**Universidad ORT Uruguay**

**Facultad de Ingeniería**

**Bernard Wand Polak**

**Programación II**

**Obligatorio 1 - Grupo N2B**





Rodrigo Camps N° 241344 Rodrigo Pistón N°261777

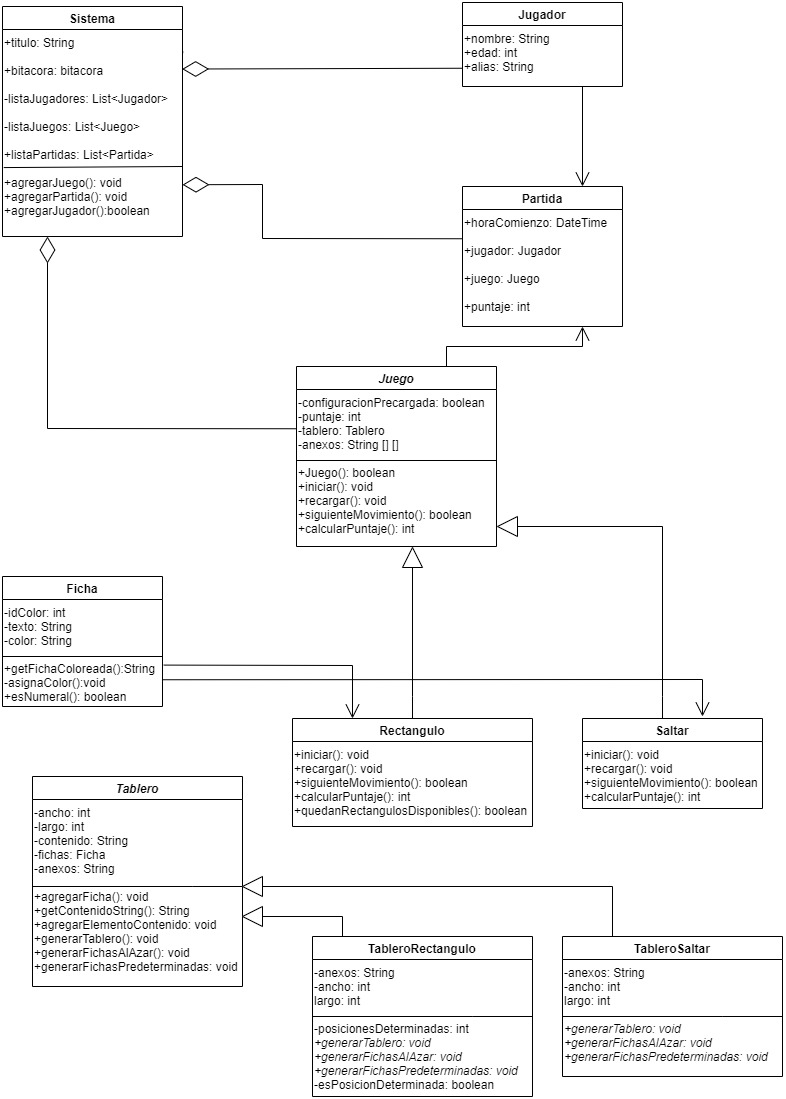
Octubre 2021

Diagrama de Clases …………………………………………………… 3

Dominio ………….…………………………………………………….... 4

Interfaz …………………………………………………………………... 30

Utilidad …………………………………………………………………… 42

Diagrama de Clases

Dominio

package dominio;

/\*\*

\*

\* @author Rodrigo Pistón(261777) | Rodrigo Camps(241344)

\*/

public class Ficha {

private int idColor;

private String texto;

private String color;

public Ficha(int id, String nombre) {

this.idColor = id;

this.texto = nombre;

this.asignaColor();

}

public String getNombre() {

return texto;

}

public String getFichaColoreada() {

return color + texto + "\u001B[0m";

}

public int getIdColor() {

return idColor;

}

public void setIdColor(int idColor) {

this.idColor = idColor;

}

@Override

public boolean equals(Object obj) {

return this.getNombre().equalsIgnoreCase(((Ficha)obj).getNombre()) &&this.getIdColor()==((Ficha)obj).getIdColor();

}

private void asignaColor() {

this.color = utilidad.Generico.COLORES[idColor];

}

public boolean esNumeral() {

return this.texto.equals("#");

}

}

package dominio;

import java.time.LocalDateTime;

/\*\*

\*

\* @author Rodrigo Pistón(261777) | Rodrigo Camps(241344)

\*/

public abstract class Juego {

private boolean configuracionPrecargada;

private int puntaje;

private Tablero tablero;

private String[][] anexos;

public Juego(boolean configuracionPrecargada) {

this.setConfiguracionPrecargada(configuracionPrecargada);

}

public int getPuntaje() {

return this.puntaje;

}

public Tablero getTablero() {

return tablero;

}

public boolean getConfiguracionPredeterminada() {

return this.configuracionPrecargada;

}

public void setPuntaje(int puntaje) {

this.puntaje = puntaje;

}

public void setConfiguracionPrecargada (boolean configuracionPrecargada) {

this.configuracionPrecargada = configuracionPrecargada;

}

public void setTablero(Tablero tablero) {

this.tablero = tablero;

}

public abstract void iniciar();

public abstract void recargar();

public abstract int calcularPuntaje();

}

package dominio;

