Universidad Autónoma de Baja California

Facultad de Ciencias Químicas e Ingeniería

### 

### 

### 

### 

### 

### 

### 

### **Proyecto Final**

### **-**Memografic

Equipo Orca

Xu Mingfan - 1274233

Edwin Alain Martinez Sanchez-1273699

Materia: Programación Orientada a Objetos

Grupo: 541

Docente: Tona Castro Claudia Gabriel

#### Tijuana, Baja California a 7 de Dic de 2021

**Índice**

Descripción del Proyecto ------------------------------------------------------------------------------- 3

Historia (en caso de que aplique)

Funcionalidades del Sistema --------------------------------------------------------------------------- 4

Gamificación

Aprendizaje

Perfil del usuario --------------------------------------------------------------------------------------- 10

Usuario administrador (en caso de que aplique)

Usuario Jugador

Lógica del sistema -------------------------------------------------------------------------------------- 11

Herramientas utilizadas ------------------------------------------------------------------------------- 12

Lenguajes de programación

Entornos/Frameworks de desarrollo

Herramientas de diseño

Diseño de la interfaz de usuario ---------------------------------------------------------------------- 13

Pantallas de interfaz de usuario

Navegabilidad

Formular la solución ----------------------------------------------------------------------------------- 14

Arquitectura de clases

Arquitectura de objetos

Implementar la solución ------------------------------------------------------------------------------- 15

ClasePrincipal

Usuario

Juego

Pruebas de ejecución ----------------------------------------------------------------------------------- 23

Pruebas de ejecución por parte de los programadores

Pruebas de ejecución con usuarios finales

Resumen de conceptos empleados de programación orientada a objetos ---------------------- 25

Video ----------------------------------------------------------------------------------------------------- 26

Repositorio ---------------------------------------------------------------------------------------------- 27

Conclusiones -------------------------------------------------------------------------------------------- 28

Referencias ---------------------------------------------------------------------------------------------- 29

**Descripción del Proyecto**

El proyecto consiste en realizar un memorama con enfoque educativo, lo que queremos hacer es que pueda ser geográfico, ya sea con banderas de países, comidas típicas o cosas que hagan referencias a otras partes del mundo.

Tenemos pensado que su funcionamiento sea dando clicks a las imágenes, si se le da click a una imagen y no es par de la imagen, las imágenes se voltean y se vuelve a buscar otra imagen.

Es un juego pensado para toda la familia, pero como se puede apreciar, prácticamente cualquier persona le puede sacar provecho al memorama, desde los que solo buscan entretenerse, hasta los que necesitan estudiar para alguna prueba o quienes quieren mejorar su

capacidad mental. Es una forma de entretenimiento sana y divertida. Creemos que no es difícil de entender ya que el juego trata de encontrar los pares de las imágenes dando clicks, estuvimos investigando y pensando que estaría bien hacerlo por puntaje y tiempo, en caso de que no podamos hacer los dos, con que podamos hacer que diga puntaje o el tiempo, podríamos decir que llegamos a nuestra meta. Alguna información extra es que visualmente queremos que sea atractivo, que tenga colores llamativos, buenas imágenes que se vean claras y que sean de un tamaño bueno.

Estuvimos investigando en consolas podríamos hacerlo y lo que encontramos es que en NetBeans es recomendable hacerlo, ya que tiene una funcionalidad parecida a Visual Studio, en el caso de Alain (miembro del equipo), en preparatoria utilizó Visual Studio lo cual es una gran ventaja ya que tiene una idea sobre cómo dar formato a la ventana donde estará el memorama.

