

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
ESCUELA DE CIENCIAS Y SISTEMAS
INTRODUCCIÓN A LA PROGRAMACIÓN Y COMPUTACIÓN 1
SECCIÓN A
SEGUNDO SEMESTRE 2022
ING. MARLON FRANCISCO ORELLANA LOPEZ
AUX. DIEGO ALEJANDRO VASQUEZ



BRANDON EDUARDO PABLO GARCIA
202112092
Guatemala, septiembre del 2022

CONTENIDO

Introducción.....	1
Objetivos.....	2
Contenido Técnico.....	3
Inicio.....	3
Desarrollo.....	13
Pruebas.....	14

INTRODUCCION

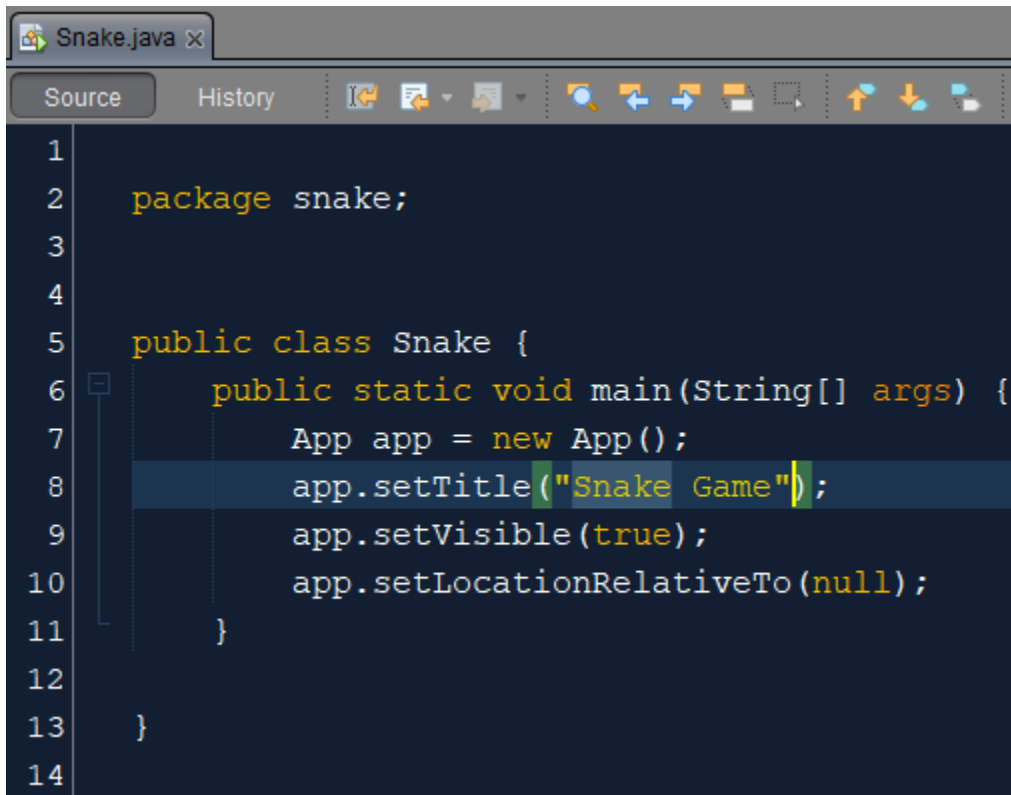
Este manual describe los pasos necesarios para cualquier persona que tenga ciertas bases de sistemas pueda realizar el código implementado en Java NetBens donde se crea un código del juego de la serpiente a través de botones y así poder implementarlo de la mejor manera. El siguiente código se explicó de la manera más detalla posible para la mejor comprensión de la persona.

OBJETIVOS

- Brindar la información necesaria para poder representar la funcionalidad técnica de la estructura, diseño y definición del aplicativo.
- Describir las herramientas utilizadas para el diseño y desarrollo del prototipo

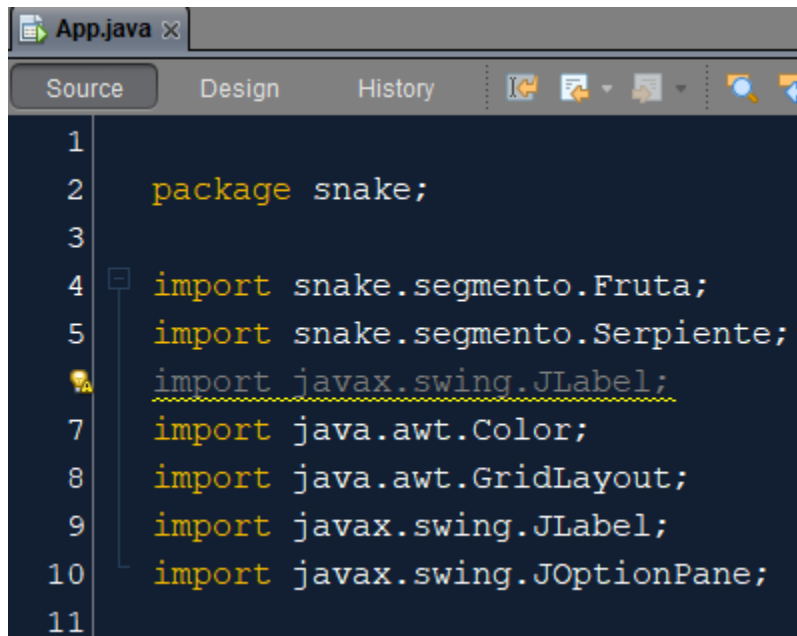
CONTENIDO TECNICO

Comienza en la clase principal en donde se declaro la otra venta como una variable para poder tener de manera mas directa cuando se utilizan los métodos set que nos ayudaron a colocar título a la ventana emergente “setTitle” y a la ubicación de la pantalla “setLocation”.



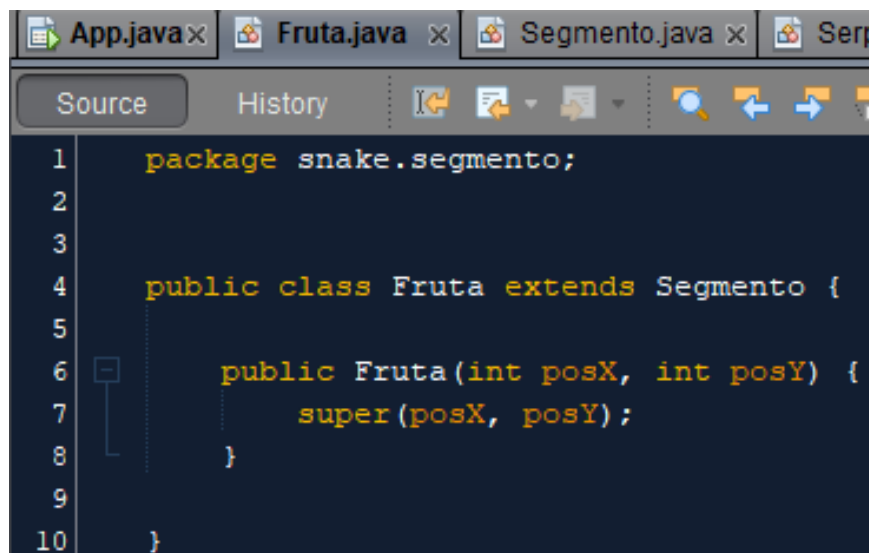
```
1
2  package snake;
3
4
5  public class Snake {
6      public static void main(String[] args) {
7          App app = new App();
8          app.setTitle("Snake Game");
9          app.setVisible(true);
10         app.setLocationRelativeTo(null);
11     }
12
13 }
14
```

En la otra pestaña del JFrame llamada App podemos encontrar todo lo que conlleva la interfaz grafica del juego, para comenzar tuvimos que importar todas las librerías que usaremos como podemos observar a continuación, adicional a eso tuvimos que crear ciertas clases para la parte de las frutas, los segmentos de la matriz y la serpiente, el en cual se utilizo POO (Programación Orientada a Objetos) y se mandaron a importar con el nombre “snake.segmento,Fruta” para las frutas y “snake.segmento.Serpiente” para la serpiente.