/\*\*

\*

\* @author Rodrigo Pistón(261777) | Rodrigo Camps(241344)

\*/

public class Jugador {

private String nombre;

private int edad;

private String alias;

public Jugador(String nombre, int edad, String alias) {

this.setNombre(nombre);

this.setEdad(edad);

this.setAlias(alias);

}

public String getNombre() {

return this.nombre;

}

public String getAlias() {

return this.alias;

}

public int getEdad() {

return this.edad;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public void setAlias(String alias) {

this.alias = alias;

}

public void setEdad(int edad) {

this.edad = edad;

}

@Override

public boolean equals(Object obj){

return this.getAlias().equalsIgnoreCase(((Jugador) obj).getAlias());

}

}

package dominio;

import java.time.LocalDateTime;

import java.util.Comparator;

/\*\*

\*

\* @author Rodrigo Pistón(261777) | Rodrigo Camps(241344)

\*/

public class Partida{

private LocalDateTime horaComienzo;

private Jugador jugador;

private Juego juego;

private int puntaje;

public Partida(Jugador jugador, Juego juego,int puntaje) {

this.setHoraComienzo(LocalDateTime.now());

this.setJuego(juego);

this.setJugador(jugador);

this.setPuntaje(puntaje);

}

public void setPuntaje(int puntaje) {

this.puntaje = puntaje;

}

public int getPuntaje() {

return puntaje;

}

public Juego getJuego() {

return juego;

}

public Jugador getJugador() {

return jugador;

}

public LocalDateTime getHoraComienzo() {

return horaComienzo;

}

private void setHoraComienzo(LocalDateTime horaComienzo) {

this.horaComienzo = horaComienzo;

}

private void setJuego(Juego juego) {

this.juego = juego;

}

private void setJugador(Jugador jugador) {

this.jugador = jugador;

}

}

package dominio;

/\*\*

\*

\* @author Rodrigo Pistón(261777) | Rodrigo Camps(241344)

\*/

public class Rectangulo extends Juego {

public Rectangulo(boolean configuracionPredeterminada) {

super(configuracionPredeterminada);

this.setTablero(new TableroRectangulo());

}

public void iniciar(){

if(!this.getConfiguracionPredeterminada()){

this.getTablero().generarFichasAlAzar();

}

recargar();

}

public void recargar(){

this.getTablero().generarTablero();

}

public boolean siguienteMovimiento(Ficha ficha,int coordenadaXInicial, int coordenadaYInicial, int rectanguloAlto, int rectanguloAncho, boolean primeraJugada) {

boolean movimientoPermitido = true;

boolean encontroAdyacente = false;

Ficha fGuion = new Ficha(0,"-");

// -- Si no se le resta 1 estaría desfasado en el array

coordenadaXInicial -= 1;

coordenadaYInicial -= 1;

int coordenadaXFinal = (coordenadaXInicial + rectanguloAlto);

int coordenadaYFinal = (coordenadaYInicial + rectanguloAncho);

Ficha[][] tableroAuxiliar = new Ficha [this.getTablero().getAncho()][this.getTablero().getLargo()];

utilidad.Generico.CopiarMatriz(this.getTablero().getFichas(), tableroAuxiliar);

// -- Si las coordenadas estan dentro del tamaño del tablero

if(coordenadaXInicial >= 0 && coordenadaXInicial < tableroAuxiliar[0].length && coordenadaYInicial >= 0 && coordenadaYInicial < tableroAuxiliar.length){

// -- Se recorre el tablero solo en las posiciones del rectángulo calculado para ahorrar iteraciones innecesarias

for(int fila = coordenadaXInicial ; fila < coordenadaXFinal && movimientoPermitido; fila++) {

for (int columna = coordenadaYInicial; columna < coordenadaYFinal && movimientoPermitido; columna++) {

// -- Si es igual a un \* o a un numeral se termina, porque estaría pisando un rectángulo

if(!tableroAuxiliar[fila][columna].equals(fGuion)){

movimientoPermitido = false;

}else{

tableroAuxiliar[fila][columna] = ficha;

}

// -- Se controlan los bordes exteriores, sin pasarse del límite del rectangulo,

// Si esta en el límite || es un rectangulo de 1 de ancho || 1 de alto && no se sobrepasa del límite de la matríz && es númeral

// -- Borde Superior

if((fila == coordenadaXInicial || rectanguloAncho == 1) && (fila - 1) >= 0 && tableroAuxiliar[fila -1][columna].esNumeral()){

encontroAdyacente = true;

// -- Borde Inferior

}else if((fila == coordenadaXFinal || rectanguloAncho == 1) && (fila + 1) < tableroAuxiliar.length && tableroAuxiliar[fila + 1][columna].esNumeral()){

encontroAdyacente = true;

// -- Borde Izquierdo

}else if((columna == coordenadaYInicial || rectanguloAlto == 1) && (columna - 1) >= 0 && tableroAuxiliar[fila][columna - 1].esNumeral()){

encontroAdyacente = true;

// -- Borde Derecho

}else if((columna == coordenadaYFinal || rectanguloAlto == 1) && (columna + 1) < tableroAuxiliar[0].length && tableroAuxiliar[fila][columna + 1].esNumeral()){

encontroAdyacente = true;

}

}

}

}else{

movimientoPermitido = false;

