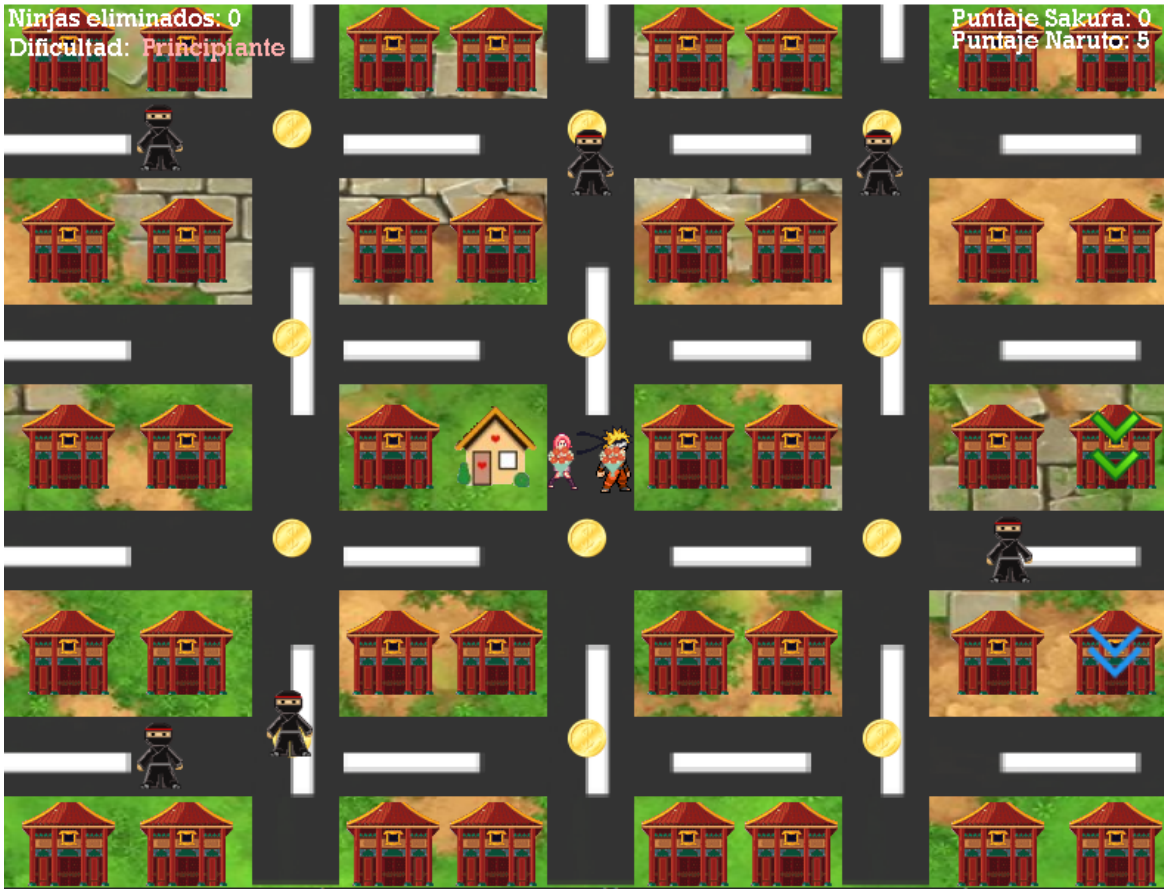
Grupo 4

TRABAJO PRÁCTICO DE PROGRAMACIÓN

Sakura Ikebana Delivery



Integrantes:

Chun Mariano 43858084-2020 marianochun01@gmail.com

Kupczyk August 43663457-2020 augustkup@gmail.com

Portillo Adriel 43240986-2020 portilloadriel200@gmail.com

Introducción

El siguiente trabajo fue realizado siguiendo el paradigma de la programación orientada a objetos. Se definieron diferentes clases para todos los objetos involucrados en el juego, delimitando sus estados y comportamientos. El programa confeccionado se trata de un videojuego en el que un personaje debe realizar entregas a diferentes casas, esquivando obstáculos y ninjas que buscarán hacer perder al jugador. Si el jugador está el suficiente tiempo con vida o llega a determinado puntaje, ganará el juego. Si son dos jugadores, sólo ganará uno de ellos al llegar a determinado puntaje, de lo contrario, se producirá un empate al llegar a cierto tiempo.

