Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Organización Computacional
Segundo semestre 2019
Catedrático: Ing. Otto Rene Escobar Leiva
Auxiliar: Christian Real

MANUAL TÉCNICO

22/11/2019

Cristian Suy - *[Coordinador] 201700918* - CSuy
Yelstin de León - *201602836* - airton47
Ricardo Pérez - *201700524* - Ricardo16X
Byron Gómez - *201700544* - ByrCas
Andrea Sáenz - *201503484* - andreadsaenz
Juan Pablo Alvarado - **201700511** - 201700511

Manual Técnico

OBJETIVOS

- Familiarizarse con los conceptos básicos de Organización Computacional.
- Aprender el manejo básico de archivos en NetBeans, para su uso posterior en el desarrollo de proyectos.
- Desarrollar métodos propios de lectura de archivos, carácter por carácter, que se utilizara para la correcta comunicación paralela.

REQUERIMIENTOS MÍNIMOS

Los requerimientos mínimos para que el programa funcione correctamente, son los siguientes:

1. JRE (java runtime enviroment).
2. Se puede ejecutar en sistema operativo Windows, mac y linux independientemente de la versión:
    a. Windows 10 (8u51 y superiores)
    b. Windows 8.x (escritorio)
    c. Windows 7 SP1
    d. Windows Vista SP2
    e. Windows Server 2008 R2 SP1 (64 bits)
    f. Windows Server 2012 y 2012 R2 (64 bits)
    g. Mac con Intel que ejecuta Mac OS X 10.8.3+, 10.9+
    h. Suse Linux Enterprise Server 10 SP2+, 11.x
    i. Suse Linux Enterprise Server 12.x (64 bits)[2] (8u31 y superiores)
    j. Ubuntu Linux 12.04 LTS, 13.x
    k. Ubuntu Linux 14.x (8u25 y superiores)
    l. Ubuntu Linux 15.04 (8u45 y superiores)
    m. Ubuntu Linux 15.10 (8u65 y superiores)
3. RAM: 128 MB
4. Espacio en disco: 124 MB para JRE; 2 MB para Java Update
5. Procesador: Mínimo Pentium 2 a 266 MHz

PLATAFORMA DE DESARROLLO

El programa se ha desarrollado en el IDE: Netbeans versión 8.2.

- JDK 8u111
- Sistema utilizado:
- Sistema operativo: Windows 10 Home de 64 bits
- Procesador x64 bits
- Java 8 Update 161 C