}

movimientoPermitido = movimientoPermitido && encontroAdyacente || primeraJugada);

if(movimientoPermitido){

utilidad.Generico.CopiarMatriz(tableroAuxiliar, this.getTablero().getFichas());

}

return movimientoPermitido;

}

public int calcularPuntaje(){

int puntaje = 0;

for (int fila = 0; fila < this.getTablero().getFichas().length; fila++) {

for (int columna = 0; columna < this.getTablero().getFichas()[0].length; columna++) {

if(this.getTablero().getFichas()[fila][columna].esNumeral()){

puntaje ++ ;

}

}

}

return puntaje;

};

public boolean quedanRectangulosDisponibles(){

boolean disponible = false;

for (int fila = 0; fila < this.getTablero().getFichas().length; fila++) {

for (int columna = 0; columna < this.getTablero().getFichas()[0].length && !disponible ; columna++) {

disponible= this.getTablero().getFichas() [fila] [columna].esNumeral();

}

}

return disponible;

}}

package dominio;

import java.time.LocalDateTime;

/\*\*

\*

\* @author Rodrigo Pistón(261777) | Rodrigo Camps(241344)

\*/

public class Saltar extends Juego{

public Saltar(boolean configuracionPredeterminada) {

super(configuracionPredeterminada);

this.setTablero(new TableroSaltar());

}

public void iniciar(){

if(!this.getConfiguracionPredeterminada()){

this.getTablero().generarFichasAlAzar();

}

this.recargar();

}

public void recargar(){

this.getTablero().generarTablero();

}

public boolean siguienteMovimiento(Ficha fichaActual,int colSeleccionada){

boolean movimientoPermitido = false;

Ficha fichaVacia = new Ficha(0," ");

Ficha [][] tableroFichas = this.getTablero().getFichas();

if(colSeleccionada >= 0 && colSeleccionada <= 3 ){

int filaOrigen = 0;

int cantidadSaltos = 0;

int filaDestino = 0;

for (int fila = tableroFichas.length -1; fila > 0 ; fila--){

// -- Si se encuentra el color en la columna, se verifica cuantas veces hay que saltar

if(tableroFichas[fila][colSeleccionada].equals(fichaActual)){

filaOrigen = fila;

for (int columna = 0; columna < tableroFichas[0].length; columna++) {

if(!tableroFichas[fila][columna].equals(fichaVacia)){

cantidadSaltos ++;

}

}

}

}

filaDestino = filaOrigen - cantidadSaltos;

if(filaDestino > 0 && filaOrigen > 4 && tableroFichas[filaDestino][colSeleccionada].equals(fichaVacia)){

movimientoPermitido = true;

for (int columna = 0; columna < tableroFichas[filaDestino].length && movimientoPermitido; columna++) {

if(tableroFichas[filaDestino][columna] == fichaActual){

movimientoPermitido = false;

}

}

if(movimientoPermitido){

tableroFichas[filaOrigen][colSeleccionada] = fichaVacia;

tableroFichas[filaDestino][colSeleccionada] = fichaActual;

}

}else{

movimientoPermitido = false;

}

}

this.getTablero().setFichas(tableroFichas);

return movimientoPermitido;

}

public int obtenerColumnaRecomendada(Ficha ficha){

Ficha[][] matrizFichas = this.getTablero().getFichas();

int columnaRecomendada = -1;

int cantidadSaltos;

int filaDestino;

boolean valido = true;

for (int fila = 5; fila < matrizFichas.length && valido; fila++) {

for (int columna = 0; columna < matrizFichas[0].length && valido; columna++) {

cantidadSaltos = 0;

if(matrizFichas[fila][columna].equals(ficha)){

for (int columnaFicha = 0; columnaFicha < matrizFichas[fila].length; columnaFicha++) {

if(!matrizFichas[fila][columna].equals(utilidad.Generico.fichaVacia)){

cantidadSaltos ++;

}

}

filaDestino = fila - cantidadSaltos;

if(filaDestino > 0 && matrizFichas[filaDestino][columna].equals(utilidad.Generico.fichaVacia) ){

for (int columnaDeFilaDestino=0; columnaDeFilaDestino < matrizFichas[filaDestino].length && columnaRecomendada == -1; columnaDeFilaDestino++) {

if(matrizFichas[filaDestino] [columnaDeFilaDestino].equals(ficha)){

valido = false;

}

}

if(valido){

columnaRecomendada = columna;

}

}

}

}

}

return columnaRecomendada;

}

public int calcularPuntaje(){

Ficha [][] tableroFichas = this.getTablero().getFichas();

Ficha fichaVacia = new Ficha(0," ");

int puntaje = 0;

for (int fila = 0; fila < tableroFichas.length && fila <= 4; fila++) {

for (int columna = 0; columna < tableroFichas[0].length; columna++) {

if(!tableroFichas[fila][columna].equals(fichaVacia)){

switch(fila){

case 0:

puntaje += 60;

break;

case 1:

puntaje += 40;

break;

case 2:

puntaje += 30;

break;

case 3:

puntaje += 20;

break;

case 4:

puntaje += 10;

break;

}

}

}

}

return puntaje;

}

}

package dominio;

import java.util.ArrayList;

import java.util.Collections;

/\*\*

\*

\* @author Rodrigo Pistón(261777) | Rodrigo Camps(241344)

