UNIVERSIDAD DE LAS FUERZAS ARMADAS (ESPE)

PROGRAMACIÓN ORIENTADA A OBJETOS.

Actividad Experimental N.° 1 Primer Parcial

Sistema de Gestión de Proyectos de TIC

AUTOR:

- Nayeli Vilaña

PARALELO:

NRC-1322

DOCENTE:

Ing. Luis Jaramillo

PERÍODO:

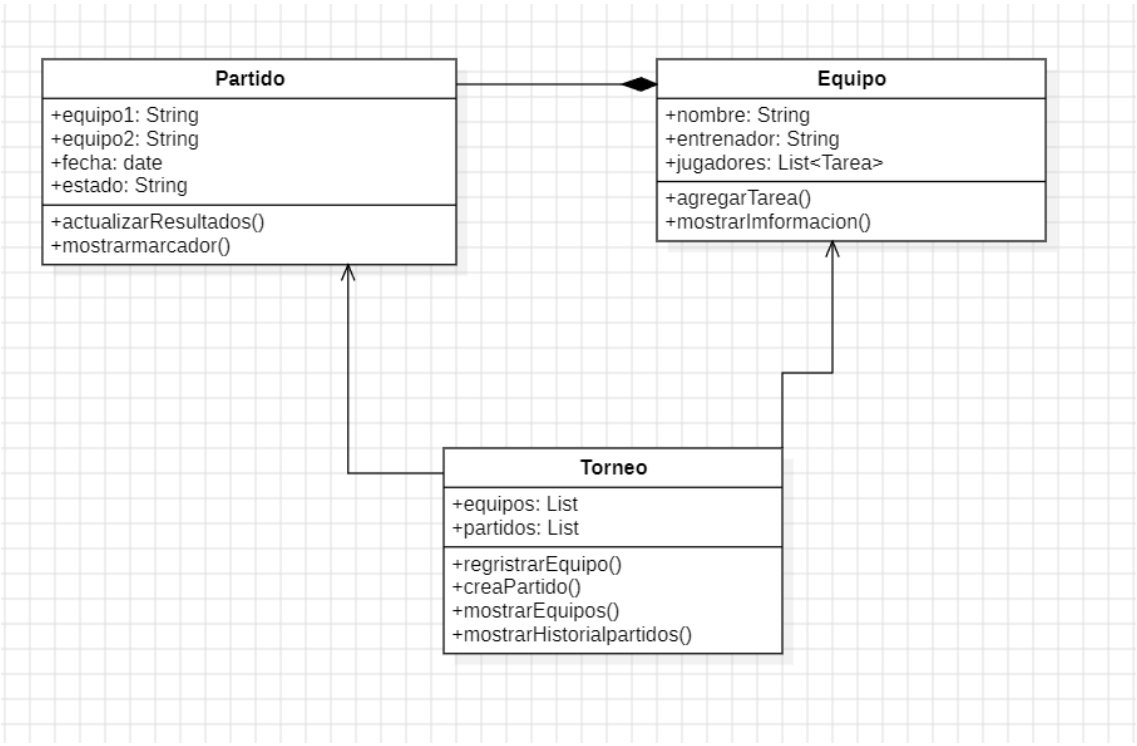
Octubre 2024 – marzo 2025

SANGOLQUI – ECUADOR

TEMA: Creación de objetos y UML

Diagrama UML

Debemos elaborar un esquema UML, el cual especifica las clases fundamentales (`Equipo`, `Partido`, `Torneo`) y las conexiones entre estas:



Ejecución del código se rige por las relaciones establecidas en el esquema. UML de la forma siguiente:

- La categoría `Torneo` incluye listas de etapas de los equipos `Equipo` y `Partido`. Esto es evidencia que los equipos y encuentros constituyen un componente esencial de un torneo.
- La clase `Partido` se vincula con la clase `Equipo` mediante las siguientes conexiones: ejemplificaciones `equipo1` y `equipo2`. Esto posibilita que los partidos incorporen a los participantes, equipos que no requieren la presencia del partido.

