

UNIVERSIDAD DE LAS FUERZAS ARMADAS

ESPE

PROGRAMACIÓN ORIENTADA A OBJETOS

ACTIVIDAD EXPERIMENTAL

GRUPO #6

INTEGRANTES: *CAROLINA ANGAMARCA

***NATHALY CASTILLO**

***VANESSA CHIRIGUAYA**

***ALFREDO LAPO**

***ALEJANDRO SÁNCHEZ**

NRC: 1322

DOCENTE: LUIS JARAMILLO

Implemente un sistema de gestión de proyectos utilizando POO. Incluya las funcionalidades de registrar proyectos, asignar tareas y mostrar su estado

Objetivo general

- Desarrollar un sistema de gestión de proyectos robusto y eficiente, utilizando los principios de la Programación Orientada a Objetos (POO), que permita registrar, asignar tareas y monitorear el estado de los proyectos de manera organizada y visual.

Objetivos Específicos

- Modelar las entidades del sistema (proyectos, tareas, usuarios) como clases en POO, estableciendo relaciones claras entre ellas (herencia, composición, agregación).
- Definir las responsabilidades de cada clase y los métodos que encapsularán su comportamiento.

INTRODUCCIÓN

La Programación Orientada a Objetos (POO) es una metodología de programación que organiza el código en torno a objetos que contienen datos (atributos) y comportamiento (métodos). Este paradigma es ideal para modelar sistemas del mundo real, como los proyectos, ya que permite representar de manera natural las entidades involucradas (proyectos, tareas, usuarios) y sus relaciones.

Un sistema de gestión de proyectos es una herramienta esencial para cualquier organización que busca planificar, organizar y controlar sus iniciativas. Al combinar la potencia de la POO con las necesidades de la gestión de proyectos, podemos crear aplicaciones altamente personalizables y eficientes.

METODOLOGÍA

Se parte con la creación de los diagramas UML, para identificar los métodos y atributos de cada clase que nos proporciona el docente, a continuación, se presenta lo solicitado y cómo quedarían los diagramas UML:

Implemente un sistema de gestión de torneos de fútbol utilizando Programación Orientada a Objetos (POO). El sistema debe permitir las siguientes funcionalidades:

- Registrar Equipos:

Cada equipo debe tener un nombre, un entrenador, y una lista de jugadores.

- Crear Partidos:

Registrar los equipos que jugarán un partido, la fecha del encuentro y el resultado final.

- Consultar Información:

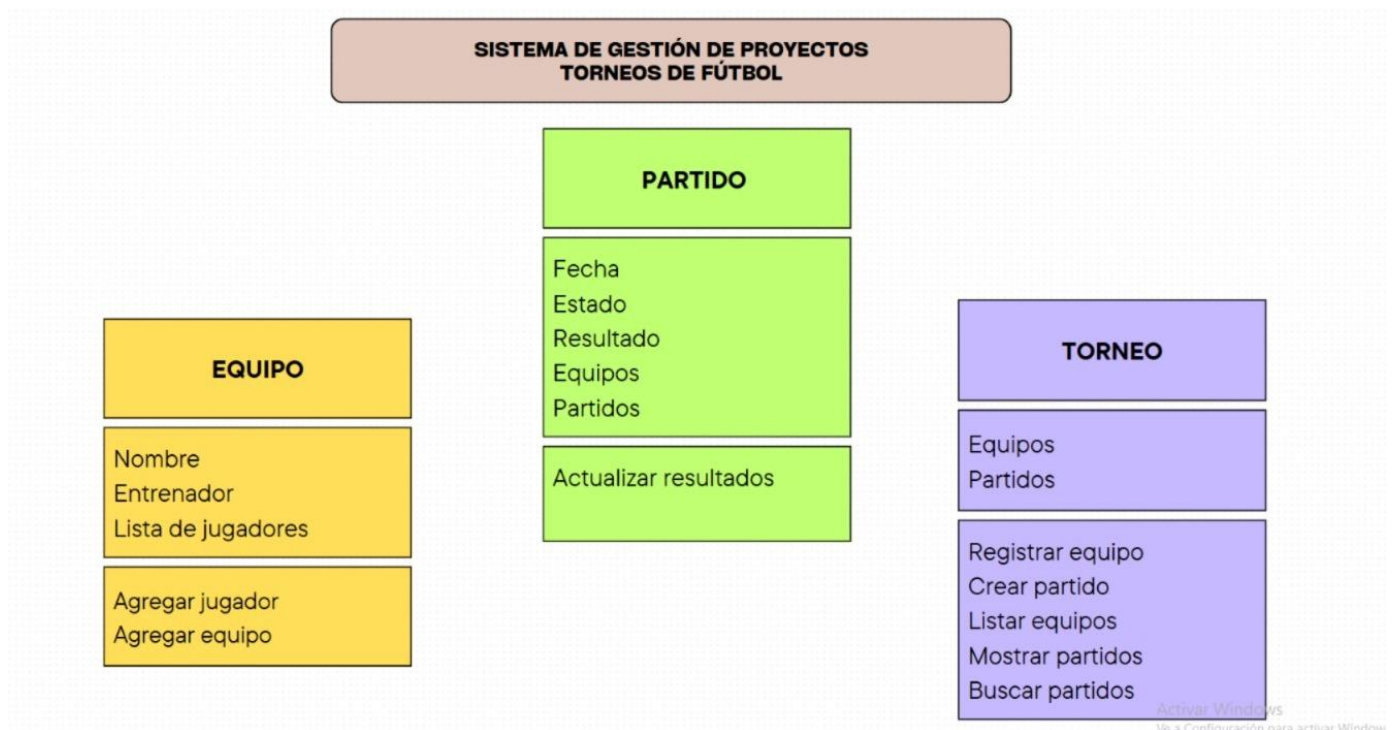
Listar todos los equipos registrados con sus respectivos jugadores.