\*/

public class Sistema {

private ArrayList<Jugador> listaJugadores;

private ArrayList<Juego> listaJuegos;

private ArrayList<Partida> listaPartidas;

public Sistema() {

this.listaJuegos = new ArrayList<Juego>();

this.listaJugadores = new ArrayList<Jugador>();

this.listaPartidas = new ArrayList<Partida>();

}

public ArrayList<Juego> getListaJuegos() {

return this.listaJuegos;

}

public ArrayList<Jugador> getListaJugadores() {

return this.listaJugadores;

}

public ArrayList<Partida> getListaPartidas() {

return this.listaPartidas;

}

public void agregarJuego(Juego juego) {

this.getListaJuegos().add(juego);

}

public void agregarPartida(Partida partida) {

this.getListaPartidas().add(partida);

}

public boolean agregarJugador(Jugador jugadorNuevo) {

boolean agregar = false;

for (final Jugador jugador : this.getListaJugadores()) {

if (jugador.equals(jugadorNuevo)) {

agregar = true;

}

}

if(!agregar){

this.getListaJugadores().add(jugadorNuevo);

}

return !agregar;

}

public ArrayList<Partida> partidasOrdenadasPuntajeAsc(){

Collections.sort(this.getListaPartidas(),(partida1, partida2) -> {

return partida1.getPuntaje() - partida2.getPuntaje();

});

return this.getListaPartidas();

}

public ArrayList<Partida> partidasOrdenadasPuntajeDesc(){

Collections.sort(this.getListaPartidas(),(partida1, partida2) -> {

return partida2.getPuntaje() - partida1.getPuntaje();

});

return this.getListaPartidas();

}

public ArrayList<Partida> partidasOrdenadasAliasDesc(){

Collections.sort(this.getListaPartidas(),(partida1, partida2) -> {

return partida1.getJugador().getAlias().compareTo(partida2.getJugador().getAlias()) ;

});

return this.getListaPartidas();

}

}

package dominio;

/\*\*

\*

\* @author Rodrigo Pistón(261777) | Rodrigo Camps(241344)

\*/

public abstract class Tablero {

private int ancho;

private int largo;

private String[][] contenido;

private Ficha [][] fichas;

/\*

En el anexo se encuentra las partes laterales del tablero, como los puntos y número de columna

\*/

private String [][] anexos = new String[4][1];

public Tablero(int largo, int ancho,String[][] anexos) {

this.largo = largo;

this.ancho = ancho;

this.anexos = anexos;

this.fichas = new Ficha[largo][ancho];

}

private String[][] getContenido() {

return contenido;

}

public String[][] getAnexos() {

return anexos;

}

public Ficha[][] getFichas() {

return fichas;

}

public int getAncho() {

return ancho;

}

public int getLargo() {

return largo;

}

public void agregarFicha(Ficha ficha,int fila,int columna) {

this.fichas[fila][columna] = ficha;

}

public String getContenidoString() {

String contenido = "\n";

for (int fila = 0; fila < this.getContenido().length; fila++) {

contenido += String.join(" ", this.getContenido()[fila]) + "\n";

}

return contenido;

}

public void setContenido(String[][] contenido) {

this.contenido = contenido;

}

public void agegarElementoContenido(String elemento,int x,int y) {

this.contenido[x][y] = elemento;

}

public void setFichas(Ficha[][] fichas) {

this.fichas = fichas;

}

public abstract void generarTablero();

public abstract void generarFichasAlAzar();

public abstract void generarFichasPredeterminadas();

}

package dominio;

/\*\*

\*

\* @author Rodrigo Pistón(261777) | Rodrigo Camps(241344)

\*/

public class TableroRectangulo extends Tablero {

private static String [][] anexos = {{},{},{},{}};

private static int ancho = 20;

private static int largo = 20;

public TableroRectangulo() {

super(ancho, largo,anexos);

this.generarFichasPredeterminadas();

}

/// -- Array de 20 posiciones, que contiene array con las fichas predeterminadas colocadas

private int [][] posicionesPredeterminadas = {{3},{0},{3},{5},{0},{16},{0},

{0},{10,11},{0},{0},{10,20},{18},{17},

{6,11,20},{17},{5},{4,11},{3,14},{15}};

public int[][] getPosicionesPredeterminadas() {

return posicionesPredeterminadas;

}

@Override

public void generarTablero() {

String [][] contenidoTablero = new String[super.getLargo() + 2][super.getAncho() + 1];

// -- Se arma los espacios sin utilizar

contenidoTablero[0][0] = " ";

contenidoTablero[1][0] = " ";

for (int fila = 0; fila < this.getFichas().length ; fila++) {

for (int columna = 0; columna < this.getFichas()[0].length ; columna++) {

contenidoTablero[fila + 2][columna + 1] = getFichas()[fila][columna].getFichaColoreada();

if(columna < 9){

contenidoTablero[0][columna+ 1] = String.format("%d", ( columna + 1)) ;

contenidoTablero[1][columna + 1] = "" ;

}else{

contenidoTablero[0][columna +1] = Character.toString( String.format("%d", (columna + 1)).charAt(0)) ;

contenidoTablero[1][columna +1] = Character.toString( String.format("%02d", (columna - 9)).charAt(1));

}

}

contenidoTablero[fila + 2][0] = String.format("%02d", (fila + 1));

}

super.setContenido(contenidoTablero);

