



INGENIERIA CIVIL EN COMPUTACION E INFORMATICA (ICCI)

## **MANUAL TALLER 03**

ELIAS MANQUE OLIVARE  
PROFESOR JONATHAN ROJAS  
PROGRAMACION AVANZADA

# CONTENIDO

1. Código
  - 1.1. Diagrama de Clases
  - 1.2. Contratos
  - 1.3. Clases
    - 1.3.1. App
    - 1.3.2. Pieza
      - 1.3.2.1. Pierna
      - 1.3.2.2. Cabeza
      - 1.3.2.3. Brazo
      - 1.3.2.4. Tórax
      - 1.3.2.5. Arma
    - 1.3.3. Lista Piezas
    - 1.3.4. Lista Robots
    - 1.3.5. Piloto
    - 1.3.6. Robot
      - 1.3.6.1. Robot Alien
      - 1.3.6.2. Robot Humano
    - 1.3.7. Equipo
    - 1.3.8. Estadísticas
    - 1.3.9. GuardarTxt
  - 1.4. Interfaz
    - 1.4.1. Sistema
    - 1.4.2. SistemaRobot
2. Archivos
  - 2.1. Texto
  - 2.2. Img.
3. Interfases Graficas
  - 3.1. Inicio
  - 3.2. Buscar Robot por Equipo
  - 3.3. Buscar Robot por Piloto
  - 3.4. Cambiar Armas
  - 3.5. Cambiar Piezas
  - 3.6. Crear Piezas
  - 3.7. Ensamblar Robot
  - 3.8. Estadísticas
  - 3.9. Simulación
  - 3.10. Victorias
4. GitHub

## 1.1. Diagrama de Clases



\*Diagrama en Digital

## 1.2. Contratos

Operación	<code>iniciarApp(Sistema sist)</code>
Descripción	Inicia la interfaz grafica del programa
@Pre	
@Post	

Operación	<code>guardarPieza(String info)</code>
Descripción	Guarda los datos ingresados de una pieza, además de discriminar que tipo es la pieza a guardar
@Pre	
@Post	Crea y guarda los datos de una pieza en listaPiezas

Operación	<code>guardarRobot(String linea)</code>
Descripción	Guarda los datos ingresados de un robot, tanto por txt como por interfaz grafica, además discrimina el tipo de robot que se debe instanciar
@Pre	
@Post	

Operación	<code>bucarListaRobotsPorPiloto(String text)</code>
Descripción	Se obtiene la lista de robot que el piloto maneja
@Pre	el piloto debe de estar registrado
@Post	

Operación	<code>buscarListaRobotPorEquipo(String text)</code>
Descripción	Se obtiene la lista de robot que realizan la mantencion por el equipo
@Pre	el equipo debe de existir en los registros
@Post	

Operación	<code>guardarCombates(String linea)</code>
Descripción	Guarda un registro de cada combate simulado
@Pre	
@Post	Guarda y crea un registro para guardarlo en registroCombates

Operación	<code>buscarNombreArma(String nombreArma)</code>
Descripción	Se obtiene el robot que esta portando el arma
@Pre	el arma debe estar registrada en listaPiezas
@Post	

Operación	<code>buscarRobot(String nombreRoboString)</code>
Descripción	Se obtiene al robot y el nombre del arma que esta usando
@Pre	el robot debe estar guardado en listaRobots
@Post	

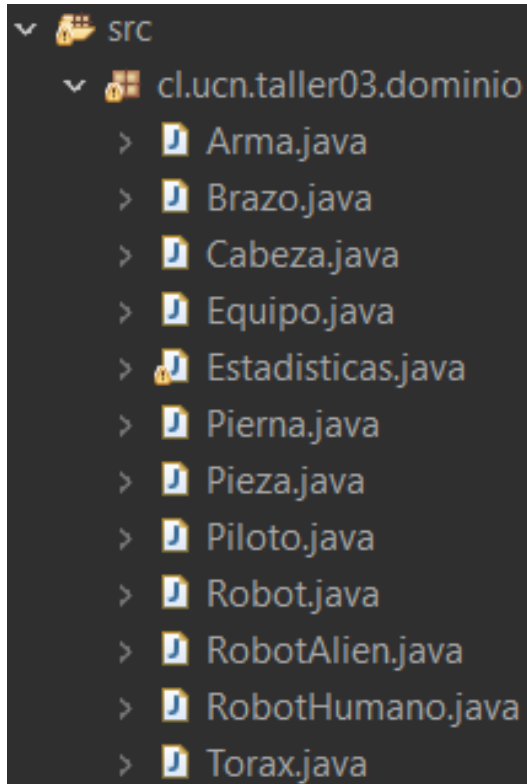
Operación	<code>cambiarArma(String arma, String nombreRobot)</code>
Descripción	Hace un cambio de arma entre robot, o simplemente instala un arma que no tenga designado un robot
@Pre	El arma y el robot deben de existir, el arma puede estar disponible o vinculada a un robot pero nunca debe ser null
@Post	Cambia las referencias de robot y de arma

Operación	<code>String buscarNombrePieza(String nombrePieza)</code>
Descripción	Obtiene la disponibilidad de la pieza
@Pre	La pieza debe estar guardada y registrada en la listaPiezas
@Post	
Operación	<code>buscarRobotPieza(String nombreRoboString, String pieza)</code>
Descripción	* Obtiene la disponibilidad de la Pieza <code>eRobot)</code>
@Pre	La pieza como el robot deben de existir en sus respectivas listas en el caso de que exista una pieza del mismo tipo a agregar serán intercambiadas
@Post	La pieza como el robot deben de existir en sus respectivas listas
@Post	Se cambian las referencias de vínculo entre la pieza y el robot

Operación	
Descripción	
@Pre	
@Post	

Operación	
Descripción	
@Pre	
@Post	

## 1.3. Clases



### 1.3.1. App

Main:

```
public static void main(String[] args) throws FileNotFoundException {  
  
    Sistema sist = new SistemaRobot();  
    Scanner leer = null;  
  
    leerArchivos(leer, sist);  
    ejecutarPaginaInicio(sist);  
  
}
```

```

private static void leerArchivos(Scanner leer, Sistema sist) throws FileNotFoundException {

    leer = new Scanner(new File("src\\cl\\ucn\\taller03\\txt\\piezas.txt"));

    while (leer.hasNextLine()) {
        String linea = leer.nextLine();
        sist.guardarPieza(linea);
    }

    leer = new Scanner(new File("src\\cl\\ucn\\taller03\\txt\\armas.txt"));

    while(leer.hasNextLine()) {
        String linea = leer.nextLine();
        sist.guardarPieza(linea);
    }

    leer = new Scanner(new File("src\\cl\\ucn\\taller03\\txt\\robots.txt"));

    while(leer.hasNextLine()) {
        String linea = leer.nextLine();
        sist.guardarRobot(linea);
    }

    leer = new Scanner(new File("src\\cl\\ucn\\taller03\\txt\\combates.txt"));

    while(leer.hasNextLine()) {
        String linea = leer.nextLine();
        sist.guardarCombates(linea);
    }

}

```

Ejecutar Pagina de Inicio: (Abre la interfaz gráfica)

```

private static void ejecutarPaginaInicio(Sistema sist) {
    sist.iniciarApp(sist);
}

```



### 1.3.2. Pieza

La clase pieza será una clase abstracta ya que hay datos en común con las demás piezas a crear como lo es el nombre, vida base, la rareza y el ataque extra. Para la vida base hay que tener en cuenta que parte en 2000 puntos.