Descripción de las clases

Clase ninja

En esta clase se define el estado y comportamiento de los ninjas que se encargaran de hacer perder al jugador.

Variables de instancia

- **Se definieron getters para todas las variables de instancias menos para la de img1**

```
private double x; // Posición en x del ninja
private double y; // Posición en y del ninja
private int ancho; // Ancho del ninja obtenido a partir de la imagen
private int alto;  // Alto del ninja obtenido a partir de la imagen
private double velocidad; // Velocidad del ninja
private Image img1; // Imagen del ninja mostrada en el juego
```

Métodos de instancia

```
public Ninja(double x, double y): Constructor de la clase Ninja
     this.x = x;
     this.y = y;
     this.velocidad = 2;
     this.img1 = Herramientas.cargarImagen("ninja.png");
     this.alto = img1.getHeight(null);
     this.ancho = img1.getWidth(null);
```

```
public void cambiarVelocidad(double velocidad): La función de este
método es la de cambiar la velocidad del ninja por la del argumento
pasado.
public void dibujar(Entorno e): La función de este método es la de
dibujar al ninja en el entorno pasado como argumento.
public void mover(double angulo): La función de este método es la de
mover al ninja aumentando o decrementando su coordenada x o su
coordenada y. Este movimiento se ejecutará de acuerdo al ángulo que
le pasemos como argumento
public boolean chocasteConElEntorno(Entorno e): Este método se
utiliza para determinar si el ninja choca con el entorno.
```

Clase rasengan

En esta clase se define el estado y el comportamiento del objeto rasengan, el cual será utilizado por el jugador para eliminar a los ninjas.

Variables de instancia

- **Se definieron getters para las variables de instancia "x","y" y tamaño**

```
private double x; // Posición en coordenada x del rasengan.
private double y; // Posición en coordenada y del rasengan.
* Usualmente las posiciones x e y toman los valores de sakura.
private double tamaño; // Tamaño o diámetro del rasengan.
private double angulo; // Ángulo del rasengan, de acuerdo a la
orientación de sakura este variara.
private double velocidad; // Velocidad a la que el rasengan es
lanzado.
private Image rasengan; // Imagen del rasengan mostrada en el juego.
```

Métodos de instancia

```
public Rasengan(double x, double y, double angulo) (Constructor de la clase
Rasengan)
            this.x = x;
            this.y = y;
            this.tamaño = 20;
            this.angulo = angulo;
            this.velocidad = 4;
            this.rasengan = Herramientas.cargarImagen("rasengan.png");
_____
```

```
public void dibujar(Entorno e) : La función de este método es la de
dibujar al rasengan en el entorno pasado como argumento.

public void mover():  La función de este método es la de mover al
rasengan por el entorno.

public boolean chocasteConElEntorno(Entorno e): La función de este
método es la de arrojar un booleano que indique si el rasengan choca
con el entorno o no.

public boolean chocasteConNinja(Ninja n): La función de este método
es la de arrojar un booleano que indique si existe colisión entre el
rasengan y algún ninja.
```

Clase Florería

En esta clase se definió el estado y comportamientos del objeto al que el jugador o los jugadores deben acudir para realizar el proceso de ir a buscar el ikebana y llevarlo a la casa indicada.

Variables de instancia

- **Se definieron getters para las variables de instancia "x" e "y".**

```
private double x; // Coordenada x de la florería
private double y; // Coordenada y de la florería
private double ancho; // Ancho de la florería
private double alto; // Alto de la florería
private Image img1; // Imagen de la florería
```

Métodos de instancia:

```
public Floreria(double x, double y) (Constructor de la Florería)
 this.x = 340;
 this.y = 300;
 this.img1= Herramientas.cargarImagen("floreria.png");
```

.

```
 public void dibujar (Entorno e): La función de este método es la de
 dibujar a la florería en el entorno pasado como argumento.
```

Clase NinjaFuerte

Esta clase posee los mismos estados y comportamientos que la clase Ninja y comparten el objetivo de hacer perder al jugador. En lo único en que se diferencian es que esta clase no posee el método cambioDeVelocidad() de la clase Ninja ya que la velocidad del objeto NinjaFuerte será constante en 2. La última diferencia que se verá plasmada en el juego es el cambio de la imagen, a diferencia de Ninja, NinjaFuerte es dibujado con la imagen de un ninja de color bordo.

