Manual del Programador

José Miguel Guzmán 20182020127
Jeasson Alfonso Suarez 20182020107
Laura Catalina Preciado 20182020122
15/07/2019

Universidad Distrital Francisco José de Caldas
Ingeniería de Sistemas
Programación orientada a objetos

Introducción

En este Manual se hace Evidencia de los procesos, requerimientos y conceptos técnicos que se necesitaron para realizar el proyecto, así como también se explicara el código fuente del programa para un mejor manejo de servicios de mantenimiento, revisión, solución de problemas, y optimización del mismo.

Requerimientos:

Para el uso adecuado del código fuente, para su manejo y optimización se necesitan los siguientes requerimientos instalados en el sistema.

- Eclipse IDE
- Windows Builder (A Preferencia del programador)

Herramientas Teóricas:

Para entender el proyecto a su totalidad se deben manejar los siguientes conceptos del paradigma de la programación orientada a objetos:

- Encapsulamiento
- Herencia
- Polimorfismo
- Abstracción
- Modularidad
- Clase
- Objeto
- Excepciones
- Archivos

Código fuente Del Programa:

- Función Main: Ejecuta el frame principal del programa.

```java
package CuatroEnLinea;

import CuatroEnLinea.Vista.VentanaInicio;

/**
 * <p> Es la Clase principal que ejecuta el frame inicial </p>
 * @author Miguel Guzman 20182020127
 * @author Jeasson Suarez 20182020107
 * @author Catalina Preciado 20182020122
 * @since 10/07/2019
 * @version 1.0
 */

public class Main {
    public static void main(String[] args) {
        VentanaInicio inicio = new VentanaInicio();
        inicio.setVisible(true);
    }

}
```

- Esta Clase perimite guardar la puntuacion de los jugadores en un archive plano

```java
package CuatroEnLinea.Logica;

import java.io.DataInputStream;

public class Archivos4EnLinea { //clase que creara y grabara los archivos del juego

    public Archivos4EnLinea() { //contructor por defecto

    }

    public void grabarArchivo(int tiempojl, String njl) {
        DataOutputStream score=null;//creacion de un objeto tipo archivo

        String nombreJugadorl= njl; //creando e inicializando los campos de registro

        int tempo=tiempojl;

        try {
            score=new DataOutputStream(new FileOutputStream("src/finalArchivo/score.txt", true));//se creara un archivo en el que se puede volve

            score.writeUTF(nombreJugadorl);//se graban los datos en un archivo score, guardando ambos jugadores de la partida
            score.writeInt(tempo);
            score.close();//se cierra la grabacion en el archivo para que sea sastisfactorio el cargue
        } catch (Exception e) {
            // TODO: handle exception
        }
    }

    public void grabarArchivo(int num) {
        DataOutputStream numArc=null;                   //creacion de un objeto tipo archivo
        try {
            numArc=new DataOutputStream(new FileOutputStream("src/finalArchivo/num.txt"));
            numArc.writeInt(num);//se graba en un archivo num un valor entero que indicara cuantas veces se ejecuto el archivo
            numArc.close();                             //cerramos la grabacion en el archivo
        } catch (Exception e) {
        }
    }
}
```

- Esta Clase Condiciona el turno del jugador

```java
package CuatroEnLinea.Logica;

public class CondicionJugador {
    //CONSTRUCTOR DE LA CLASE
    public CondicionJugador(){
    }
    /**
     * @return Retorna verdadero si juega el jugador 2
     */
    //METODO QUE PERMITE EL INTERCAMBIO DE TURNOS DE LOS JUGADORES
    public boolean condicion(int n){
        if(n%2==1){
            return true;
        }else{
            return false;
        }
    }
}
```

- Es la clase jugador de la cual se derivan los diferentes tipos de jugadores

```java
package CuatroEnLinea.Logica;

public class Jugador {
    //ATRIBUTOS
    protected String Nombre;
    //CONSTRUCTOR
    public Jugador(){
    }
    //METODOS SETTERS AND GETTERS
    public String getNombre() {
        return Nombre;
    }

    public void setNombre(String nombre) {
        this.Nombre = nombre;
    }
}
```

