el símbolo que queremos imprimir. Con el vector ITEMS, hacemos un for que indique si el atributo ITEMS es iguala 0, menor igual a 2, incrementa. Asignándole un random iniciándola en la posición 0, se indica las coordenadas y el atributo ítems e imprimirá ese tablero.

COLLISIONS:

```
public int[] collisions(String[][] array, int rowY, int columnX){
```

Este método interactuar con las paredes internas del tablero, las cuales indicaran que si al recorrer cada coordenada de la matriz se encuentra con un "*" este retornará el movimiento y punteo dependiendo el símbolo, así como se muestra a continuación.

```
int ret[] = {1, 0};
```

MOVIMIENTO PUNTAJE

REGENERATEITEM:

```
public void regenerateItem(String[][] array, String symbol){
```

Utilizada para regenerar los ítems en una nueva posición aleatoria. Método para mostrar e imprimir la matriz generado con ítems y personaje aleatorio.

COLORBOARD:

```
public String colorBoard(String item) {
```

Utilizada para colorear el tablero y cada uno de los ítems, utilizando un switch.

SHOWBOARD:

```
public void showBoard(String[][] array) {
```

Método para mostrar e imprimir la matriz generado con ítems y personaje aleatorio.

MOVETO:

```
public void moveTo(char direction){
```

Método en donde decidirá en que dirección se moverá el personaje. Si DIRECTION es igual a "W" este disminuirá su posición y aumentara en los movimientos. Se realiza el mismo análisis con "A", "S" y" D".

RUN:

```
public void run() {
```

Método utilizado para ejecutar el juego integrando todo lo antes hecho se ejecutara el juego.

LLAMANDO LA CLASE GAME AL PROYECTO.

```
Game game = new Game(filas, columnas);
game.run();
```

HISTORIAL

Ejecutando el case 2.

Con String GAMEINFO y los métodos de la clase GAME recolectamos la información e imprimimos.

SALIR

Ejecutando el case 3.

Ejecutará esto gracias a un boolean y terminará el juego e imprimiendo un mensaje en pantalla donde indicara que a finalizado.

RECOMENDACIONES

Es recomendable utilizar el software con las especificaciones mínimas y tener un lugar en donde poder plasmar las ideas y de esta forma tener más claro lo que se quiere plantear en código.