- **Se definieron getters para las variables de instancia “x” e ”y”.**

Clase Calle

En esta clase se definió 8 variables con sus respectivas funciones para crear las calles en donde solo deben caminar Sakura, Naruto, los ninjas y donde solo deben posicionarse las monedas.

Las variables de instancia son:

```
       private double x; // donde me indica la posición de X de la calle
       private double y; // me indica la posición en Y de la calle
       private boolean tipo; // nos indica si una calle es horizontal o vertical,
True para horizontal y False para vertical. Si es horizontal dibuja la img1 si no
la img 2
       private Image img1; //Imagen de la calle horizontal
       private Image img2; //Imagen de la calle vertical
```

Hicimos un método Calle con las variables double y boolean como parámetros y luego llamamos a todas las variables.

```
public Calle(double x, double y, double ancho, double alto, double angulo, boolean
tipo) {
```

Importamos al java.awt.Image y al Import entorno.herramientas para para poder cargar la imagen y mostrarla en el entorno. Con el siguiente código pudimos cargar las imágenes.

```
       this.img1= Herramientas.cargarImagen("CALLE.png");
       this.img2= Herramientas.cargarImagen("CALLE2.png");
```

Por otro lado, Importamos el Entorno, import entorno.Entorno para poder dibujar la imagen en donde hicimos un método dibujar en donde hicimos un If que la variable tipo nos indicaba si una calle era vertical dibujaba la img1 si no dibujaba la img2.

```
              if(tipo) {
                    e.dibujarImagen(this.img1,this.x ,this.y,0.0, 1.2);
             } else {
                    e.dibujarImagen(this.img2,this.x,this.y, 0.0,1.3);
             }
```

Luego después de haber creado la clase Calle fuimos a la clase juego y llamamos a la clase Calle para poder ponerle los parámetros correspondientes a cada calle, para las calles horizontales usamos los siguientes parámetros.

// Horizontales

```
calle = new Calle(400,90, 800,50,0,true);
             calle1 = new Calle(400,230, 800,50,0, true);
             calle2 = new Calle(400,370, 800,50,0, true);
```

```
            calle3 = new Calle(400,510, 800,50,0, true);
```

Y para las calles verticales usamos los siguientes parámetros.

// Verticales

```
calle4 = new Calle(200,100, 50,1000,0, false);
            calle5 = new Calle(400,100, 50,1000,0, false);
            calle6 = new Calle(600,100, 50,1000,0, false);
```

Y desde el método Tick dibujamos a las calles con de la siguiente manera:

//dibujo las calles

```
calle.dibujar(entorno);
            calle1.dibujar(entorno);
            calle2.dibujar(entorno);
            calle3.dibujar(entorno);
            calle4.dibujar(entorno);
            calle5.dibujar(entorno);
            calle6.dibujar(entorno);
```

En un principio nos encontramos con dificultades como en dónde queríamos hacer un arreglo de calles pero no nos salía y cambiamos a utilizar las variables de cada calle y seleccionarles los parámetros fijos de cada una de ellas. Por otro lado la segunda dificultad que tuvimos fue que nos dibujaba las imágenes de calles cortadas ya que las tomaba desde el medio de la imagen y por eso tuvimos que poner 400 en X para las horizontales y ahí agregarle escala para que complete la anchura de la imagen y con las verticales no se dibujaban bien, se me superpone en la imagen 1 y por eso recurrimos al if donde me dibujaba la img1 si era horizontal si no dibujaba la img2 si era vertical y ahí funcionó correctamente.

Clase Sakura

Esta clase fue implementada para que el objeto principal (Sakura) tuviera sus propias variables de instancia junto con sus propios métodos. Estas variables de instancia son todas privadas y se las debe llamar desde el método public Sakura.

