MANUAL TÉCNICO
JUEGO: SUPER AUTO PETS

SOFTWARE UTILIZADO:

- Lenguaje: **Java 17 con openJDK 17**
- IDE utilizado: **JetBrains IntelliJ IDEA Ultimate**
- Sistema operativo: **Fedora Linux 36**
- Herramienta de gestión y construcción de proyecto: **Maven, version: 4.0.0**
- Version: **1.0 – SNAPSHOT**
- Paquetería: **Jar**

REQUISITOS PARA EJECUTAR EL PROGRAMA:

- Es necesario tener instalado el programa Java en la última versión.

ARCHIVO: Main.java:

```
import Pets.*;
import Engine.*;
import Utils.ArrayUtils;

public class Main {

	public static void main(String[] args) throws
CloneNotSupportedException {

		MainMenu mainMenu = new MainMenu();
		mainMenu.startGame();

	}

}
```

ARCHIVO: Utils/ArrayUtils.java:

```
package Utils;

import Pets.Pet;

public class ArrayUtils {

    public static Pet[] cloner(Pet[] pets) {
        Pet[] clonedPets = new Pet[pets.length];
        for (int i = 0; i < pets.length; i++) {
            clonedPets[i] = pets[i].clone();
        }
        return clonedPets;
    }

    public static boolean isEmpty(Pet[] pets) {
        boolean arrayIsEmpty = true;
        for (Pet pet : pets) {
            if (pet != null) {
                arrayIsEmpty = false;
                break;
            }
        }
        return arrayIsEmpty;
    }
}
```

ARCHIVO: Utils/InputHandler.java:

```
package Utils;

import java.util.Scanner;

public class InputHandler {
    static Scanner input = new Scanner(System.in);

    public static int askNumberRange(int lMin, int lMax) {
        int enteredNumber = 0;
        boolean inputError;

        do {
            try {
                enteredNumber = input.nextInt();
```

```
                if ((enteredNumber >= lMin) && (enteredNumber <=
lMax)) {
                    inputError = false;
                } else {
                    inputError = true;
                    System.out.println("El valor ingresado debe ser
entre " + lMin + " y " + lMax);
                    System.out.println("Intenta de nuevo");
                }

            } catch (Exception ex) {
                inputError = true;
                System.out.println("El valor ingresado no es valido,
intenta de nuevo");
                input.next();

            }

        } while (inputError);
        return enteredNumber;
    }

    public static int askNumber(String message) {
        int enteredNumber = 0;
        boolean inputError;

        do {
            try {
                System.out.print(message);
                enteredNumber = input.nextInt();
                inputError = false;

            } catch (Exception ex) {
                inputError = true;
                System.out.println("El valor ingresado no es valido,
intenta de nuevo");
                input.next();

            }

        } while (inputError);
        return enteredNumber;
```

```
    }

}
```

