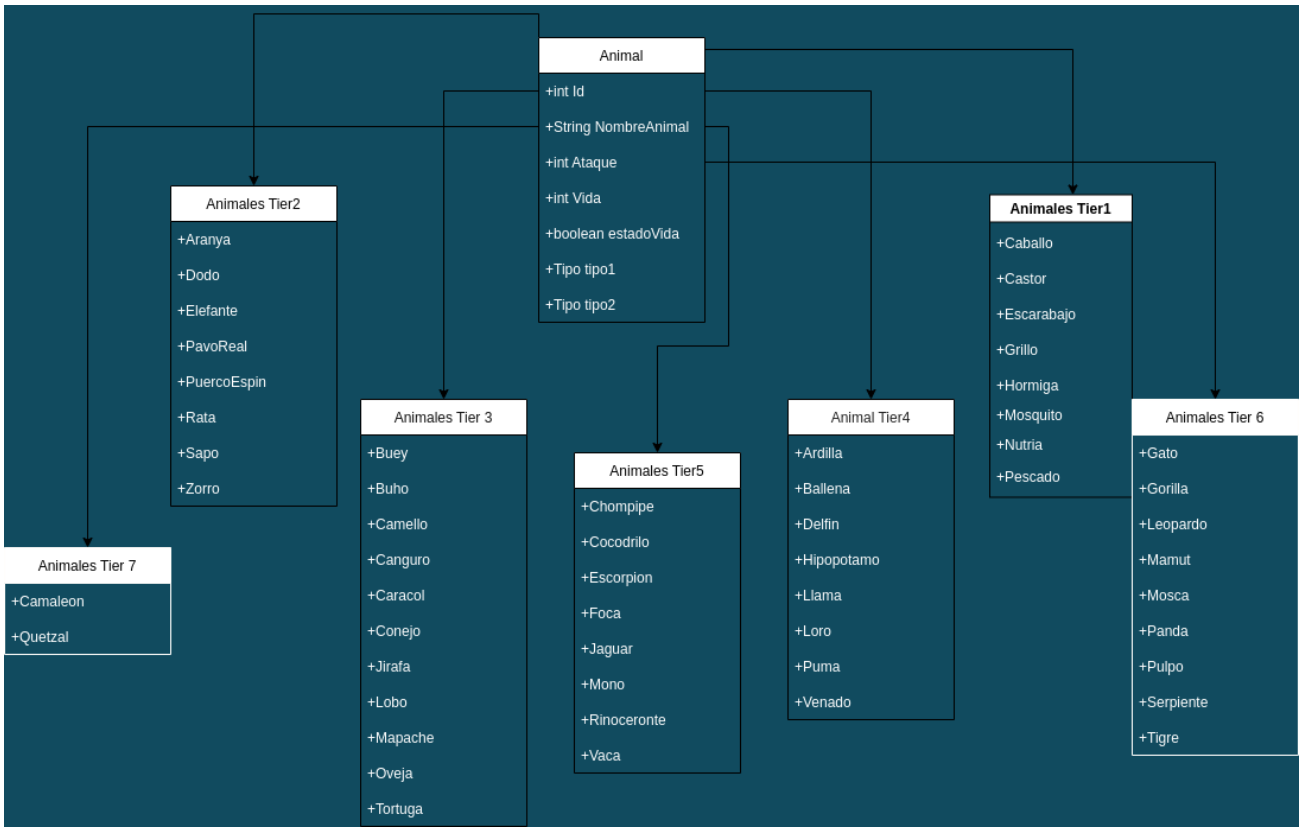
UNIVERSIDAD SAN CARLOS DE GUATEMALA
CENTRO UNIVERSITARIO DE OCCIDENTE.
DIVISION DE CIENCIAS DE LA INGENIERIA
CATEDRATICO: ING OLIVER SIERRA
CURSO: IPC1
SECCIÓN: B

MANUAL TÉCNICO PROYECTO 1

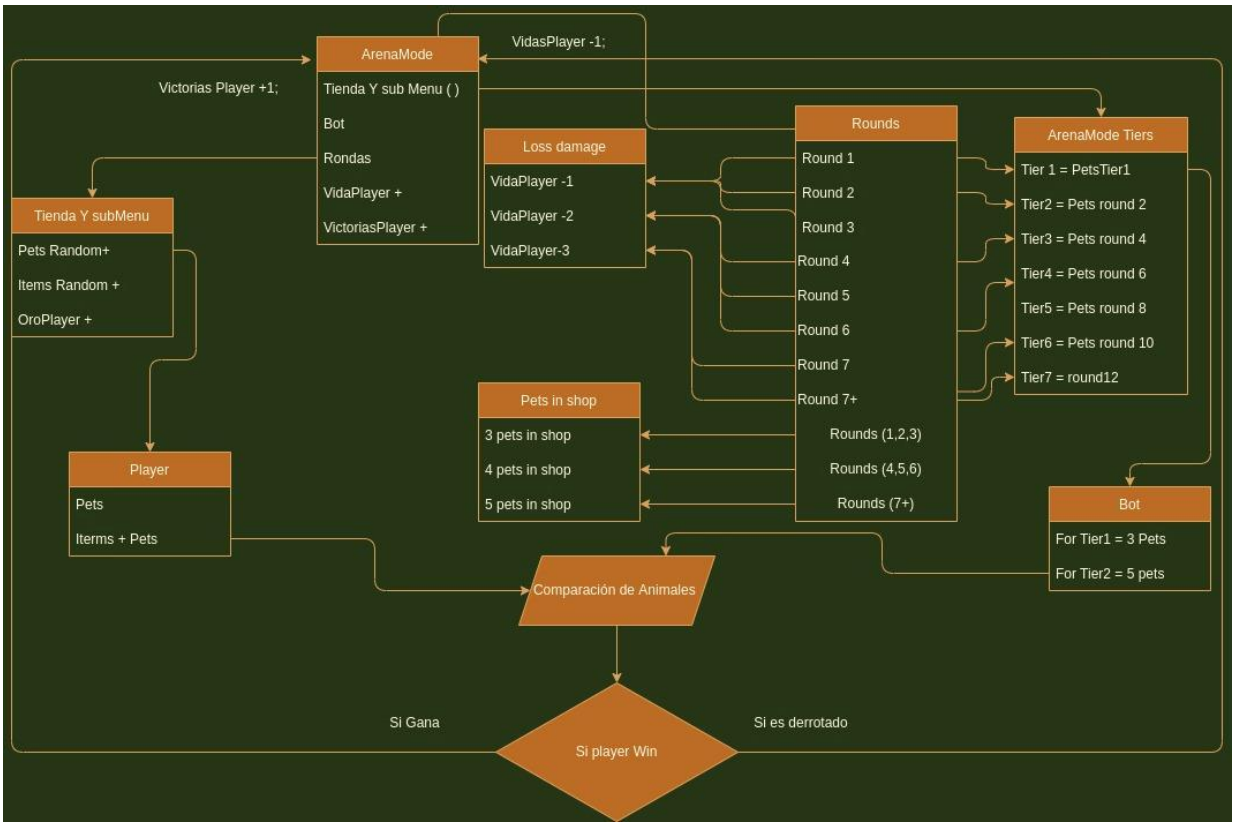
NOMBRES Y APELLIDOS
ENMER OSWALDO SANDOVAL MAZARIEGOS/ 202031773