- Es la clase que extiende de jugador y permite organizar las mejores puntuaciones

```
package CuatroEnLinea.Logica;
//claase que implementa una interfas para poder ordenar los jugadores de mejor a peor
public class jugadorPuntaje extends Jugador implements Comparable<jugadorPuntaje> {
    private int tiempo;                 //tiempo que tarda jugando
    //EL ATRIBUTO NOMBRE LO HEREDA DE JUGADOR
    public jugadorPuntaje() {
    }
    public int getTiempo() {    //getters y setters de variables de clase
        return tiempo;
    }
    public void setTiempo(int tiempo) {
        this.tiempo = tiempo;
    }
    @Override
    public int compareTo(jugadorPuntaje o) {//metodo implementado
        if(o.getTiempo()>this.tiempo) { //comparaciones
            return -1;
        }else if(o.getTiempo()>tiempo) {
            return 0;
        }else {
            return 0;
        }

    }

}
```

- Esta clase permite seleccionar un simbolo para cada jugador

```
package CuatroEnLinea.Logica;

import java.net.URL;

public class SelectorImagen {
    /**
     * @see El arreglo de imagenes se usa en los frames de juego
     */
    //ATRIBUTOS CON LAS IMAGENES PARA CADA JUGADOR
    String urlJ1 = "D:/ECLIPSE/CuatroEnLinea/src/images/imJ1.jpg";
    String urlJ2 = "D:/ECLIPSE/CuatroEnLinea/src/images/imJ2.jpg";
    String Imagen[] = new String[2];
    //CONSTRUCTOR DE LA CLASE QUE ME GUARDA LAS IMAGENES EN UN ARREGLO
    public SelectorImagen(){
        Imagen[0]=urlJ1;
        Imagen[1]=urlJ2;
    }
    //METODOS SETTER AND GETTER
    public String[] getImagen() {
        return Imagen;
    }

    public void setImagen(String[] imagen) {
        Imagen = imagen;
    }

}
```

- El Frame Game es el frame del juego Jugador vs Computadora

```
package CuatroEnLinea.Vista;

import java.awt.BorderLayout;

public class Game extends JFrame {
    // ATRIBUTOS E INSTANCIAS PARA LA CLASE
    SelectorImagen imgn = new SelectorImagen();
    CondicionJugador condicion = new CondicionJugador(); // constructor vacio
    TerminarGame terminar = new TerminarGame();
    private int juega = 0;
    private String nombre;
    private int jugadaJ1;
    String[] selector = new String[3];
    String urlRevers = "D:/ECLIPSE/EjemploCuatroEL/src/images/revers.jpg";
    Winner winner = new Winner();
    private boolean ocupado[][];
    private boolean bloqueo[][];
    private boolean color[][];
    // SE CREAN LOS BOTONES
    JButton A1 = new JButton("");
    JButton A2 = new JButton("");
    JButton A3 = new JButton("");
    JButton A4 = new JButton("");
    JButton A5 = new JButton("");
    JButton A6 = new JButton("");
    JButton A7 = new JButton("");
```

```
    // METODO QUE PERMITE VACIAR LAS OPCIONES DE LA MAQUINA
    public void vaciar() { // al iniciar finalizar vital
        ocupado = new boolean[6][7];
        color = new boolean[6][7];
        bloqueo = new boolean[6][7];
        for (int i = 0; i < 6; i++) {
            for (int j = 0; j < 7; j++) {
                bloqueo[i][j] = true;
                ocupado[i][j] = false;
                color[i][j] = false;
                bloqueo[0][j] = false; // primera fila
            }
        }
    }

    // DIBUJA
    public void dibujar(int i, int j) {
        if (!(bloqueo[i][j])) {
            if (!(ocupado[i][j])) {
                if (i != 5) {
                    bloqueo[i + 1][j] = false;
                }
                color[i][j] = true;
                ocupado[i][j] = true;
                logica(i, j);
                perdedor();
            }
        }
    }
```