En esta clase, se crearon las variables de instancia:

```
·         private double x : coordenada x de Sakura
·         private double y : coordenada y de Sakura
```

```
·         private double ancho : tamaño de ancho de Sakura
·         private double alto : tamaño de alto de Sakura
·         private double velocidad : velocidad de Sakura
·         private Image imagenSakura : Imagen de Sakura estática
·         private Image imagenSakuraArriba : Imagen de Sakura cuando se dirige
hacia arriba
·         private Image imagenSakuraDerecha : Imagen de Sakura cuando se dirige
hacia la derecha
·         private Image imagenSakuraIzquierda : Imagen de Sakura cuando se dirige
hacia la izquierda
·         private Image imagenSakuraAbajo : Imagen de Sakura cuando se dirige hacia
abajo
```

Se definieron getters para las variables x e y. También se define un getter para el alto y el ancho.

Métodos de instancia:

```
Método public Sakura(double x, double y, double velocidad) : Este es el constructor
de la clase Sakura.
```

```
this.x = x;
this.y = y;
this.velocidad = velocidad;
this.alto = imagenSakura.getHeight(null);
this.ancho = imagenSakura.getWidth(null);
this.imagenSakura = Herramientas.cargarImagen("SakuraQuieta.png");
this.imagenSakuraArriba = Herramientas.cargarImagen("SakuraArriba.png");
this.imagenSakuraDerecha = Herramientas.cargarImagen("SakuraDer.png");
this.imagenSakuraIzquierda = Herramientas.cargarImagen("SakuraIzq.png");
this.imagenSakuraAbajo = Herramientas.cargarImagen("SakuraAbajo.png");
```

```
Método public void dibujar(Entorno e): En este método se dibuja a Sakura con su
respectiva imagen y también se actualiza la imagen dependiendo de la dirección a la
que esta se dirija.
Método public void moverIzquierda() {: Con este método, Sakura se logra mover a la
izquierda.
public void moverDerecha() :  Con este método, Sakura se logra mover a la derecha.
public void moverAbajo() {  Con este método, Sakura se logra mover abajo.
public void moverArriba() {  Con este método, Sakura se logra mover arriba.
public boolean chocasteConNinja(Ninja n) {: Con este método, se detecta si Sakura
choca con algún ninja.
public boolean enCasaMarcada(Flecha f) {: Con este método, se identifica a qué casa
debe entregar las flores Sakura.
public boolean tocasteMoneda(Moneda m) {: Con este método, se detecta si Sakura
toca alguna moneda del juego.
```

En esta clase, la dificultad que tuvimos fue darle el movimiento a Sakura junto con sus respectivas colisiones, para esto primero propusimos encasillar el movimiento en determinadas coordenadas x e y, primeramente con un if verificamos los limites del entorno. El primer if es para restringir en x los movimientos de izquierda a derecha, y en los movimientos de arriba a abajo restringimos en y. Luego si Sakura esta en una intersección se restringe en y, si se encuentra entrecalles se restringe en x.

Clase Casa

En esta clase se crearon y definieron las posiciones y casas en donde debería hacer la entrega las flores Sakura.

Las variables de instancias son:

```
public class Casa
	private double x; // me indica la posición de X de la casa
	private double y; // me indica la posición de Y de la casa
	private double ancho; //Indica el ancho de la casa
	private double alto; // indica el alto de la casa
	private Image img1; //Imagen de la casa
```

Hicimos un método Casa donde le pasamos los parámetros de x e y.

```
public Casa(double x, double y)
```

Y también hicimos un método para poder dibujar la casa es su respectiva posición.

```
public void dibujar (Entorno e)
```

desde la clase Juego pusimos dos variables, una variable con matrices para las casas y dos variables con arreglos para las posiciones de las casas.

```
private Casa casas[][];
private int posXCasas[] = {50,130,270,340,470,540,680,760};
private int posYCasas[] = {30,160,300,440,570};
```

Luego inicializamos la matriz de las casas con 8 columnas y 5 filas y lo recorrimos para que cada casa se dibuje en su respectiva posición.

Luego en el método Tick hicimos un doble for de i y j y fuimos recorriendo para que se vayan dibujando las casas en los arreglos de [I] y [j]

Para la clase Casa nos encontramos con varias dificultades, ya que, cada casa tendría que recibir diferentes parámetros y posiciones, en donde, en un principio creamos variables para cada casa y salió bien pero el código iba a hacer muy extendido y dijimos vamos a hacer un arreglo pero no funcionaba solo dibujaba 4 casas de 8 y en cualquier posición, hasta que en la ultima opcion pensamos en las matrices y ahi fue donde el codigo quedo sencillo y eficaz para cada que dibuje cada casa.