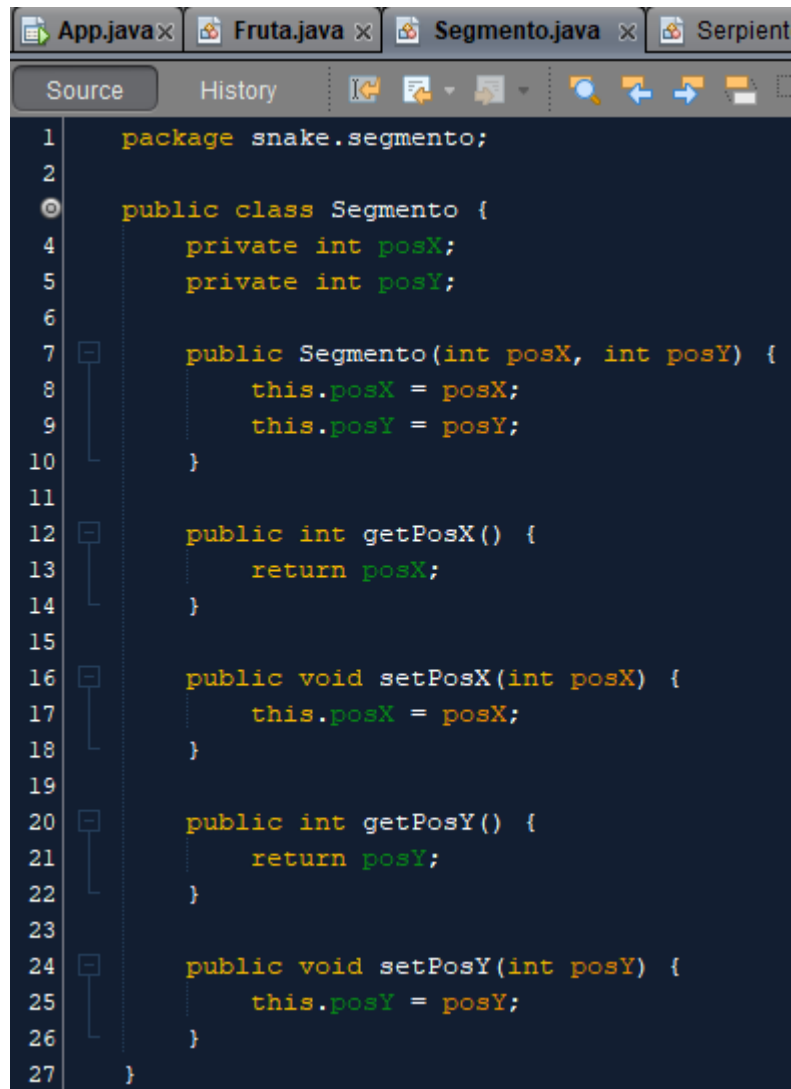
```
1
2  package snake;
3
4  import snake.segmento.Fruta;
5  import snake.segmento.Serpiente;
6  import javax.swing.JLabel;
7  import java.awt.Color;
8  import java.awt.GridLayout;
9  import javax.swing.JLabel;
10 import javax.swing.JOptionPane;
11
```

En la parte de las frutas se colocaron las posiciones donde aparecerá dentro de la matriz que se utilizará.



```
1  package snake.segmento;
2
3
4  public class Fruta extends Segmento {
5
6      public Fruta(int posX, int posY) {
7          super(posX, posY);
8      }
9
10 }
```

Esta pestaña se utilizó para colocar cada segmento y la posición que tendrá en un plano de dos dimensiones utilizando el nombre de variable “posX” y “posy” y declararlos como un int en una clase publica y así poder retornarla con la opción “return”.



```
1 package snake.segmento;
2
3 public class Segmento {
4     private int posX;
5     private int posY;
6
7     public Segmento(int posX, int posY) {
8         this.posX = posX;
9         this.posY = posY;
10    }
11
12    public int getPosX() {
13        return posX;
14    }
15
16    public void setPosX(int posX) {
17        this.posX = posX;
18    }
19
20    public int getPosY() {
21        return posY;
22    }
23
24    public void setPosY(int posY) {
25        this.posY = posY;
26    }
27 }
```

Se creo una clase publica llamada serpiente donde se declaro su velocidad, el largo y su vida a lo largo de todo el juego para eso se utilizó un volatile int que nos ayudo a designarle al segmento un largo de 25 al igual que el ancho cuanto esté vivo.

Para eso fueron necesaria usa POO y crear variables públicas con las palabras mencionadas anteriormente. También se designo la movilidad de cada parte usando las posiciones en X y Y para que se pueda mover en la parte llamada segmentos junto con su largo y sus segmentos.

```
App.java x Serpiente.java x
Source History
1 package snake.segmento;
2
3 public class Serpiente {
4
5     private final Segmento[] segmentos;
6     private volatile int velocidad;
7     private volatile int largo;
8     private volatile boolean viva;
9
10    public Serpiente() {
11        segmentos = new Segmento[25];
12        segmentos[0] = new Segmento(5, 5);
13        largo = 1;
14        viva = true;
15    }
16
17    public Segmento[] getSegmentos() {
18        return segmentos;
19    }
20
21    public int getVelocidad() {
22        return velocidad;
23    }
24
25    public void setVelocidad(int velocidad) {
26        this.velocidad = velocidad;
27    }
```

```
App.java x Serpiente.java x
Source History
28
29    public void moverArriba() {
30        moverSegmentos();
31        segmentos[0].setPosY(segmentos[0].getPosY() - 1);
32    }
33
34    public void moverAbajo() {
35        moverSegmentos();
36        segmentos[0].setPosY(segmentos[0].getPosY() + 1);
37    }
38
39    public void moverIzquierda() {
40        moverSegmentos();
41        segmentos[0].setPosX(segmentos[0].getPosX() - 1);
42    }
43
44    public void moverDerecha() {
45        moverSegmentos();
46        segmentos[0].setPosX(segmentos[0].getPosX() + 1);
47    }
48
49    public void moverSegmentos() {
50        for (int i = largo - 1; i > 0; i--) {
51            segmentos[i].setPosY(segmentos[i - 1].getPosY());
52            segmentos[i].setPosX(segmentos[i - 1].getPosX());
53        }
54    }
```



```

48
49     public void moverSegmentos() {
50         for (int i = largo - 1; i > 0; i--) {
51             segmentos[i].setPosY(segmentos[i - 1].getPosY());
52             segmentos[i].setPosX(segmentos[i - 1].getPosX());
53         }
54     }
55
56     public int getLargo() {
57         return largo;
58     }
59
60     public boolean puedeCrecer() {
61         return largo < segmentos.length;
62     }
63
64     public boolean isViva() {
65         return viva;
66     }
67
68     public void setViva(boolean viva) {
69         this.viva = viva;
70     }
71
72 }

```

Continuamente de bajo de importar las clases se utilizó el método `javax.swing.JFrame` para poder declarar variables y así poder llamar las variables que se utilizaron en las demás pestañas para así poder crear la serpiente y lo que conlleva.

```

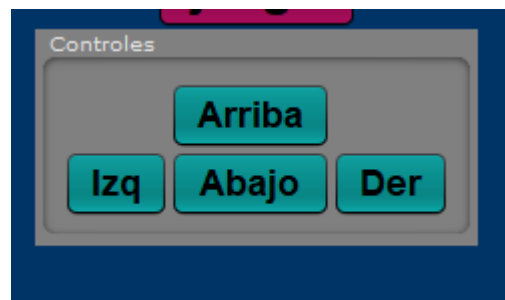
12
13     public class App extends javax.swing.JFrame {
14
15         private JLabel[][] cuadrículaGamePane;
16         private Serpiente serpiente;
17         private Fruta fruta;
18         private Thread hiloRepintar;
19         private Thread hiloMoverSerpiente;
20         private volatile int direccion;
21         private volatile int velocidad = 1000;
22         private volatile int modificadorDeDificultad;
23         private volatile boolean juegoActivo = false;
24         private final int ARRIBA = 0;
25         private final int ABAJO = 1;
26         private final int DERECHA = 2;
27         private final int IZQUIERDA = 3;
28
29         public App() {
30             initComponents();
31         }
32     }

```

Se le colocó un nombre de variable a cada botón y poder indicarle que acción realizara con las variables que declaramos como lo son las direcciones, esos movimientos serán los que realizara la serpiente dentro del cuadro.

```
32  @SuppressWarnings("unchecked")
33  Generated Code
227
228  private void formKeyPressed(java.awt.event.KeyEvent evt) {
229
230  }
231
232  private void btnArribaActionPerformed(java.awt.event.ActionEvent evt) {
233      direccion = ARRIBA;
234  }
235
236  private void btnAbajoActionPerformed(java.awt.event.ActionEvent evt) {
237      direccion = ABAJO;
238  }
239
240  private void btnDerechaActionPerformed(java.awt.event.ActionEvent evt) {
241      direccion = DERECHA;
242  }
243
244  private void btnIzquierdaActionPerformed(java.awt.event.ActionEvent evt) {
245      direccion = IZQUIERDA;
246  }
```

Gráficamente se miran de la siguiente manera.