CÓDIGO POR UTILIZAR

```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String username = tf_username.getText();
        String password = tf_password.getText();

        if(username.equals(LogIn.admin.getUsername()) && password.equals(LogIn.admin.getPassword())){
            AdminForm admin_form = new AdminForm();
            this.dispose();
            admin_form.setVisible(true);
        }else{
            for(Jugador j: LogIn.admin.getListaJugadores()){
                int a = 0;
                if(j.getUsername().equals(tf_username.getText()) && j.getPassword().equals(password)){
                    if(LogIn.admin.getListaJugadores().get(a).isAceptado()==true){
                        PlayerForm player_form = new PlayerForm();
                        this.dispose();
                        loggedPlayer = j;
                        player_form.setVisible(true);

                    }else{
                        JOptionPane.showMessageDialog(null,"La solicitud del jugador aún no ha sido aprovada por el administador!");
                    }
                    return;
                }
                a++;
            }
            JOptionPane.showMessageDialog(null,"El usuario ingresado no existe o la contraseña es incorrecta!");
```

```
     */
    public EliminarUsuarioForm() {
        initComponents();
        this.setLocationRelativeTo(null);
    }

    public void desplegarLista(){
        int a = LogIn.admin.getListaJugadores().size();
        String[] b = new String[a];
        int c = 0;
        System.out.println(a);
        if(!LogIn.admin.getListaJugadores().isEmpty()){
            for(Jugador j: LogIn.admin.getListaJugadores()){
                b[c] = j.getUsername();
                c++;
            }
        }
        jList1.setModel(new javax.swing.AbstractListModel<String>() {
            //String[] strings = { "Item 1", "Item 2", "Item 3", "Item 4", "Item 5" };
            public int getSize() { return b.length; }
            public String getElementAt(int i) { return b[i]; }
            });

        jList1.addListSelectionListener(new javax.swing.event.ListSelectionListener() {
        public void valueChanged(javax.swing.event.ListSelectionEvent evt) {
            jList1ValueChanged(evt);
            }
        });

        jList1.repaint();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code

    private void formWindowOpened(java.awt.event.WindowEvent evt) {
        // TODO add your handling code here:
        desplegarLista();
    }

    private void jList1ValueChanged(javax.swing.event.ListSelectionEvent evt) {
        // TODO add your handling code here:
```

```
public class PantallaJuego extends JFrame implements Runnable {
    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        LogIn login = new LogIn();
        this.dispose();
        login.setVisible(true);
    }

    private void formWindowOpened(java.awt.event.WindowEvent evt) {
        // TODO add your handling code here:
        if(LogIn.loggedPlayer!=null){
            tf_user.setText(LogIn.loggedPlayer.getUsername());
            tf_time.setText(String.valueOf(LogIn.loggedPlayer.getTiempo()));
            tf_score.setText(String.valueOf(LogIn.loggedPlayer.getPuntos()));
            tf_level.setText(String.valueOf(LogIn.loggedPlayer.getNivel()));
        }else{
            LogIn login = new LogIn();
            this.dispose();
            login.setVisible(true);
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new PlayerForm().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
```

```
public PantallaJuego() {
        /*
         * Creando las variables para el uso del hilo y el envío de datos.
         */
        coorCuerpo = new Coordenadas(0, 0);
        movSnake = new Thread(coorCuerpo);
        juego = new Thread(this);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 550, 419);
        contentPane = new JPanel();
        contentPane.setBackground(new Color(153, 153, 204));
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setLocationRelativeTo(null);
        setContentPane(contentPane);

        tablero = new JLabel[12][12];
        contentPane.setLayout(null);
        panel = new JPanel();
        panel.setBounds(10, 10, 360, 360);
        /* Inicializando la interfaz */
        panel.setBorder(new LineBorder(Color.WHITE));
        panel.setBackground(Color.DARK_GRAY);
        contentPane.add(panel);
        panel.setLayout(null);

        JLabel lblTiempo = new JLabel("Tiempo: 0:0");
        lblTiempo.setBounds(380, 93, 144, 14);
        contentPane.add(lblTiempo);
        tiempo = new Tiempo(lblTiempo);
        mostrarTiempo = new Thread(tiempo);

        JLabel lblJugador = new JLabel("Jugador:");
        lblJugador.setBounds(380, 10, 144, 14);
        contentPane.add(lblJugador);

        lblPunteo = new JLabel("Punteo: 0");
        lblPunteo.setBounds(380, 174, 144, 14);
        contentPane.add(lblPunteo);

        lblPausa = new JLabel("PAUSA");
        lblPausa.setVisible(false);
        lblPausa.setFont(new Font("Segoe UI", Font.PLAIN, 20));
        lblPausa.setHorizontalAlignment(SwingConstants.CENTER);
        lblPausa.setBounds(380, 349, 144, 21);
        contentPane.add(lblPausa);
```

```
            for (int i = 0; i < tablero.length; i++) {
                for (int j = 0; j < tablero.length; j++) {
                    JLabel casilla = new JLabel();
                    casilla.setOpaque(true);
                    casilla.setSize(30, 30);
                    casilla.setLocation((i) * 30, (j) * 30);
                    casilla.setBorder(new LineBorder(Color.WHITE));
                    casilla.setBackground(Color.RED);
                    /* Tablero de Juego */
                    tablero[i][j] = new JLabel();
                    tablero[i][j] = casilla;
                    /* Fin Tablero Juego */