Quetzaltenango, 18 de Abril de 2,022

PATRONES REPETITIVOS



Pudimos establecer una super clase heredada a todos los animales la super clase cuenta con un id está nos servirá para identificar a nuestra mascota, nombre del animal nos sirve para identificar que animal es, como el Grillo su nombre es Grillo como tal, un valor de tipo entero en Ataque estos valores ya están definidos desde las bases entonces las establecimos perfectamente, también un valor de tipo entero en Vida es la del animal como tal, una variable booleana de estado de vida nos ayudara a identificar si el animal está vivo o muerto, luego tenemos los tipos son clases usadas con polimorfismo donde estableces si el animal es de tipo acuático, terrestre entre otros.



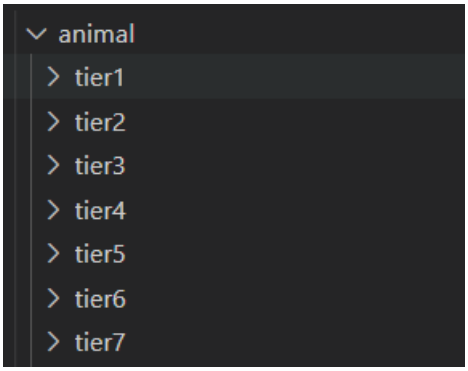
Dividimos los animales por Tiers para poder generarlos por rondas, organizamos perfectamente y creamos, tenemos la tienda donde generamos un animal random, con un item, la tienda tiene una su variable oro, que cada vez que se accede a ella siempre tenemos 10 de oro, Al estár en la ronda 1,2,3 generamos aleatoriamente 3 animales y si estamos en la ronda 4,5,6 generamos aleatoriamente 4 animales, luego de la 7 ronda en adelante generamos 5 animales en la tienda, cada uno tiene un valor de 3 y por generar aleatoriamente los animales se descuenta 1 de oro.

Animal.java

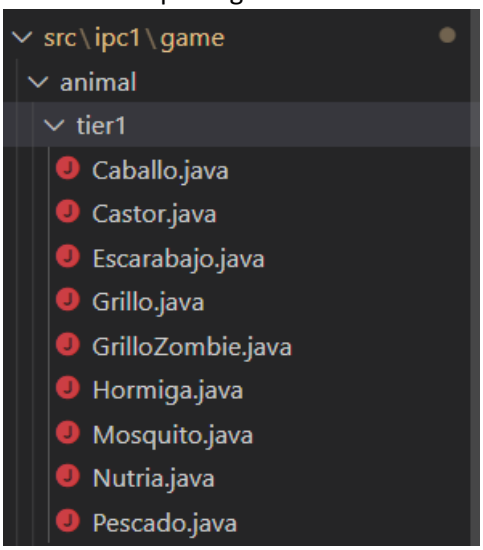
src > ipc1 > game > animal > Animal.java > {} ipc1.game.animal

```java
package ipc1.game.animal;

import ipc1.game.animal.tipos.Tipos;

public abstract class Animal implements Cloneable {
    private String nombreAnimal;
    private int ataque, vida, id;
    private double nivel;
    private boolean estadoVida;
    public Tipos tipo1, tipo2;

    // public enum Tipo {insecto, terrestre, acuatico, volador, mamifero, domestico,
    // solitario, desertico, reptil};
    // Grillo (tipo.insecto, terreste)

    public Animal() {

    }

    public Animal(int id, String nombreAnimal, int ataque, int vida, double nivel, boolean estadoVida, Tipos tipo1,
            Tipos tipo2) {
        this.id = id;
        this.nombreAnimal = nombreAnimal;
        this.ataque = ataque;
        this.vida = vida;
        this.nivel = nivel;
        this.estadoVida = estadoVida;
        this.tipo1 = tipo1;
        this.tipo2 = tipo2;
    }
```

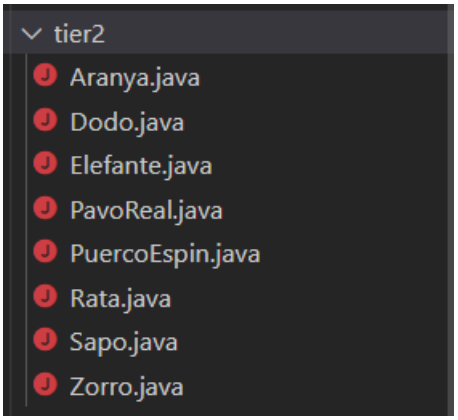
Está es nuestra suer clase con todos sus getters y setters de id, nombreAnimal, ataque, vida, nive, estadovida, tipo1 y tipo 2., tenemos que identificar perfectamente que estamos usando la Interfaz clonable, esto para que no apunte al mismo espacio en Memoria o para que no apunte al mismo Hash.



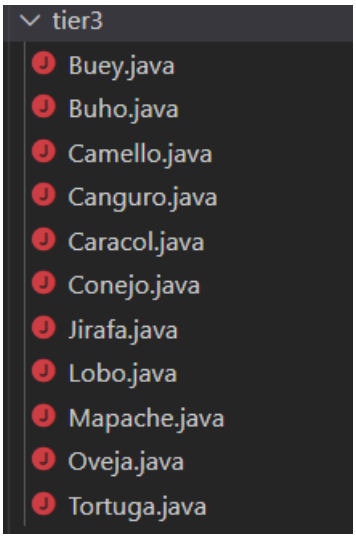
Tenemos los package de los Tier de Animales divididos por cada 1.



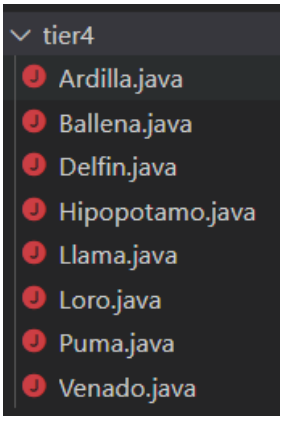
Nos encontramos con el Tier1 con las clases caballo, castor, escarabajo, grillo, grillozombie, hormiga, mosquito, nutria, pescado, todos estos heredados de la super clase Animal.