En el botón mas importante de todos es el botón que sirve para comenzar toda la acción que hace que aparezca las frutas y la serpiente.

Aquí se utilizó un if para hacer que la serpiente se pueda mover y así poder indicarle a donde ir, también aparecerá la velocidad que se le indica para usar un hilo para que se pueda ir repintando durante todo el transcurso del juego y así poder usar la matriz que es lo principal del código

```

App.java x
Source Design History
private void btnIniciarActionPerformed(java.awt.event.ActionEvent evt) {
249     if (this.hiloMoverSerpiente != null && this.hiloMoverSerpiente.isAlive() || this.hiloRepintar != null && this.hiloRepintar.isAlive()) {
250         return;
251     }
252     GridLayout gamePanelLayout = (GridLayout) this.gamePanel.getLayout();
253     int cantidadCol = 10;
254     int cantidadFilas = 10;
255     serpiente = new Serpiente();
256     gamePanelLayout.setColumns(cantidadCol);
257     gamePanelLayout.setRows(cantidadFilas);
258     cuadrriculaGamePane = new JLabel[cantidadFilas][cantidadCol];
259     gamePanel.removeAll();
260     for (int i = 0; i < cuadrriculaGamePane.length; i++) {
261         for (int j = 0; j < cuadrriculaGamePane[i].length; j++) {
262             cuadrriculaGamePane[i][j] = new JLabel();
263             cuadrriculaGamePane[i][j].setOpaque(true);
264             cuadrriculaGamePane[i][j].setBackground(Color.white);
265             gamePanel.add(cuadrriculaGamePane[i][j]);
266         }
267     }
268     modificadorDeDificultad = this.Dificultad.getSelectedIndex() + 1;
269     velocidad = 1000;
270     this.lbltamanio.setText(String.valueOf(serpiente.getLargo()));
271     this.lblvelocidad.setText(String.valueOf(velocidad));
272     nuevaFruta();
273     gamePanel.validate();
274     juegoActivo = true;
275     direccion = -1;
276     hiloRepintar = new Thread(() -> {
277         repintar();
278     });
279     hiloRepintar.start();
280     hiloMoverSerpiente = new Thread(() -> {
281         mover();
282     });
283     hiloMoverSerpiente.start();
}

```

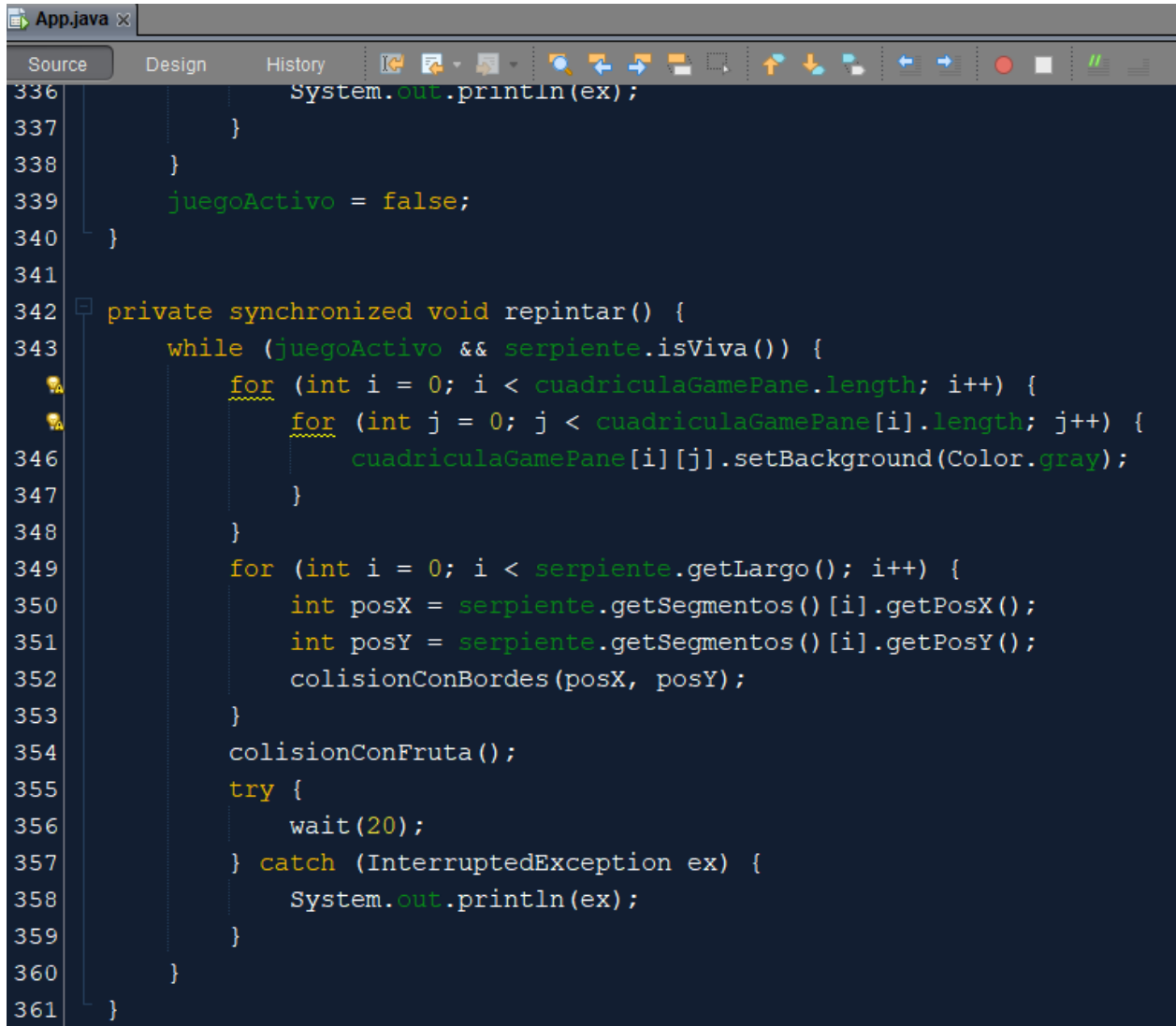
Aquí utilizando el método void mover se le indica la dirección de la serpiente para hacer esto posible se utilizó un while junto con un switch y así poder designar que movimiento realizara la serpiente dentro del cuadro junto con un try que nos indicara la velocidad del mismo.

```

App.java x
Source Design History
321 private synchronized void mover() {
322     while (juegoActivo && serpiente.puedeCrecer()) {
323         switch (direccion) {
324             case IZQUIERDA ->
325                 serpiente.moverIzquierda();
326             case ARRIBA ->
327                 serpiente.moverArriba();
328             case DERECHA ->
329                 serpiente.moverDerecha();
330             case ABAJO ->
331                 serpiente.moverAbajo();
332         }
333         try {
334             wait(velocidad);
335         } catch (InterruptedException ex) {
336             System.out.println(ex);
337         }
338     }
}

