

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS

Ing. William Estuardo Escobar Argueta

Aux. Javier Oswaldo Mirón Cifuentes

Aux. Hector Josue Orozco Salazar



USAC.
TRICENTENARIA
Universidad de San Carlos de Guatemala

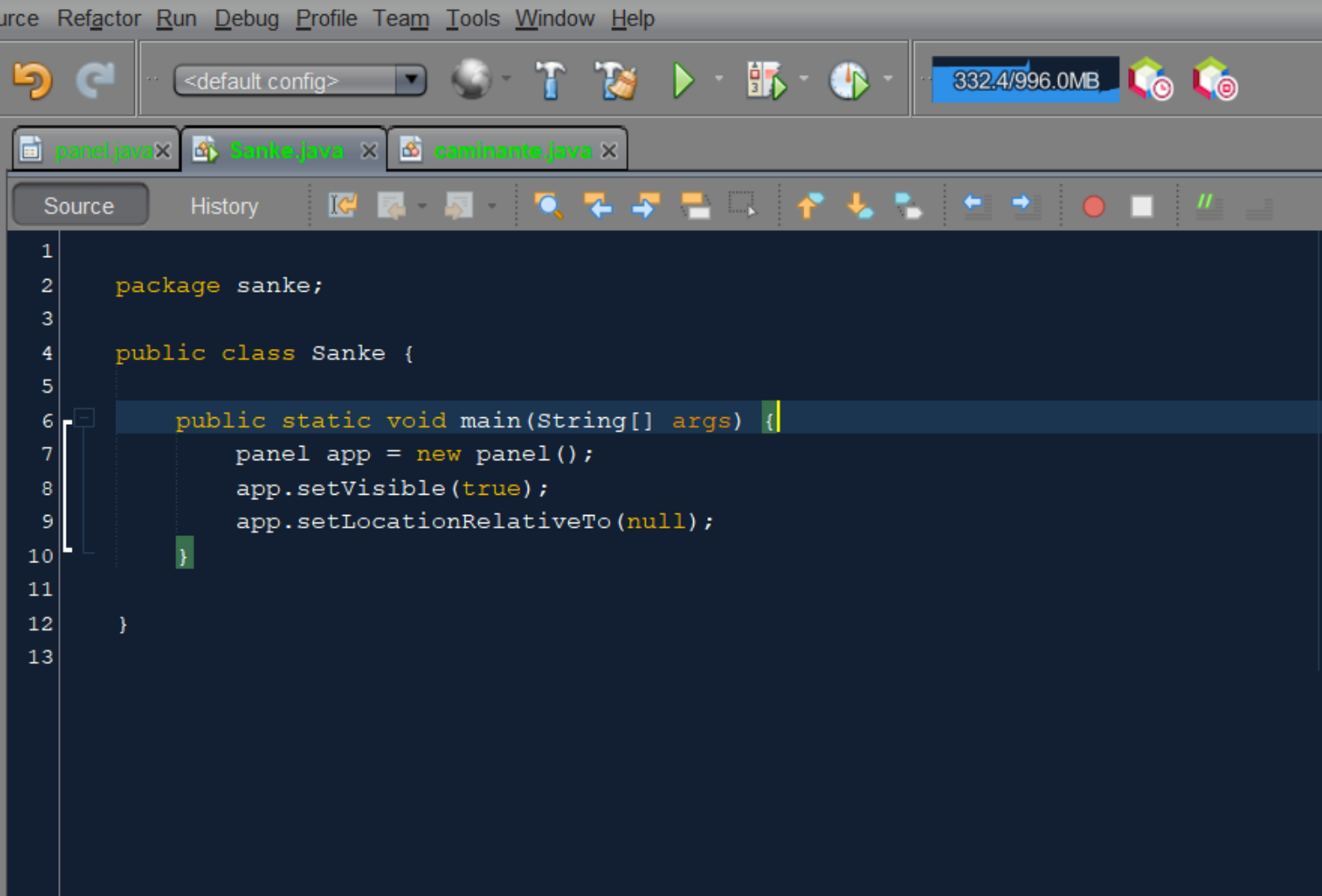
Segundo semestre 2022

Practica 2

Clase main

La clase main cuenta con un set visible para poder mostrar el panel donde se encuentra el juego y un set location relative to null para poder centrar la interfaz grafica.