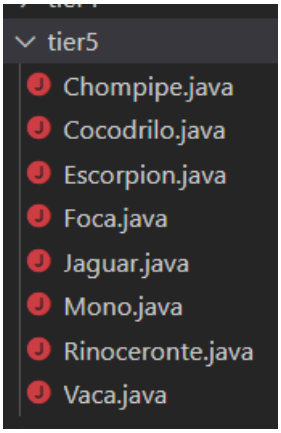
Nos encontramos con el Tier2 con las clases aranya, dodo, elefante, pavo real, puerco espin, rata, sapo, zorro todos estos heredados de la super clase Animal.



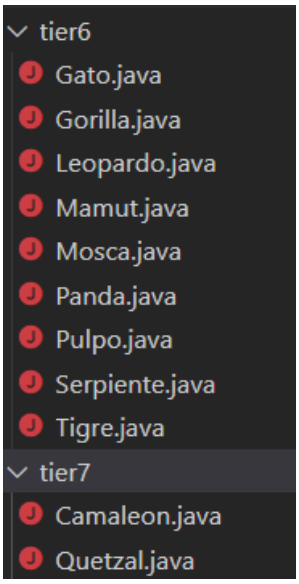
En el Tier3 encontramos a las clases de animales como el buey, buho, camello, canguro, caracol, conejo, jirafa, lobo, mapache, oveja, tortuga heredados de la superclase Animal.



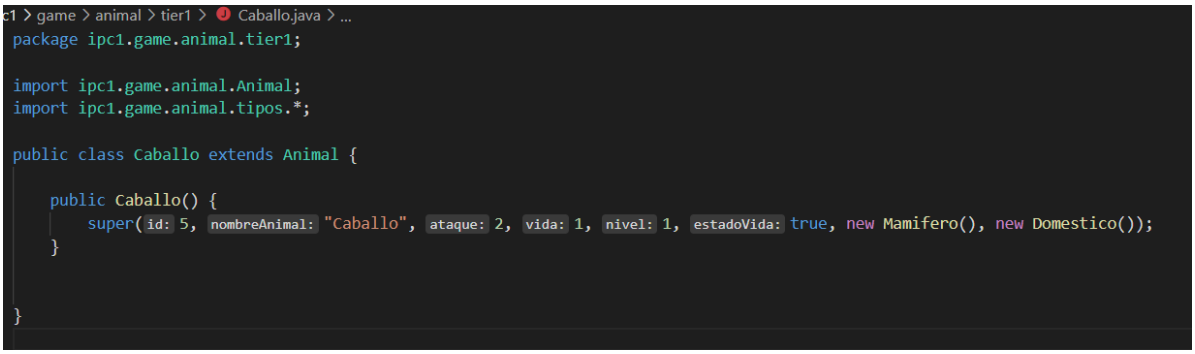
En el Tier4 tenemos las clases de animales como la ardilla, ballena, delfin, hipopotamo, llama, loro, puma, venados heredados de la superclase Animal.



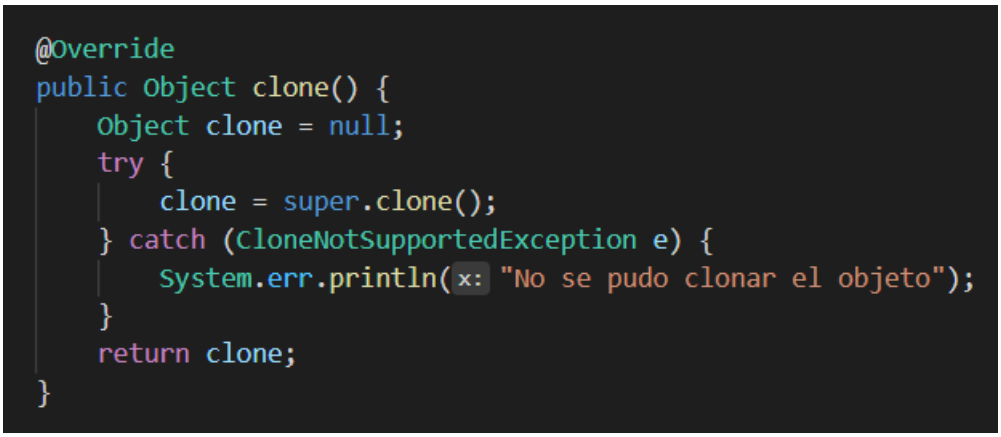
En el Tier5 nos encontramos a las clases Chompipe, cocodrilo, escorpion, foca, jaguar, mono, rinoceronte, vaca heredados de la super clase Animal.



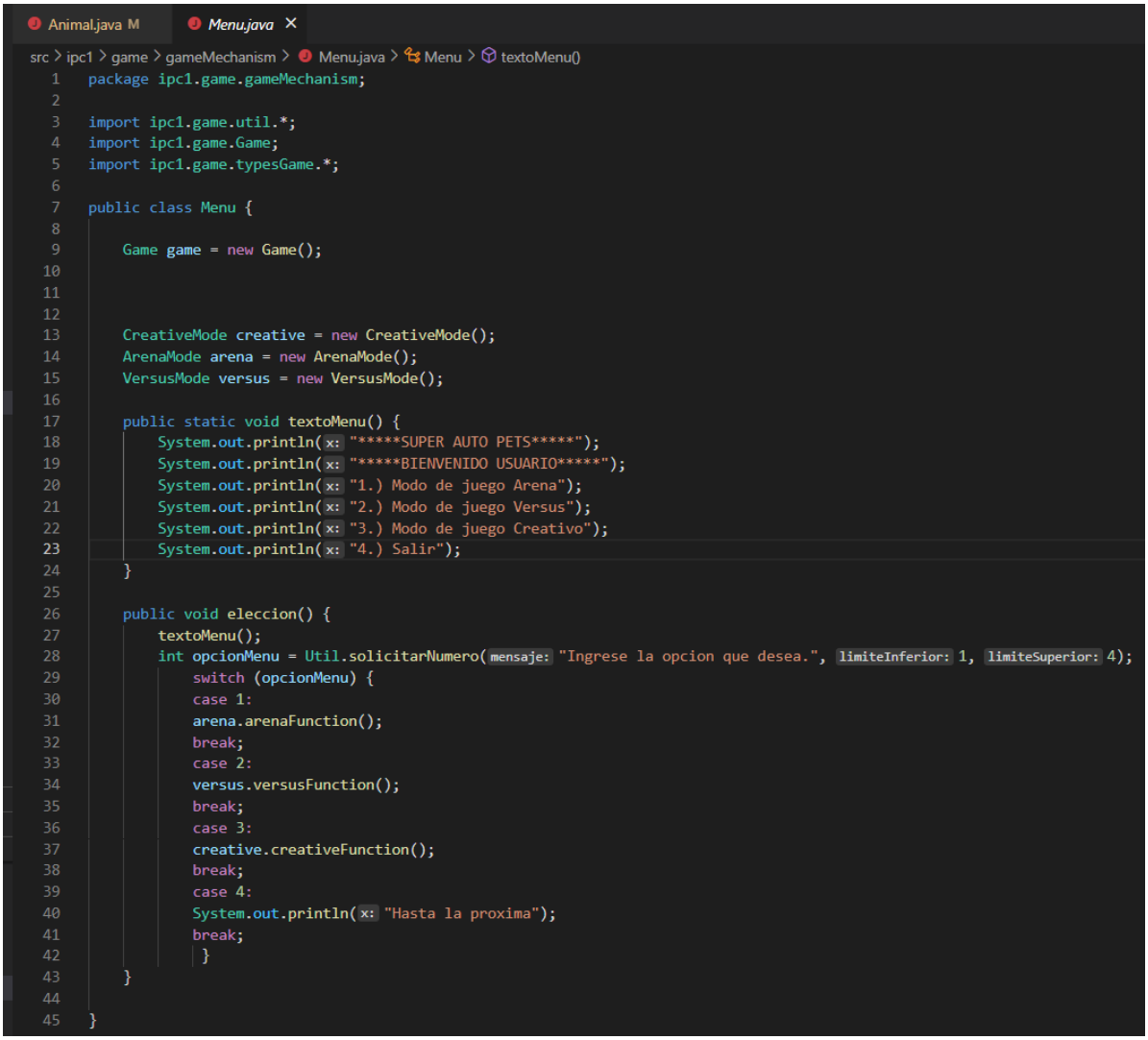
En el Tier6 nos encontramos con las clases heredadas de nuestra super clase Animal, como gato, gorila, leopardo, mamut, mosca, panda, pulpo, serpiente, tigre y en Tier 7 tenemos al Camaleon y el quetzal.