El método añadirMasVidaBase() solo genera una suma dependiendo la rareza de la pieza, dándole un poco mas de ventaja al robot.

```
public class Pieza {  
  
    private String nombre;  
    private int vidaBase;  
    private String rareza;  
    private int ataqueExtra;  
  
    public Pieza(String nombre) {  
        this.nombre = nombre;  
        this.vidaBase = 2000;  
    }  
  
    public void añadirMasVidaBase(int vidaExtra) {  
        this.vidaBase += vidaExtra;  
    }  
}
```

#### 1.3.2.1 Pierna (extends Pieza)

La clase pierna solo agrega un “powerUp” de velocidad

```
public class Pierna extends Pieza {  
  
    private int velocidad;  
  
    public Pierna(String nombre, int velocidad) {  
        super(nombre);  
        this.velocidad = velocidad;  
        // TODO Auto-generated constructor stub  
    }  
}
```

### 1.3.2.2 Cabeza (extends Pieza)

Para la clase de “Cabeza” añade velocidad + vida al robot

```
public class Cabeza extends Pieza {  
  
    private int velocidad;  
    private int vida;  
  
    public Cabeza(String nombre, int velocidad, int vida) {  
        super(nombre);  
        this.velocidad = velocidad;  
        this.vida = vida;  
    }  
}
```

### 1.3.2.3 Brazo

```
public class Brazo extends Pieza {  
  
    private int ataque;  
  
    public Brazo(String nombre, int ataque) {  
        super(nombre);  
        this.ataque = ataque;  
    }  
}
```

### 1.3.2.4. Tórax

```
public class Torax extends Pieza {  
  
    private int vida;  
  
    public Torax(String nombre, int vida) {  
        super(nombre);  
        this.vida = vida;  
        // TODO Auto-generated constructor stub  
    }  
}
```

### 1.3.2.5. Arma

Con el tema del arma decidí usarla como un extend de Pieza, porque al mostrar las estadísticas todo el ataque también debe estar en la suma, en el caso de la vida se debe de obviar.

```
public class Arma extends Pieza{  
  
    private int daño;  
    private int velocidadDeAtaque;  
  
    public Arma(String nombre, int daño, int velocidadAtaque) {  
        super(nombre);  
        this.daño = daño;  
        this.velocidadDeAtaque = velocidadAtaque;  
    }  
}
```

### 1.3.3 Lista Piezas

Considerando cómo funcionan las ArrayList donde puedes añadir una gran cantidad de objetos decidí que sería más controlable una clase contenedora ya que a esta desde un principio puedo darle un tamaño fijo que en este caso sería de 5 además el otro beneficio que me da usar contenedores es poder crear métodos que sean específicos para implementar en el programa y no genéricos como en la ArrayList

```
public class ListaPiezas {  
  
    private int max;  
    private int cantidad;  
    private Pieza[] lista;  
  
    public ListaPiezas(int max) {  
        this.max = max;  
        this.cantidad = 0;  
        this.lista = new Pieza[max];  
    }  
  
    public boolean agregarPieza(Pieza p) {  
  
    public boolean eliminar(String nombre) {  
  
    public boolean verificarExiste(Pieza p) {  
  
    public Pieza getIndex(int i) {  

```

Agregar Pieza:

```
    public boolean agregarPieza(Pieza p) {  
        if (cantidad < max) {  
            lista[cantidad] = p;  
            cantidad++;  
            return true;  
        }  
        return false;  
    }
```

Eliminar:

```
public boolean eliminar(String nombre) {  
  
    Pieza[] listaAux = new Pieza[max];  
    int agregar = 0;  
  
    for(int i = 0; i < cantidad ; i++) {  
        if(!lista[i].getNombre().equalsIgnoreCase(nombre)) {  
            listaAux[agregar] = lista[i];  
            agregar++;  
        }  
    }  
  
    cantidad = agregar;  
    this.lista = listaAux;  
    return true;  
}
```

Verificar Existe:

```
public boolean verificarExiste(Pieza p) {  
    for (int i = 0; i < cantidad; i++) {  
        if (lista[i].equals(p)) {  
            return true;  
        }  
    }  
    return false;  
}
```

Get index:

```
public Pieza getIndex(int i) {  
    return lista[i];  
}
```

### 1.3.4. Lista Robots

```
public class ListaRobots {  
  
    private int max;  
    private int cantidad;  
    private Robot[] lista;  
  
    public ListaRobots(int max) {  
        this.max = max;  
        cantidad = 0;  
        lista = new Robot[max];  
    }  
  
    public boolean agregarRobot(Robot r) {
```

Agregar Robot:

```
        if(cantidad < max) {  
            lista[cantidad] = r;  
            cantidad++;  
            return true;  
        }return false;  
    }
```

### 1.3.5. Piloto

En la clase piloto tendrá una lista robot con un máximo de 10 (solo le puse 10 pero perfectamente podrían ser mas o menos, es solo para darle un tamaño)

```
public class Piloto {  
  
    private String piloto;  
    private ListaRobots lista;  
    private Equipo equipo;  
  
    public Piloto(String piloto) {  
        super();  
        this.piloto = piloto;  
        this.equipo = null;  
        lista = new ListaRobots(10);  
    }  
}
```

### 1.3.6. Robot

En la clase robto se define si o si la lista piezas en 5 ya que el robot solo puede portar contando en piezas y arma un total de 5 obejetos

```
public class Robot {  
  
    private String nombre;  
    private Estadisticas estadisticas;  
    private ListaPiezas lista;  
  
    public Robot(String nombre) {  
        this.nombre = nombre;  
        lista = new ListaPiezas(5);  
    }  
}
```