ARCHIVO: Engine/ArenaGame.java:

```
package Engine;
import Engine.Players.*;
import Pets.*;
import Utils.ArrayUtils;

public class ArenaGame {

    PlayerUser playerUser = new PlayerUser();

    AiPlayer ai = new AiPlayer();
    PreFight preparator = new PreFight(playerUser);
    int round = 1;


    public void PlayArena() throws CloneNotSupportedException {

        while (playerKeepPlaying()){
            selectBattles();

preparator.prepareFightMenu(playerUser.getPlayerPets(),round);
            TeamsFight tFight = new
TeamsFight(ArrayUtils.cloner(playerUser.getPlayerPets()),ArrayUtils.
cloner(ai.getAiPlayerPets()));
            tFight.startTeamFight();
            preparator.selection = 0;
        }

    }

    private boolean playerKeepPlaying(){
        return preparator.selection != 1;

    }

    private void selectBattles(){
        int random_int = (int)Math.floor(Math.random()*(3-1+1)+1);
```

```
    if (random_int == 1){battle1();}
    else if (random_int == 2){battle2();}
    else if (random_int == 3){battle3();}

}

private void battle1(){
    playerUser.setPlayerPets(new Fish(), 0);
    playerUser.setPlayerPets(new Cricket(), 1);
    playerUser.setPlayerPets(new Mosquito(), 2);
    playerUser.setPlayerPets(new Otter(), 3);
    playerUser.setPlayerPets(new Frog(), 4);

    ai.setAiPlayerPets(new Beetle(),0);
    ai.setAiPlayerPets(new Dodo(),1);
    ai.setAiPlayerPets(new Elephant(),2);
    ai.setAiPlayerPets(new Cricket(),3);
    ai.setAiPlayerPets(new Mosquito(),4);
}
private void battle2(){
    playerUser.setPlayerPets(new Camel(), 0);
    playerUser.setPlayerPets(new Rat(), 1);
    playerUser.setPlayerPets(new Raccoon(), 2);
    playerUser.setPlayerPets(new Giraffe(), 3);
    playerUser.setPlayerPets(new Sheep(), 4);

    ai.setAiPlayerPets(new Snail(),0);
    ai.setAiPlayerPets(new Dodo(),1);
    ai.setAiPlayerPets(new Turtle(),2);
    ai.setAiPlayerPets(new Wolf(),3);
    ai.setAiPlayerPets(new Ox(),4);

}
private void battle3(){
    playerUser.setPlayerPets(new Fox(), 0);
    playerUser.setPlayerPets(new Ox(), 1);
    playerUser.setPlayerPets(new Kangaroo(), 2);
    playerUser.setPlayerPets(new Deer(), 3);
    playerUser.setPlayerPets(new Parrot(), 4);

    ai.setAiPlayerPets(new Hipo(),0);
    ai.setAiPlayerPets(new Dolphin(),1);
```

```
        ai.setAiPlayerPets(new Cougar(),2);
        ai.setAiPlayerPets(new Jaguar(),3);
        ai.setAiPlayerPets(new Llama(),4);

    }
}
```

ARCHIVO: Engine/IndividualFight.java:

```
package Engine;
import Pets.*;

public class IndividualFight {

    private Pet petA;
    private Pet petB;

    public IndividualFight(Pet petA, Pet petB) {
        this.petA = petA;
        this.petB = petB;
    }

    public Pet startFight(){

        System.out.println(petA.getName()+" pelea con:
"+petB.getName());

        do {
            petA.setHp((petA.getHp() - petB.getAttack()));
            petB.setHp((petB.getHp() - petA.getAttack()));
        } while (noWinner());

        showWinner();

        // Return the winner pet or null if draw
        if (petA.getHp() > 0){
            return petA;
        } else if (petB.getHp() > 0) {
            return petB;
        } else {
            return null;
        }
    }
```

```
    private boolean noWinner() { return petA.getHp() > 0 &&
petB.getHp() > 0; }

    public void showWinner(){
        if (petA.getHp() > 0){
            System.out.println("Gana "+petA.getName());
        } else if (petB.getHp() > 0) {
            System.out.println("Gana "+petB.getName());
        } else {
            System.out.println("Empate! Ambas mascotas mueren");
        }
    }

    public Pet getWinner(){
        if (petA.getHp() > 0){
            return petA;
        } else if (petB.getHp() > 0) {
            return petB;
        } else {
            return null;
        }

    }
}
```