¨Para cada clase de cada animal realizamos el mismo proceso, exceptuando los datos que ya están definidos para cada animal, realizamos un constructor y modificamos la clase super.



En nuestra super clase hacemos un método que es el de la interfaz clone, nos servirá para no apuntar al mismo Hashcode o al mismo espacio en memoria visto y explicado en los diagramas UML.



En nuestra clase Menú llamamos a nuestros objetos como lo son creative, arena, versus nos ayudaran a referirnos a que modo de juego queremos y en está clase trabajamos lo que es el texto y funcionalidad del Menú.

```java
package ipc1.game.gameMechanism;


public class SubMenu {
    public static int opcionMenu, oro = 10, vidaPlayer = 10, victorias = 0;

    public SubMenu(){

    }

    public void texto(int vidaPlayer, int victorias){
        System.out.println( x: "*****TIENDA DE ANIMALES******");
        System.out.println("                                                    Tiene " + vidaPlayer +" vidas");
        System.out.println("                                                    Tiene " + oro +" de oro");
        System.out.println("                                                    Tiene " + victorias +" victorias");

    }
}
```

En nuestra clase sub-Menú incluimos un método básico que nos servirá para los 3 modos de juego.

Animal.java M | SubMenu.java M | *ArenaMode.java* ×

src > ipc1 > game > typesGame > ArenaMode.java > {} ipc1.game.typesGame

```java
package ipc1.game.typesGame;


public class ArenaMode {
    public ArenaMode(){

    }

    public void arenaFunction(){

```

Modo de juego Arena.

Animal.java M | SubMenu.java M | CreativeMode.java M ×

src > ipc1 > game > typesGame > CreativeMode.java > CreativeMode >

```java
package ipc1.game.typesGame;

public class CreativeMode {

    public static void creativoFuntion(){
    }

    public void creativeFunction(){

    }
}
```

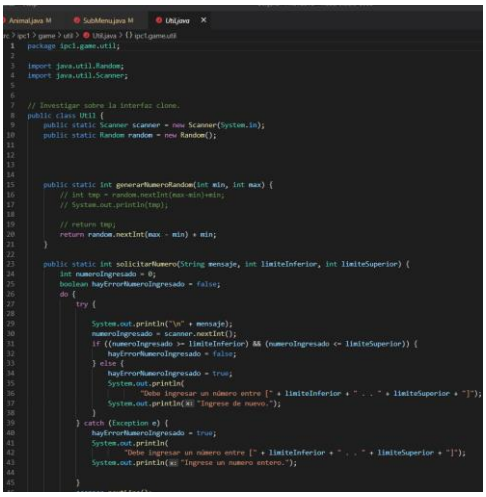
Modo de juego para Creativo.

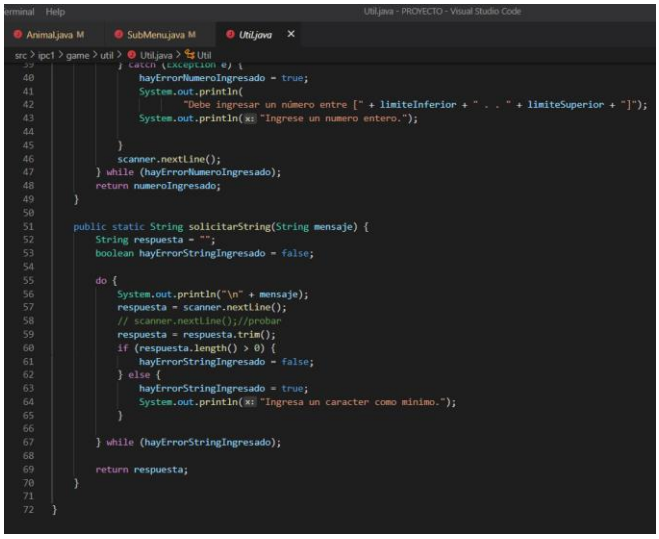
Animal.java M | SubMenu.java M | *VersusMode.java* ×