### 1.3.7. Equipo

En el caso de la clase equipos es lo mismo que en la clase piloto igualmente definí la lista robot en 10 pero puede ser otra cantidad

```
public class Equipo {  
  
    private String nombreEquipo;  
    private ListaRobots lista;  
    private Piloto piloto;  
  
    public Equipo(String nombreEquipo) {  
        this.nombreEquipo = nombreEquipo;  
        lista = new ListaRobots(10);  
    }  
}
```

### 1.3.8. Estadísticas

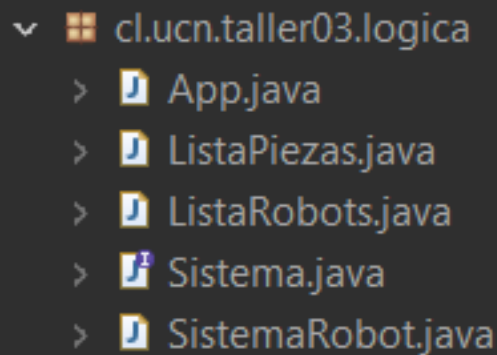
```
public class Estadisticas {  
  
    private int ataque;  
    private int vida;  
    private int velocidad;  
    private int velocidadAtaque;  
    private int daño;  
    private Robot robot;  
  
    public Estadisticas(Robot r) {  
        this.robot = r;  
    }  
}
```

### 1.3.9. GuardarTxt

```
public class GuardarTxt {  
  
    public void guardarTxtPiezas(List<Pieza> lista) throws IOException {..  
    public void guardarTxtArmas(List<Pieza> listaPiezas) throws IOException {..  
    public void guardarCombates(List<String> lista) throws IOException {..  
    public void guardarRobots(List<Robot> lista) throws IOException {..  
    private String generarPiezas(Robot r) {..  
}
```

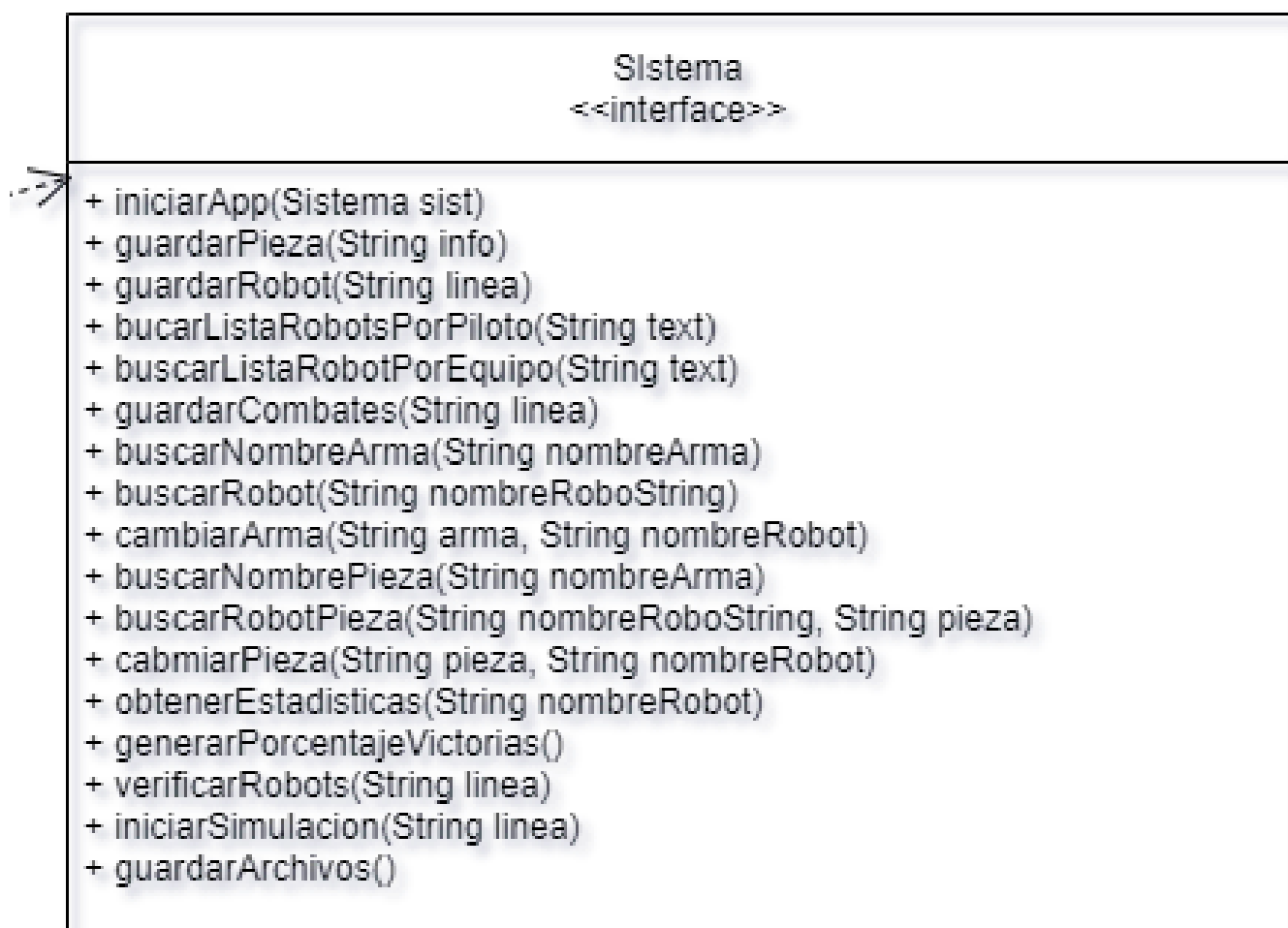


## 1.4 Interfaz



```
cl.ucn.taller03.logica
├── App.java
├── ListaPiezas.java
├── ListaRobots.java
├── Sistema.java
└── SistemaRobot.java
```

### 1.4.1 Sistem



**1.4.2. SistemaRobot** (implements Sistema) El orden en que se leerá esta parte, comenzara por cada contrato y desde ese contrato se seguirá la forma de ejecucion.

```
public class SistemaRobot implements Sistema {

    private List<Pieza> listaPiezas;
    private List<Piloto> listaPilotos;
    private List<Equipo> listaEquipos;
    private List<Robot> listaRobots;
    private List<String> registroCombates;
    private Inicio app;

    public SistemaRobot() {
        listaPiezas = new ArrayList<Pieza>();
        listaRobots = new ArrayList<Robot>();
        listaPilotos = new ArrayList<Piloto>();
        listaEquipos = new ArrayList<Equipo>();
        registroCombates = new ArrayList<String>();
    }
}
```