Mostrar el historial de partidos jugados, incluyendo los equipos participantes y los resultados.


- Actualizar Resultados:

Cambiar el estado de un partido de "Pendiente" a "Finalizado", añadiendo el marcador.

DIAGRAMA UML:



Luego de desarrollar el diagrama UML, mismo que nos dará una guía adecuada para realizar el código de forma organizada, implementamos los códigos en Java para cada clase, procurando que cumplan con la función solicitada. Las pestañas a crear serán: Main, Equipo, Partido y Torneo. Adjuntamos evidencias del proceso ya que el código se encontrará subido a GitHub al terminar el proceso:

 **OnlineGDB**
online compiler and debugger for c/c++

Welcome, **Alejandro Sánchez** ▲

Actividad Experimental 1 - Grupo 6

Create New Project

My Projects

Classroom **new**

Learn Programming

Programming Questions


Upgrade

Logout ▾

Run Debug Stop Share Save {} Beautify

Main.java Equipo.java Partido.java Torneo.java

```
1- import java.util.ArrayList;
2- import java.util.HashSet;
3- import java.util.List;
4- import java.util.Scanner;
5- import java.util.Set;
6
7- // Clase Equipo
8- class Equipo {
9-     private String nombre;
10-    private String entrenador;
11-    private List<String> jugadores;
12
13-    public Equipo(String nombre, String entrenador) {
14-        this.nombre = nombre;
15-        this.entrenador = entrenador;
16-        this.jugadores = new ArrayList<>();
17-    }
18
19-    public String getNombre() {
20-        return nombre;
21-    }
22
23-    public void agregarJugador(String jugador) {
24-        jugadores.add(jugador);
25-    }
26-}
```

 **OnlineGDB**
online compiler and debugger for c/c++

Welcome, **Alejandro Sánchez** ▲

Actividad Experimental 1 - Grupo 6

Create New Project

My Projects

Classroom **new**

Learn Programming

Programming Questions

Upgrade

Logout ▾

Run Debug Stop Share Save

Main.java Equipo.java Partido.java Torneo.java

```
1- import java.util.ArrayList;
2- import java.util.HashSet;
3- import java.util.List;
4- import java.util.Scanner;
5- import java.util.Set;
6
7- // Clase Equipo
8- class Equipo {
9-     private String nombre;
10-    private String entrenador;
```

--- Sistema de Gestión de Torneos ---

1. Registrar equipo
2. Crear partido
3. Listar equipos
4. Mostrar historial de partidos
5. Actualizar resultado de un partido
6. Salir

Seleccione una opción: 1
Nombre del equipo: BSC
Nombre del entrenador: Alfaro
¿Cuántos jugadores desea registrar? 2
Nombre del jugador 1: Nathy
Nombre del jugador 2: Vannesa
Equipo registrado: BSC