}

@Override

public void generarFichasAlAzar() {

Ficha ficha;

for (int fila = 0; fila < this.getFichas().length; fila++) {

for (int columna = 0; columna < this.getFichas()[fila].length; columna++) {

if(utilidad.Entrada.getRandom(0,20) == 0){

ficha = new Ficha(0,"\*");

}else{

ficha = new Ficha(0,"-");

}

this.agregarFicha(ficha, fila, columna);

}

}

}

@Override

public void generarFichasPredeterminadas() {

Ficha ficha;

for (int fila = 0; fila < this.getFichas().length; fila++) {

for (int columna = 0; columna < this.getFichas()[0].length; columna++) {

if( this.esPosicionPredeterminada(fila, columna)){

ficha = new Ficha(0,"\*");

}else{

ficha = new Ficha(0,"-");

}

this.agregarFicha(ficha, fila, columna);

}

}

}

private boolean esPosicionPredeterminada(int fila, int columnaSeleccionada) {

boolean generar = false;

// -- Para no tratar con la posición 0

columnaSeleccionada += 1;

for (int columna = 0; columna < this.getPosicionesPredeterminadas()[fila].length && !generar; columna++) {

if(this.getPosicionesPredeterminadas()[fila][columna] == columnaSeleccionada){

generar = true;

}

}

return generar;

}

}

package dominio;

/\*\*

\*

\* @author Rodrigo Pistón(261777) | Rodrigo Camps(241344)

\*/

public class TableroSaltar extends Tablero{

private static String [][] anexos = {{"60","40","30","20","10"},{},{""},{"1","2","3","4"}};

private static int ancho = 11;

private static int largo = 4;

public TableroSaltar() {

super(ancho, largo,anexos);

this.generarFichasPredeterminadas();

}

@Override

public void generarTablero() {

String [][] contenidoTablero = new String[(super.getLargo() \* 2) + 2][(super.getAncho()\* 2) + 2];

int puntuacion = 60;

int posicion = 1;

int filaTableroFichas = 0;

int columnaTableroFichas = 0;

boolean filaPar = false;

boolean columnaPar = false;

for (int fila = 0; fila < contenidoTablero.length ; fila++) {

filaPar = (fila + 1)% 2 == 0;

for (int columna = 0; columna < contenidoTablero[0].length ; columna++) {

columnaPar = (columna + 2) % 2 == 0;

if(columna == 0 ){

if(filaPar && puntuacion > 0){

contenidoTablero[fila][columna] = Integer.toString(puntuacion)+ " " ;

if(puntuacion == 60){

puntuacion -= 20;

}else{

puntuacion -= 10;

}

}else{

contenidoTablero[fila][columna] = " ";

}

}else if(fila == contenidoTablero.length - 1){

if(columnaPar && posicion <= 4){

contenidoTablero[fila][columna] = Integer.toString(posicion);

posicion ++;

}else{

contenidoTablero[fila][columna] = " ";

}

}

else if(filaPar){

if(!columnaPar){

contenidoTablero[fila][columna] = "|";

}else{

if(this.getFichas()[filaTableroFichas][columnaTableroFichas]!= null){

contenidoTablero[fila][columna] = this.getFichas()[filaTableroFichas][columnaTableroFichas].getFichaColoreada();

}

columnaTableroFichas ++;

}

}

else if(columnaPar){

contenidoTablero[fila][columna] = "-";

}else if(!columnaPar) {

contenidoTablero[fila][columna] = "+";

}

}

if(filaPar){

filaTableroFichas ++;

columnaTableroFichas = 0;

}

}

super.setContenido(contenidoTablero);

}

@Override

public void generarFichasAlAzar() {

// -- Valores entre el 1 y el 4

int offset = (int) (Math.random()\*(4-0)) + 1;

Ficha ficha;

/// -- Una mejor forma es revisar la posición de la columna anterior, si coincide, no se pone,y se pone en la siguiente.

for (int fila = 0; fila < super.getFichas().length ; fila++) {

for (int columna = 0; columna < super.getFichas()[0].length ; columna++) {

ficha = super.getFichas()[fila][columna];

if(!ficha.equals(new Ficha(0," "))){

int valorActual = ficha.getIdColor() + offset;

if(valorActual > 4){

int diferencia = (valorActual - 4);

ficha.setIdColor(diferencia);

}else{

ficha.setIdColor(valorActual);

}

super.agregarFicha(ficha, fila, columna);

}

}

}

}

@Override

public void generarFichasPredeterminadas() {

Ficha fRoja = new Ficha(1,"#");

Ficha fAzul = new Ficha(2,"#");

Ficha fVerde = new Ficha(3,"#");

Ficha fAma = new Ficha(4,"#");

Ficha fVacia = new Ficha(0," ");

Ficha [][] listaFichas = {{fRoja,fAzul,fVerde,fAma},{fAzul,fRoja,fAma,fVerde},{fVerde,fAma,fAzul,fRoja},{fAma,fVerde,fRoja,fAzul}};

for (int fila = 0; fila < this.getFichas().length; fila++) {

for (int columna = 0; columna < this.getFichas()[0].length; columna++) {

if(fila > 6){

this.agregarFicha(listaFichas[fila - 7][columna], fila, columna);

}else{

this.agregarFicha(fVacia, fila, columna);

}

}

}

}

}

Interfaz

package interfaz;

import dominio.Sistema;

import java.util.InputMismatchException;

import java.util.Scanner;