Iniciar interfaz gráfica “Inicio”

```
@Override
public void iniciarApp(Sistema sist) {
    app = new Inicio(sist);
    app.setVisible(true);
}
```

## Guardar Pieza:

El método guardar pieza discrimina si lo que se creará es una arma, brazo, tórax, etc... además cada vez que instancia y guarda el objeto correspondiente asigna la vida extra por la rareza y el ataque extra igualmente.

```
@Override
public void guardarPieza(String info) {

    String[] datos = info.split(",");
    String nombre = datos[0];
    String tipoDePieza = datos[2];

    int vidaExtra = generarVidaExtra(datos[1]);
    // Define un extra de puntos de vida dependiendo la rareza de la
    // pieza
    int dañoExtra = generarDañoExtra(datos[1]);
    // Define un extra de ataque dependiendo la rareza de la pieza

    Pieza buscar = verificarPieza(nombre);

    if (buscar == null) {
        // CREAR Y GUARDAR EL ARMA
        // INGRESADA////////////////////////////////////
        if (datos.length == 3) {
            int daño = Integer.parseInt(datos[1]);
            int velocidadAtaque = Integer.parseInt(datos[2]);

            Pieza nuevaArma = new Arma(nombre, daño, velocidadAtaque);
            listaPiezas.add(nuevaArma);

            // CREA Y GUARDA PIERNAS////////////////////////////////////
        } else if (tipoDePieza.equals("P")) {

            // nombre
            String rareza = datos[1];
            // tipo de pieza
            int velocidad = Integer.parseInt(datos[3]);

            Pieza nuevaPierna = new Pierna(nombre, velocidad);

            nuevaPierna.añadirMasVidaBase(vidaExtra);
            nuevaPierna.setRareza(rareza);
            nuevaPierna.setAtaqueExtra(dañoExtra);

            listaPiezas.add(nuevaPierna);

            // CREA Y GUARDA BRAZOS////////////////////////////////////
        } else if (tipoDePieza.equals("B")) {

            // nombre
            String rareza = datos[1];
            // tipo de pieza
            int ataque = Integer.parseInt(datos[3]);

            Pieza nuevoBrazo = new Brazo(nombre, ataque);

            nuevoBrazo.añadirMasVidaBase(vidaExtra);
            nuevoBrazo.setRareza(rareza);
            nuevoBrazo.setAtaqueExtra(dañoExtra);

            listaPiezas.add(nuevoBrazo);

            // CREA Y GUARDA TORAX////////////////////////////////////
        } else if (tipoDePieza.equals("T")) {

            // nombre
            String rareza = datos[1];
            // tipo de pieza
            int vida = Integer.parseInt(datos[3]);

            Pieza nuevoTorax = new Torax(nombre, vida);

            nuevoTorax.añadirMasVidaBase(vidaExtra);
            nuevoTorax.setRareza(rareza);
            nuevoTorax.setAtaqueExtra(dañoExtra);

            listaPiezas.add(nuevoTorax);

            // CREA Y GUARDA UNA CABEZA //////////////////////////////////////
        } else if (tipoDePieza.equals("C")) {

            // nombre = datos[0]
            String rareza = datos[1];
            // tipo de pieza = datos[2]
            int velocidad = Integer.parseInt(datos[3]);
            int vida = Integer.parseInt(datos[4]);

            Pieza nuevaCabeza = new Cabeza(nombre, velocidad, vida);

            nuevaCabeza.añadirMasVidaBase(vidaExtra);
            nuevaCabeza.setRareza(rareza);
            nuevaCabeza.setAtaqueExtra(dañoExtra);

            listaPiezas.add(nuevaCabeza);

        }
    }

    /////////////////////////////////////////////////// RECUERDA BORRAR ESTA PARTE
    mostrarLista();
    System.out.println("Tamaño = " + listaPiezas.size());
}
```

Generar daño extra:

Dependiendo de la rareza de la pieza convierte ese valor a números, los que darán una ventaja al robot.

```
private int generarDañoExtra(String tipoDePieza) {  
    if (tipoDePieza.equals("PP")) {  
        return 200;  
    } else if (tipoDePieza.equals("PE")) {  
        return 100;  
    } else {  
        return 0;  
    }  
}
```

Generar vida extra:

Al igual que el anterior método entrega puntos extras por la rareza de la pieza

```
// Generar estadísticas nuevas  
private int generarVidaExtra(String tipo) {  
    if (tipo.equals("PP")) {  
        return 3000;  
    } else if (tipo.equals("PE")) {  
        return 2000;  
    } else {  
        return 1000;  
    }  
}
```

Guardar robot:

Genera un robot de un alien y lo agrega a lista robots

Genera un robot humano con la condición de que no esté registrado, de lo contrario setea al equipo y al

En esta parte crea y designa al piloto, pero si ya existe se agrega al robot a la lista propia del piloto, pasa lo mismo con el equipo

```
public void guardarRobot(String linea) {

    String[] datos = linea.split(",");

    String nombreRobot = datos[0];
    String tipoRobot = datos[6];

    // Se crean al piloto y al Equipo a la vez

    if (tipoRobot.equals("A")) {
        String claseRobot = datos[7];
        int[] powerUp = puntosExtrasVidaDaño(claseRobot);

        if (verificarRobot(nombreRobot) == null) {
            Robot nuevo = new RobotAlien(nombreRobot, claseRobot, powerUp[0], powerUp[1]);
            designarPiezasRobot(nuevo, datos);
            listaRobots.add(nuevo);
        }
    } else if (tipoRobot.equals("H")) {
        String nombrePiloto = datos[7];
        String nombreEquipo = datos[8];

        Piloto nuevoPiloto = verificarPiloto(nombrePiloto);
        Equipo nuevoEquipo = verificarEquipo(nombreEquipo);

        Robot nuevo = verificarRobot(nombreRobot);
        if (nuevo == null) {
            Robot nuevoRobot = new RobotHumano(nombreRobot);

            if (nuevoPiloto == null) {
                nuevoPiloto = new Piloto(nombrePiloto);
                nuevoPiloto.getLista().agregarRobot(nuevoRobot);
            } else {
                nuevoPiloto.getLista().agregarRobot(nuevoRobot);
            }

            if (nuevoEquipo == null) {
                nuevoEquipo = new Equipo(nombreEquipo);
                nuevoEquipo.getLista().agregarRobot(nuevoRobot);
            } else {
                nuevoEquipo.getLista().agregarRobot(nuevoRobot);
            }

            nuevoPiloto.setEquipo(nuevoEquipo);
            nuevoEquipo.setPiloto(nuevoPiloto);

            RobotHumano setearRobot = (RobotHumano) nuevoRobot;
            setearRobot.setEquipoMantencion(nuevoEquipo);
            setearRobot.setNombrePiloto(nuevoPiloto);

            designarPiezasRobot(nuevoRobot, datos);

            listaRobots.add(nuevoRobot);
            listaPilotos.add(nuevoPiloto);
            listaEquipos.add(nuevoEquipo);
        } else {
            RobotHumano setearRobot = (RobotHumano) nuevo;

            if (nuevoPiloto == null) {
                nuevoPiloto = new Piloto(nombrePiloto);
                nuevoPiloto.getLista().agregarRobot(setearRobot);
            } else {
                nuevoPiloto.getLista().agregarRobot(setearRobot);
            }

            if (nuevoEquipo == null) {
                nuevoEquipo = new Equipo(nombreEquipo);
                nuevoEquipo.getLista().agregarRobot(setearRobot);
            } else {
                nuevoEquipo.getLista().agregarRobot(setearRobot);
            }

            nuevoPiloto.setEquipo(nuevoEquipo);
            nuevoEquipo.setPiloto(nuevoPiloto);

            setearRobot.setEquipoMantencion(nuevoEquipo);
            setearRobot.setNombrePiloto(nuevoPiloto);

            listaRobots.add(setearRobot);
        }
    }

    System.out.println(listaRobots.toString());
}
```