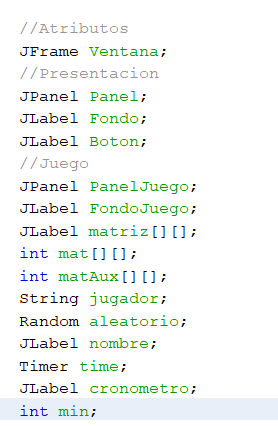
**Funcionalidades del Sistema**

- Uso de clases

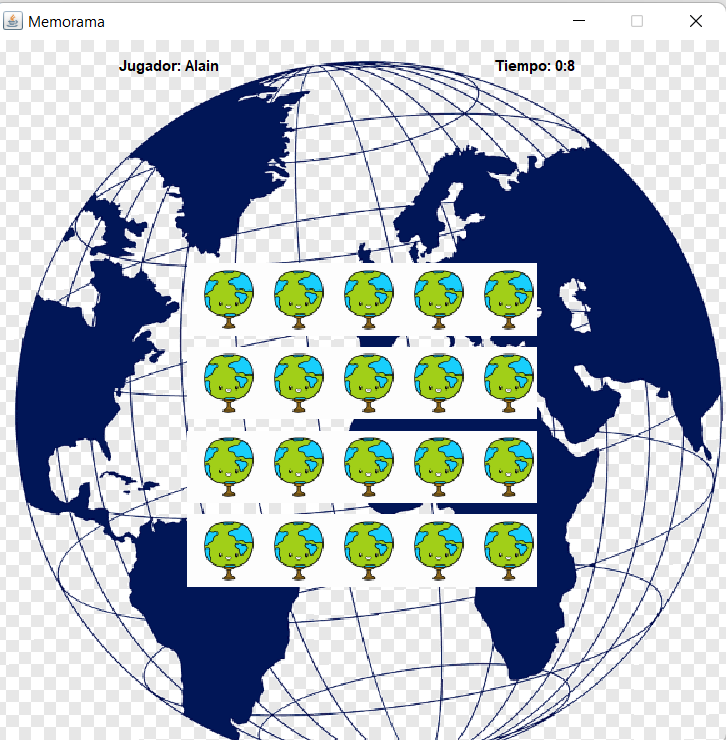


- Uso de métodos

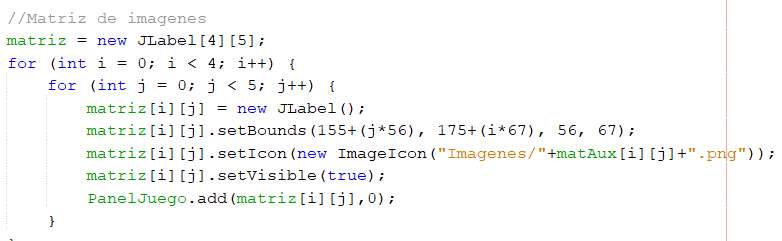
- Uso de atributos



- Crear una ventana que contenga las imágenes del juego



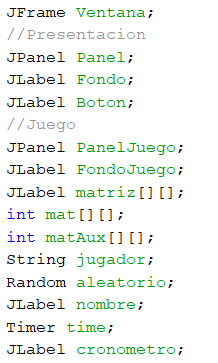
- Uso del tema de matrices



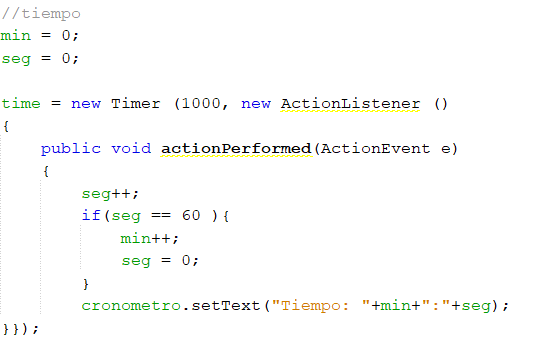
- Uso de JFrame

- Uso de JLabel

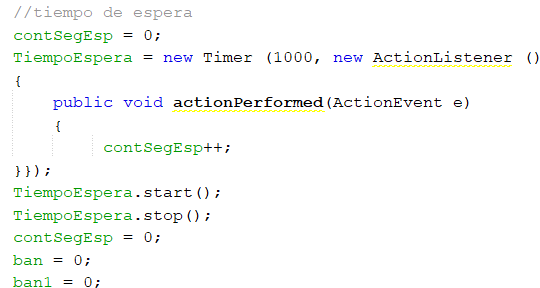
- Uso de JPanel



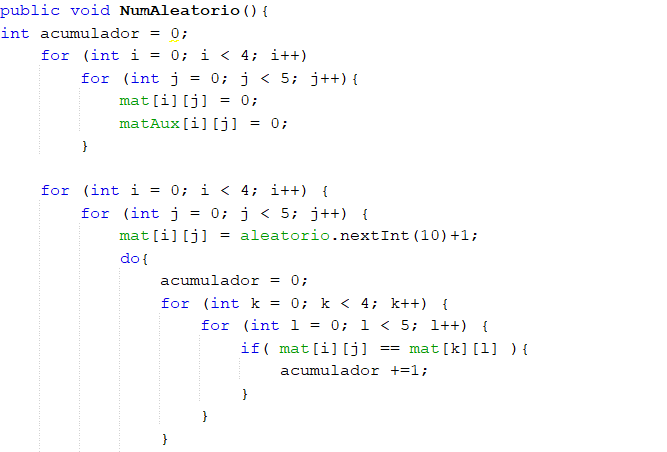
- Cronómetro



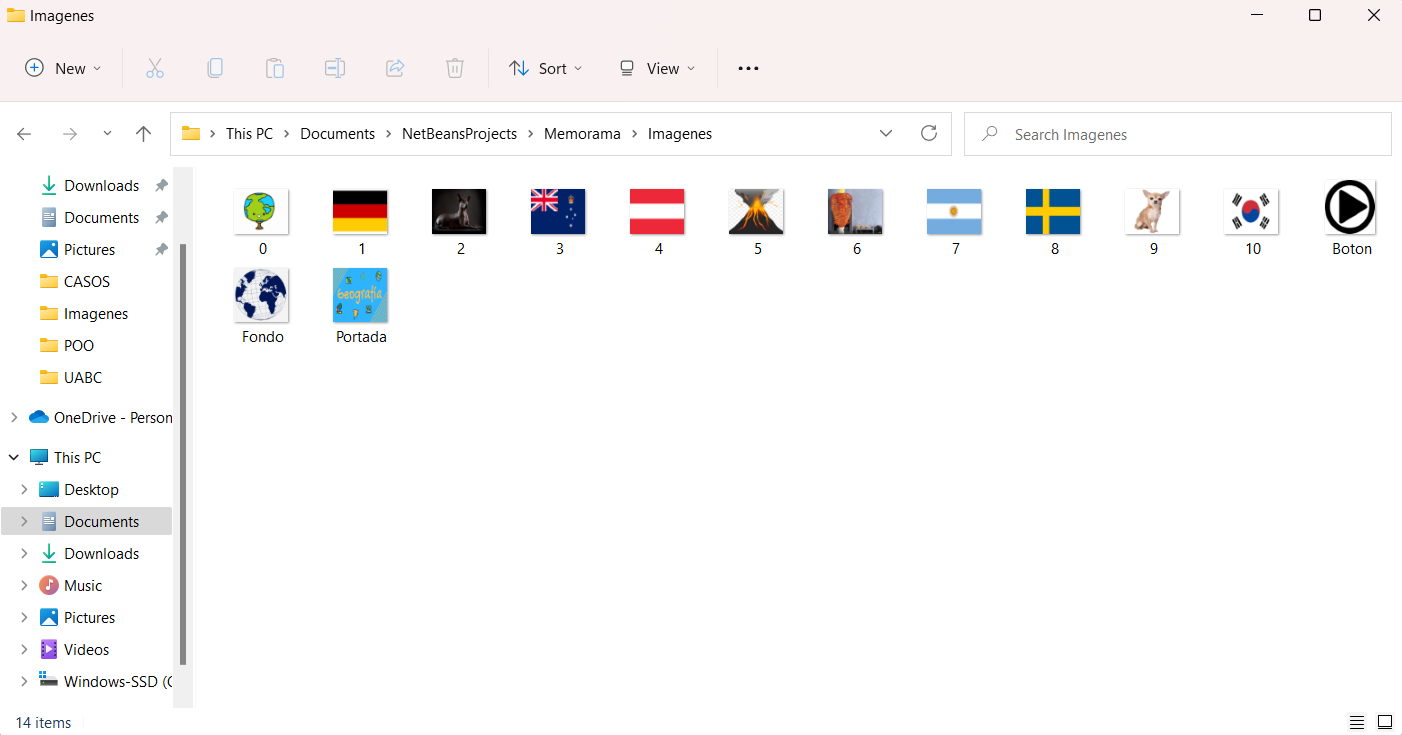
- Si las imágenes no son pares, habrá un cierto tiempo de que las imágenes se vuelvan a voltear