ARCHIVO: Engine/TeamsFight.java:

```
package Engine;
import Pets.*;

public class TeamsFight {

    private Pet[] teamA;
    private Pet[] teamB;

    public TeamsFight(Pet[] teamA, Pet[] teamB) {
        this.teamA = teamA;
        this.teamB = teamB;
    }

    public void startTeamFight() throws CloneNotSupportedException {
```

```
        int indexA = 0;
        int indexB = 0;

        do {

            Pet avaliableA = getFirstAvailablePet(teamA);
            Pet avaliableB = getFirstAvailablePet(teamB);

            IndividualFight fight = new IndividualFight(avaliableA,
avaliableB);
            Pet winnerPet = fight.startFight();


            if (winnerPet == null) {

                teamA[indexA] = null;
                indexA += 1;
                teamB[indexB] = null;
                indexB += 1;
//                System.out.println("Draw");

            } else if (winnerPet.equals(avaliableA)){

                System.out.println(String.format("Ha ganado la
mascota del equipo A: %s",avaliableA.getClass().getSimpleName()));
                teamB[indexB] = null;
                indexB += 1;

            } else if (winnerPet.equals(avaliableB)){

                System.out.println(String.format("Ha ganado la
mascota del equipo B: %s",avaliableB.getClass().getSimpleName()));
                teamA[indexA] = null;
                indexA += 1;

            }

        } while (teamsHavePets());

        showFinalStats();

    }
```

```
    public Pet getFirstAvailablePet(Pet[] team) throws
CloneNotSupportedException {
        Pet available = null;

        for (Pet pet : team) {
            if (pet != null) {
                available = pet;
                break;
            }
        }
        return available;
    }

    private boolean teamsHavePets() {
        boolean isFirstTeamAlive = teamHasPets(teamA);
        boolean isSecondTeamAlive = teamHasPets(teamB);
        return isFirstTeamAlive && isSecondTeamAlive;
    }

    private boolean teamHasPets(Pet[] team) {
        boolean teamIsAlive = false;
        for (Pet pet : team) {
            if (pet != null) {
                teamIsAlive = true;
                break;
            }
        }
        return teamIsAlive;
    }

    private void showFinalStats(){
        int leftPets = 0;
        Pet[] winnerTeam = null;
        //show winner team
        if (teamHasPets(teamA)){
            System.out.println("El equipo A ha ganado esta ronda!");
            winnerTeam = teamA;
        } else if (teamHasPets(teamB)){
            System.out.println("El equipo B ha ganado esta ronda!");
            winnerTeam = teamB;
        } else {
```

```
            System.out.println("Empate!");
        }

        if (winnerTeam == teamA){
            for (Pet pet:winnerTeam) {
                if (pet != null){leftPets += 1;}
            }
            System.out.println(String.format("En el equipo ganador A
han sobrevivido %s mascotas:",leftPets));

            for (Pet pet : winnerTeam) {
                if (pet != null) {
                    System.out.println(String.format("Animal: %s |
Ataque: %s | Vida: %s", pet.getName(), pet.getAttack(),
pet.getHp()));
                }
            }
        } else if (winnerTeam == teamB){
            for (Pet pet:winnerTeam) {
                if (pet != null){leftPets += 1;}
            }
            System.out.println(String.format("En el equipo ganador B
han sobrevivido %s mascotas:",leftPets));

            for (Pet pet : winnerTeam) {
                if (pet != null) {
                    System.out.println(String.format("Animal: %s |
Ataque: %s | Vida: %s", pet.getName(), pet.getAttack(),
pet.getHp()));
                }
            }
        }
    }
}
```