DE 15

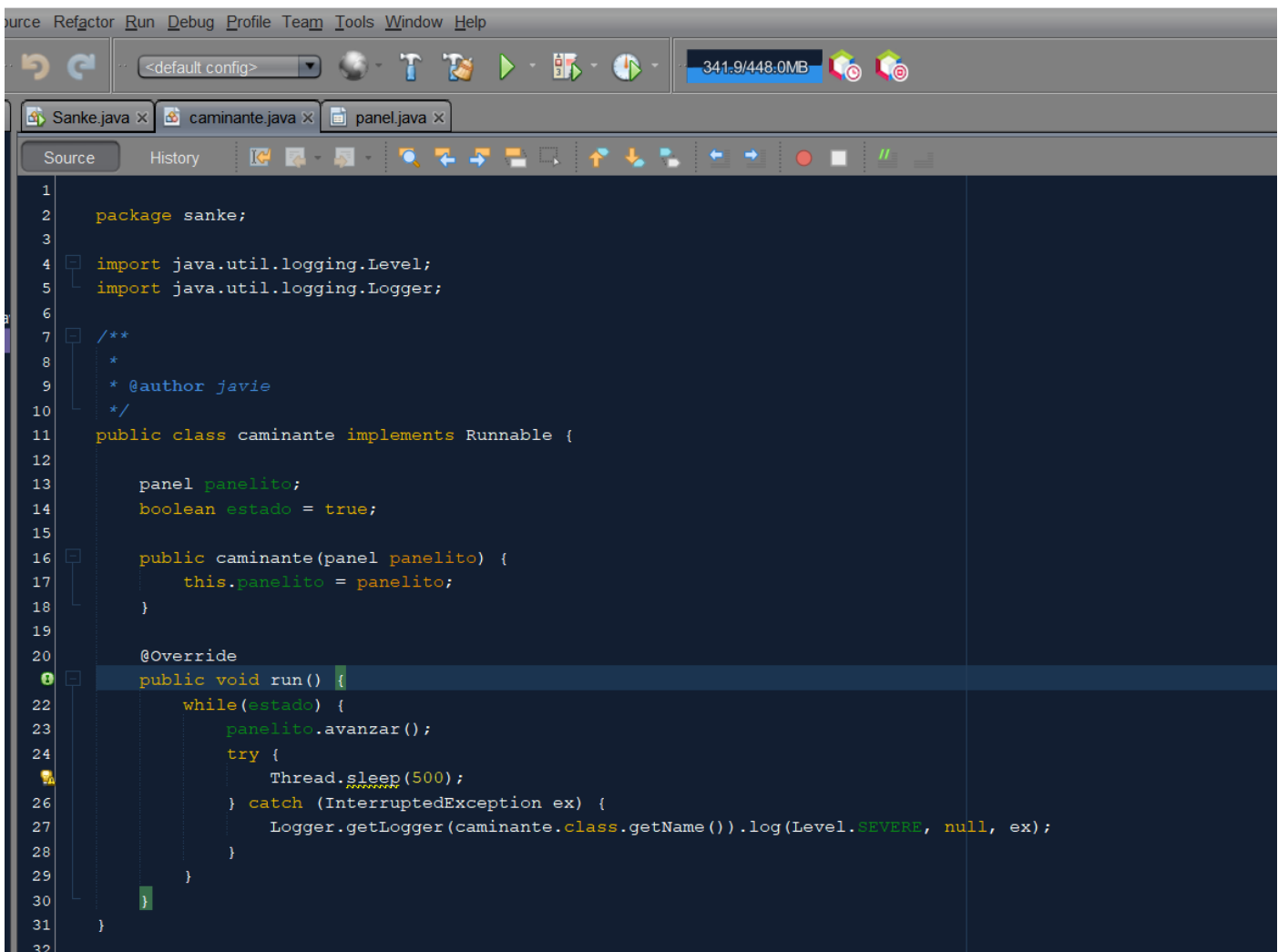
A screenshot of an IDE window showing the code for the Sanke class. The window has a menu bar with 'Source', 'Refactor', 'Run', 'Debug', 'Profile', 'Team', 'Tools', 'Window', and 'Help'. Below the menu bar is a toolbar with various icons, including a configuration dropdown set to '<default config>', a memory usage indicator showing '332.4/996.0MB', and several icons for running and debugging. The editor area shows three tabs: 'panel.java', 'Sanke.java', and 'caminante.java'. The 'Sanke.java' tab is active, displaying the following code:

```
1 package sanke;
2
3
4 public class Sanke {
5
6     public static void main(String[] args) {
7         panel app = new panel();
8         app.setVisible(true);
9         app.setLocationRelativeTo(null);
10    }
11
12 }
13
```