```

Aquí se mostrará cuando el juego ya este activo se utilizó el void de nuevo nombrado de la manera repintar que hará que se muestre la cuadrícula la cual será de color gris a la misma vez que la posición x y de la serpiente en cada segmento de la matriz donde tendrá la colisión con la fruta que comerá.



```
336         System.out.println(ex);
337     }
338 }
339 juegoActivo = false;
340 }
341
342 private synchronized void repintar() {
343     while (juegoActivo && serpiente.isViva()) {
344         for (int i = 0; i < cuadrículaGamePane.length; i++) {
345             for (int j = 0; j < cuadrículaGamePane[i].length; j++) {
346                 cuadrículaGamePane[i][j].setBackground(Color.gray);
347             }
348         }
349         for (int i = 0; i < serpiente.getLargo(); i++) {
350             int posX = serpiente.getSegmentos()[i].getPosX();
351             int posY = serpiente.getSegmentos()[i].getPosY();
352             colisionConBordes(posX, posY);
353         }
354         colisionConFruta();
355         try {
356             wait(20);
357         } catch (InterruptedException ex) {
358             System.out.println(ex);
359         }
360     }
361 }
```

Usando un void llamado colisionConBordes se realizó la parte que sucede cuando la serpiente choca contra la pared para eso se usó un if junto con un else para ver si cumple la condición de choque en la posición X y Y del plano, de lo contrario si llegar a chocar se usó el método JOptionPane que mostrara un mensaje al usuario diciéndole que la serpiente ha muerto.

```
App.java x
Source Design History
359     }
360 }
361
362
363 private void colisionConBordes(int posX, int posY) {
364     if (posY >= 0 && posX >= 0 && posY < cuadrículaGamePane.length && posX < cuadrículaGamePane[0].length) {
365         cuadrículaGamePane[posY][posX].setBackground(Color.green);
366     } else {
367         serpiente.setViva(false);
368         juegoActivo = false;
369         JOptionPane.showMessageDialog(null, "La serpiente ha muerto", this.getTitle(), JOptionPane.INFORMATION_MESSAGE);
370     }
371 }
```

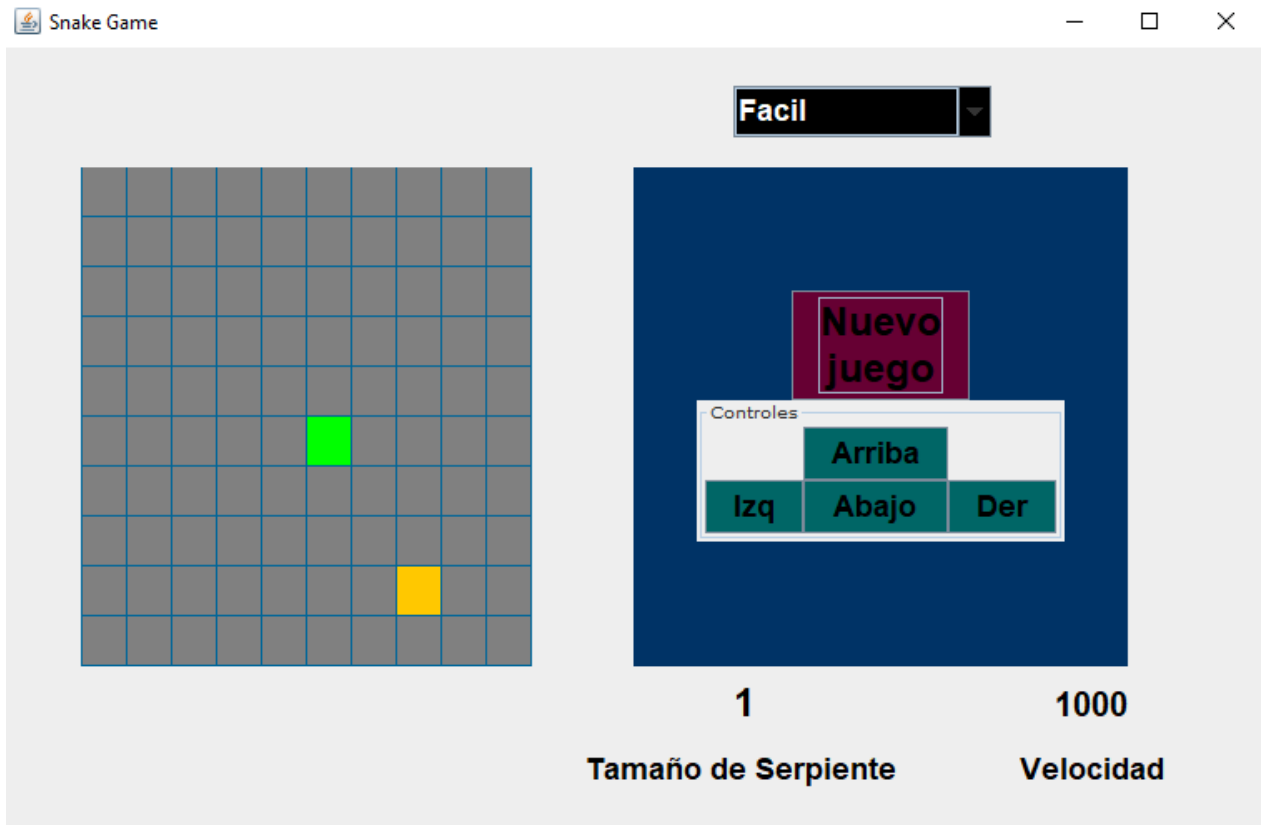
De la misma manera que los demás se usó el método void con el nombre colisionConFruta donde se declararon las posiciones de las frutas a la misma manera que la serpiente para que no salga fuera del plano y así poder estar dentro de los segmentos designados, para eso se usó el método get y lo demás que necesitamos. La fruta será mostrada y para eso se llamó al panel denominado cuadrículaGamePane y la posición de la fruta.

```
App.java x
Source Design History
371     }
372 private void colisionConFruta() {
373     int posXFruta = fruta.getPosX();
374     int posYFruta = fruta.getPosY();
375
376     int posXCabeza = serpiente.getSegmentos()[0].getPosX();
377     int posYCabeza = serpiente.getSegmentos()[0].getPosY();
378
379     cuadrículaGamePane[posYFruta][posXFruta].setBackground(Color.orange);
380 }
```

De la misma manera que los demás se usó el método void con el nombre nuevaFruta donde se declararon las posiciones de las frutas a la misma manera que la serpiente que aparecerá después de ser comida por la serpiente, para que dicha fruta apareciera se utilizó el método random para que aparezca dentro del cuadro a lo largo de las posiciones X y Y.

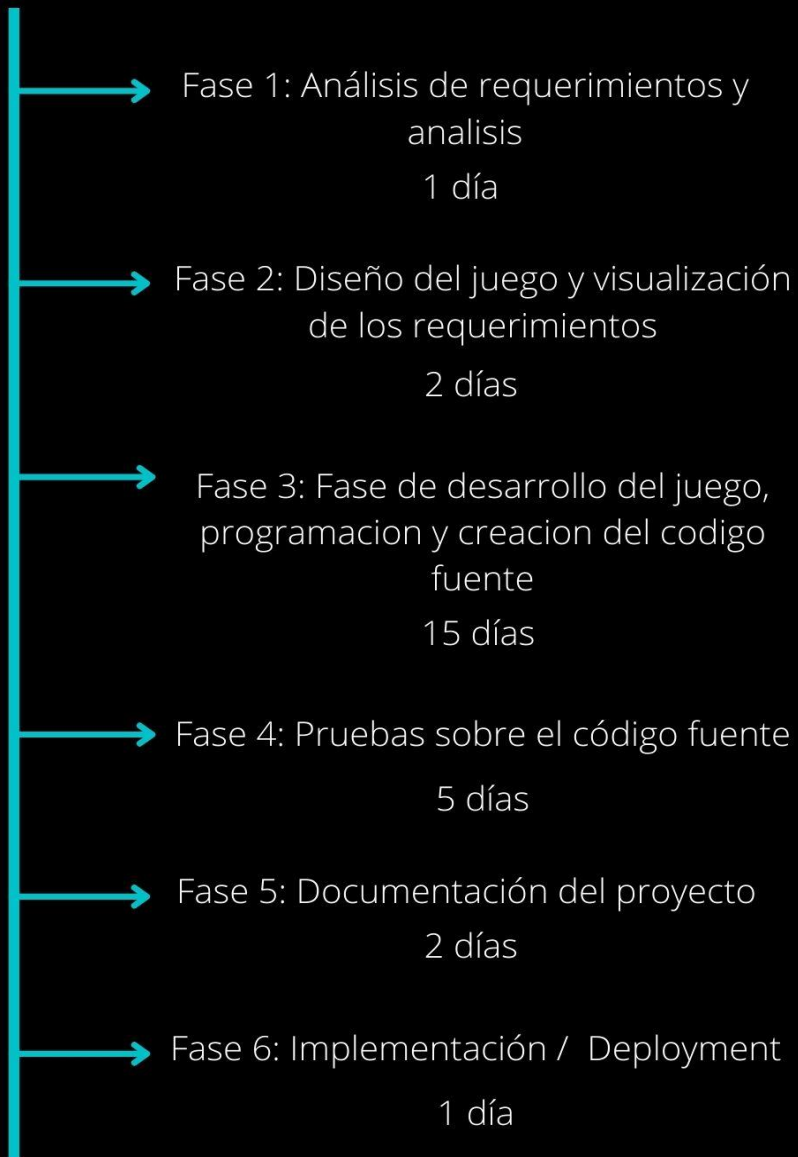
```
App.java x
Source Design History
380     }
381     private void nuevaFruta() {
382         int largoY = cuadrículaGamePane.length - 1;
383         int largoX = cuadrículaGamePane[0].length - 1;
384
385         int posX = (int) Math.round(Math.random() * largoX);
386         int posY = (int) Math.round(Math.random() * largoY);
387
388         fruta = new Fruta(posX, posY);
389     }
```