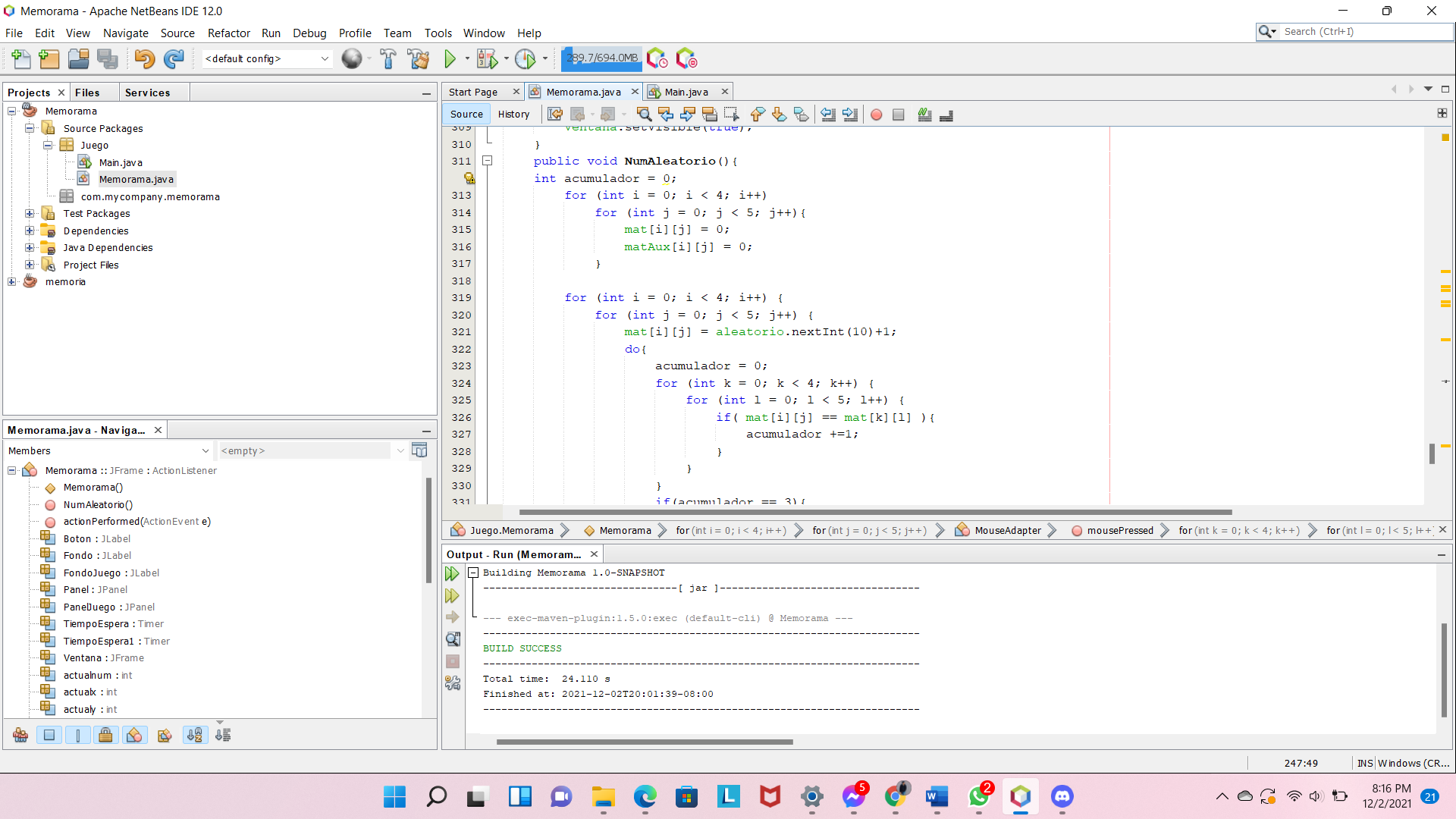
- Cada vez que se abra el juego, las imágenes tomarán distinto lugar



- En el archivo donde esté el programa, se creará uno donde puedan estar las imágenes .png



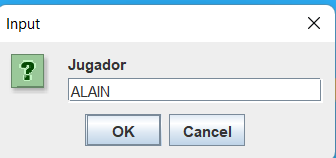
- Aplicación NetBeans



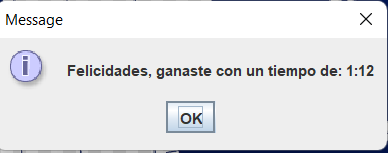
- Se reiniciará el juego cuando se gane para que nuevamente se pueda jugar

- Al iniciar el juego, se abrirá una ventana donde tenga un botón que diga iniciar

- Se abrirá una ventana en donde pida nombre del jugador y lo guarde



- Al completar todos los pares, se abrirá una ventana donde salga un mensaje y diga el tiempo con el cual terminó

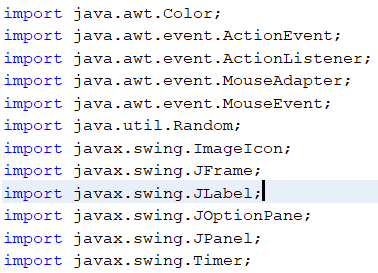


- Si se escoge la primera imagen y luego la siguiente es distinta, aunque se de clicks a las demás imágenes, no hará nada y se tendrá que esperar

- Uso del modificador public



- Uso de librerías



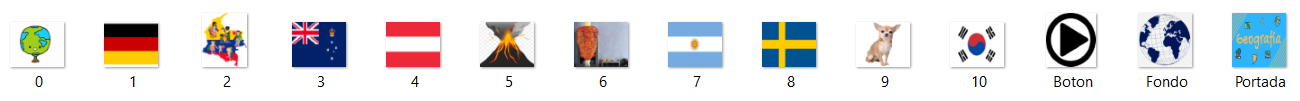
- Trabajar la mente, agilizar la memoria botón de jugar