Verificar robot:

```
private Robot verificarRobot(String nombre) {  
    for (Robot r : listaRobots) {  
        if (r.getNombre().equals(nombre)) {  
            return r;  
        }  
    }  
    return null;  
}
```

Buscar lista robot por nombre del piloto:

Este método solo verifica si existe el piloto, en el caso de existir llama al siguiente método

```
public String bucarListaRobotsPorPiloto(String text) {  
  
    Piloto buscarPilot = buscarPiloto(text);  
  
    if (buscarPilot == null) {  
        return "Piloto no encontrado!";  
    } else {  
        return generarListaStringRobots(buscarPilot);  
    }  
  
}
```

Generar lista:

Este método esta encargado de generar el string completo de todos los robots que tenga el piloto ingresado

```
private String generarListaStringRobots(Piloto p) {  
  
    String lineaString = "Nombre del Piloto: " + p.getPiloto() + " \n";  
  
    for (int i = 0; i < p.getLista().getCantidad(); i++) {  
        lineaString += ("Robot [" + (i + 1) + "] " + p.getLista().index(i).getNombre() + "\n");  
    }  
  
    return lineaString;  
  
}
```

Generar lista por equipo:

Es la misma lógica que se realizó en el anterior método

```
public String buscarListaRobotPorEquipo(String text) {
    Equipo buscarEquipo = buscarEquipoRobot(text);

    if (buscarEquipo == null) {
        return "Equipo no encontrado";
    } else {
        return generarListaStringRobotsEquipo(buscarEquipo);
    }
}
```

Generar lista de robot por equipo:

```
private String generarListaStringRobotsEquipo(Equipo buscarEquipo){

    String datosString = "Nombre del equipo: " + buscarEquipo.getNombreEquipo() + "\n";

    for (int i = 0; i < buscarEquipo.getLista().getCantidad(); i++) {

        datosString += "Robot[" + (i + 1) + "]" + buscarEquipo.getLista().index(i).getNombre() + "\n";

    }
    return datosString;
}
```

Guardar Combate:

Guarda toda la línea del archivo de texto

```
public void guardarCombates(String linea) {
    registroCombates.add(linea);
}
```

Buscar Arma:

Se encarga de avisar por la interfaz grafica si esa arma ingresada es usada por algún robot, o si esta disponible o si no existe.

```
public String buscarNombreArma(String nombreArma) {

    for (Robot r : listaRobots) {
        for (int i = 0; i < r.getList().getCantidad(); i++) {
            Pieza buscar = r.getList().getIndex(i);
            if (buscar instanceof Arma) {
                if (buscar.getNombre().equalsIgnoreCase(nombreArma)) {
                    return "Usada por: " + r.getNombre();
                }
            }
        }
    }

    for (Pieza p : listaPiezas) {
        if (p instanceof Arma) {
            if (p.getNombre().equalsIgnoreCase(nombreArma)) {
                return "Disponible";
            }
        }
    }

    return "No encontrada";
}
```

Buscar Robot:

Muestra por la interfaz gráfica el arma que esta usando el robot ingresado

```
public String buscarRobot(String nombreRoboString) {

    String arma = "Null";

    for (Robot r : listaRobots) {
        if (r.getNombre().equalsIgnoreCase(nombreRoboString)) {
            for (int i = 0; i < r.getList().getCantidad(); i++) {
                Pieza buscar = r.getList().getIndex(i);
                if (buscar instanceof Arma) {
                    arma = buscar.getNombre();
                }
            }

            return "Find- Arma Inst.: " + arma;
        }
    }

    return "No encontrado";
}
```



Cambiar arma:

Cambia las referencias de las armas en el robot

Busca el arma y busca el robot para luego por cada lista interna de los robots buscar el arma y eliminar su referencia.

Al final asignará el arma al robot

```
public void cambiarArma(String arma, String nombreRobot) {  
  
    // Eliminar el arma de donde este en los robots  
  
    Robot cambiar = null;  
    Pieza armaEncontrada = null;  
  
    for (Robot r : listaRobots) {  
        if (r.getNombre().equalsIgnoreCase(nombreRobot)) {  
            cambiar = r;  
        }  
    }  
  
    for (Pieza p : listaPiezas) {  
        if (p.getNombre().equalsIgnoreCase(arma)) {  
            armaEncontrada = p;  
        }  
    }  
  
    for (Robot r : listaRobots) {  
        if (r.getList().verificarExiste(armaEncontrada)) {  
            r.getList().eliminar(armaEncontrada.getNombre());  
        }  
    }  
  
    cambiar.getList().agregarPieza(armaEncontrada);  
  
}
```

Buscar pieza:

Genera un string con el nombre del robot en donde esta instalado

```
public String buscarNombrePieza(String nombrePieza) {  
  
    for (Robot r : listaRobots) {  
        for (int i = 0; i < r.getList().getCantidad(); i++) {  
            Pieza buscar = r.getList().getIndex(i);  
            if (!(buscar instanceof Arma)) {  
                if (buscar.getNombre().equalsIgnoreCase(nombrePieza)) {  
                    return "Usada por: " + r.getNombre();  
                }  
            }  
        }  
    }  
  
    for (Pieza p : listaPiezas) {  
        if (!(p instanceof Arma)) {  
            if (p.getNombre().equalsIgnoreCase(nombrePieza)) {  
                return "Disponible";  
            }  
        }  
    }  
  
    return "No encontrada";  
  
}
```