Para culminar con un interfaz grafica como se le muestra a continuación.



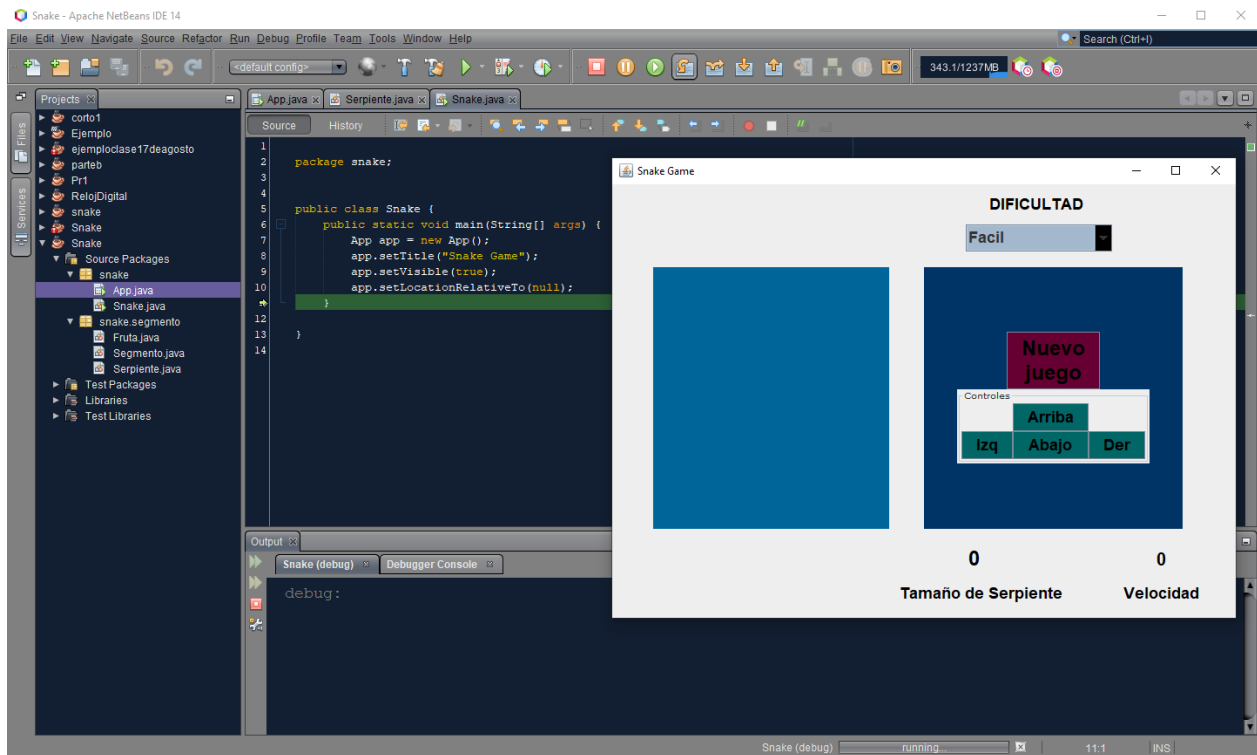
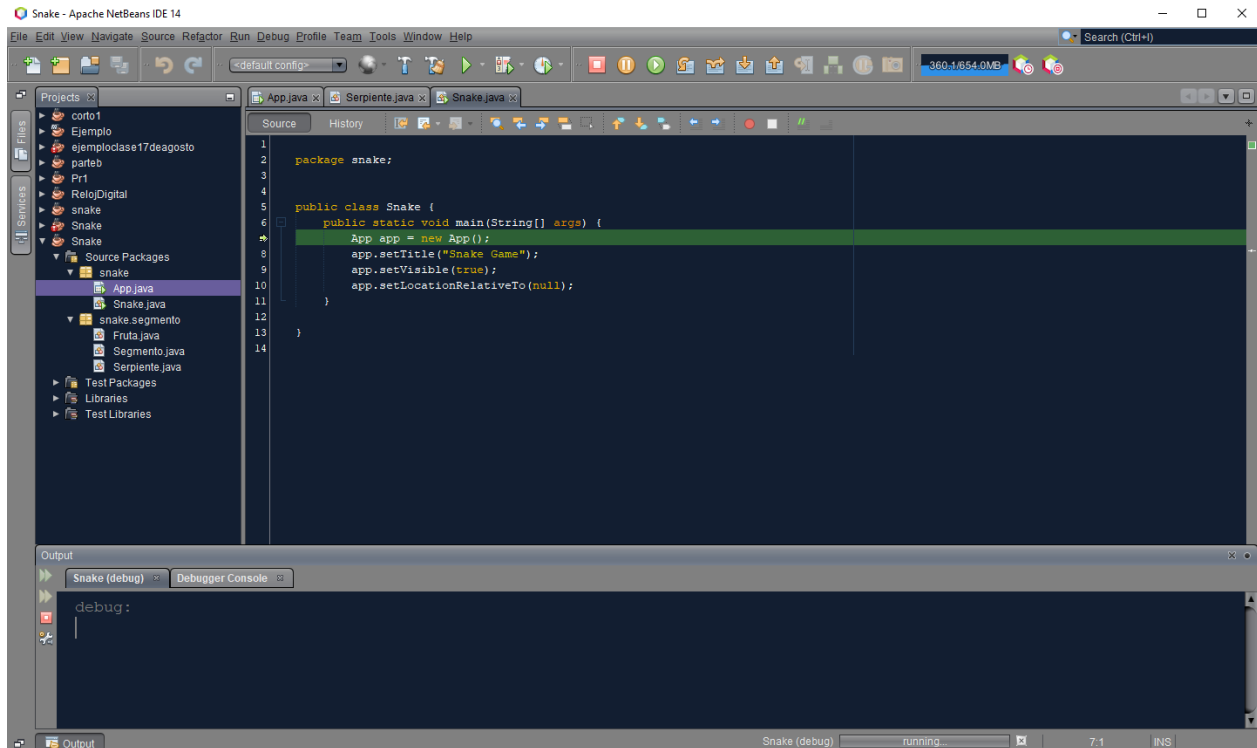
Agenda de proyecto