ARCHIVO: Engine/MainMenu.java:

```
package Engine;

import Utils.InputHandler;

public class MainMenu {

    private void greeter() {
        System.out.println("Bienvenid@ a");
        System.out.println("""
                    _____                                          _
_____     _      \s
                   / ____|                           /\\         | |
|  __ \\   | |     \s
                  | (___  _   _ _ __   ___ _ __     /  \\  _   _| |_
___   | |__) |__| |_ ___\s
                   \\___ \\| | | | '_ \\ / _ \\ '__|   / /\\ \\| | |
| __/ _ \\  |  ___/ _ \\ __/ __|
                   ____) | |_| | |_) |  __/ |     / ____ \\ |_| | ||
(_) | | |  |  __/ |_\\__ \\
                  |_____/ \\__,_| .__/ \\___|_|    /_/
\\_\\__,_|\\__\\___/  |_|   \\___|\\__|___/
                               | |
\s
                               |_|
\s
                """);
    }

    public void startGame() throws CloneNotSupportedException {
        greeter();
        mainMenu();
    }

    public void mainMenu() throws CloneNotSupportedException {
        int selection = 0;
        while (2 != selection) {
            System.out.println("##### MENU PRINCIPAL #####");
            System.out.println("¿Qué quieres hacer?");
            System.out.println("1: Correr demo de modo arena");
            System.out.println("2: Salir del juego");
            System.out.print("Eleccion? > ");
```

```
            selection = InputHandler.askNumberRange(1, 2);
            checkOption(selection);
        }
    }

    private void checkOption(int selection) throws
CloneNotSupportedException {
        switch (selection) {
            case 1:
                ArenaGame agame = new ArenaGame();
                agame.PlayArena();
                break;
            case 2:
                System.out.println("Saliendo del juego...");
                System.exit(0);
                break;
        }
    }

}
```

ARCHIVO: Engine/PreFight.java:

```
package Engine;

import Engine.Players.*;
import Pets.*;
import Utils.*;

public class PreFight {

    MainMenu mainMenu = new MainMenu();
    PlayerUser player;
    int availableGold;
    int selection = 0;

    public PreFight(PlayerUser player) {
        this.player = player;
    }

    public void prepareFightMenu(Pet[] pets, int round) throws
CloneNotSupportedException {
```

```
        availableGold = 10;
        while (1 != selection) {
            System.out.println("\n##### JUGANDO MODO ARENA #####");
            System.out.println("(Aviso: Este modo de juego esta
incompleto, \nsolo se efectuaran batallas generadas aleatoriamente
hasta que se desee salir)\n");
            System.out.println("Que deseas hacer?:\n" +
                    "1: Ser espectador de una pelea demo\n" +
                    "2: Salir de este modo\n");

            System.out.print("Eleccion? > ");
            selection = InputHandler.askNumberRange(1, 2);
            checkOption(selection);
        }
    }

    private void checkOption(int selection) throws
CloneNotSupportedException {
        switch (selection) {
            case 1:
                System.out.println("Mostrando pelea...");
                break;
            case 2:
                System.out.println("Regresando a menu
principal...");
                mainMenu.startGame();
                break;
        }
    }

    private void showPets(Pet[] pets) {
        if (ArrayUtils.isEmpty(pets)){
            System.out.println("|| No tienes mascotas en tu equipo!
||");
        } else {
            System.out.println("|| Tu equipo tiene las mascotas:
||");
            for (int i = 0; i < player.getPlayerPets().length; i++)
{
                if (pets[i] != null) {
```

```
                    System.out.println(String.format("\t|| Animal:
%s | Ataque: %s | Vida: %s ||", pets[i].getName(),
pets[i].getAttack(), pets[i].getHp()));
                }
            }
        }
    }

}
```

ARCHIVO: Engine/Players/AiPlayer.java:

```
package Engine.Players;
import Pets.*;

public class AiPlayer {

    Pet[] aiPlayerPets = new Pet[5];
    int round = 1;
    int lives = 10;
    int wins = 0;

    public Pet[] getAiPlayerPets() {
        return aiPlayerPets;
    }

    public void setAiPlayerPets(Pet pet, int index) {
        this.aiPlayerPets[index] = pet;
    }

    public int getRound() {
        return round;
    }

    public void setRound(int round) {
        this.round = round;
    }

    public int getLives() {
        return lives;
    }

    public void setLives(int lives) {
```

```
        this.lives = lives;
    }

    public int getWins() {
        return wins;
    }

    public void setWins(int wins) {
        this.wins = wins;
    }

}
```