<

--- Sistema de Gestión de Torneos ---

1. Registrar equipo
2. Crear partido
3. Listar equipos
4. Mostrar historial de partidos
5. Actualizar resultado de un partido
6. Salir

Seleccione una opción: 1
Nombre del equipo: LDU
Nombre del entrenador: Gareca
¿Cuántos jugadores desea registrar? 2
Nombre del jugador 1: Bladimir
Nombre del jugador 2: Alejandro
Equipo registrado: LDU

The screenshot displays the OnlineGDB interface. On the left is a sidebar with navigation links: 'Welcome, Alejandro Sánchez', 'Actividad Experimental 1 - Grupo 6', 'Create New Project', 'My Projects', 'Classroom' (marked as 'new'), 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area shows a Java code editor with files 'Main.java', 'Equipo.java', 'Partido.java', and 'Torneo.java'. The code in 'Equipo.java' defines a class 'Equipo' with attributes 'nombre' and 'entrenador'. The console output shows the program running and displaying a menu of options for managing the tournament. The menu includes options like 'Registrar equipo', 'Crear partido', 'Listar equipos', 'Mostrar historial de partidos', 'Actualizar resultado de un partido', and 'Salir'. The program also displays the current state of the tournament, including the date '2024-12-05' and the result '5-0'.

```
1- import java.util.ArrayList;
2- import java.util.HashSet;
3- import java.util.List;
4- import java.util.Scanner;
5- import java.util.Set;
6
7- // Clase Equipo
8- class Equipo {
9-     private String nombre;
10-    private String entrenador;
11-}
12
13- // Clase Partido
14- class Partido {
15-    private String fecha;
16-    private String resultado;
17-    private String estado;
18-    private String equipos;
19-}
20
21- // Clase Torneo
22- class Torneo {
23-    private List<Partido> partidos;
24-    private Set<Equipo> equipos;
25-}
26
27- // Metodo principal
28- public static void main(String[] args) {
29-    Torneo torneo = new Torneo();
30-    Scanner scanner = new Scanner(System.in);
31-    while (true) {
32-        System.out.println("Ingrese el resultado (ejemplo: 2-1): 5-0");
33-        String resultado = scanner.nextLine();
34-        torneo.actualizarResultado(resultado);
35-        System.out.println("Partidos registrados:");
36-        torneo.mostrarHistorial();
37-        System.out.println("Seleccione una opción: 4");
38-        int opcion = scanner.nextInt();
39-        switch (opcion) {
40-            case 1:
41-                registrarEquipo(torneo);
42-                break;
43-            case 2:
44-                crearPartido(torneo);
45-                break;
46-            case 3:
47-                listarEquipos(torneo);
48-                break;
49-            case 4:
50-                mostrarHistorial(torneo);
51-                break;
52-            case 5:
53-                actualizarResultado(torneo);
54-                break;
55-            case 6:
56-                salir();
57-                break;
58-        }
59-    }
60-}
61
62- // Metodo para registrar equipo
63- public static void registrarEquipo(Torneo torneo) {
64-    System.out.println("Nombre del equipo 1: BSC");
65-    System.out.println("Nombre del equipo 2: LDU");
66-    System.out.println("Ingrese el resultado (ejemplo: 2-1): 5-0");
67-    String resultado = scanner.nextLine();
68-    torneo.actualizarResultado(resultado);
69-}
70
71- // Metodo para crear partido
72- public static void crearPartido(Torneo torneo) {
73-    System.out.println("Fecha: 2024-12-05");
74-    System.out.println("Estado: Finalizado");
75-    System.out.println("Resultado: 5-0");
76-    torneo.actualizarResultado(resultado);
77-}
78
79- // Metodo para listar equipos
80- public static void listarEquipos(Torneo torneo) {
81-    System.out.println("Equipos registrados:");
82-    torneo.mostrarEquipos();
83-}
84
85- // Metodo para mostrar historial de partidos
86- public static void mostrarHistorial(Torneo torneo) {
87-    System.out.println("Historial de partidos:");
88-    torneo.mostrarHistorial();
89-}
90
91- // Metodo para actualizar resultado de un partido
92- public static void actualizarResultado(Torneo torneo) {
93-    System.out.println("Ingrese el resultado (ejemplo: 2-1): 5-0");
94-    String resultado = scanner.nextLine();
95-    torneo.actualizarResultado(resultado);
96-}
97
98- // Metodo para salir
99- public static void salir() {
100-    System.out.println("¡Hasta luego!");
101-}
102
103- ...Program finished with exit code 0
104- Press ENTER to exit console.
```