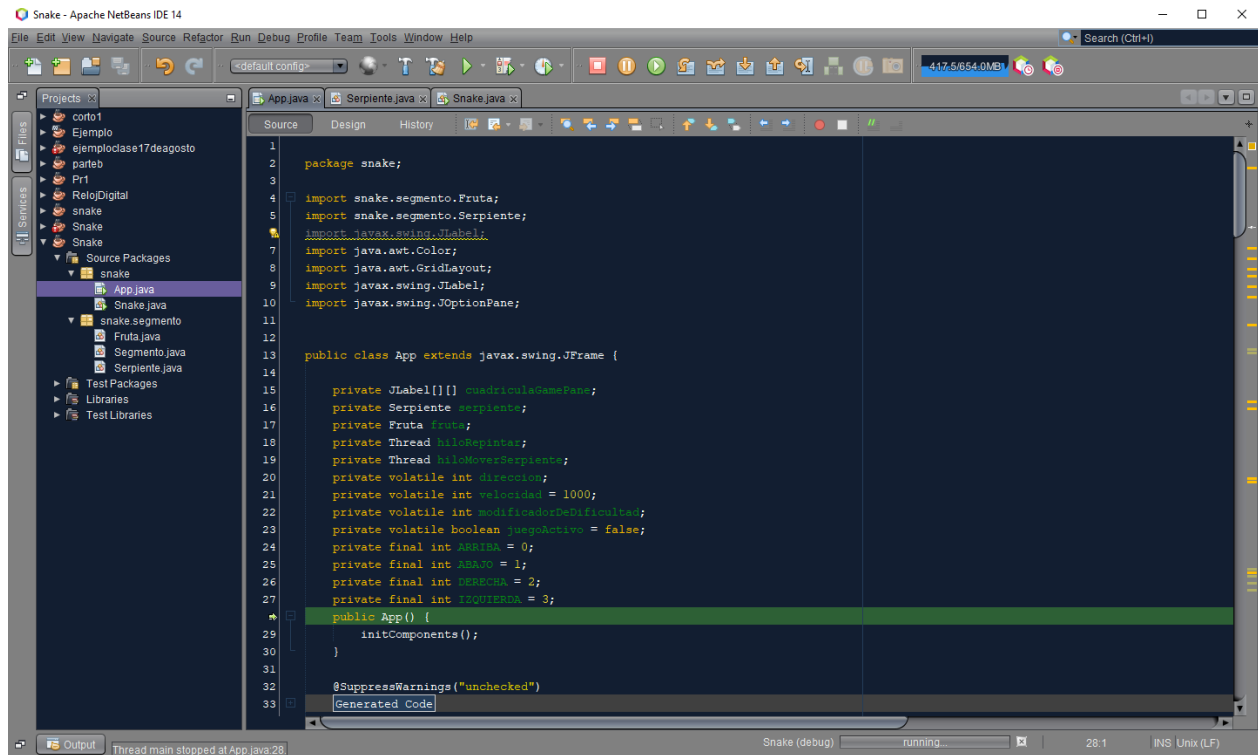
PRACTICA 2 SNAKE



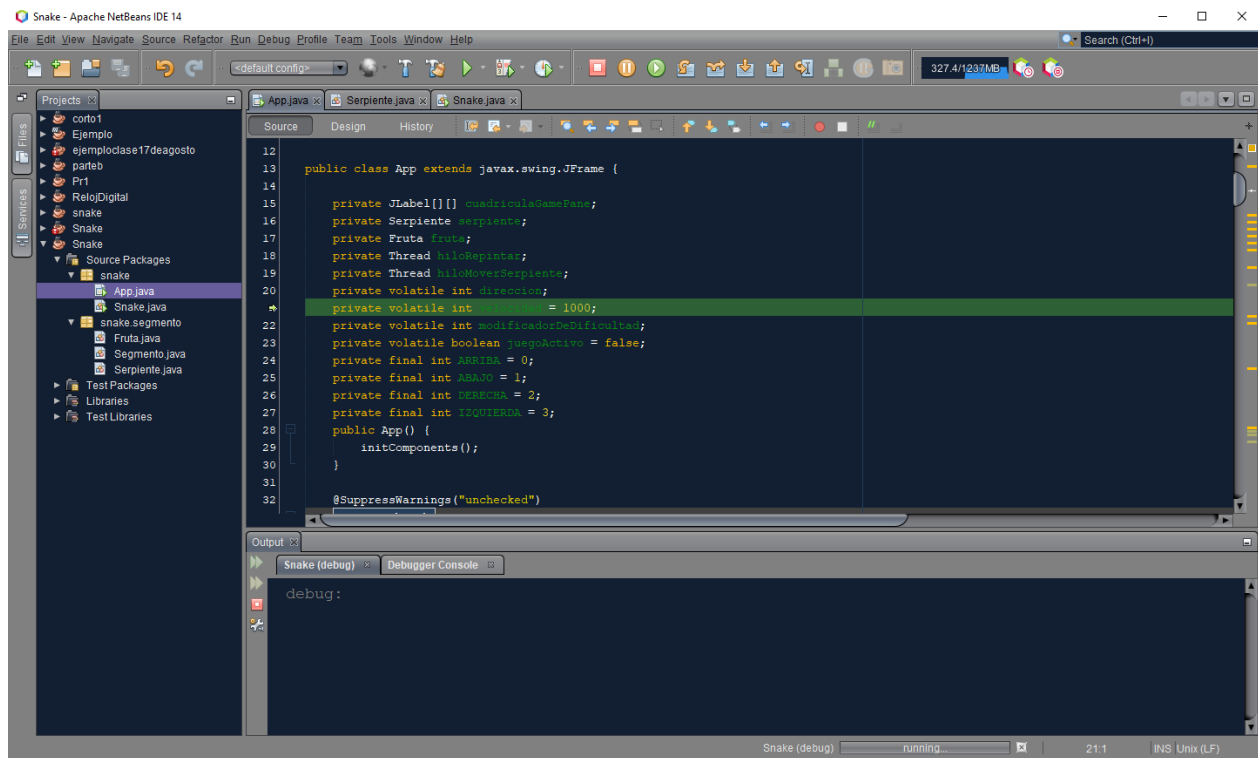
PRUEBAS

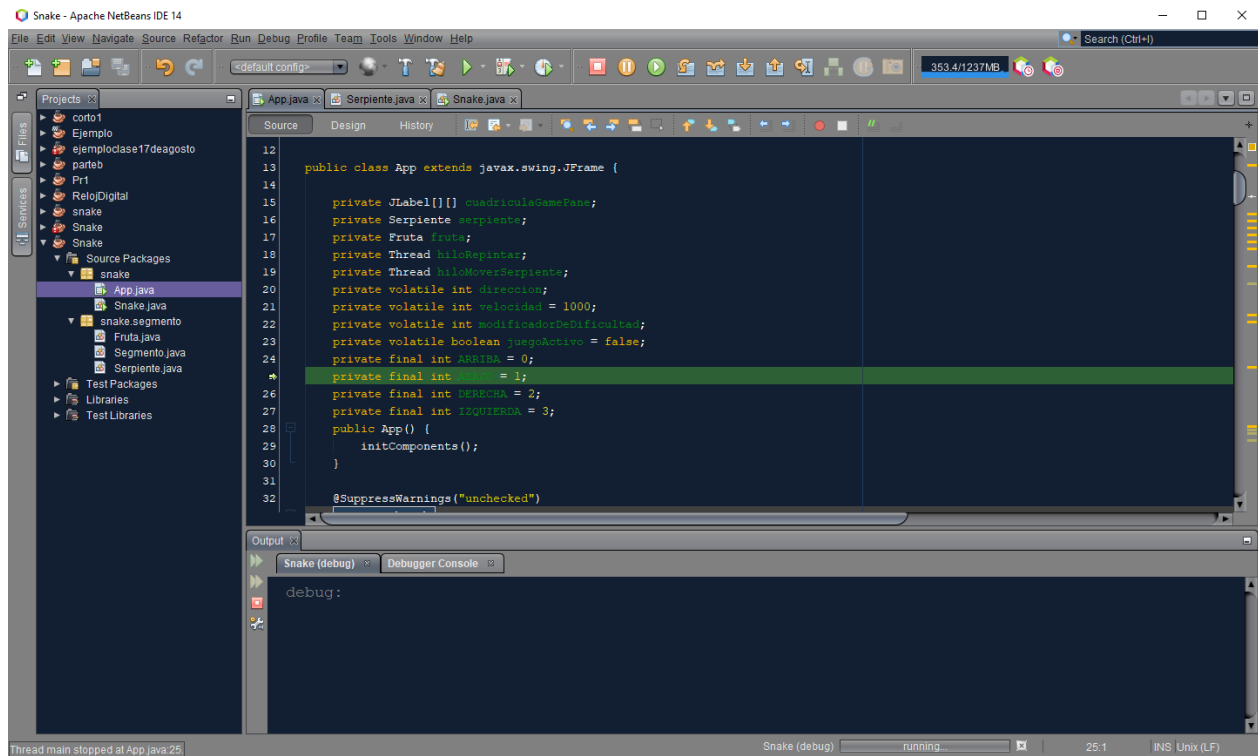
Se verifico si entra correcta a la venta de la App.