- Fondo del juego

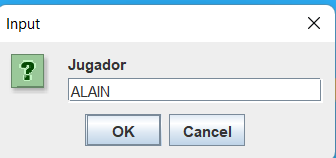
- Portada del menú



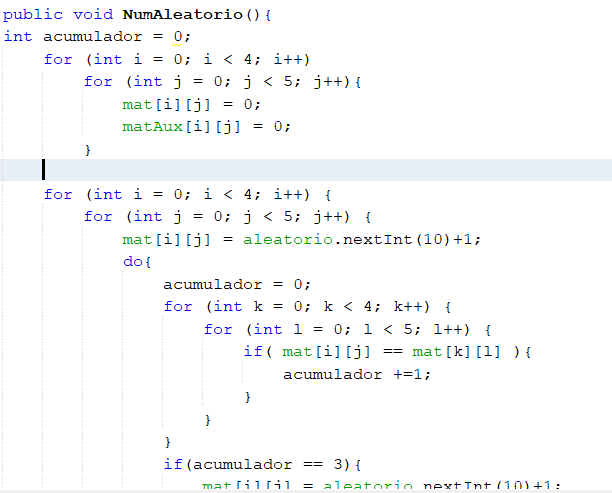
- Imágenes claras, no con color fuerte



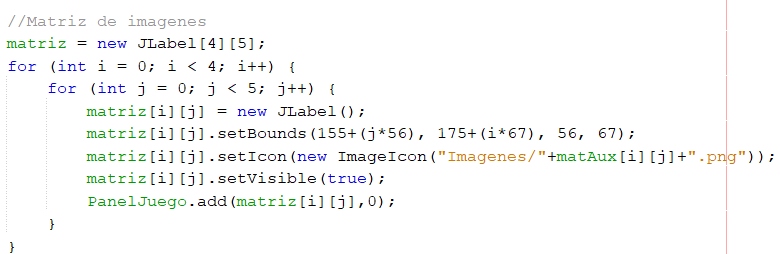
- Opción para escribir nombre de jugador



- Se usará matriz para los números aleatorios



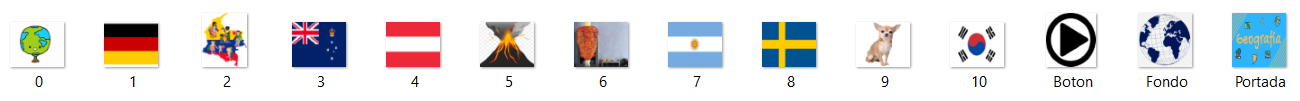
- Se usará matriz para las imágenes



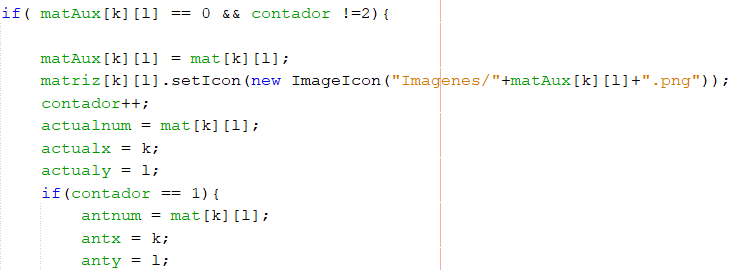
- Habrá tiempo para saber cuánto dura el jugador en realizar el juego



- Se usan medidas específicas para las imágenes



- Se usarán condiciones



- Se usarán algunos for



- Anotación @Override



- Uso de setVisible



- Uso de herencia



- No se podrán ver las imágenes en caso de fallar, ya que habrá tiempo de espera

- Diseño de la interfaz de operación: compile y ejecute, seleccione el nivel, el juego de

iconos comienza, si el usuario no está seguro de la ubicación del mismo icono, haga

clic en el mensaje

- Reanudar el área de prueba del juego

- Clasificación de rendimiento del jugador

- setForeground para poner color a la letra



- Cree una variable miembro para determinar el número de cuadrados en el área de prueba

- El valor del éxito es el número de cuadrados con el mismo icono encontrados por el usuario.

**Perfil del usuario**

Todos en nuestro proyecto pueden jugar, la edad mínima puede ser de 6 años o más, porque esta edad es la época dorada de la memoria, se fortalecerá la memoria del país y / o las características del país, en cuanto a educación, memoria. Los juegos pueden mejorar la capacidad de memoria de los niños, juzgar el color y la forma de la imagen.

La experiencia de cómputo simplemente es que sepa manejar el mouse, saber como funciona, no es difícil pero en caso de que se pueda complicar, explicaremos lo básico para poder jugar. Para que sea juego sea más divertido, tenemos pensado en que la persona que haga mejor puntaje y/o tiempo, dar un obsequio por haberlo hecho rápido y por haber tenido preferencia en poder hacer uso de nuestro proyecto.

Usuario administrador

En el usuario administrador se necesita que tenga conocimiento de Java y de programación orientada a objetos ya que el proyecto está hecho con temas vistos en clase, en caso de no saber mucho, para más sencillo hay videos en YouTube donde explican muy bien lo básico para poder entender. Si una persona con conocimientos de programación orientada a objetos quiere ayudar a implementar algo en nuestro proyecto es bienvenido.

Usuario Jugador

La función que tiene es simplemente jugar, encontrar rápido las imágenes para así tener mejor tiempo en cuanto a otros resultados.

**Lógica del sistema**

Para empezar, el programa está hecho en base a dos clases, la principal y donde contiene todo lo que hará el memorama, una de las primeras cosas que se hizo fue poner las librerías que se usarán en el programa.

En la clase del memorama se pudo usar herencia para poder poner JFrame y también ActionListener. JFrame se utiliza para que se pueda convertir en una ventana y se implementa ActionListener para que esté pendiente cuando el usuario de click.

Luego teniendo ya lo primordial, se empieza describiendo todo lo que se va a usar, dentro de la parte del diseño y algunos datos que se usan para poder iniciar a la hora del tiempo y de las matrices, ya que el programa fue hecho por medio de matrices.

Primero se hace la ventana, luego el panel de presentación, el fondo, los botones, por ejemplo el botón para poder iniciar a jugar se hizo mediante una imagen, el panel del juego, nombre de jugador que es donde la persona que jugará va a escribir su nombre, el bloque de tiempo, que en pocas palabras es el cronómetro.

Se crea la matriz lógica, se usaron 10 parejas, donde tiene 4 filas y 5 columnas, después se crea la matriz donde va a contener las imágenes, para poder hacer lo de las imágenes, en la carpeta donde se guarda la información del programa, se creó una carpeta donde se pusieron todas las imágenes que se usaron, se editó también el tamaño de las imágenes para que pudieran quedar mejor en la ventana.