Buscar en el robot el mismo tipo de pieza ingresada:

Compara las clases de la pieza ingresada en el caso de coincidir muestra el nombre de la pieza instalada en el robot  
ejemplo si en el robot hay un brazo instalado y la clase del brazo coincide con la clase de la pieza encontrada genera el string para verlo en la interfaz grafica

```
public String buscarRobotPieza(String nombreRoboString, String pieza) {  
    String arma = "Null";  
  
    Pieza buscarPieza = null;  
  
    for (Pieza p : listaPiezas) {  
        if (p.getNombre().equalsIgnoreCase(pieza)) {  
            buscarPieza = p;  
        }  
    }  
  
    for (Robot r : listaRobots) {  
        if (r.getNombre().equalsIgnoreCase(nombreRoboString)) {  
            for (int i = 0; i < r.getLista().getCantidad(); i++) {  
                if (!(r.getLista().getIndex(i) instanceof Arma)) {  
                    if (r.getLista().getIndex(i).getClass() == buscarPieza.getClass()) {  
                        System.out.println("NOMBRE PIEZA: " + r.getLista().getIndex(i).getNombre());  
                        arma = r.getLista().getIndex(i).getNombre();  
                    }  
                }  
            }  
        }  
        return "Find- Pieza Inst.: "+arma;  
    }  
}  
  
return "No encontrado";
```

Cambiar pieza:

Genera la misma función que al cambiar de arma.

```
public void cabmiarPieza(String pieza, String nombreRobot) {

    // Eliminar la pieza de donde este en los robots

    Robot cambiar = null;
    Pieza armaEncontrada = null;

    for (Pieza p : listaPiezas) {
        if (p.getNombre().equalsIgnoreCase(pieza)) {
            armaEncontrada = p;
        }
    }

    for (Robot r : listaRobots) {
        if (r.getNombre().equalsIgnoreCase(nombreRobot)) {
            cambiar = r;
            for(int i = 0; i < r.getLista().getCantidad(); i++) {
                if(r.getLista().getIndex(i).getClass() == armaEncontrada.getClass()) {
                    r.getLista().eliminar(r.getLista().getIndex(i).getNombre());
                }
            }
        }
    }

    for (Robot r : listaRobots) {
        if (r.getLista().verificarExiste(armaEncontrada)) {
            r.getLista().eliminar(armaEncontrada.getNombre());
        }
    }

    cambiar.getLista().agregarPieza(armaEncontrada);

}
```

Obtener Estadísticas:

```
public String obtenerEstadisticas(String nombreRobot) {
    String datos = "No encontrado!";

    for (Estadisticas e : listaEstadisticas) {
        if (e.getRobot().getNombre().equalsIgnoreCase(nombreRobot)) {

            String nombreR = e.getRobot().getNombre();
            int vida = e.getVida();
            int ataque = e.getAtaque();
            int velocidad = e.getVelocidad();
            int velocidadAtaque = e.getVelocidadAtaque();
            int daño = e.getDaño();

            datos = nombreR + ": \n" + "Vida Total : " + vida + "\nAtaque : " + ataque + "\nVelocidad : "
                + velocidad + "\nVelocidad de ataque : " + velocidadAtaque + "\nDaño : " + daño;

        }
    }

    return datos;
}
```

Generar Porcentaje de Victorias:

```
public String generarPorcentajeVictorias() {

    int totalRegistros = registroCombates.size();
    double humanos = 0;
    double alien = 0;

    for (String s : registroCombates) {
        String datos[] = s.split(",");
        if (datos[6].equalsIgnoreCase("H")) {
            humanos++;
        } else if (datos[6].equalsIgnoreCase("A")) {
            alien++;
        }
    }

    double porcentaje = (humanos / totalRegistros) * 100;
    double porcentajeAlie = (alien / totalRegistros) * 100;
    return "Porcentaje de victorias: \nHumanos: " + porcentaje + "%\nAlien: " + porcentajeAlie + "%";
}
```

Guardar Txt

Guardar Pieza:

```
public void guardarTxtPiezas(List<Pieza> lista) throws IOException {

    List<Pieza> listaPiezas = new ArrayList<Pieza>();

    FileWriter piezas = new FileWriter("src\\cl\\ucn\\taller03\\txt\\piezas.txt");

    for (Pieza p : lista) {
        if (!(p instanceof Arma)) {
            listaPiezas.add(p);
        }
    }

    int ultimafila = listaPiezas.size() - 1;

    for (int i = 0; i < listaPiezas.size(); i++) {

        Pieza aux = listaPiezas.get(i);

        if (aux instanceof Cabeza) {
            Cabeza c = (Cabeza) aux;

            String nombre = c.getNombre();
            String rareza = c.getRareza();
            String tipo = "C";
            int velocidad = c.getVelocidad();
            int vida = c.getVida();

            String linea = nombre + "," + rareza + "," + tipo + "," + velocidad + "," + vida

            ;

            if (ultimafila == i) {
                piezas.write(linea);
            } else {
                piezas.write(linea + "\n");
            }
        } else if (aux instanceof Pierna) {
            Pierna p = (Pierna) aux;

            String nombre = p.getNombre();
            String rareza = p.getRareza();
            String tipo = "P";
            int velocidad = p.getVelocidad();

            String linea = nombre + "," + rareza + "," + tipo + "," + velocidad;

            if (ultimafila == i) {
                piezas.write(linea);
            } else {
                piezas.write(linea + "\n");
            }
        } else if (aux instanceof Brazo) {
            Brazo b = (Brazo) aux;

            String nombre = b.getNombre();
            String rareza = b.getRareza();
            String tipo = "B";
            int ataque = b.getAtaque();

            String linea = nombre + "," + rareza + "," + tipo + "," + ataque;

            if (ultimafila == i) {
                piezas.write(linea);
            } else {
                piezas.write(linea + "\n");
            }
        } else if (aux instanceof Torax) {
            Torax t = (Torax) aux;

            String nombre = t.getNombre();
            String rareza = t.getRareza();
            String tipo = "T";
            int vida = t.getVida();

            String linea = nombre + "," + rareza + "," + tipo + "," + vida;

            if (ultimafila == i) {
                piezas.write(linea);
            } else {
                piezas.write(linea + "\n");
            }
        }
    }

    piezas.close();

}
```