Clase Shuriken

Esta clase se creó con el objetivo de otorgarles shurikens a los ninjas. Se le asignan variables y métodos.

En esta clase se crearon las siguientes variables de instancia:

```
private double x;  (coordenada x del shuriken)
private double y;  (coordenada y del shuriken)
private double tamaño; (tamaño del shuriken)
private double angulo; (angulo del shuriken)
private double velocidad; (velocidad del shuriken)
private Image shuriken; (imágen del shuriken)
```

Se definieron getters para tamaño, x e y.

Métodos de instancia:

```
public Shuriken(double x, double y, double angulo) (Constructor de la clase
Shuriken)
            this.x = x;
            this.y = y;
            this.tamaño = 20;
            this.angulo = angulo;
            this.velocidad = 3;
            this.shuriken = Herramientas.cargarImagen("shuriken.png");
```

```
public void dibujar(Entorno e)  (Con este método se dibuja el shuriken)
```

```
public void mover()  (Con este método se le asigna movimiento al shuriken)
public boolean chocasteConElEntorno(Entorno e)  (Con este método se
verifica si un shuriken choca con el entorno)
public boolean chocasteConSakura(Sakura s) (Con este método se verifica si
un shuriken choca a Sakura)
public boolean chocasteConNaruto(Naruto n) (Con este método se verifica si
un shuriken choca a Naruto)
```

Clase Flecha

En esta clase, se crea la flecha que marca la florería cuando el jugador debe ir a buscar flores y otra flecha que indica dónde debe realizar la entrega. Se le asignan variables y métodos.

Variables de instancia:

```
      private double x; (coordenada x de la flecha)
      private double y; (coordenada y de la flecha)
      private Image flecha; (Imágen de la flecha)
```

Se definieron getters para las variables x e y.

Métodos de instancia:

```
public Flecha(double x, double y) (Este es el constructor de la clase Flecha )
            this.x = x;
            this.y = y;
            this.flecha = Herramientas.cargarImagen("Flecha.png");
```

```
public void dibujar(Entorno e) //Con este método se dibuja a la flecha
```

Clase Moneda

En esta clase, se le asignan variables de instancia y métodos al objeto Moneda.

Variables de instancia:

```
      private double x; (coordenada x de la moneda)
      private double y;  (coordenada y de la moneda)
      private Image moneda; (Imagen de la moneda)
```

Se definieron getters para las variables x e y.

Métodos de instancia:

```
public Moneda(double x, double y) (Este es el constructor de la clase Moneda)

            this.x = x;
            this.y = y;
            this.moneda = Herramientas.cargarImagen("moneda.png");
```

```
public void dibujar(Entorno e) // Con este método se dibujan las monedas.
```

Clase Naruto

En esta clase se le asignan sus respectivas variables y métodos de instancia a Naruto (

Modo de 2 jugadores).

Variables de instancia:

```
       private double x; (coordenada x de naruto)
      private double y;  (coordenada y de naruto)
      private double ancho; (ancho de naruto)
      private double alto; (alto de naruto)
      private double velocidad; (velocidad de naruto)
      private Image imagenNaruto; (Imagen de Naruto quieto)
      private Image imagenNarutoArriba; (Imagen de Naruto dirigiéndose arriba)
      private Image imagenNarutoDerecha; (Imagen de Naruto dirigiéndose a la
derecha)
      private Image imagenNarutoIzquierda; (Imagen de Naruto dirigiéndose a la
izquierda)
      private Image imagenNarutoAbajo; (Imagen de Naruto dirigiéndose abajo)
```

Se definieron getters para las variables x e y. También se definen getters para el alto y el ancho de Naruto.