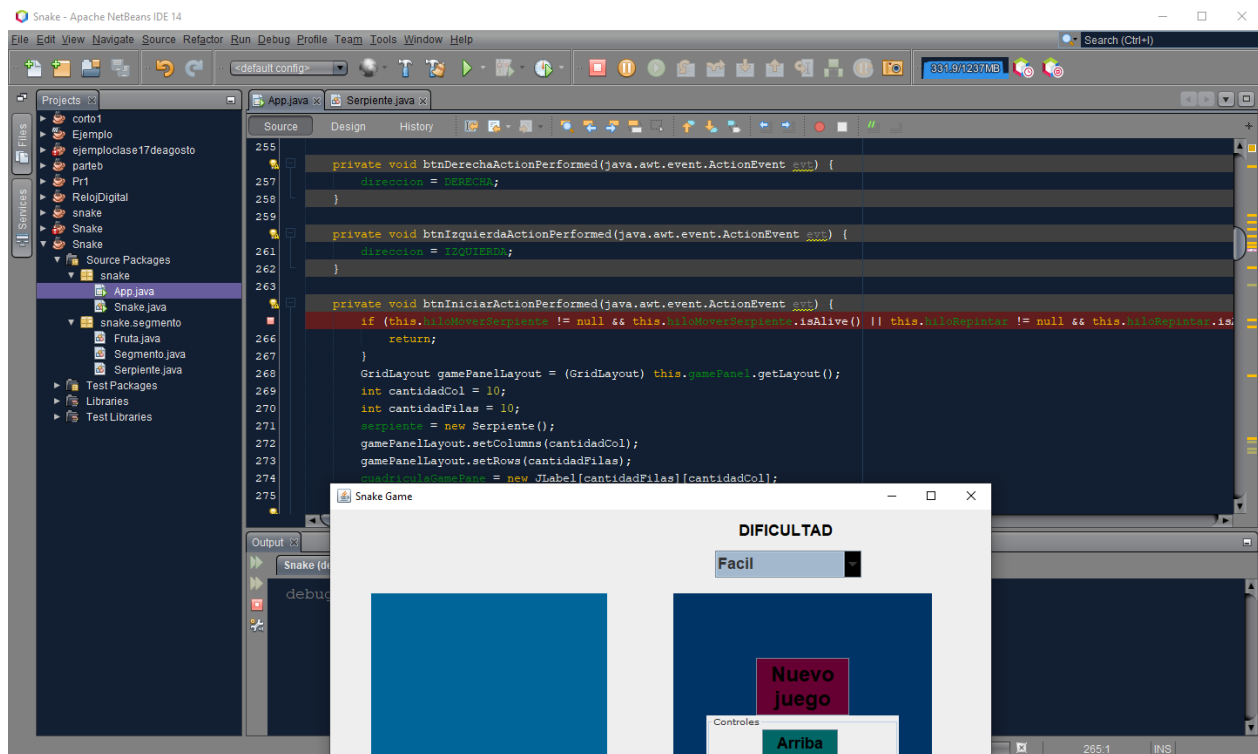


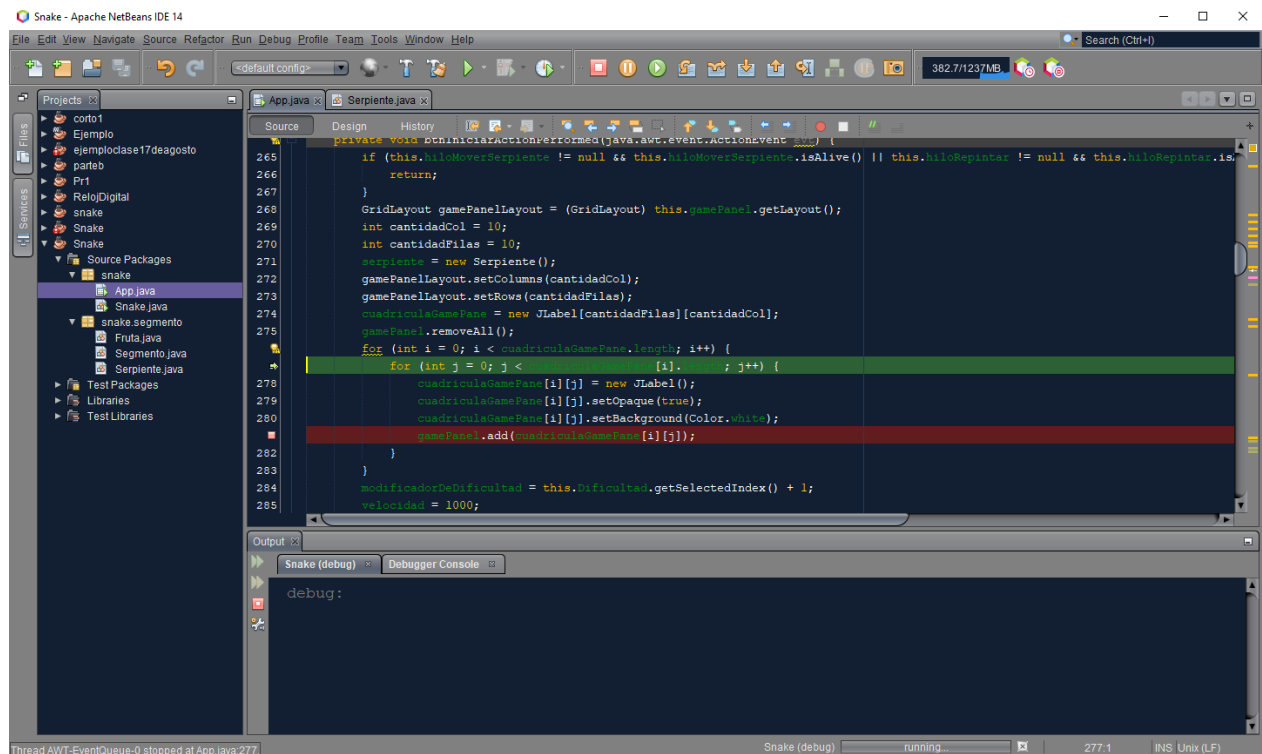
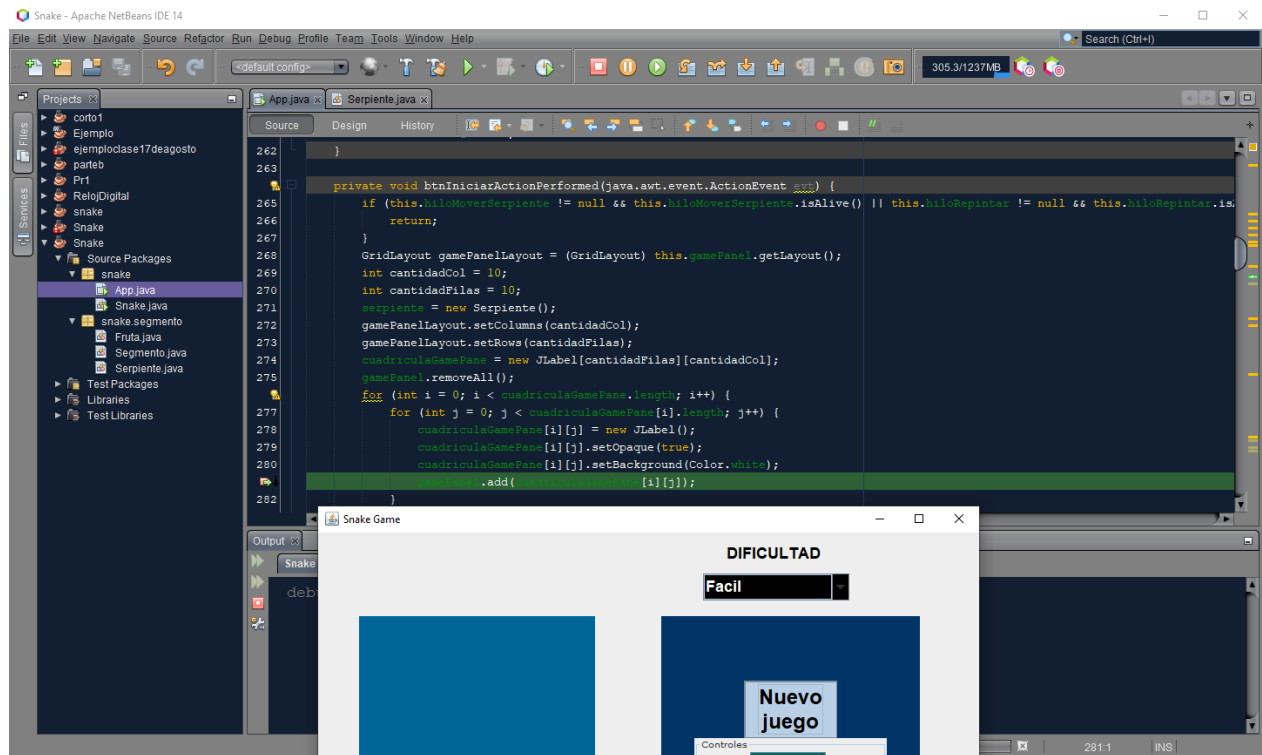
Verificando si reconoce cada objeto dentro del JFrame.