ARCHIVO: Engine/Players/PlayerUser.java:

```
package Engine.Players;
import Pets.*;

public class PlayerUser {

    Pet[] playerPets = new Pet[5];
    int lives = 10;
    int wins = 0;

    public Pet[] getPlayerPets() {
        return playerPets;
    }

    public void setPlayerPets(Pet pet, int index) {
        this.playerPets[index] = pet;
    }

    public int getLives() {
        return lives;
    }

    public void setLives(int lives) {
        this.lives = lives;
    }

    public int getWins() {
        return wins;
    }
```

```
    public void setWins(int wins) {
        this.wins = wins;
    }
}
```

ARCHIVO: Pets/Pet.java:
En este caso, se creó una clase abstracta de la que todos los animales heredan, para evitar la sobrecarga de información en este documento, solo se mostrara el codigo de algunos animales

```
package Pets;

public abstract class Pet implements Cloneable{

    int attack;
    int hp;
    int level;
    int xp;
    int tier;
    String name;

    public int getAttack() { return attack; }

    public int getHp() { return hp; }

    public int getTier(){ return tier; }

    public void setHp(int hp) {
        this.hp = hp;
    }

    public String getName() {
        return name;
    }

    @Override
    public Pet clone() {
        try {
            return (Pet) super.clone();
        } catch (CloneNotSupportedException e) {
            throw new AssertionError();
```

```
        }
    }
}
```

ARCHIVO: Pets/Ant.java:

```
package Pets;

public class Ant extends Pet implements Insect, Land{

    public Ant() {
        attack = 2;
        hp = 1;
        level = 1;
        xp = 1;
        tier = 1;
        name = "Hormiga";
    }

}
```

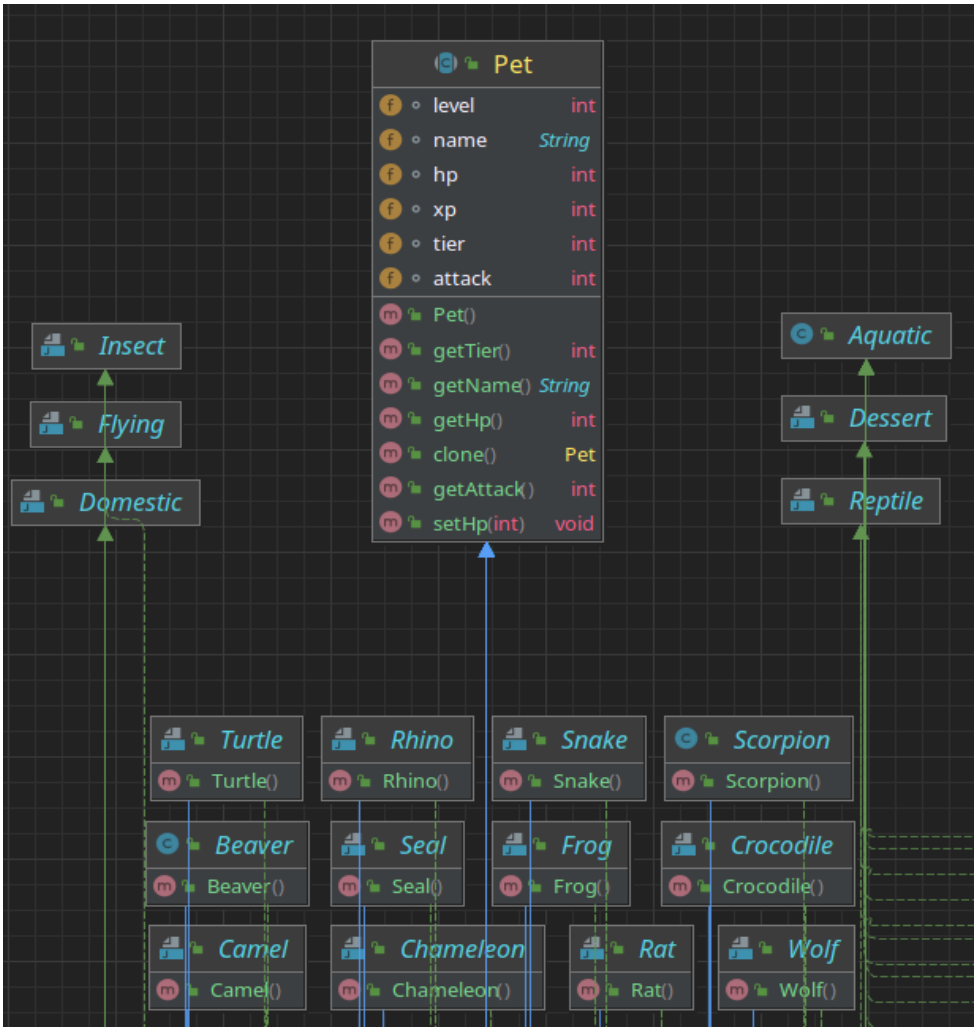
ARCHIVO: Pets/Cricket.java:

```
package Pets;

public class Cricket extends Pet implements Insect{

    public Cricket() {
        attack = 1;
        hp = 2;
        level = 1;
        xp = 1;
        tier = 1;
        name = "Grillo";

    }
}
```