Movimientos y funciones

Para la parte de el movimiento de la serpiente siempre trasladamos la información a las diferentes clases, por medio de panel, para avanzar la serpiente usamos un hilo con un sleep para que sea notorio el movimiento que ira haciendo.

IDE 15

A screenshot of an IDE window titled 'IDE 15'. The interface includes a menu bar (Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help), a toolbar with icons for undo, redo, run, and other IDE functions, and a tab bar showing three open files: 'Sanke.java', 'caminante.java', and 'panel.java'. The 'caminante.java' file is active, displaying the following Java code:

```
1 package sanke;
2
3
4 import java.util.logging.Level;
5 import java.util.logging.Logger;
6
7 /**
8  *
9  * @author javie
10  */
11 public class caminante implements Runnable {
12
13     panel panelito;
14     boolean estado = true;
15
16     public caminante(panel panelito) {
17         this.panelito = panelito;
18     }
19
20     @Override
21     public void run() {
22         while(estado) {
23             panelito.avanzar();
24             try {
25                 Thread.sleep(500);
26             } catch (InterruptedException ex) {
27                 Logger.getLogger(caminante.class.getName()).log(Level.SEVERE, null, ex);
28             }
29         }
30     }
31 }
32
```

Código de la serpiente

Tenemos diferentes tipos de variables para guardar posiciones de la matriz de jables y otra para guardar la velocidad de la serpiente.

```
16 //jlabel -> botones, txt
17 JLabel[][] cuadriculaJuego;
18
19 String direccion = "der";
20
21 // hilos
22 private Thread hiloRepintar;
23
24 //borde del tablero
25 int izquierda = 0;
26 int derecha = 10;
27
28 //
29 int x = 1;
30 int y = 1;
31
32 //posicion inicial del snake
33 int snakeX = 5;
34 int snakeY = 5;
35
36 //fruta
37 int frutax = 1;
38 int frutay = 1;
39
40 // velocidad del snake
41 int velocidad = 1000;
42
43 int contadorfrutas = 0;
44
45 /**
46  * Creates new form panel
47  */
48 public panel() {
49     initComponents();
50     snakeGame();
51 }
```

pintamos la matriz para lo cual utilizamos dos for y hacemos visible la serpiente frente a la matriz para ello hacemos opaquamos el panel

```
52
53 public void snakeGame() {
54
55     // visibilidad del panel
56     this.jPanell.setVisible(true);
57
58     //matriz del juego
59     cuadrículaJuego = new JLabel[10][10];
60
61     // un doble for para manejar la matriz
62     for (int i = 0; i < cuadrículaJuego.length; i++) {
63         for (int j = 0; j < cuadrículaJuego.length; j++) {
64
65             //poder crear un label en esa poscion
66             cuadrículaJuego[i][j] = new JLabel();
67             cuadrículaJuego[i][j].setOpaque(true);
68
69             cuadrículaJuego[i][j].setBackground(Color.white);
70
71             // posicion x y posicion y
72             // tamaño de cada cuadro
73             cuadrículaJuego[i][j].setBounds(x, y, 50, 50);
74
75             x = x + 51;
76             this.jPanell.add(cuadrículaJuego[i][j]);
77         }
78         y = y + 51;
79         x = 1;
80     }
81
82     // pintar el snake
83     cuadrículaJuego[snakeX][snakeY].setBackground(Color.red);
84
85     // 1,1 de color verde
86     cuadrículaJuego[frutaX][frutaY].setBackground(Color.green);
87
```