src > ipc1 > game > typesGame > VersusMode.java > { } ipc1.game.types

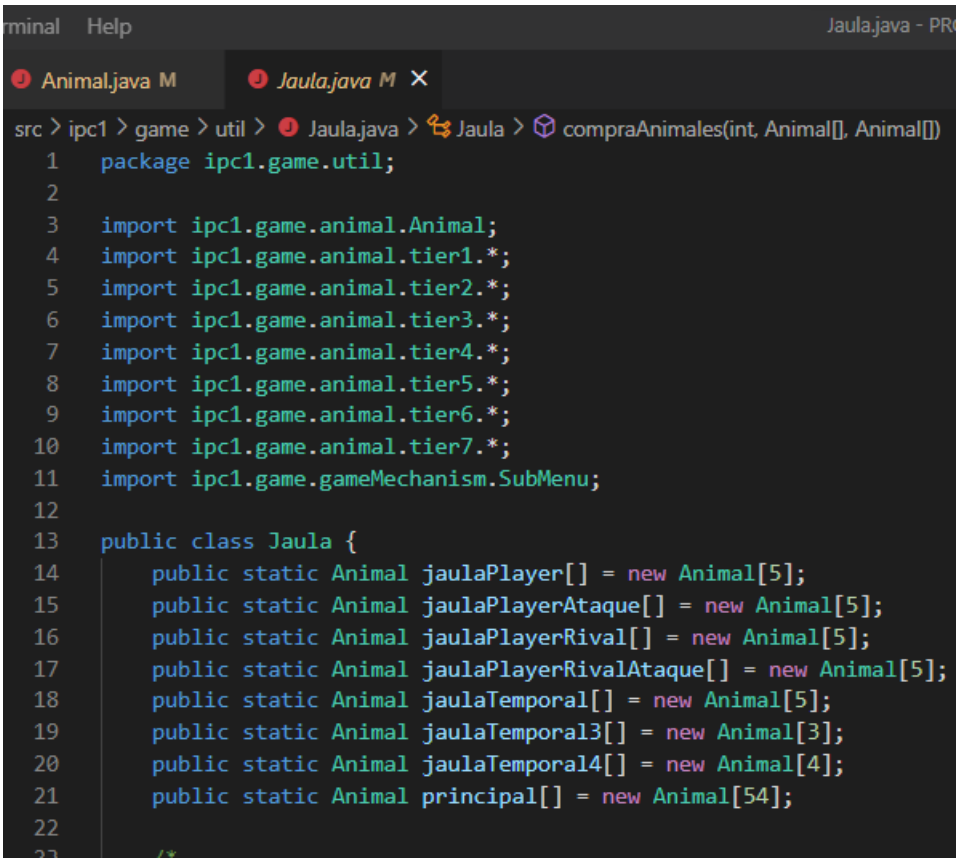
```java
package ipc1.game.typesGame;

import ipc1.game.util.Jaula;

public class VersusMode {
    private int rondas = 1;

    public VersusMode(){

    }

    public void versusFunction(){

        //System.out.println("Player 2");
        //Jaula.generarJaula(rondas);
        //Jaula.llenadoPlayerRival(rondas);
        Jaula.ataque(rondas);
    }
}

```

Modo de juego para el Versus.





Parte de la clase Util nos ayudara a solicitar un entero, o un String.



Nuestra clase Jaula, es la mayor parte de funcionalidad del juego, de aquí definimos los arreglos que nos servirán más adelante, el jaulaPlayer nos servirá para poder llenar nuestro arreglo de la tienda, el jaulaPlayerAtaque es una clonación del arreglo del equipo escogido y así mismo pasa con el rival, tenemos arreglos temporales para poder generarlos aleatoriamente.

```
public static Animal jaula[] = new Animal[] {
        principal[0] = new Hormiga(), principal[1] = new Pescado(), principal[2] = new Mosquito(),
        principal[3] = new Grillo(), principal[4] = new Castor(), principal[5] = new Caballo(),
        principal[6] = new Nutria(), principal[7] = new Escarabajo(), principal[8] = new Sapo(),
        principal[9] = new Dodo(), principal[10] = new Elefante(), principal[11] = new PuercoEspin(),
        principal[12] = new PavoReal(), principal[13] = new Rata(), principal[14] = new Zorro(),
        principal[15] = new Aranya(), principal[16] = new Camello(), principal[17] = new Mapache(),
        principal[18] = new Jirafa(), principal[19] = new Tortuga(), principal[20] = new Caracol(),
        principal[21] = new Oveja(), principal[22] = new Conejo(), principal[23] = new Lobo(),
        principal[24] = new Buey(), principal[25] = new Canguro(), principal[26] = new Buho(),
        principal[27] = new Venado(), principal[28] = new Loro(), principal[29] = new Hipopotamo(),
        principal[30] = new Delfin(), principal[31] = new Puma(), principal[32] = new Ballena(),
        principal[33] = new Ardilla(), principal[34] = new Llama(), principal[35] = new Foca(),
        principal[36] = new Jaguar(), principal[37] = new Escorpion(), principal[38] = new Rinoceronte(),
        principal[39] = new Mono(), principal[40] = new Cocodrilo(), principal[41] = new Vaca(),
        principal[42] = new Chompipe(), principal[43] = new Panda(), principal[44] = new Gato(),
        principal[45] = new Tigre(), principal[46] = new Serpiente(), principal[47] = new Mamut(),
        principal[48] = new Leopardo(), principal[49] = new Gorilla(), principal[50] = new Pulpo(),
        principal[51] = new Mosca(), principal[52] = new Quetzal(), principal[53] = new Camaleon()
};
```

Generamos nuestro arreglo Jaula, es el arreglo que tiente almacenado todas nuestras mascotas.

```java
private static void randomJaula(int tierMaximo, int cantidadAnimales) {
    int numrandom1 = 0;
    for (int j = 0; j < cantidadAnimales; j++) {
        numrandom1 = Util.generarNumeroRandom( min: 0, tierMaximo);
        if (numrandom1 < Jaula.jaula.length) {
            switch (cantidadAnimales) {
                case 3:
                    Jaula.jaulaTemporal3[j] = (Animal) Jaula.jaula[numrandom1].clone();
                    break;
                case 4:
                    Jaula.jaulaTemporal4[j] = (Animal) Jaula.jaula[numrandom1].clone();
                    break;
                case 5:
                    Jaula.jaulaTemporal[j] = (Animal) Jaula.jaula[numrandom1].clone();
                    break;
            }
        }
    }
    if (cantidadAnimales <= 3) {
        for (int i = 0; i < Jaula.jaulaTemporal3.length; i++) {
            System.out.println(i + "" + Jaula.jaulaTemporal3[i]);
        }
    } else if (cantidadAnimales <= 4) {
        for (int i = 0; i < jaulaTemporal4.length; i++) {
            System.out.println(i + "" + Jaula.jaulaTemporal4[i]);
        }
    } else if (cantidadAnimales <= 5) {
        for (int i = 0; i < jaulaTemporal.length; i++) {
            System.out.println(i + "" + Jaula.jaulaTemporal[i]);
        }
    }
}
```

Esté método nos ayuda a llenar los arreglos temporales que tenemos para generar automáticamente pasandole un TierMaximo y un int de Cantidad de Animales, esté método nos servirá más adelante.

```java
public static Animal[] getSeleccionArray(int rondas) {
    if (rondas <= 3) {
        return jaulaTemporal3;
    }
    if (rondas < 7) {
        return jaulaTemporal4;
    }
    return jaulaTemporal;
}
```

Tenemos esté método que nos devuelve el arreglo dependiendo en el numero de ronda que vayamos.