/\*\*

\*

\* @author Rodrigo Pistón(261777) | Rodrigo Camps(241344)

\*/

public class MenuPrincipal {

public static Sistema sistema = new Sistema();

public void mostrarOpciones(){

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

System.out.println("| [1] Registrar Jugador |");

System.out.println("| [2] Jugar a Saltar |");

System.out.println("| [3] Jugar a Rectangulo|");

System.out.println("| [4] Bitácora |");

System.out.println("| [5] Salir |");

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

public void mostrarOpcionesBitacora(){

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

System.out.println(" Seleccione orden de vista: ");

System.out.println("| [1] Por alias creciente |");

System.out.println("| [2] Por alias decreciente |");

System.out.println("| [3] Por puntaje creciente |");

System.out.println("| [4] Por puntaje decreciente |");

System.out.println("| [5] Volver |");

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

public void mostrarMenu(){

boolean salir = false;

int opcion;

while (!salir) {

mostrarOpciones();

try {

opcion = utilidad.Entrada.leerInt("una de las opciones", 0, 5);

switch (opcion) {

case 1:

MenuSecundario.RegistrarJugador();

break;

case 2: case 3:

if(this.sistema.getListaJugadores().size() == 0){

System.out.println("Debe de ingresar un jugador primero para elegir cualquier juego.");

}else{

MenuSecundario.CrearPartida(opcion);

}

break;

case 4:

if(this.sistema.getListaPartidas().size() == 0){

System.out.println("No hay partidas registradas en la bitacora.");

}else{

mostrarOpcionesBitacora();

opcion = utilidad.Entrada.leerInt("una de las opciones", 0, 5);

if(opcion != 5){

MenuSecundario.MostrarBitacora(opcion);

}

}

break;

case 5:

System.out.println("Fin de la ejecución");

salir = true;

break;

default:

System.out.println("Debes seleccionar solo números entre 1 y 5");

}

} catch (InputMismatchException e) {

System.out.println("Debes de ingrasar un número");

}

}

}

}

package interfaz;

import dominio.Ficha;

import utilidad.Entrada;

import dominio.Juego;

import dominio.Jugador;

import dominio.Partida;

import dominio.Rectangulo;

import dominio.Saltar;

import java.util.Collections;

import java.util.InputMismatchException;

import java.util.List;

import static utilidad.Entrada.esNumero;

/\*\*

\*

\* @author Rodrigo Pistón(261777) | Rodrigo Camps(241344)

\*/

public class MenuSecundario {

public static void RegistrarJugador(){

try{

String nombre = Entrada.leerString("nombre.");

String alias = Entrada.leerString("alias.");

int edad = Entrada.leerInt(" edad",0,99);

// -- Hacer las validaciones al input al nombre

Jugador jugador = new Jugador(nombre,edad,alias);

boolean jugadorAgregado = MenuPrincipal.sistema.agregarJugador(jugador);

if(jugadorAgregado){

System.out.println("Jugador agregado correctamente!");

}else{

System.out.println("El jugador ya existe en el sistema.");

}

}catch(InputMismatchException e){

System.out.println("Error al agregar al jugador");

}

}

private static Jugador SeleccionarJugador(){

Jugador jugador = null;

boolean entradaValida = false;

while(!entradaValida){

System.out.println("Seleccione un jugador:");

List<Jugador> listaJugadores = MenuPrincipal.sistema.getListaJugadores();

for(int posicion = 0; posicion < listaJugadores.size(); posicion ++ ){

System.out.printf("[%d] %s\n",(posicion + 1), listaJugadores.get(posicion).getNombre());

}

int posicionSeleccioanda = Entrada.leerInt("jugador",0,listaJugadores.size() + 1);

if(listaJugadores.size() >= posicionSeleccioanda - 1){

jugador = listaJugadores.get(posicionSeleccioanda - 1);

entradaValida = true;

}else{

System.out.println("Debe de seleccionar un jugador de la lista.");

}

}

return jugador;

}

/\*\*

\*

\* @return True:Configuración Predeterminada, False:Configuración al Azar

\*/

private static boolean SeleccionarConfiguracion(){

// -- Se selecciona la configuración

System.out.println("Seleccione configuracion: \n\t-[1]Predetermianda\n\t-[2]Azar");

int ingresoConfiguracion = Entrada.leerInt(" 1 o 2",0,3);

return ingresoConfiguracion == 1;

}

public static void CrearPartida(int opcion){

// -- Se selecciona el jugador

Jugador jugadorSeleccionado = SeleccionarJugador();

Juego nuevoJuego;

switch(opcion){

case 2:

nuevoJuego = jugarSaltar();

break;

case 3:

nuevoJuego = jugarRectangulo();

break;

default:

System.out.println("Ningun juego seleccionado.");

return;

}

// -- Se crea la nueva partida

Partida partida = new Partida(jugadorSeleccionado,nuevoJuego,nuevoJuego.calcularPuntaje());

MenuPrincipal.sistema.agregarPartida(partida);

}

private static Juego jugarRectangulo (){

Rectangulo juegoRectangulo = new Rectangulo(SeleccionarConfiguracion());

Ficha[] listaFichas = {new Ficha(1,"#"),new Ficha(2,"#"),new Ficha(3,"#"),new Ficha(4,"#")};

String [] arrayEntrada = {"","","",""};

String entrada = "";