Después de crear la matriz lógica, se inicia con el tiempo, empieza siendo 0, también cuando los segundos lleguen a 60 se vaya sumando como un minuto, el tiempo de espera también se pone para que en el tiempo cuando de click el jugador y la pareja sea o no, haya un lapso de tiempo, en ese lapso aunque se den clicks, también uno de sus usos es que con el tiempo, desaparece o voltea las cartas. A raíz de esto, se crea el evento de los clicks, al momento que se click a la carta, se mandan a llamar las cartas, para diferenciarlas más rápido y sencillo, se les puso nombre con números desde el 1 al 10.

Se crea también el tiempo de que al momento de terminar, se pueda reiniciar el juego para poder jugar de nuevo. También se crea la parte lógica de lo que son las matrices, la función es que manda a poner las imagenes, si se tienen las 20 cartas completas, mandará un mensaje diciendo "Felicidades, ganaste con un tiempo de: ", a lo cual muestra el tiempo en el que se paró el cronómetro. Seguido de eso, se reinicia todo, se reinicia el tiempo, las cartas, para volver a jugar.

Al final de haber hecho todo esto, se hace la parte de la matriz que haga que todo sea aleatorio, cada vez que se inicie el juego las cartas no aparecerán en el mismo orden, estarán en otros lugares.

**Herramientas utilizadas**

Lenguajes de programación

El lenguaje de programación utilizado fue java, ya que queríamos conocer un poco mas de java porque anteriormente no lo habíamos usado en otras materias.

Entornos/Frameworks de desarrollo

La herramienta de programación utilizada fue NetBeans, en un inicio queríamos usar Eclipse pero no pudimos, luego quisimos usar Intellij pero tampoco no sabíamos muy bien, investigamos un poco de NetBeans y vimos que era más sencillo hacer el juego.

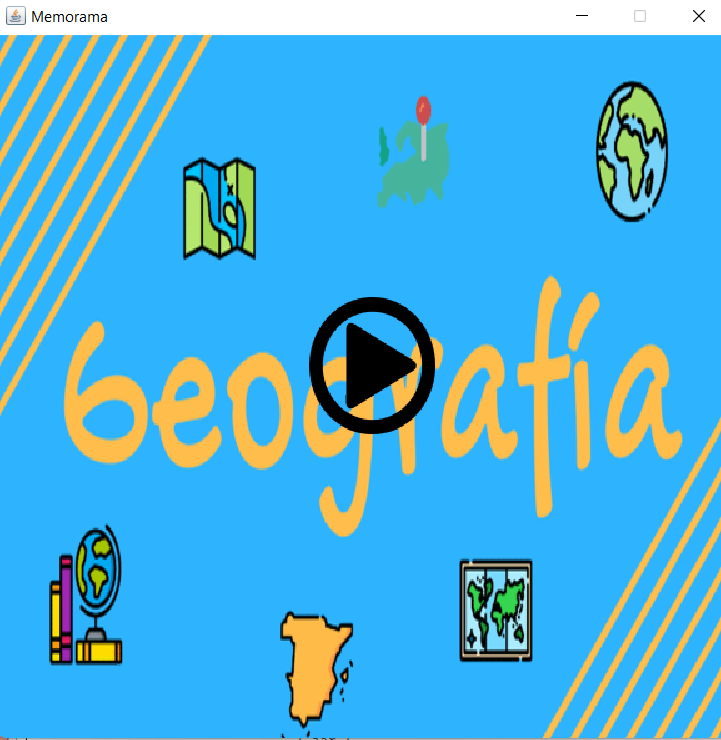
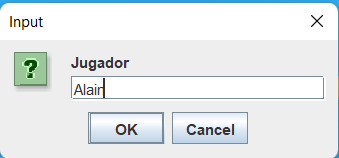
Herramientas de diseño

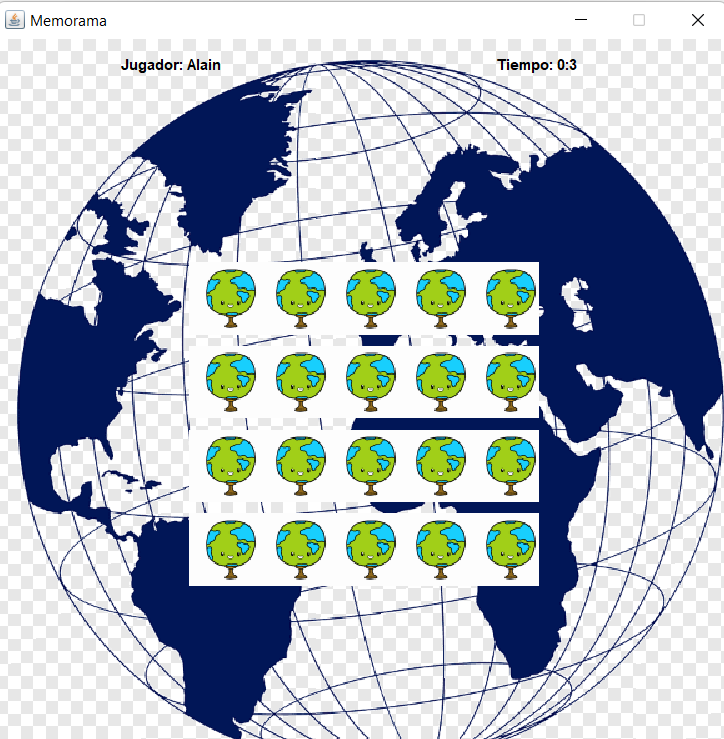
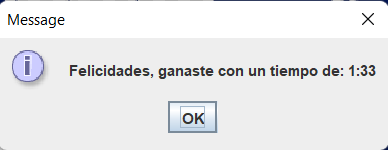
Herramientas de diseño no usamos, ninguno de nosotros ha usado algo similar y se nos hizo más sencillo usar solo imágenes.