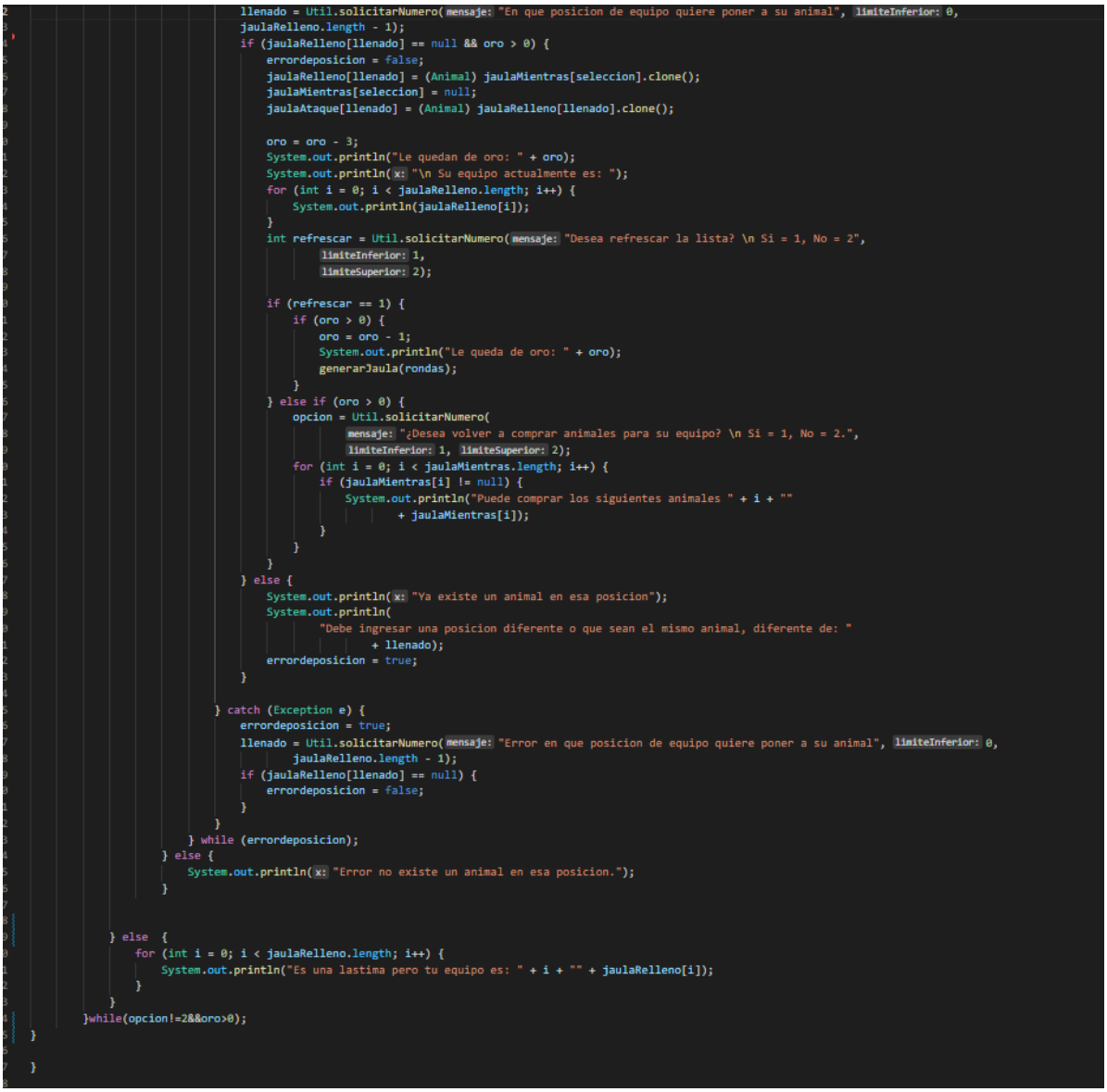
```java
public static void generarJaula(int rondas) {
    switch (rondas) {
        case 1:
            randomJaula( tierMaximo: 7,  cantidadAnimales: 3);
            break;
        case 2:
        case 3:
            randomJaula( tierMaximo: 15,  cantidadAnimales: 3);
            break;
        case 4:
        case 5:
            randomJaula( tierMaximo: 26,  cantidadAnimales: 4);
            break;
        case 6:
            randomJaula( tierMaximo: 34,  cantidadAnimales: 4);
            break;
        case 7:
            randomJaula( tierMaximo: 34,  cantidadAnimales: 5);
            break;
        case 8:
        case 9:
            randomJaula( tierMaximo: 42,  cantidadAnimales: 5);
            break;
        case 10:
        case 11:
            randomJaula( tierMaximo: 51,  cantidadAnimales: 5);
            break;
        case 12:
            randomJaula( tierMaximo: 53,  cantidadAnimales: 5);
            break;
    }
}
```

Tenemos esté método que en base al numero de rondas que tengamos llamamos al método randomJaula haciendole una comparación con un Switch dándole un numero de Tier Maximo que sería el máximo numero del animal por Tier y la cantidad de animales generados.

```java
public static void ventaAnimales(int rondas, Animal[] jaulaVacia, int oro) {
    if (rondas > 1) {
        int venta = Util.solicitarNumero( mensaje: "¿Desea vender algun animal de su equipo? \n Si = 1, No = 2.",  limiteInferior: 1,  limiteSuperior: 2);
        if (venta == 1) {
            for (int i = 0; i < jaulaVacia.length; i++) {
                if (jaulaVacia[i] != null) {
                    System.out.println(i + "" + jaulaVacia[i]);
                }
            }
            int ventaPos = Util.solicitarNumero( mensaje: "¿Que animal desea vender conforme al indice que se le dio?",  limiteInferior: 0,
                    jaulaVacia.length - 1);
            jaulaVacia[ventaPos] = null;
            oro = oro + 1;
            System.out.println("Vendio correctamente al jugador tiene de oro: " + oro);
        }
    }
}
```