- El Frame Game2 es el frame del jugador 1 vs jugador 2

```
package CuatroEnLinea.Vista;

import java.awt.BorderLayout;

public class Game2 extends JFrame {
    //ATRIBUTOS E INSTANCIAS PERTINENTES
    SelectorImagen imgn = new SelectorImagen();
    CondicionJugador condicion = new CondicionJugador();
    TerminarGame terminar = new TerminarGame();
    private int juega = 0;
    private int jugadaJ1 = 0;
    private int jugadaJ2 = 0;
    private String nombre;
    private String nombre2;
    String[] selector = new String[3];
    String urlRevers = "D:/ECLIPSE/CuatroEnLinea/src/images/revers.jpg";
    Winner winner = new Winner();
    //CREA LOS BOTONES
    JButton A1 = new JButton("");
    JButton A2 = new JButton("");
    JButton A3 = new JButton("");
    JButton A4 = new JButton("");
    JButton A5 = new JButton("");
    JButton A6 = new JButton("");
    JButton A7 = new JButton("");

    /**
     * Create the frame.
     */
    public Game2() {
        //CREA Y MODIFICA EL FRAME
        setResizable(false);
        setType(Type.UTILITY);
        setTitle("4 En Linea");
        setFont(new Font("Lucida Calligraphy", Font.BOLD | Font.ITALIC, 18));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setBounds(600, 0, 800, 800);
        Panel contentPane = new Panel("/images/Jugadores.jpg");
        getContentPane().add(contentPane);
        contentPane.setLayout(null);

        //POSICIONA LOS BOTONES EN EL FRAME
        A1.setIcon(new ImageIcon(urlRevers));
        A1.setBounds(105, 603, 83, 83);
        contentPane.add(A1);
        A1.setEnabled(false);

        A2.setIcon(new ImageIcon(urlRevers));
        A2.setBounds(188, 603, 83, 83);
        contentPane.add(A2);
        A2.setEnabled(false);
```

- **La clase Game:** ejecuta el juego contra la computadora, la clase tiene la misma base que la clase Game2, con unas dos diferencias marcadas, la primera esta dentro del constructor de la clase, ya que no hace uso de la clase CondiciónJugador que indica quien juega.

```
//ESTE PROCESO SE REPITE PARA CADA BOTON
//ES UN ACTION LISTENER PARA EL BOTON A1
A1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        if (p1) {//SI EL BOTON ESTA HABILITADO ENTONCES
            dibujar(0, 0);//ENVIA LOS DATOS CON EL CUAL LA COMPUTADORA JUEGA
            A1.setIcon(new ImageIcon(imgn.getImagen()[1]));//ENVIA EL SIMBOLO DEL JUGADOR 1
            A1.setToolTipText(nombre);//ENVIA EL IDENTIFICADOR AL BOTON CON EL NOMBRE DE USUARIO
            p1 = false;//INHABILITA EL BOTON
            juega++;//AUMENTA LA VARIABLE CONTADORA PARA EL SIGUIENTE TURNO
            jugadaJ1++;//AUMENTA LA VARIABLE CONTADORA PARA LAS JUGADAS REALIZADAS POR EL USUARIO
            winner.setUno(A1.getToolTipText());//ENVIA A WINNER LA JUGADA A SU RESPECTIVA POSICION EN LA MATRIZ
            rdbtnJ1.setVisible(true);//INTERCALA LOS RADIOBUTTON
            rdbtnJC.setVisible(false);
        }
        if (winner.Ganador()) {//PREGUNTA SI HAY UN GANADOR
            terminar.Botones1();//FINALIZA EL PROCESO DE TODOS LOS BOTONES
            Resultados vr = new Resultados();//GENERA LA VENTANA GANADORA
            vr.setNombreJ(A1.getToolTipText());//ENVIA EL NOMBRE DEL GANADOR
            vr.setScore(jugadaJ1);//ABRE LA VENTANA
            vr.setVisible(true);
            setVisible(false);
        }
    }
});
```

- La segunda diferencia está en los métodos usados para que la maquina determine posiciones validas en el juego, para esto hace uso de tres matrices que se inicializan con la función vaciar.

```
public void vaciar() {
    ocupado = new boolean[6][7];
    color = new boolean[6][7];
    bloqueo = new boolean[6][7];
    for (int i = 0; i < 6; i++) {
        for (int j = 0; j < 7; j++) {
            bloqueo[i][j] = true; //para respetar las reiglas del 4 en linea
            ocupado[i][j] = false; //indica que posicion esta ocupada
            color[i][j] = false; //indica cual jugador tiene el lugar ocupado
            bloqueo[0][j] = false; // desbloquea primera fila
        }
    }
}
```