**Diseño de la interfaz de usuario**

Pantallas de interfaz de usuario

Navegabilidad

**Formular la solución**

Arquitectura de clases

Diagrama de clases de toda la aplicación/juego. Separados en arquitectura de TRES CAPAS:

1. Interfaz de usuario
2. Datos
3. Lógica

Esto es, se debe poner:

a) Un diagrama de clases por CADA CAPA. Con sus RELACIONES ENTRE CLASES.

b) Un diagrama de clases GENERAL (incluye las tres capas.). Con sus RELACIONES

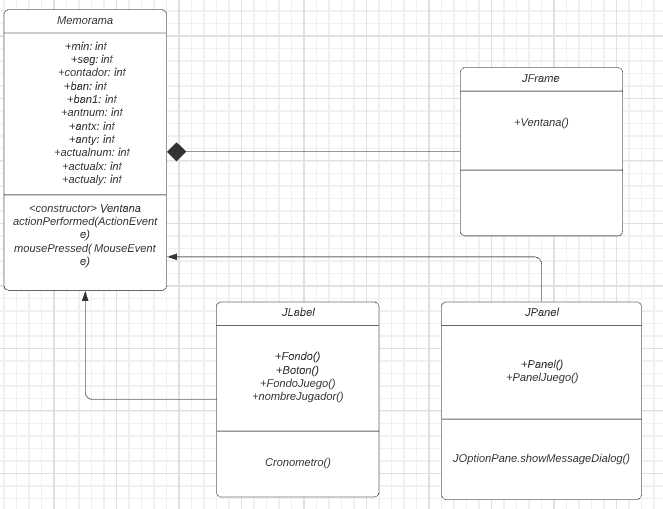
ENTRE CLASES.

Debe incluir:

* Herencia (simple, múltiple)
* Composición/Agregación
* Uso/dependencia

Arquitectura de objetos

Poner una arquitectura de OBJETOS, esto es, por cada clase CUANTOS Y CUALES objetos (instancias) se crearon.



**Implementar la solución**

Incluir código utilizado, separado por clases. Cada inciso anidado a este punto será una clase diferente.

**ClasePrincipal**

package Juego;

/\*\*

\*

\* @author Alain

\*/

public class Main {

public static void main(String []args){

Memorama obj = new Memorama();

}

}

**Juego**

package Juego;

/\*\*

\*

\* @author Alain

\*/

import java.awt.Color;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.util.Random;

import javax.swing.ImageIcon;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.Timer;

public class Memorama extends JFrame implements ActionListener{

//Atributos

JFrame Ventana;

//Presentacion

JPanel Panel;

JLabel Fondo;

JLabel Boton;

//Juego

JPanel PanelJuego;

JLabel FondoJuego;

JLabel matriz[][];

int mat[][];

int matAux[][];

String jugador;

Random aleatorio;

JLabel nombre;

Timer time;

JLabel cronometro;

int min;

int seg;

int contador;

Timer TiempoEspera;

int contSegEsp;

Timer TiempoEspera1;

int ban;

int ban1;

int antnum;

int antx;

int anty;

int actualnum;

int actualx;

int actualy;

//Constructor

public Memorama(){

//Ventana

Ventana = new JFrame("Memorama");

Ventana.setSize(600, 600);

Ventana.setLayout(null);

Ventana.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Ventana.setLocationRelativeTo(null);

Ventana.setResizable(false);

//JpanelPresen

Panel = new JPanel();

Panel.setSize(Ventana.getWidth(), Ventana.getHeight());

Panel.setLocation(0, 0);

Panel.setLayout(null);

Panel.setVisible(true);

//FondoPresen

Fondo = new JLabel();

Fondo.setSize(Ventana.getWidth(), Ventana.getHeight());

Fondo.setLocation(0, 0);

Fondo.setIcon(new ImageIcon("Imagenes/Portada.png"));

Fondo.setVisible(true);

Panel.add(Fondo,0);

//Boton

Boton = new JLabel();

Boton.setSize(110, 130);

Boton.setLocation(250, 200);

Boton.setIcon(new ImageIcon("Imagenes/Boton.png"));

Boton.setVisible(true);

Panel.add(Boton,0);

//Jpanel Juego

PanelJuego = new JPanel();

PanelJuego.setSize(Ventana.getWidth(), Ventana.getHeight());

PanelJuego.setLocation(0, 0);

PanelJuego.setLayout(null);

PanelJuego.setVisible(true);

//Fondo

FondoJuego = new JLabel();

FondoJuego.setSize(Ventana.getWidth(), Ventana.getHeight());

FondoJuego.setLocation(0, 0);

FondoJuego.setIcon(new ImageIcon("Imagenes/Fondo.png"));

FondoJuego.setVisible(true);

PanelJuego.add(FondoJuego,0);

//nombre del jugador

nombre = new JLabel();

nombre.setSize(150, 20);

nombre.setLocation(100, 10);

nombre.setForeground(Color.BLACK);

nombre.setVisible(true);

PanelJuego.add(nombre,0);

//Nombre del jugador

cronometro = new JLabel();

cronometro.setSize(150, 20);

cronometro.setLocation(Ventana.getWidth()-200, 10);

cronometro.setForeground(Color.BLACK);

cronometro.setVisible(true);

PanelJuego.add(cronometro,0);

//Matriz logica

mat = new int[4][5];

matAux = new int[4][5];

aleatorio = new Random();

this.NumAleatorio();

//Matriz de imagenes

matriz = new JLabel[4][5];

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 5; j++) {

matriz[i][j] = new JLabel();

matriz[i][j].setBounds(155+(j\*56), 175+(i\*67), 56, 67);

matriz[i][j].setIcon(new ImageIcon("Imagenes/"+matAux[i][j]+".png"));

matriz[i][j].setVisible(true);

PanelJuego.add(matriz[i][j],0);

}

}

//tiempo

min = 0;

seg = 0;

time = new Timer (1000, new ActionListener ()

{

public void actionPerformed(ActionEvent e)

{

seg++;

if(seg == 60 ){

min++;

seg = 0;

}

cronometro.setText("Tiempo: "+min+":"+seg);

}});

//tiempo de espera

contSegEsp = 0;

TiempoEspera = new Timer (1000, new ActionListener ()

{

public void actionPerformed(ActionEvent e)

{

contSegEsp++;

}});

TiempoEspera.start();

TiempoEspera.stop();

contSegEsp = 0;

ban = 0;

ban1 = 0;

//Click de las cartas

contador = 0;