Está función nos sirve para la venta de animales si los tenemos en nuestro equipo,

```java
public static void compraAnimales(int rondas, Animal[] jaulaRelleno, Animal[] jaulaAtaque) {
    int oro = 10;
    int llenado, opcion;
    boolean errordeposicion;
    Animal[] jaulaMientras = getSeleccionArray(rondas);
    if (rondas == 1) {
        opcion = Util.solicitarNumero(mensaje: "¿Desea comprar animales para su equipo? \n Si = 1, No = 2.", limiteInferior: 1, limiteSuperior: 2);
    } else {
        opcion = Util.solicitarNumero(mensaje: "¿Desea comprar animales para su equipo o modificarlos? \n Si = 1, No = 2.",
        limiteInferior: 1, limiteSuperior: 2);
    }
    //  if (rondas > 1 && opcion == 1) {
        //      ventaAnimales(rondas, jaulaRelleno, oro);
        //  }
    do {
        if (opcion == 1) {

            int seleccion = Util.solicitarNumero(
                mensaje: "Que animal desea comprar indique el animal conforme al indice que se le dio", limiteInferior: 0,
                jaulaMientras.length - 1);
                if (jaulaMientras[seleccion] != null) {
                    do {
                        try {

                            llenado = Util.solicitarNumero(mensaje: "En que posicion de equipo quiere poner a su animal", limiteInferior: 0,
                            jaulaRelleno.length - 1);
                            if (jaulaRelleno[llenado] == null && oro > 0) {
                                errordeposicion = false;
                                jaulaRelleno[llenado] = (Animal) jaulaMientras[seleccion].clone();
                                jaulaMientras[seleccion] = null;
                                jaulaAtaque[llenado] = (Animal) jaulaRelleno[llenado].clone();

                                oro = oro - 3;
                                System.out.println("Le quedan de oro: " + oro);
                                System.out.println(x: "\n Su equipo actualmente es: ");
                                for (int i = 0; i < jaulaRelleno.length; i++) {
                                    System.out.println(jaulaRelleno[i]);
                                }
                                int refrescar = Util.solicitarNumero(mensaje: "Desea refrescar la lista? \n Si = 1, No = 2",
                                        limiteInferior: 1,
                                        limiteSuperior: 2);

                                if (refrescar == 1) {
                                    if (oro > 0) {
                                        oro = oro - 1;
                                        System.out.println("Le queda de oro: " + oro);
                                        generarJaula(rondas);
                                    }
                                } else if (oro > 0) {
                                    opcion = Util.solicitarNumero(
                                            mensaje: "¿Desea volver a comprar animales para su equipo? \n Si = 1, No = 2.",
                                            limiteInferior: 1, limiteSuperior: 2);
                                    for (int i = 0; i < jaulaMientras.length; i++) {
                                        if (jaulaMientras[i] != null) {
                                            System.out.println("Puede comprar los siguientes animales " + i + ""
                                                    + jaulaMientras[i]);
                                        }
                                    }
                                }
                            } else {
                                System.out.println(x: "Ya existe un animal en esa posicion");
                                System.out.println(
                                        "Debe ingresar una posicion diferente o que sean el mismo animal, diferente de: "
                                                + llenado);
                                errordeposicion = true;
                            }
```



```
                llenado = Util.solicitarNumero( mensaje: "En que posicion de equipo quiere poner a su animal", limiteInferior: 0,
                jaulaRelleno.length - 1);
                if (jaulaRelleno[llenado] == null && oro > 0) {
                    errordeposicion = false;
                    jaulaRelleno[llenado] = (Animal) jaulaMientras[seleccion].clone();
                    jaulaMientras[seleccion] = null;
                    jaulaAtaque[llenado] = (Animal) jaulaRelleno[llenado].clone();

                    oro = oro - 3;
                    System.out.println("Le quedan de oro: " + oro);
                    System.out.println( x: "\n Su equipo actualmente es: ");
                    for (int i = 0; i < jaulaRelleno.length; i++) {
                        System.out.println(jaulaRelleno[i]);
                    }
                    int refrescar = Util.solicitarNumero( mensaje: "Desea refrescar la lista? \n Si = 1, No = 2",
                            limiteInferior: 1,
                            limiteSuperior: 2);

                    if (refrescar == 1) {
                        if (oro > 0) {
                            oro = oro - 1;
                            System.out.println("Le queda de oro: " + oro);
                            generarJaula(rondas);
                        }
                    } else if (oro > 0) {
                        opcion = Util.solicitarNumero(
                                mensaje: "¿Desea volver a comprar animales para su equipo? \n Si = 1, No = 2.",
                                limiteInferior: 1, limiteSuperior: 2);
                        for (int i = 0; i < jaulaMientras.length; i++) {
                            if (jaulaMientras[i] != null) {
                                System.out.println("Puede comprar los siguientes animales " + i + ""
                                        + jaulaMientras[i]);
                            }
                        }
                    }
                } else {
                    System.out.println( x: "Ya existe un animal en esa posicion");
                    System.out.println(
                            "Debe ingresar una posicion diferente o que sean el mismo animal, diferente de: "
                                    + llenado);
                    errordeposicion = true;
                }

            } catch (Exception e) {
                errordeposicion = true;
                llenado = Util.solicitarNumero( mensaje: "Error en que posicion de equipo quiere poner a su animal", limiteInferior: 0,
                        jaulaRelleno.length - 1);
                if (jaulaRelleno[llenado] == null) {
                    errordeposicion = false;
                }
            }
        } while (errordeposicion);
    } else {
        System.out.println( x: "Error no existe un animal en esa posicion.");
    }


} else  {
    for (int i = 0; i < jaulaRelleno.length; i++) {
        System.out.println("Es una lastima pero tu equipo es: " + i + "" + jaulaRelleno[i]);
    }
}
}while(opcion!=2&&oro>0);
}

}
```

Está función nos sirve para comprar los animales está nos sive para el modo versus, y poder llenar nuestros arreglos de ataque.

```
public static void llenadoPlayer(int rondas) {
    compraAnimales(rondas, jaulaPlayer, jaulaPlayerAtaque);
}

public static void llenadoPlayerRival(int rondas) {
    compraAnimales(rondas, jaulaPlayerRival, jaulaPlayerRivalAtaque);
}
```