Métodos de instancia:

```
public Naruto(double x, double y, double velocidad) (Constructor de Naruto)
      this.x = x;
      this.y = y;
      this.velocidad = velocidad;
      this.imagenNaruto = Herramientas.cargarImagen("narutoQuieto.png");
      this.imagenNarutoArriba = Herramientas.cargarImagen("narutoArriba.png");
      this.imagenNarutoDerecha = Herramientas.cargarImagen("narutoDerecha.png");
this.imagenNarutoIzquierda= Herramientas.cargarImagen("narutoIzquierda.png");
      this.imagenNarutoAbajo = Herramientas.cargarImagen("narutoAbajo.png");
      this.alto = imagenNaruto.getHeight(null);
      this.ancho = imagenNaruto.getWidth(null);
```

Los diferentes métodos implementados son los mismos que en la clase Sakura, con la particularidad de cambiar el objeto de Sakura por Naruto.

Clase RamoFlores

Con esta clase, le asignamos variables y métodos de instancia al objeto RamoFlores.

Variables de instancia:

```
      private double x; (coordenada x del Ramo de flores)
      private double y; (coordenada y del Ramo de flores)
      private Image imagenRamo; (Imagen del Ramo de flores)
```

Métodos de instancia:

```
public RamoFlores(double x,double y) (Este es el constructor de la clase
RamoFlores)
            this.x = x;
            this.y = y;
            imagenRamo = Herramientas.cargarImagen("flores.png");
}
```

```
public void dibujar(Entorno e) { (con este método se dibuja el ramo de flores)
```

Clase Juego

Esta es la clase principal del juego. Desde esta clase se ejecuta el juego. Se asignaron muchas variables de instancia junto con métodos de instancia.

Variables de instancia:

```
       private Entorno entorno;
      private Rasengan rasengan[];
      private Calle calle;
      private Calle calle1;
      private Calle calle2;
      private Calle calle3;
      private Calle calle4;
      private Calle calle5;
      private Calle calle6;
      private Ninja ninjas[];
      private Sakura sakura;
      private Casa casas[][];
      private Moneda monedas[][];
```

```
	private Floreria floreria;
	private int posXMonedas[] = {202,402,602};
	private int posYMonedas[] = {84,226,362,498};
	private int posicionesNinjas[][];
	private int pedidoXSakura; (coordenada x del pedido que debe entregar
Sakura)
	private int pedidoYSakura; (coordenada y del pedido que debe entregar
Sakura)

	private int posXCasas[] = {50,130,270,340,470,540,680,760};
	private int posYCasas[] = {30,160,300,440,570};
	private boolean entregaHecha = true;
	private boolean ikebanaBuscado = false;
	private int posXFlecha; (coordenada x de la flecha)
	private int posYFlecha; (coordenada y de la flecha)
	private boolean perdido = false;
	private int clockNinjasNormales[];
	private int clockNinjasFuertes[];
	private int clockShuriken; (esta variable sirve de cronometro para hacer
reaparecer al shuriken cada cierto tiempo)
	private Flecha flechaSakura;
	private Image fondo;
	private Image fondoMenu;
	private RamoFlores floresSakura;
	private int puntajeSakura;
	private int tiempo;
	private int ninjasEnJuego; (Esta variable sirve para determinar cuantos
ninjas hay en juego)
	private int ninjasEliminados; (Esta variable sirve para determinar que
ninjas fueron eliminados)
	private boolean ganarSakura = false;
	private String dificultad = "Principiante";
	private Image gameover;
	private Shuriken shuriken;
	private Image fotoGanoSakura;
	private boolean unJugador = false;
	private NinjaFuerte ninjasFuertes[]; (array de ninjas fuertes)
	private int rasenganANinjaFuerte[];  (array de rasengan a ninja fuerte)
	private int ninjasNuevos = 0;
	private int posShurikenAleatorio; (variable de posicion del shuriken)
```

Variables de instancia para segundo jugador:

```
 private Naruto naruto;
 private boolean ganarNaruto = false;
private boolean empate = false;
private boolean dosJugadores = false;
private int puntajeNaruto; (puntaje de Naruto)
private RamoFlores floresNaruto; (Ramo de flores de Naruto)
private Flecha flechaNaruto; (Flecha de Naruto)
private boolean entregaHechaNaruto = true;
private boolean ikebanaBuscadoNaruto = false;
private int posXFlechaNaruto; (coordenada x de la flecha de Naruto)
private int posYFlechaNaruto; (coordenada y de la flecha de Naruto)
private int pedidoXNaruto; (coordenada x del pedido de Naruto)
private int pedidoYNaruto; (coordenada y del pedido de Naruto)
private Image fotoGanoNaruto; (Imagen cuando gana Naruto)
private Image fotoEmpate; (Imagen cuando se empata)
```

Métodos de instancia:
public Juego(): En este método se inicializan las variables de instancia del juego.