- Al ejecutar el constructor se puede ver la clase dibujar y dos parámetros que indican cual es el botón donde el usuario hace clic, a continuación, se muestra la función dibujar, que se encarga de llenar las matrices color, bloqueo y ocupado en la posición i, j con usuario (maquina o jugador), dar valor true a la fila superior para que la maquina pueda jugar en esa posición e indicar que esa posición esta está ocupada.

```
public void dibujar(int i, int j) {
    if (!(bloqueo[i][j])) {
        if (!(ocupado[i][j])) {
            if (i != 5) {
                bloqueo[i + 1][j] = false; //desbloquear la parte superior
            }
            color[i][j] = true;//jugador humano
            ocupado[i][j] = true;
            logica(i, j);
            perdedor();
        }
    }
}
```

- Luego de guardar los datos del jugador se ejecuta la función encargada de encontrar posiciones validas alrededor del jugador, a continuación, se muestra un pedazo del código donde se muestra la condición mas simple, la que hace que la maquina se ubique sobre la ficha del jugador.

```
public void logica(int i, int j) {
    boolean a = false;
    int fila = i;
    int columna = j;
    a = pensar(i, j);

    while (!a) { //si ya dibujo ficha se deja de ejecutar de lo contrario ejecutese hasta dibujar
        i = fila;
        j = columna;
        int random;
        if (i % 2 == 0 & j % 3 == 0) { //evita que entre mas veces a la condicion de dibujar arriba
            random = (int) (Math.random() * 5);
        } else {
            random = (int) (Math.random() * 5 + 1);
        }

        if (random == 0 & !a) { //dibujar arriba
            if (i != 5) {
                if (!bloqueo[i + 1][j] & !ocupado[i + 1][j]) {
                    a = pc(i + 1, j);
                }
            }
        }
```

- Las funciones a continuación son la extensión de la anterior, ya que cumplen con la función anterior, pero que tienen otras condiciones, en el método pensar por ejemplo tiene la función de evitar que el jugador complete 4 fichas seguidas, hace esto evaluando si hay tres fichas seguidas del mismo color o sea 3 trues seguidos y pasando el dato a la función pc de donde tiene que ubicar la ficha para evitar que gane el jugador.

```
public boolean pensar(int i, int j) {
    boolean a = false;
    a = pensar();
    if (!a) {
        if (j <= 5 & j >= 2 & !a) { //solo es posible hacer esta jugada en este rango
            if (ocupado[i][j] & ocupado[i][j - 1] & ocupado[i][j - 2]) { // winer en lateral izquierda-derecha
                if (color[i][j] & color[i][j - 1] & color[i][j - 2]) { // 3 fichas jugador seguidas
                    if (!ocupado[i][j + 1] & !bloqueo[i][j + 1]) {
                        a = pc(i, j + 1);
                    }
                }
            }
        }
        if (j >= 1 & j <= 4 & !a) {
            if (ocupado[i][j] & ocupado[i][j + 1] & ocupado[i][j + 2]) { // winer
                                                                         // en
                                                                         // lateral
                                                                         // derecha-izquierda
                if ((color[i][j]) & (color[i][j + 1]) & (color[i][j + 2])) { // 3 fichas jugador seguidas
                    if (!ocupado[i][j - 1] & !bloqueo[i][j - 1]) {
  public boolean pensar() {
      boolean a = false;

      for (int i = 0; i < 6; i++) {
          for (int j = 0; j < 7; j++) {
              if (!a) {
                  if (j < 4) { // compara las posiciones de lado derecha a izquierda
                      if (ocupado[i][j] & ocupado[i][j + 1]
                              & ocupado[i][j + 2]) {
                          if (!color[i][j] & !color[i][j + 1]
                                  & !color[i][j + 2]) {
                              if (!ocupado[i][j + 3] & !bloqueo[i][j + 3]) {
                                  a = pc(i, j + 3);//lado derecho
                              }
                          }
                      }
                  }
```

- Finalmente, la función más simple, pero la que se relaciona estrechamente con la función lógica (que analiza las posiciones), ya que dibuja el botón que la computadora eligió y le asigna valores a las tres matrices con la cual manejábamos toda la lógica.

```
public boolean pc(int i, int j) {

    color[i][j] = false;
    ocupado[i][j] = true;
    if (i != 5) {
        bloqueo[i + 1][j] = false;
    }
    if (i == 0 & j == 0) {
        A1.setIcon(new ImageIcon(imgn.getImagen()[0]));
        A1.setToolTipText(lbNombreComp.getText());
        p1 = false;
        juega++;
        winner.setUno(A1.getToolTipText());
        rdbtnJC.setVisible(true);
        rdbtnJ1.setVisible(false);
    }
```