CONCLUSIONES

- La POO es ideal para modelar sistemas complejos: Al representar entidades del mundo real como objetos, la POO facilita la comprensión y el mantenimiento del código. Los sistemas de gestión de proyectos basados en POO son altamente personalizables: Gracias a la modularidad y reutilización del código, se pueden adaptar a las necesidades específicas de cada organización.
- La POO mejora la eficiencia y productividad: Al organizar el código de manera lógica, se reducen los errores y se acelera el desarrollo. Los sistemas de gestión de proyectos

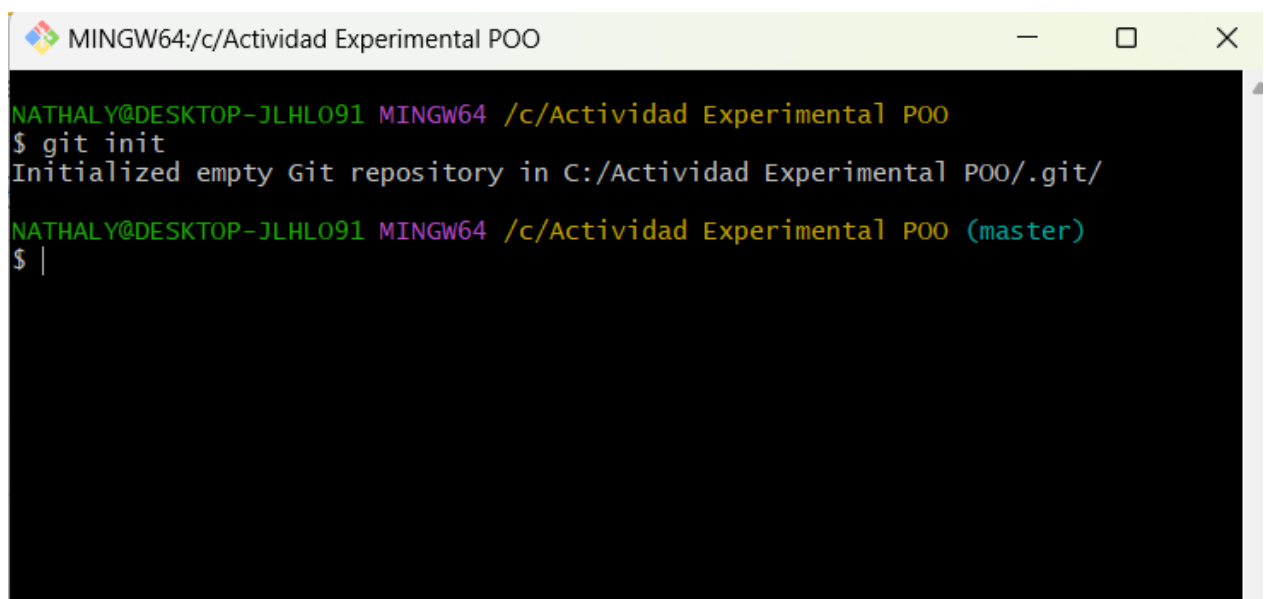
basados en POO son escalables: Pueden crecer y adaptarse a medida que aumentan los proyectos y los usuarios.

RECOMENDACIONES

- Planificación detallada: Antes de comenzar a codificar, es fundamental realizar un análisis detallado de los requisitos del sistema y crear un diseño de clases bien estructurado.
Elección del lenguaje de programación adecuado: Considera lenguajes como Java, que ofrecen un fuerte soporte para la POO.
- Realización de pruebas exhaustivas: Asegúrate de que el sistema funcione correctamente en diferentes escenarios y corrige cualquier error que se encuentre.
- Consideración de la interfaz de usuario: Diseña una interfaz intuitiva y fácil de usar para que los usuarios puedan interactuar con el sistema de manera eficiente.