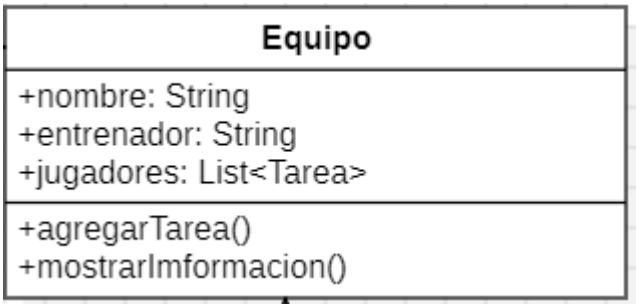
. Etapa de resolución del ejercicio

Clase Equipo:

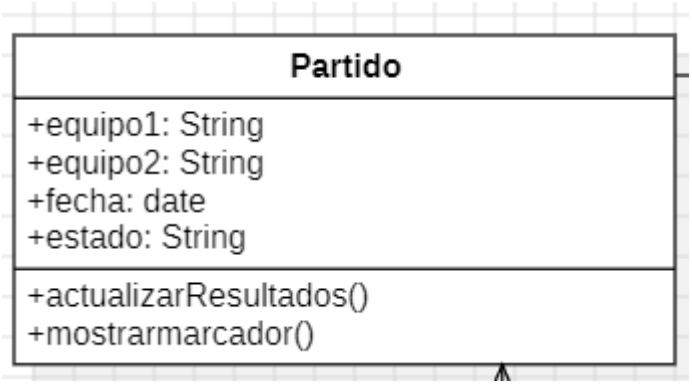
Tipo de Equipo:

La clase Equipo simboliza a cada equipo participante en el torneo. Cada grupo posee:
Una denominación (como " Los Tigres Dorados").
Un entrenador (como " Roberto Martínez").
Una lista de futbolistas (como " Antonio López", " Luis Torres",” Alejandro Pérez“, entre otros).
Constructor: Este procedimiento se aplica para establecer un nuevo equipo con su denominación, nombre y supervisor.

Método de añadir jugador: Facilita la incorporación de jugadores a la lista del equipo.
Técnica toString: Presenta todos los datos del equipo en formato texto.



Clase Partido:
Simboliza un enfrentamiento entre dos equipos. Cada alianza posee:
Existen dos equipos (equipo 1 y equipo 2).
Una fecha para conocer cuándo se realiza el partido.
Una condición (comienza como "Pendiente" y puede ser modificada a "Finalizado").
Estructurador: Establece una nueva partida entre dos equipos en una fecha concreta.
Procedimiento para actualizarResultado: Facilita añadir el resultado y señalar el encuentro como "Completo".
Un marcador, como "3-1".
Procedimiento toString: Presenta datos sobre el partido político



La clase Torneo:
Tiene la responsabilidad de gestionar todo el torneo. Es similar al "líder" del sistema.
y lleva a cabo diversas actividades:
Registrar equipos.
Establecer encuentros entre equipos.
Revelar equipos que están registrados.
Presentar el registro de partidos.
Constructor: Aborda las listas de equipos y encuentros como si fueran vacías.
Procedimiento de registro de Equipo: Facilita la incorporación de equipos a la lista del torneo.
Procedimiento para crear Partido: Establece un nuevo encuentro entre dos equipos inscritos.
Método de exhibición de Equipos: Emite datos acerca de todos los equipos.
Procedimiento de mostrarHistorial de Partidos: Presenta todos los encuentros registrados, incluyendo aquellos que se han jugado.
su condición y marcador.

Clase Guia Principal (Sistema que gestionara la clase Torneo)

Creación de un Torneo.
registrar cada equipo y sus jugadores y su entrenador.

Debemos mostrar los equipos y partidos.

Debemos ir actualizando el marcador de el encuentro es decir actualizar el historial.

Los pasos que debemos seguir son:

Ir registrando la lista de jugadores.

Creamos el encuentro entre estos dos equipos.