Ficha ficha;

int contListaFichas = 0;

int rectanguloPosicionX = 0;

int rectanguloPosicionY = 0;

int rectanguloAlto = 0;

int rectanguloAncho = 0;

int cantidadRectangulos = 0;

boolean primeraJugada = true;

boolean terminaJuego = false;

juegoRectangulo.iniciar();

while(!terminaJuego){

// -- Para cortar un poco el rastro en pantalla y que parezca que refresca las mismas posiciones

System.out.println("\n\n\n\n\n\n\n\n\n\n");

// -- Se va alternando entre fichas

ficha = listaFichas[contListaFichas];

// -- Se imprime el tablero a pantalla

System.out.println(juegoRectangulo.getTablero().getContenidoString() + "\n");

// -- Se lee la entrada con una función utilitaria

entrada = utilidad.Entrada.leerString("los siguientes valores [Posición X] [Posición Y] [Alto] [Ancho]\nen una sola línea númerica o presiones X para finalizar:");

// -- Se convierte la entrada en un array con split, se utiliza expresiones regulares para separar por espacios o tabuladores,

// para imperdir que tome mal la entrada con muchos espacios

arrayEntrada = entrada.trim().split("\\s+\\t|\\t\\s+|\\s+");

try

{

if(arrayEntrada.length == 4){

rectanguloPosicionX = Integer.parseInt(arrayEntrada[0]);

rectanguloPosicionY = Integer.parseInt(arrayEntrada[1]);

rectanguloAlto = Integer.parseInt(arrayEntrada[2]);

rectanguloAncho = Integer.parseInt(arrayEntrada[3]);

if(!juegoRectangulo.siguienteMovimiento(ficha,rectanguloPosicionX,rectanguloPosicionY,rectanguloAlto,rectanguloAncho,primeraJugada)){

System.out.println("Error: Movimiento invalido.");

}else{

cantidadRectangulos ++;

primeraJugada = false;

};

// -- Solo se cambia de color cuando es correcta la entrada

if(contListaFichas == 3){

contListaFichas = 1;

}else{

contListaFichas ++;

}

}else if(!arrayEntrada[0].equals("X"))

{

System.out.println("\nError: Se ingresaron datos invalidos, debe ingresar al menos 4 valores");

}

}catch(Exception e){

System.out.println("\nError: Se ingresaron datos invalidos, solo puede ingresar números");

}

juegoRectangulo.recargar();

System.out.printf("[ \* El puntaje es: %d \* ]\n",juegoRectangulo.calcularPuntaje());

if(arrayEntrada[0].equals("X") || arrayEntrada[0].equals("x") || cantidadRectangulos >= 10 || !juegoRectangulo.quedanRectangulosDisponibles()){

terminaJuego = true;

System.out.println("Fin del juego!");

}

}

return juegoRectangulo;

}

private static Saltar jugarSaltar(){

Saltar juegoSaltar = new Saltar(SeleccionarConfiguracion());

String entrada = "";

Ficha[] listaFichas = {new Ficha(1,"#"),new Ficha(2,"#"),new Ficha(3,"#"),new Ficha(4,"#")};

int contListaFichas = 0;

int puntaje = 0;

int columnaIngresada = 0;

int columnaRecomendada = 0;

boolean terminaJuego = false;

Ficha ficha;

juegoSaltar.iniciar();

// -- Se itera hasta que se presione X

while(!terminaJuego){

// -- Para cortar un poco el rastro en pantalla y que parezca que refresca las mismas posiciones

System.out.println("\n\n\n\n\n\n\n\n\n\n");

// -- Se va alternando entre fichas

ficha = listaFichas[contListaFichas];

// -- Se imprime el tablero a pantalla

System.out.println(juegoSaltar.getTablero().getContenidoString());

// -- Se indica que ficha va a ser la siguiente en mover

System.out.printf("\nSiguiente color de ficha a mover: %s \n",ficha.getFichaColoreada());

// -- Se lee la entrada del usuario

entrada = utilidad.Entrada.leerString("columna, [A] para ayuda o [X] para finalizar");

if(esNumero(entrada)){

columnaIngresada = Integer.parseInt(entrada) - 1;

if(!juegoSaltar.siguienteMovimiento(ficha,columnaIngresada)){

System.out.println("Error: Movimiento invalido.");

};

}else if(!entrada.equals("X") && !entrada.equals("x") && !entrada.equals("A") && !entrada.equals("a")){

System.out.println("Error, debe de ingresar una columna para realizar la siguiente jugada o X para finalizar");

}

columnaRecomendada = juegoSaltar.obtenerColumnaRecomendada(ficha);

if(columnaRecomendada == -1){

terminaJuego = true;

System.out.println("No quedan mas movimientos!");

}else{

if(entrada.equals("A") || entrada.equals("a")){

System.out.printf("Columna recomendada:%d",columnaRecomendada);

}else{

if(contListaFichas == 3){

contListaFichas = 1;

}else{

contListaFichas ++;

}

// -- Se muestra puntaje en pantalla

}

}

juegoSaltar.recargar();

if(entrada.equals("X") || entrada.equals("x")){

terminaJuego = true;

}

}

System.out.println("Fin del juego!");

return juegoSaltar;