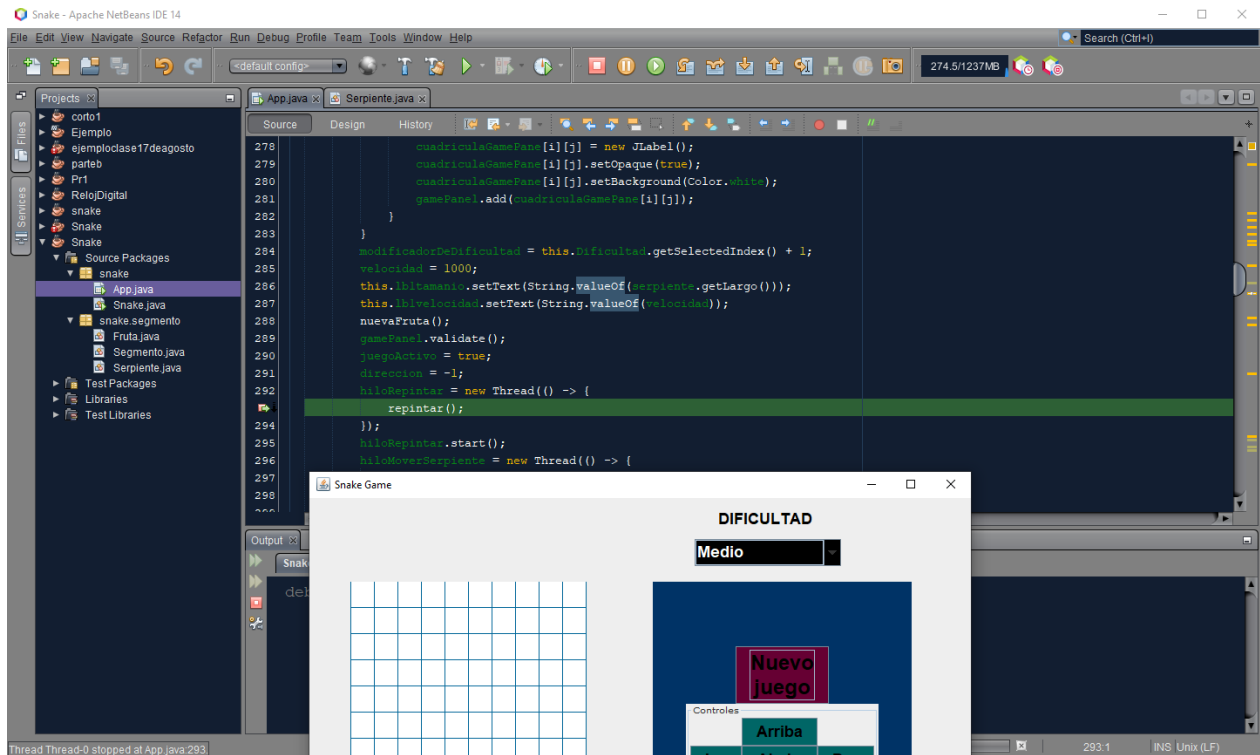


Comprobando la utilidad del botón iniciar y su código.

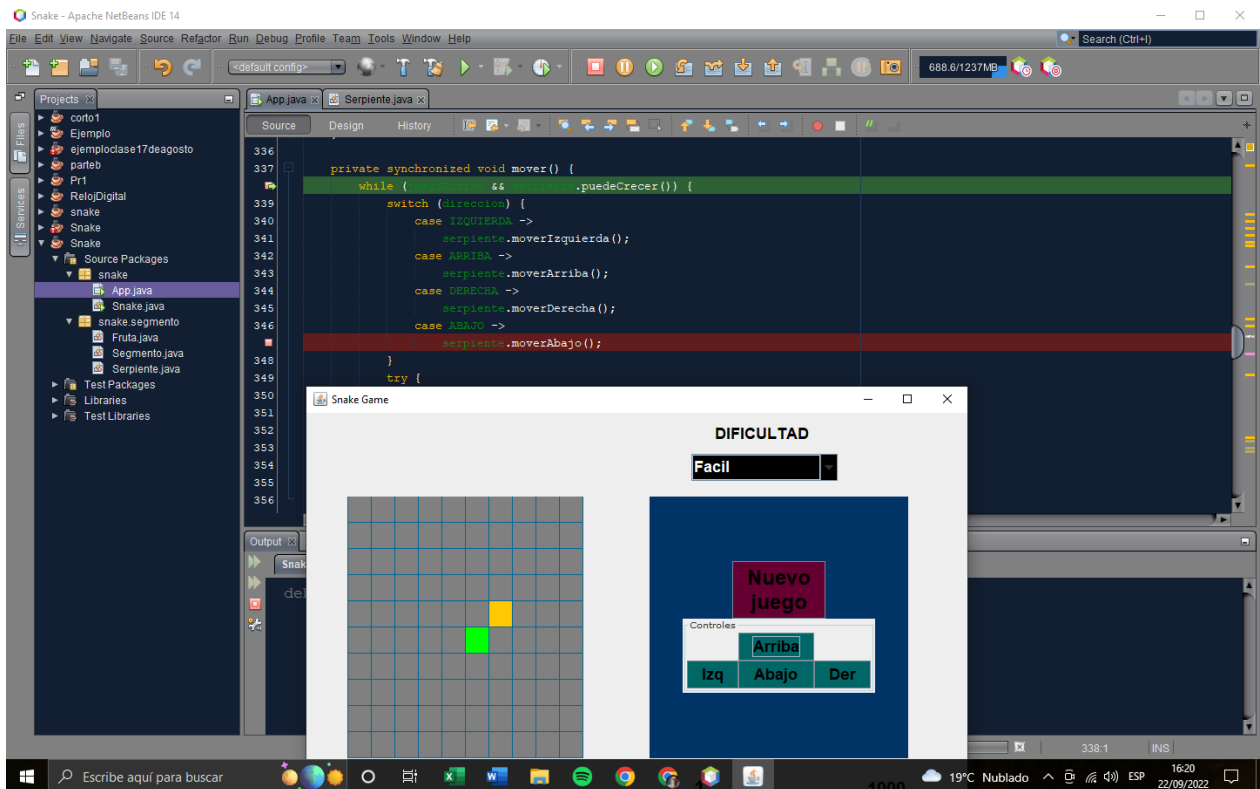




Verificación del repinto del tablero.



Funcionamiento de los botones que dirigirá a la serpiente.



Colisión con las paredes de la serpiente.