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 5; j++) {

matriz[i][j].addMouseListener(new MouseAdapter() {

@Override

public void mousePressed( MouseEvent e){

for (int k = 0; k < 4; k++) {

for (int l = 0; l < 5; l++) {

if(e.getSource() == matriz[k][l]){

if( matAux[k][l] == 0 && contador !=2){

matAux[k][l] = mat[k][l];

matriz[k][l].setIcon(new ImageIcon("Imagenes/"+matAux[k][l]+".png"));

contador++;

actualnum = mat[k][l];

actualx = k;

actualy = l;

if(contador == 1){

antnum = mat[k][l];

antx = k;

anty = l;

}

TiempoEspera1 = new Timer (500, new ActionListener ()

{

public void actionPerformed(ActionEvent e)

{

if(contador == 2 && ban1 == 0 ){

TiempoEspera.restart();

ban1=1;

}

if( contador == 2 && contSegEsp == 2 ){

TiempoEspera.stop();

contSegEsp = 0;

if( matAux[actualx][actualy] == matAux[antx][anty] ){

matAux[actualx][actualy] = -1;

matAux[antx][anty] = -1;

matriz[actualx][actualy].setIcon(new ImageIcon("Imagenes/"+matAux[actualx][actualy]+".png"));

matriz[antx][anty].setIcon(new ImageIcon("Imagenes/"+matAux[antx][anty]+".png"));

contador = 0;

//Funcion para ganar

int acum = 0;

for (int m = 0; m < 4; m++) {

for (int n = 0; n < 5; n++) {

if(matAux[m][n] == -1)

acum++;

}

}

if(acum == 20 ){

time.stop();

JOptionPane.showMessageDialog(Ventana, "Felicidades, ganaste con un tiempo de: "+min+":"+seg);

for (int m = 0; m < 4; m++) {

for (int n = 0; n < 5; n++) {

mat[m][n] = 0;

matAux[m][n] = 0;

matriz[m][n].setIcon(new ImageIcon("Imagenes/"+matAux[m][n]+".png"));

}

}

NumAleatorio();

min = 0;

seg = 0;

time.restart();

}

}

for (int m = 0; m < 4; m++) {

for (int n = 0; n < 5; n++) {

System.out.print(matAux[m][n]+" ");

if( matAux[m][n] != 0 && matAux[m][n]!=-1 ){

matAux[m][n] = 0;

matriz[m][n].setIcon(new ImageIcon("Imagenes/"+matAux[m][n]+".png"));

System.out.println("Adios carta");

contador = 0;

}

}

System.out.println("");

}

TiempoEspera1.stop();

ban1=0;

}

}});

if( ban == 0 ){

TiempoEspera1.start();

ban = 1;

}

if(contador == 2)

TiempoEspera1.restart();

}

}

}

}

}

});

}

}

Boton.addMouseListener(new MouseAdapter() {

@Override

public void mousePressed(MouseEvent e){

jugador = JOptionPane.showInputDialog(Ventana, "Jugador", "Por favor, escribe tu nombre" );

while(jugador == null || jugador.compareTo("Escribe aqui")==0 || jugador.compareTo("")==0){

jugador = JOptionPane.showInputDialog(Ventana, "Debes ingresar tu nombre","Por favor, escribelo");

}

nombre.setText("Jugador: "+jugador );

time.start();

Panel.setVisible(false);

Ventana.add(PanelJuego);

PanelJuego.setVisible(true);

}});

Ventana.add(Panel);

Ventana.setVisible(true);

}

public void NumAleatorio(){

int acumulador = 0;

for (int i = 0; i < 4; i++)

for (int j = 0; j < 5; j++){

mat[i][j] = 0;

matAux[i][j] = 0;

}

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 5; j++) {

mat[i][j] = aleatorio.nextInt(10)+1;

do{

acumulador = 0;

for (int k = 0; k < 4; k++) {

for (int l = 0; l < 5; l++) {

if( mat[i][j] == mat[k][l] ){

acumulador +=1;

}

}

}

if(acumulador == 3){

mat[i][j] = aleatorio.nextInt(10)+1;

}

}while( acumulador == 3 );

}

}

}

@Override

public void actionPerformed(ActionEvent e) {

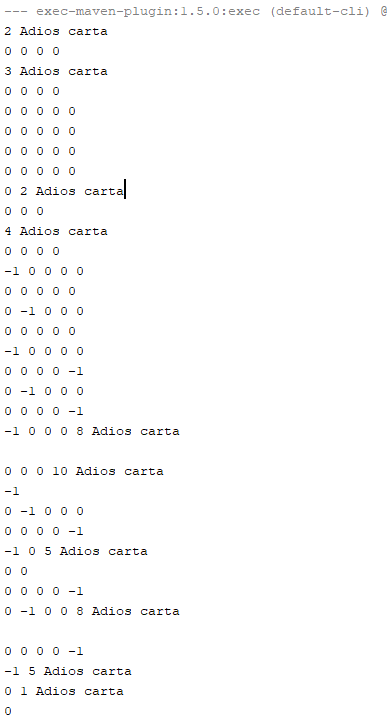
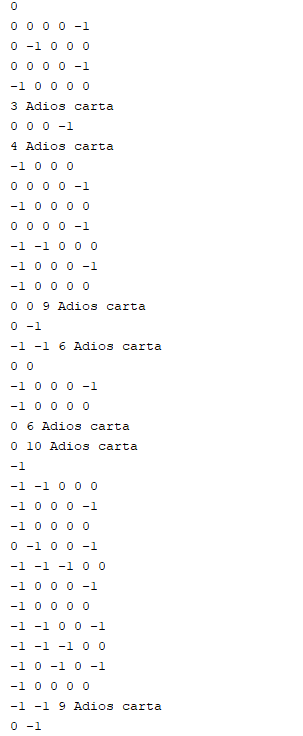
}