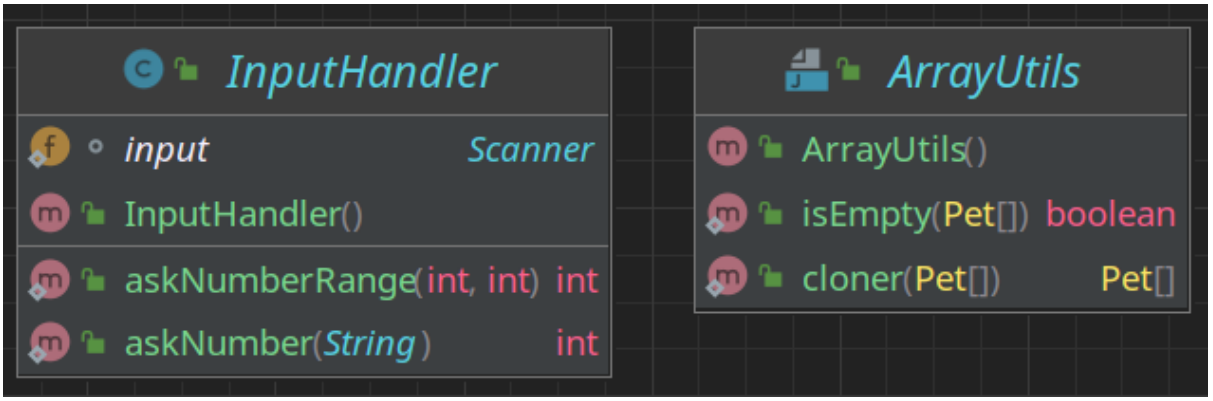
ARCHIVO: Pets/Dodo.java:
package Pets;