para el movimiento utilizamos ifs para validar el botón seleccionado para correspondientemente cambiar la dirección de la serpiente

```
90
91 // funcion de mover el snake
92 public void mover() {
93     while (true) {
94         this.requestFocus(true);
95         if (this.direccion.equals("arr")) {
96             snakeX--;
97             //pintar
98             cuadrículaJuego[snakeX][snakeY].setBackground(Color.red);
99             //despintar
100            cuadrículaJuego[snakeX+1][snakeY].setBackground(Color.white);
101        } else if (this.direccion.equals("aba")) {
102            snakeX++;
103            //pintar
104            cuadrículaJuego[snakeX][snakeY].setBackground(Color.red);
105            //despintar
106            cuadrículaJuego[snakeX-1][snakeY].setBackground(Color.white);
107        } else if (this.direccion.equals("der")) {
108            snakeY++;
109            //pintar
110            cuadrículaJuego[snakeX][snakeY].setBackground(Color.red);
111            //despintar
112            cuadrículaJuego[snakeX][snakeY - 1].setBackground(Color.white);
113        } else if (this.direccion.equals("izq")) {
114            snakeY--;
115            //pintar
116            cuadrículaJuego[snakeX][snakeY].setBackground(Color.red);
117            //despintar
118            cuadrículaJuego[snakeX][snakeY + 1].setBackground(Color.white);
119        }
120
121        for (int i = 0; i < cuadrículaJuego.length; i++) {
122            for (int j = 0; j < cuadrículaJuego.length; j++) {
123                if (cuadrículaJuego[i][9]==cuadrículaJuego[snakeX][snakeY] || cuadrículaJuego[j][9]==cuadricu
124                JOptionPane.showMessageDialog(null, "Hubo un choque");
125            }
126        }
127    }
128 }
```

para mover la serpiente utilizaremos otro hilo y creamos eventos a la hora de pulsar un botón en el panel para cambiar la dirección de la serpiente

```
145         try {
146             Thread.sleep(velocidad);
147         } catch (InterruptedException ex) {
148             Logger.getLogger(panel.class.getName()).log(Level.SEVERE, null, ex);
149         }
150     }
151 }
152
153 public void avanzar() {
154 }
155
156 /**
157  * This method is called from within the constructor to initialize the form.
158  * WARNING: Do NOT modify this code. The content of this method is always
159  * regenerated by the Form Editor.
160  */
161 @SuppressWarnings("unchecked")
162 Generated Code
163
164 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
165
166     hiloRepintar = new Thread(() -> {
167         this.mover();
168     });
169     hiloRepintar.start();
170 }
171
172 private void formKeyPressed(java.awt.event.KeyEvent evt) {
173     System.out.println("hola");
174 }
175
176 private void btt_ArribaActionPerformed(java.awt.event.ActionEvent evt) {
177     this.requestFocus(true);
178     this.direccion = "arr";
179     historial.add("Preciono ARRIBA");
180 }
```

cada botón tendrá la misma función de poder cambiar la dirección de la serpiente y utilizaremos focus para que el programa se centre en la instrucción correspondiente

```
329 private void btt_AbajoActionPerformed(java.awt.event.ActionEvent evt) {  
331     this.requestFocus(true);  
332     historial.add("Preciono ABAJO");  
333     this.direccion = "aba";  
334 }  
335  
336 private void btt_DerechaActionPerformed(java.awt.event.ActionEvent evt) {  
337     this.requestFocus(true);  
338     historial.add("Preciono DERECHA");  
339     this.direccion = "der";  
340 }  
341  
342 private void btt_IzquierdaActionPerformed(java.awt.event.ActionEvent evt) {  
343     this.requestFocus(true);  
344     historial.add("Preciono IZQUIERDA");  
345     this.direccion = "izq";  
346 }  
347  
348 private void btt_HistorialActionPerformed(java.awt.event.ActionEvent evt) {  
349     for (String cadena : historial) {  
350         System.out.println(cadena);  
351     }  
352 }  
353  
354 /**  
355  * @param args the command line arguments  
356  */  
357 // public static void main(String args[]) {  
358 //     /* Set the Nimbus look and feel */  
359 //     Look and feel setting code (optional)  
360 //  
361 //     /* Create and display the form */  
362 //     java.awt.EventQueue.invokeLater(new Runnable() {  
363 //         public void run() {  
364 //             new panel().setVisible(true);  
365 //         }  
366 //     });  
367 // }  
368
```