Se le asignan valores a las diferentes variables antes declaradas. A los ninjas se le asignan diferentes posiciones en el eje x e y.
Se inicializa el array de casas junto con el de floreria y monedas.
Al final del método se da inicio al juego.

public void tick(): Este método es el más importante de la clase Juego y va a ser ejecutado en todo momento. Se actualiza el estado interno del juego para simular el paso del tiempo.
A continuación se describe todo lo implementado en este método de forma cronológica.
Se crea una pantalla de inicio para seleccionar el modo 1 jugador o 2 jugadores.
Se dibujan las diferentes calles dentro del juego.
Se dibujan los ramos para Sakura y Naruto.
Se dibujan las casas.
Se dibuja la florería.
Se dibujan las monedas.
Se dibuja la flecha para Sakura.
Se dibuja la flecha para Naruto (para dos jugadores)
Se determina si el Ikebana fue buscado
Se verifica si Sakura y/o Naruto están en la casa marcada.

Se dibujan las flechas para Sakura y Naruto
Se dibuja el array de Ninjas ya antes creado.
Se crea el movimiento de los ninjas.
El ninja que choque con el entorno es eliminado y se lo inicializa otro nuevo en la posición inicial.
Se crea la reaparición de los ninjas, es decir, se comprueba si hay algún ninja eliminado y cada cierto tiempo hace que reaparezcan.
Se dibujan los ninjas mas fuertes que requieren dos rasengan para morir. Dibujamos un ninja para el nivel intermedio y agregamos dos para el nivel experto.
Volvemos a dibujar a los ninjas fuertes si chocan con el entorno.
Si los ninjas fuertes están eliminados y pasó cierto tiempo, los inicializamos de nuevo.
Creamos los movimientos de Sakura.
Creamos los movimientos de Naruto si es que hay dos jugadores.
Creamos el lanzamiento del rasengan de Naruto si es que hay dos jugadores.
Creamos el lanzamiento del rasengan de Sakura.
Se crean los shurikens para los ninjas.
Un ninja aleatorio es capaz de lanzar este shuriken.
Verificamos si el rasengan de naruto chocó con algún ninja y los eliminamos.
Verificamos si hay rasengan y choca con el entorno lo eliminamos.
Verificamos si hay rasengan y no choca con el entorno, lo dibujamos y movemos.
Verificamos si Sakura toca alguna moneda y le sumamos puntos.
Verificamos si Naruto toca alguna moneda y le sumamos puntos si es que hay dos jugadores.
Verificamos la colisión de Sakura con algún ninja para la opción 1 jugador.
Verificamos la colisión de Naruto y Sakura con algún ninja para la opción de 2 jugadores.
Verificamos si Sakura cumplió con los requisitos para ganar el juego.
Verificamos si Sakura o Naruto cumplieron con los requisitos para ganar el juego o si hay un empate.
Creamos y dibujamos los datos en pantalla: puntaje y ninjas eliminados.
Creamos y dibujamos en pantalla la dificultad en la que se encuentra el jugador.
Aumentamos la velocidad base del ninja a un nivel intermedio y luego a un nivel experto
Finalmente cargamos las imágenes del ganador, si hubo empate y la imagen cuando pierde.

Conclusiones

Luego de la elaboración de este trabajo podemos sintetizar distintas reflexiones a partir del mismo. En primer lugar, este trabajo práctico nos sirvió para aplicar todo lo visto hasta ahora en la cursada de la materia, nuestro entendimiento sobre la programación orientada a objetos y del lenguaje java a nuestro juicio mejoró notablemente a partir del armado de este videojuego. Por último consideramos que dar uso de git para elaborar el trabajo fue

de gran ayuda para comprender más esta poderosa herramienta, sin duda nos será de gran utilidad en futuros programas en el ámbito académico y laboral.