                    panel.add(casilla);
                }
            }

            cuerpo = new ArrayList<Coordenadas>();
            cuerpo.add(new Coordenadas(6, 5));
            cuerpo.add(new Coordenadas(7, 5));
            cuerpo.add(new Coordenadas(8, 5));

            panel.revalidate();
            panel.repaint();

            envioParalelo = new Envio_Datos(X, tablero);
            enviarDatos = new Thread(envioParalelo);
            juego.start();
            movSnake.start();
            enviarDatos.start();
            mostrarTiempo.start();
            pintarSerpiente(1);
            pintarComida();
            nivel1();
        }

        private void azules() {
            Obstaculos = new ArrayList<>();
            String coordenadas1;
            for (int i = 0; i < tablero.length; i++) {
                for (int j = 0; j < tablero.length; j++) {
                    if(tablero[i][j].getBackground() == Color.BLUE) {
                        coordenadas1 = i + "," + j;
                        Obstaculos.add(coordenadas1);
                    }
                }
```

```
public void pintarComida() {
        PX = (int) (Math.random() * 10) + 1;
        PY = (int) (Math.random() * 10) + 1;
        while (tablero[PX][PY].getBackground() == Color.BLACK || tablero[PX][PY].getBackground() == Color.BLUE) {
                PX = (int) (Math.random() * 10) + 1;
                PY = (int) (Math.random() * 10) + 1;
        }
        tablero[PX][PY].setBackground(Color.LIGHT_GRAY);
        tablero[PX][PY].updateUI();
}

public void pintarSerpiente(int direccion) {
        if (direccion != 0) {
                limpiar();
                azules();

                switch (direccion) {
                case 1: // Izquierda
                        if (X == 0) {
                                X = 12;
                        }
                        X -= 1;
                        break;
                case 2: // Derecha
                        if (X >= 11) {
                                X = 0;
                        } else {
                                X += 1;
                        }
                        break;
                case 3: // Arriba
                        if (Y == 0) {
                                Y = 12;
                        }
                        Y -= 1;
                        break;
                case 4: // Abajo
                        if (Y >= 11) {
                                Y = 0;
                        } else {
                                Y += 1;
                        }
                        break;
                }
```

```
int ActualX = X, ActualY = Y;
int TempX, TempY;

for (int i = 0; i < cuerpo.size(); i++) {
        tablero[ActualX][ActualY].setBackground(Color.BLACK);
        TempX = cuerpo.get(i).ActualX;
        TempY = cuerpo.get(i).ActualY;
        cuerpo.get(i).ActualX = ActualX;
        cuerpo.get(i).ActualY = ActualY;
        ActualX = TempX;
        ActualY = TempY;
}

/*
 * Detectando colisiones con las paredes
 */
/*for (int i = 0; i < tablero.length; i++) {
        for (int j = 0; j < tablero.length; j++) {
                if (tablero[i][j].getBackground() == Color.BLUE) {
                        if (cuerpo.get(0).ActualX == j && cuerpo.get(0).ActualY == i) {
                                limpiar();
                                direccion = 0;
                                JOptionPane.showMessageDialog(null, "GAME OVER");
                        }
                }
        }
}*/

if(Obstaculos.size() > 0){
        String primero_serpiente = cuerpo.get(0).ActualX + "," + cuerpo.get(0).ActualY;
        boolean colision1 = Obstaculos.contains(primero_serpiente);
        if(colision1){
                JOptionPane.showMessageDialog(null, "GAME OVER");
        }
}

// Detectando Colisi□n con la Comida
if (PX == cuerpo.get(0).ActualX && PY == cuerpo.get(0).ActualY) {
        switch (direccion) {
        case 1: // Izquierda
                PX--;
                break;
        case 2: // Derecha
                PX++;
                break;
        case 3: // Abajo
```

```
                                PY++;
                                break;
                        case 4: // Arriba
                                PY--;
                                break;
                        }
                        punteo += 5;
                        lblPunteo.setText("Punteo: " + String.valueOf(punteo));
                        cuerpo.add(new Coordenadas(PX, PY));
                        pintarComida();
                } else {
                        /*
                         * Detectando colisiones con el cuerpo
                         */
                        for (int i = 1; i < cuerpo.size(); i++) {
                                if (cuerpo.get(0).ActualX == cuerpo.get(i).ActualX
                                                && cuerpo.get(0).ActualY == cuerpo.get(i).ActualY) {
                                        limpiar();
                                        direccion = 0;
                                        JOptionPane.showMessageDialog(null, "GAME OVER");
                                }
                        }
                }

                /*Reinicio de tamaño de snake si el punteo es igual a 50 o 100*/
                if(punteo == 50 && !cambioPunteo) {
                        cambioPunteo = true;
                        limpiar();
                        nivel2();
                        reiniciar();
                        nivel++;
                }else if(punteo == 100 && cambioPunteo) {
                        cambioPunteo = false;
                        limpiar();
                        nivel3();
                        reiniciar();
                        nivel++;
                }
        }
}
```