Debemos ir actualizando el marcador según vallan avanzando el partido esto será en el historial que debe ir actualizándose.

```java
public class SistemaGestionTorneo {
    public static void main(String[] args) {
        Torneo torneo = new Torneo();

        // Registrar equipos
        Equipo equipo1 = new Equipo("Real Madrid", "Carlo Ancelotti");
        equipo1.agregarJugador("Benzema, Juano, Carlitos, MegaPorter,Messi, Pele");
        equipo1.agregarJugador("Modric, Pepe, Josue, Javito, Don Joshep, Maradona");

        Equipo equipo2 = new Equipo("Barcelona", "Xavi Hernandez");
        equipo2.agregarJugador("Lewandowski, Malotetron, Optimus, Bumbulbee, Megatron");
        equipo2.agregarJugador("Pedri, Shokwave, Soudnwave, Chitor, Rattramp, Rino");

        torneo.registrarEquipo(equipo1);
        torneo.registrarEquipo(equipo2);


        torneo.crearPartido(equipo1, equipo2, "2024-12-10");


        torneo.mostrarEquipos();


        torneo.mostrarHistorialPartidos();


        torneo.partidos.get(0).actualizarResultado("3-1");
        System.out.println("\nHistorial actualizado:");
        torneo.mostrarHistorialPartidos();
    }
}
```

```
put - Experimento 1 (run)
run:
Equipos registrados:
Equipo: Real Madrid, Entrenador: Carlo Ancelotti, Jugadores: [Benzema, Juano, Carlitos, MegaPorter,Messi, Pele, Modric, Pepe, Josue, Javito, Don Joshep, Maradona]
Equipo: Barcelona, Entrenador: Xavi Hernandez, Jugadores: [Lewandowski, Malotetron, Optimus, Bumbulbee, Megatron, Pedri, Shokwave, Soudnwave, Chitor, Rattramp, Rino]
Historial de partidos:
Partido: Real Madrid vs Barcelona, Fecha: 2024-12-10, Estado: Pendiente, Marcador: 0-1

Historial actualizado:
Historial de partidos:
Partido: Real Madrid vs Barcelona, Fecha: 2024-12-10, Estado: Finalizado, Marcador: 3-1
BUILD SUCCESSFUL (total time: 0 seconds)
```

```java
class Partido {
    public Equipo equipo1;
    public Equipo equipo2;
    public String fecha;
    public String estado;
    public String marcador;

    public Partido(Equipo equipo1, Equipo equipo2, String fecha) {
        this.equipo1 = equipo1;
        this.equipo2 = equipo2;
        this.fecha = fecha;
        this.estado = "Pendiente";
        this.marcador = "0-1";
    }

    public void actualizarResultado(String marcador) {
        this.marcador = marcador;
        this.estado = "Finalizado";
    }

    @Override
    public String toString() {
        return "Partido: " + equipo1.nombre + " vs " + equipo2.nombre + ", Fecha: " + fecha +
                ", Estado: " + estado + ", Marcador: " + marcador;
    }
}
```

```java
import java.util.ArrayList;
import java.util.List;

class Equipo {
    public String nombre;
    public String entrenador;
    public List<String> jugadores;

    public Equipo(String nombre, String entrenador) {
        this.nombre = nombre;
        this.entrenador = entrenador;
        this.jugadores = new ArrayList<>();
    }

    public void agregarJugador(String jugador) {
        jugadores.add(jugador);
    }

    @Override
    public String toString() {
        return "Equipo: " + nombre + ", Entrenador: " + entrenador + ", Jugadores: " + jugadores;
    }
}
```

```java
import java.util.ArrayList;
import java.util.List;

class Torneo {
    public List<Equipo> equipos;
    public List<Partido> partidos;

    public Torneo() {
        this.equipos = new ArrayList<>();
        this.partidos = new ArrayList<>();
    }

    public void registrarEquipo(Equipo equipo) {
        equipos.add(equipo);
    }

    public void crearPartido(Equipo equipo1, Equipo equipo2, String fecha) {
        Partido partido = new Partido(equipo1, equipo2, fecha);
        partidos.add(partido);
    }

    public void mostrarEquipos() {
        System.out.println("Equipos registrados:");
        for (Equipo equipo : equipos) {
            System.out.println(equipo);
        }
    }

    public void mostrarHistorialPartidos() {
        System.out.println("Historial de partidos:");
        for (Partido partido : partidos) {
            System.out.println(partido);
        }
    }
}
```