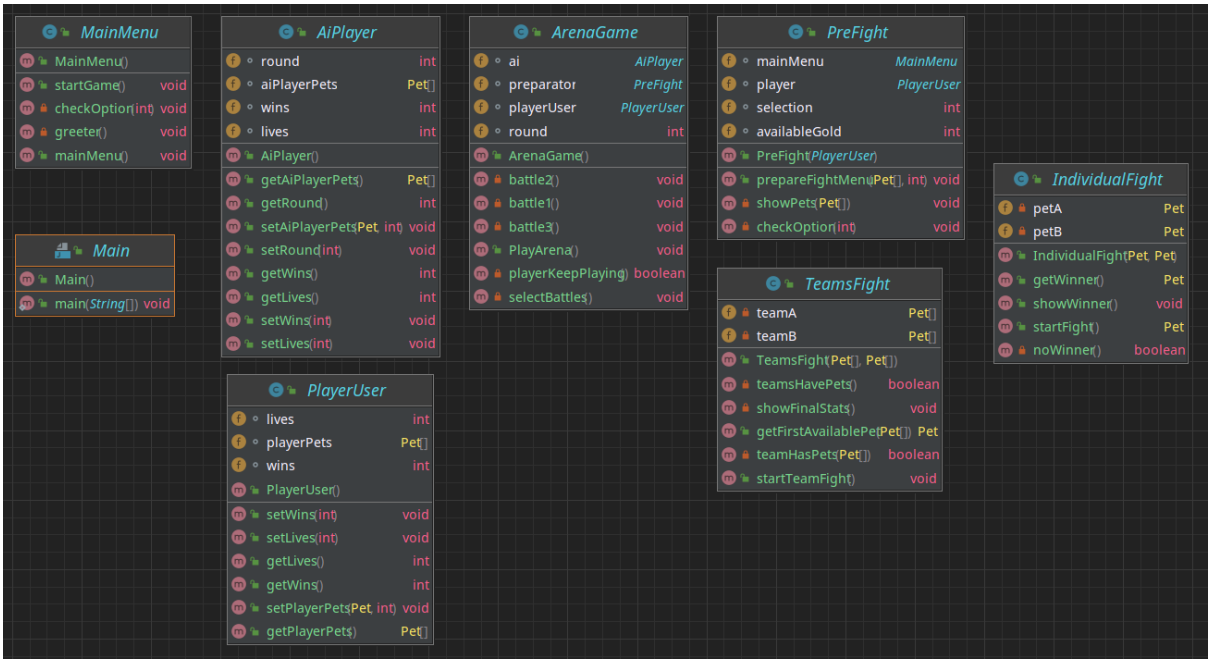
```
public class Dodo extends Pet implements Flying{
    public Dodo() {
        attack = 2;
        hp = 3;
        level = 1;
        xp = 1;
        tier = 2;
        name = "Dodo";
    }
}
```

DIAGRAMA DE CLASES:



En el caso de las mascotas, no se han colocado todas para evitar sobrecarga de información


InputHandler
input Scanner
InputHandler()
askNumberRange(int, int) int
askNumber(String) int
ArrayUtils
ArrayUtils()
isEmpty(Pet[]) boolean
cloner(Pet[]) Pet[]


MainMenu
MainMenu()
startGame() void
checkOption(int) void
greeter() void
mainMenu() void
Main
Main()
main(String[]) void
AiPlayer
round int
aiPlayerPets Pet[]
wins int
lives int
AiPlayer()
getAiPlayerPets() Pet[]
getRound() int
setAiPlayerPets(Pet, int) void
setRound(int) void
getWins() int
getLives() int
setWins(int) void
setLives(int) void
PlayerUser
lives int
playerPets Pet[]
wins int
PlayerUser()
setWins(int) void
setLives(int) void
getLives() int
getWins() int
setPlayerPets(Pet, int) void
getPlayerPets() Pet[]
ArenaGame
ai AiPlayer
preparator PreFight
playerUser PlayerUser
round int
ArenaGame()
battle2() void
battle1() void
battle3() void
PlayArena() void
playerKeepPlaying() boolean
selectBattles() void
PreFight
mainMenu MainMenu
player PlayerUser
selection int
availableGold int
PreFight(PlayerUser)
prepareFightMenu(Pet[], int) void
showPets(Pet[]) void
checkOption(int) void
TeamsFight
teamA Pet[]
teamB Pet[]
TeamsFight(Pet[], Pet[])
teamsHavePets() boolean
showFinalStats() void
getFirstAvailablePet(Pet[]) Pet
teamHasPets(Pet[]) boolean
startTeamFight() void
IndividualFight
petA Pet
petB Pet
IndividualFight(Pet, Pet)
getWinner() Pet
showWinner() void
startFight() Pet
noWinner() boolean