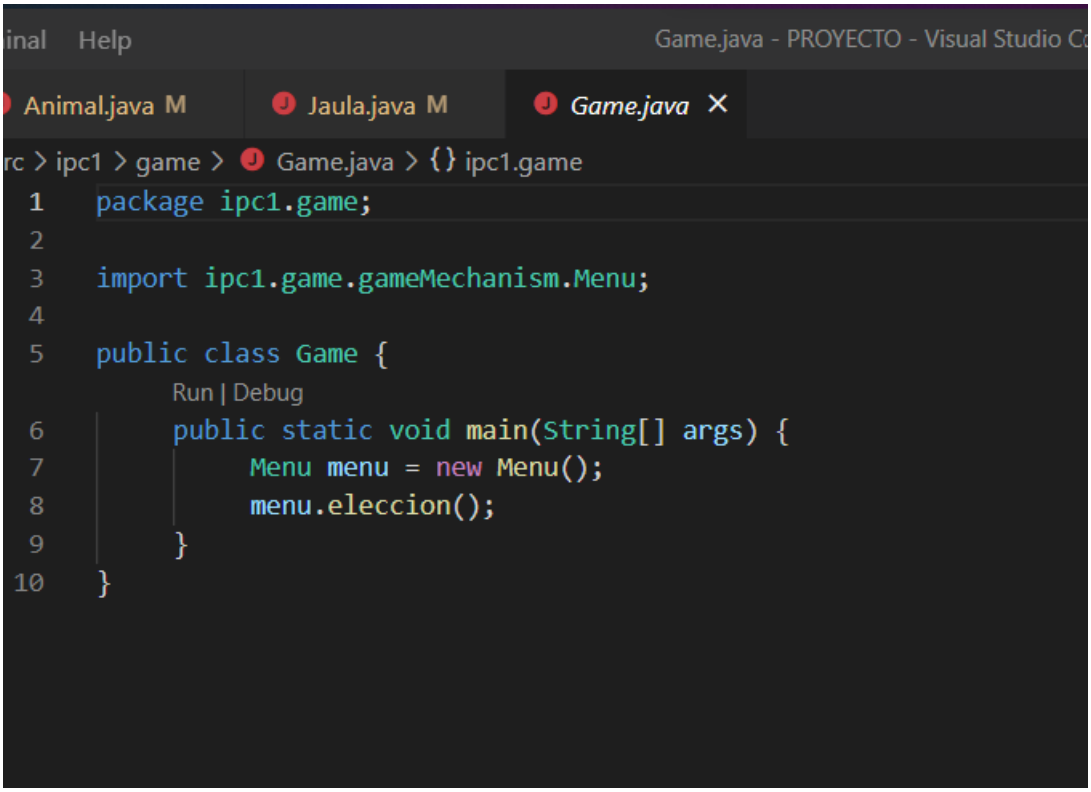
Solo llamamos a la función CompraAnimales y ya podemos llenar nuestros arreglos

```java
public static void ataque(int rondas) {
    int vida = 10;
    int victorias = 0;
    do {
        System.out.println( x: "****** PLAYER 1 ******");
        System.out.println();
        sub.texto(vida, victorias);
        generarJaula(rondas);
        llenadoPlayer(rondas);
        System.out.println();
        System.out.println( x: "****** PLAYER 2 ******");
        System.out.println();
        sub.texto(vida, victorias);
        System.out.println();
        generarJaula(rondas);
        llenadoPlayerRival(rondas);
        System.out.println();
        System.out.println( x: "******INICIA LA PELEA*****");
        for (int i = 0; i < Jaula.jaulaPlayerAtaque.length; i++) {
            for (int j = 0; j < Jaula.jaulaPlayerRival.length; j++) {
                if (Jaula.jaulaPlayerAtaque[i] != null && Jaula.jaulaPlayerRival[j] != null) {
                    System.out.println("Ataca el animal " + Jaula.jaulaPlayerAtaque[i].getNombreAnimal());
                    System.out.println(
                            "Tiene de ataque " + Jaula.jaulaPlayerAtaque[i].getAtaque() + " y de vida tiene "
                                    + Jaula.jaulaPlayerAtaque[i].getVida());
                    Jaula.jaulaPlayerAtaque[i]
                            .setVida(Jaula.jaulaPlayerAtaque[i].getVida() - Jaula.jaulaPlayerRival[j].getAtaque());
                    if (Jaula.jaulaPlayerAtaque[i].getVida() <= 0) {
                        Jaula.jaulaPlayerAtaque[i].setVida( vida: 0);
                        Jaula.jaulaPlayerAtaque[i].setEstado( estadoVida: false);
                    }
                    System.out.println("\n Recibio de daño  " + jaulaPlayerRival[j].getAtaque()
                            + " por lo tanto tiene de vida " + jaulaPlayer[i].getVida());
                    if (Jaula.jaulaPlayerAtaque[i].getVida() > 0) {
                        System.out.println(
                                "El animal " + jaulaPlayerAtaque[i].getNombreAnimal() + " vuelve a recibir daño");
                        Jaula.jaulaPlayerAtaque[i]
                                .setVida(Jaula.jaulaPlayerAtaque[i].getVida()
                                        - Jaula.jaulaPlayerRival[j].getAtaque());
                    }
                }
                // Boolean Cuando termine la pelea todos vivos Si el animal muere un nuevo
                // arreglo con la posicion -1
                // Atributo vivo o muerto si este animal 0 Se muere YA pasaria a la posicion de
                // | mueve todos los otros animales.
                // 1 del arreglo
                /* */
            }
        }
        rondas = rondas + 1;
        // for (int i = 0; i < jaulaPlayer.length; i++) {

        // }
    } while (vida > 0);
    // Implementar por for o copyOfRange
    // Cuando muera llamar al copyOfRange.@interface
    // Si la longitud es == 0 entonces el otro gano, o perdio.

    // Arrays.copyOf(jaula, 2);
```

En esté método es el método de Ataque hacemos comparaciones, de la vida y el ataque del personaje.

```java
public static void contraAtaque() {
    System.out.println( x: "El rival ataca");
    System.out.println();
    for (int i = 0; i < jaulaPlayer.length; i++) {
        for (int j = 0; j < jaulaPlayerRival.length; j++) {
            if (jaulaPlayer[i] != null && jaulaPlayerRival[j] != null) {
                Jaula.jaulaPlayerRival[j]
                        .setVida(Jaula.jaulaPlayer[i].getAtaque() - Jaula.jaulaPlayerRival[j].getVida());
                if (Jaula.jaulaPlayerRival[j].getVida() <= 0) {
                    Jaula.jaulaPlayerRival[j].setVida( vida: 0);
                    Jaula.jaulaPlayerRival[j].setEstado( estadoVida: false);
                }
                System.out.println("Ataca " + jaulaPlayer[i]);
                System.out.println(jaulaPlayerRival[j] + " El principal ataco");
            }
        }
    }
}

public static void incrementaNivel(Animal[] mientras, int seleccion, int llenado) {
    for (int i = 0; i < jaulaPlayer.length; i++) {
        jaulaPlayer[llenado] = mientras[seleccion];
        jaulaPlayer[llenado].setNivel(jaulaPlayer[llenado].getNivel() + 0.5);
        System.out.println(
                "El nivel de: " + jaulaPlayer[llenado] + " es: " + jaulaPlayer[llenado].getNivel());
        if (jaulaPlayer[llenado].getNivel() == 2) {
            jaulaPlayer[llenado].setAtaque(jaulaPlayer[llenado].getAtaque() + 1);
            jaulaPlayer[llenado].setVida(jaulaPlayer[llenado].getVida() + 1);
            System.out.println( x: "Se le sumo 1 de vida y 1 de ataque");
        }
    }
}
```

En este método incrementamos la vida y es el contrataque del Player 2.



Y en nuestro Main llamamos el menú.eleccion().