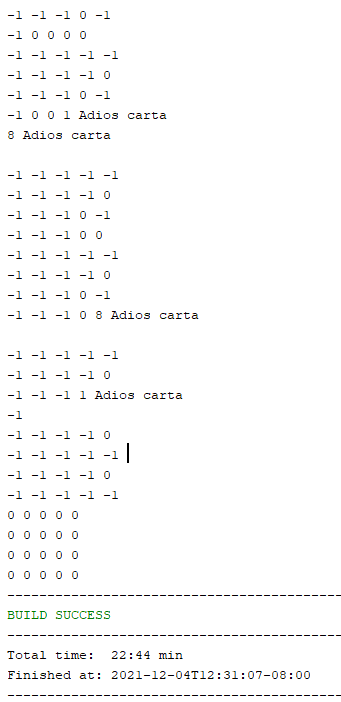
}

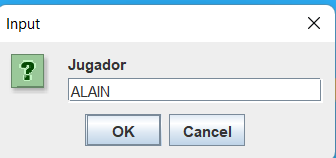
**Pruebas de ejecución**

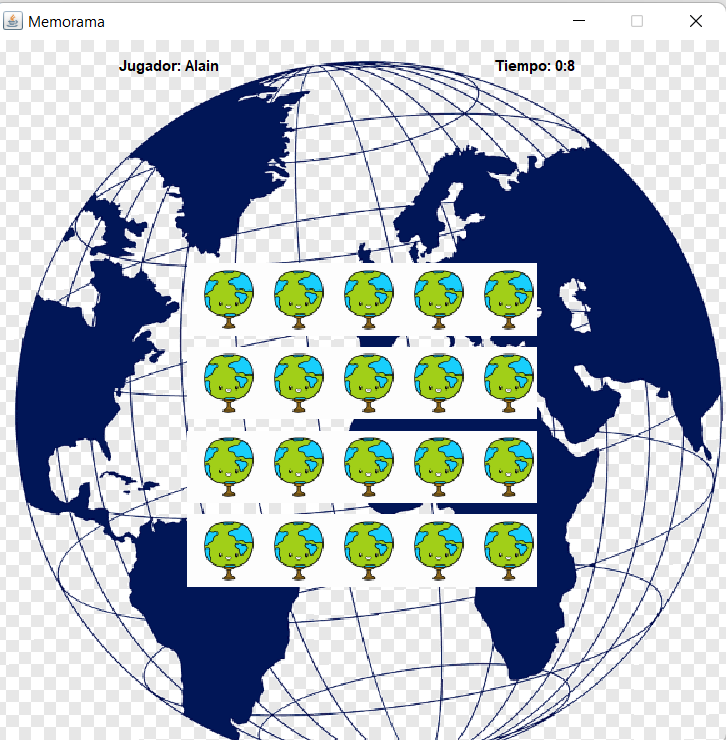
Pruebas de ejecución por parte de los programadores

Pruebas de ejecución con usuarios finales







**Explicación breve de los conceptos de POO empleados en el proyecto**

**Clases normales**

Usamos solamente 2 clases, una llamada Memorama y otra Main, en la clase llamada Memorama hicimos todas las indicaciones sobre hacer la ventana, cronómetro y lo que ocupamos, en la clase Main pusimos el código para que pueda funcionar lo que escribimos.

**Interfaces**

ActionListener en Java es una clase que es responsable de manejar todos los eventos de acción, como cuando el usuario hace clic en un componente.

Un ActionListener puede ser utilizado por la palabra clave implements para la definición de clase. También se puede usar por separado de la clase creando una nueva clase que lo implemente pero nosotros decidimos hacerla en la misma clase Memorama.

**Jerarquía: Herencia múltiple**

Sabemos que java no permite herencia múltiple pero en realidad las clases derivadas de varias clases base pueden conseguirse de una forma segura usando interfaces.

En extends usamos JFrame, JFrame es una clase utilizada en Swing para generar ventanas sobre las cuales añadir distintos objetos con los que podrá interactuar o no el usuario.

**Persistencia**

Lo que entendimos de persistencia es sobre poder cargar archivos, en este caso poder cargar el archivo donde tenemos las imágenes con las cuales usamos en el proyecto.

**Componentes gráficos**

Usamos solamente imágenes, queríamos usar animaciones para que se viera mas entretenido pero no supimos bien usar las herramientas para crear una animación.

**Video**

* <https://drive.google.com/file/d/14guE9JZRqIpUNViGPvEuLtmvtmauB0Ko/view?usp=sharing>

**Repositorio**

* <https://github.com/Alain2401/Practicas-Poo-Lab> (Edwin Alain Martinez Sanchez)
* <https://github.com/mingfanxu/practica>(Mingfan Xu)

**Conclusiones**

(Martinez Sanchez Edwin Alain)

En conclusión, fue un proyecto no muy sencillo de hacer, ya que era complicado poder estar a las mismas horas mi compañero y yo para poder hacer el proyecto juntos, siento que era sencillo hacerlo juntos ya que así podiamos dar ideas de como hacerlo para entender mejor. Al final, logramos hacer el proyecto, nos hubiese gustado hacer más cosas pero creemos que así como lo dejamos está muy bien para todas las personas.

En si la clase me gustó porque aprendí un nuevo lenguaje de programación que antes no había usado, fue complicado al inicio porque no lo entendía muy bien pero estuve investigando en videos y preguntando a amigos de mi grupo.

（Mingfan Xu）

En primer lugar, este proyecto no es muy sencillo, en el proceso de finalización solicitamos mucha información y videos. Creo que primero debemos enumerar el orden lógico, y luego podremos comprender y completar el trabajo más rápidamente una vez que se haya establecido el orden. Este curso se debe a que aprendí un nuevo lenguaje de programación que no había usado antes. Aunque es muy complicado al principio, encontraré conocimientos interesantes más adelante. Al mismo tiempo, también encontré muchas de mis fallas. Durante el estudio, también les pregunté a amigos de mi grupo, espero consolidar los conocimientos básicos de programación en el futuro.

**Referencias**

* Canelo, M. M. (2021, 10 septiembre). ¿Qué es la Programación Orientada a Objetos? Profile Software Services. Recuperado 5 de diciembre de 2021, de <https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/>
* Java面向对象编程知识点总结. (2018, 29 abril). LangInteger. Recuperado 5 de diciembre de 2021, de <https://langinteger.github.io/2018/04/29/java-oop/?fbclid=IwAR2DPF8OWJXZ7yVq6y34X8FhGgZhMJnkMMd1kcTX-Q_X90pNpSPVHIaYBZ0>
* Jackson, G. S. (2017, 20 noviembre). ¿Qué es la herencia múltiple en Java? Techlandia. Recuperado 6 de diciembre de 2021, de <https://techlandia.com/herencia-multiple-java-info_269416/>