## Guardar txt Armas

```
public void guardarTxtArmas(List<Pieza> listaPiezas) throws IOException {  
  
    List<Pieza> listaArmas = new ArrayList<Pieza>();  
  
    FileWriter armas = new FileWriter("src\\cl\\ucn\\taller03\\txt\\armas.txt");  
  
    for (Pieza p : listaPiezas) {  
        if (p instanceof Arma) {  
            listaArmas.add(p);  
        }  
    }  
  
    int ultimaLinea = listaArmas.size() - 1;  
  
    for (int i = 0; i < listaArmas.size(); i++) {  
  
        Arma a = (Arma) listaArmas.get(i);  
  
        String nombre = listaArmas.get(i).getNombre();  
        int daño = a.getDaño();  
        int velocidadAtaque = a.getVelocidadDeAtaque();  
  
        String linea = nombre + "," + daño + "," + velocidadAtaque;  
  
        if (ultimaLinea == i) {  
            armas.write(linea);  
        } else {  
            armas.write(linea + "\n");  
        }  
    }  
  
    armas.close();  
  
}
```

## Guardar Combates:

```
public void guardarCombates(List<String> lista) throws IOException {  
  
    int ultimaLinea = lista.size() - 1;  
  
    FileWriter combates = new FileWriter("src\\cl\\ucn\\taller03\\txt\\combates.txt"  
);  
  
    for (int i = 0; i < lista.size(); i++) {  
  
        if (ultimaLinea == i) {  
            combates.write(lista.get(i));  
        } else {  
            combates.write(lista.get(i) + "\n");  
        }  
    }  
  
    combates.close();  
  
}
```

```

public void guardarRobots(List<Robot> lista) throws IOException {

    int ultimaLinea = lista.size() - 1;

    FileWriter robots = new FileWriter("src\\cl\\ucn\\taller03\\txt\\robots.txt");

    for (int i = 0; i < lista.size(); i++) {

        Robot r = lista.get(i);

        String nombre = r.getNombre();

        if (r instanceof RobotHumano) {

            RobotHumano rHumano = (RobotHumano) r;

            String piezas = generarPiezas(r);
            String tipo = "H";
            String nombrePiloto = rHumano.getNombrePiloto().getPiloto();
            String equipo = rHumano.getEquipoMantencion().getNombreEquipo();

            String linea = nombre + piezas + "," + tipo + "," + nombrePiloto + "," + equipo;

            if (ultimaLinea == i) {
                robots.write(linea);
            } else {
                robots.write(linea + "\n");
            }

        } else if (r instanceof RobotAlien) {

            RobotAlien rAlien = (RobotAlien) r;

            String piezas = generarPiezas(r);
            String tipo = "A";
            String clase = rAlien.getClase();

            String linea = nombre + piezas + "," + tipo + "," + clase;

            if (ultimaLinea == i) {
                robots.write(linea);
            } else {
                robots.write(linea + "\n");
            }

        }

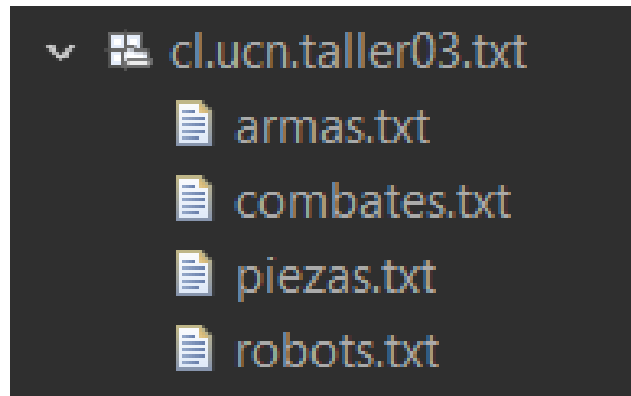
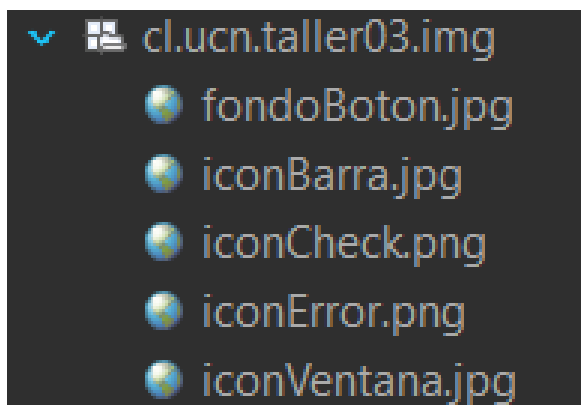
    }

    robots.close();

}

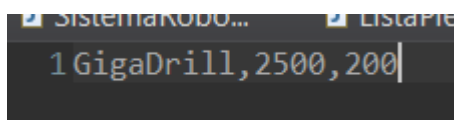
```

## 2. Archivos

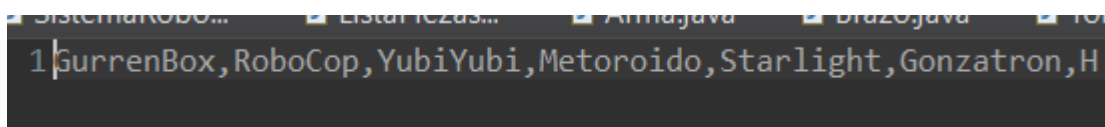


### 2.1. Txt

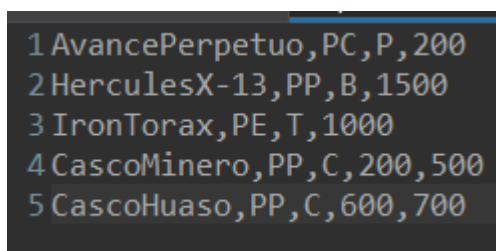
Arma:



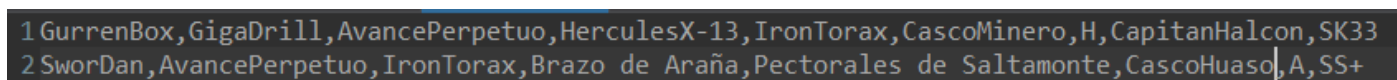
Combates:



Piezas

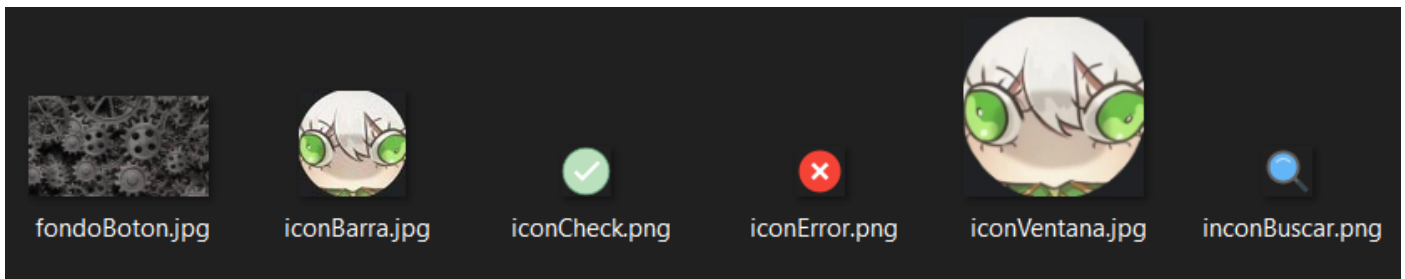


Arma





## 2.2 Imágenes




## 3. Interfaces Graficas

Interfaz de inicio



Añadir Pieza:

## Ensamblar Robot:

 Ensamblador

**Nombre Robot**

**Pierna**

**Brazo**

**Torax**

**Cabeza**

**Arma**

**Tipo**  
Robot Humano ▾

**Clase**  
SS+ ▾


**Piloto**

**Equipo**

Ensamblar

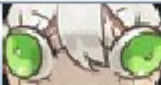
Atras

## Cambiar Pieza:

 Cambiar Pieza

**Ingresa la Pieza**


**Ingresa el Robot**

 Busc...

Cambiar Pieza

Atras

## Cambiar Arma

 Cambiar Arma

**Ingrese el Arma**

**Ingrese el Robot**

Buscar

Cambiar Arma

Atras

## Buscar robot por piloto:

 Buscar Robot

**Ingresa el nombre del Piloto**



Atras

Buscar robot por equipo:

 Buscar Robot

**Ingresa el nombre del Equipo**



**Atras**

Mostrar Estadísticas:

Buscar Robot

 Ingresa el nombre del Robot

Atras

Victorias:

%Victorias



Porcentaje de victorias:  
Humanos: 29.0%  
Alien: 71.0%


Atras

Simulación:

Simulacion

Robot Aliens	Robot Humanos
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>

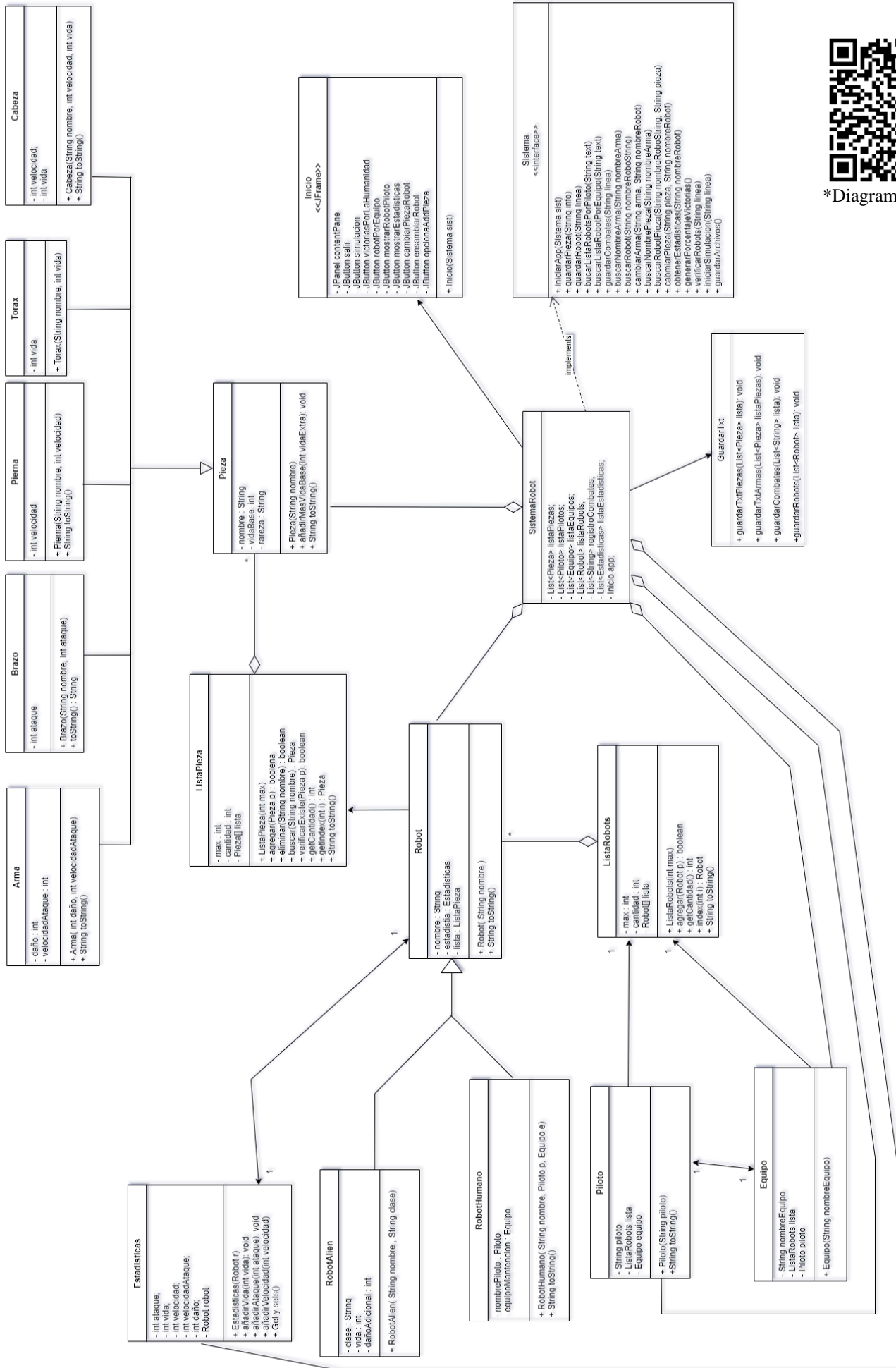
Verificar Robots

 Iniciar Simulacion

Ganador:

Atras

# Diagrama de Clases Versión Final:



\*Diagrama en Digital

## 4 GitHub

Se puede presionar el QR para acceder a GitHub web



main 1 branch 0 tags Go to file Add file <> Code

updateTxT

46006b0 1 minute ago 19 commits

folder	.settings	Initial commit	17 days ago
folder	bin/cl/ucn/taller03	updateTxT	1 minute ago
folder	src/cl/ucn/taller03	Ultima Update	1 hour ago
file	.classpath	Initial commit	17 days ago
file	.gitattributes	Initial commit	17 days ago
file	.project	Update	yesterday

Help people interested in this repository understand your project by adding a README.

Add a README