}

public static void MostrarBitacora(int opcion){

System.out.println("Juego | Alias | Edad | Puntaje | Fecha ");

switch(opcion){

case 1:

for(Partida partida : MenuPrincipal.sistema.partidasOrdenadasAliasDesc()){

System.out.printf("%10s| %10s | %6d | %8d | %10s \n",partida.getJuego().getClass().getSimpleName(),partida.getJugador().getAlias(),partida.getJugador().getEdad(),partida.getJuego().getPuntaje(),partida.getHoraComienzo());

}

break;

case 2:

for(Partida partida : MenuPrincipal.sistema.partidasOrdenadasPuntajeDesc()){

System.out.printf("%s | %s | %d | %d | %s \n",partida.getJuego().getClass().getSimpleName(),partida.getJugador().getAlias(),partida.getJugador().getEdad(),partida.getJuego().getPuntaje(),partida.getHoraComienzo());

}

break;

}

}

}

package interfaz;

/\*\*

\*

\* @author Rodrigo Pistón(261777) | Rodrigo Camps(241344)

\*/

public class Solitario {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

MenuPrincipal menuPrincipal = new MenuPrincipal();

menuPrincipal.mostrarMenu();

}

}

Utilidad

package utilidad;

import java.util.InputMismatchException;

import java.util.Random;

import java.util.Scanner;

/\*\*

\*

\* @author Rodrigo Pistón(261777) | Rodrigo Camps(241344)

\*/

public class Entrada {

public static String leerString(String texto){

Scanner input = new Scanner(System.in);

String textoIngresado = "";

boolean entradaValida = false;

while (!entradaValida){

try {

System.out.printf("Ingrese %s\n" ,texto);

textoIngresado = input.nextLine().trim();

if(textoIngresado.equals("")){

System.out.print("No puede ingresar un valor vacío, ");

}else{

entradaValida = true;

}

}catch (InputMismatchException e) {

System.out.print("Error, entrada inválida, ");

}

}

return textoIngresado;

}

public static int leerInt(String texto,int minimo,int maximo){

Scanner input = new Scanner(System.in);

boolean entradaValida = false;

int numeroIngresado = 0;

while (!entradaValida) {

try {

System.out.println("Ingrese " +texto);

numeroIngresado = input.nextInt();

if(minimo != 0 && numeroIngresado <= minimo){

System.out.printf("Debes de ingresar un número de %s mayor a %d\n",texto,minimo);

}else if(maximo != 0 && numeroIngresado >= maximo){

System.out.printf("Debes de ingresar un número de %s menor a %d\n",texto,maximo);

}else{

entradaValida = true;

}

} catch (InputMismatchException e) {

System.out.printf("Debes ingresar un valor de %s valido.\n",texto );

input.nextLine();

}

}

return numeroIngresado;

}

/\*\*

\*

\* @param texto

\* @return

\*/

public static boolean leerBoolean(String texto){

Scanner input = new Scanner(System.in);

String textoIngresado = "";

boolean entradaValida = false;

while (!entradaValida) {

try {

System.out.printf("Ingrese %s\n" ,texto);

textoIngresado = input.nextLine().trim();

if(textoIngresado.equals("")){

System.out.print("No puede ingresar un valor vacío, ");

}else{

entradaValida = true;

}

} catch (InputMismatchException e) {

System.out.print("Error, entrada inválida, ");

}

}

return textoIngresado == "Y" || textoIngresado == "y" || textoIngresado == "S" || textoIngresado == "s" ;

}

public static boolean esNumero(String str) {

try {

Integer.parseInt(str);

return true;

} catch(NumberFormatException e){

return false;

}

}

public static int getRandom(int min, int max) {

return new java.util.Random().nextInt(max - min + 1) + min;

}

}

package utilidad;

import dominio.Ficha;

/\*\*

\*

\* @author Rodrigo Pistón(261777) | Rodrigo Camps(241344)

\*/

public class Generico {

// -- 0:RESET - 1:ROJO - 2:AZUL - 3:VERDE - 4:AMARILLO

public static String [] COLORES = {"\u001B[0m","\u001B[31m","\u001B[34m","\u001B[32m","\u001B[33m"};

public static Ficha fichaVacia = new Ficha(0," ");

// 0:RESET | 1:[#] ROJO | 2: [#] AZUL | 3: [#] VERDE | 4: [#] AMARILLO | 5: [-] Sin color | 6: [\*] Sin Color

public static String [] caracteresColores = {

" ",

COLORES[1]+"#"+ COLORES[0],

COLORES[2]+"#"+ COLORES[0],

COLORES[3]+"#"+ COLORES[0],

COLORES[4]+"#"+ COLORES[0],

"-",

"\*"

};

public static String RetornarColor(int codigo){

return caracteresColores[codigo];

}

public static String ResolverTextoColor(int codigo){

String textoColor = "";

switch(codigo){

case 1:

textoColor = COLORES[1] + "[R] Rojo";

break;

case 2:

textoColor = COLORES[2] + "[A] Azul";

break;

case 3:

textoColor = COLORES[3] + "[V] Verde";

break;

case 4:

textoColor = COLORES[4] + "[M] Amarillo";

break;

}

return textoColor + COLORES[0];

}

public static void CopiarMatriz(Ficha[][] original, Ficha[][] copia){

for (int fila = 0; fila < original.length; fila++) {

for (int columna = 0; columna < original[0].length; columna++) {

copia[fila][columna] = original[fila][columna];

}

}

}

public static boolean EsNumeral(int codigo){

return caracteresColores[codigo].contains("#");

}

}