EVIDENCIAS:

Proceso para subir a la plataforma GitHub.



```
MINGW64:/c/Actividad Experimental POO
NATHALY@DESKTOP-JLHLO91 MINGW64 /c/Actividad Experimental POO
$ git init
Initialized empty Git repository in C:/Actividad Experimental POO/.git/
NATHALY@DESKTOP-JLHLO91 MINGW64 /c/Actividad Experimental POO (master)
$ |
```

```
MINGW64:/c/Actividad Experimental POO
NATHALY@DESKTOP-JLHLO91 MINGW64 /c/Actividad Experimental POO
$ git init
Initialized empty Git repository in C:/Actividad Experimental POO/.git/

NATHALY@DESKTOP-JLHLO91 MINGW64 /c/Actividad Experimental POO (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    AE1_Grupo6_NRC_1322 (2).pdf
    Equipo.java
    Main (1).java
    Partido.java
    Torneo.java

nothing added to commit but untracked files present (use "git add" to track)

NATHALY@DESKTOP-JLHLO91 MINGW64 /c/Actividad Experimental POO (master)
$ |
```

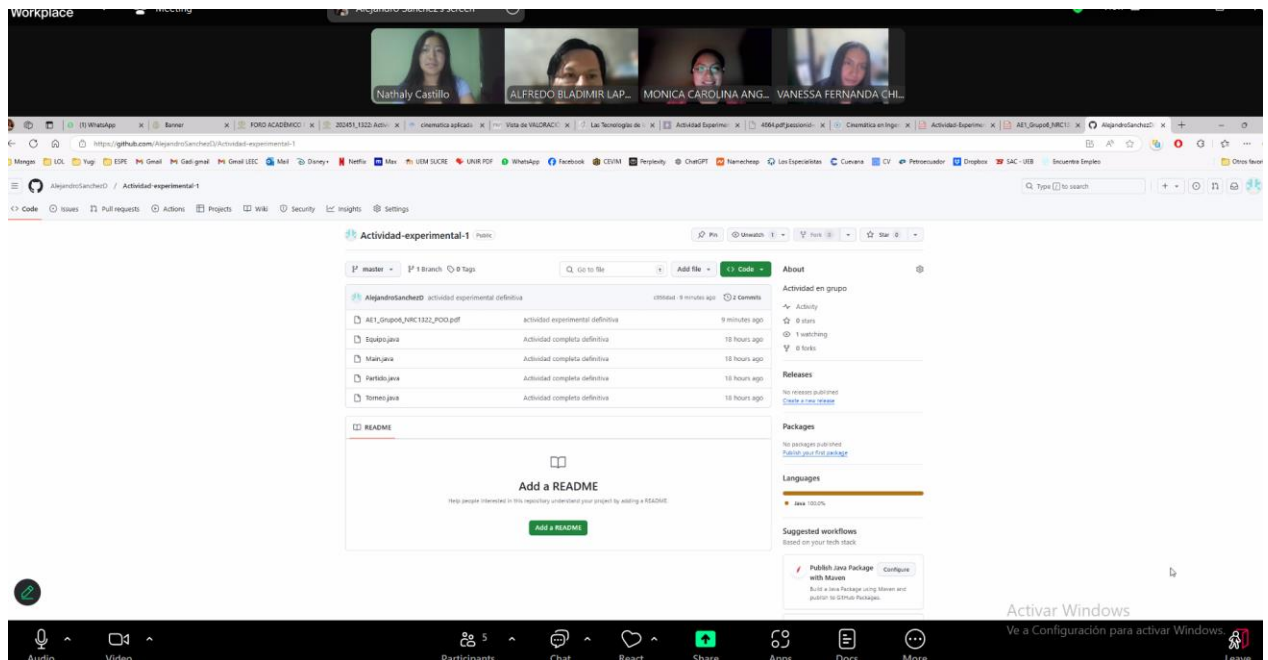
```
create mode 100644 AE1_Grupo6_NRC_1322 (2).pdf
create mode 100644 Equipo.java
create mode 100644 Main (1).java
create mode 100644 Partido.java
create mode 100644 Torneo.java

NATHALY@DESKTOP-JLHLO91 MINGW64 /c/Actividad Experimental POO (master)
$ git remote add origin git@github.com:Nathaly-Castillo/Actividad-experimental-1
.git

NATHALY@DESKTOP-JLHLO91 MINGW64 /c/Actividad Experimental POO (master)
$ git push -u origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 361.47 KiB | 2.51 MiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:Nathaly-Castillo/Actividad-experimental-1.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

NATHALY@DESKTOP-JLHLO91 MINGW64 /c/Actividad Experimental POO (master)
$
```


Reunión del grupo



REFERENCIAS BIBLIOGRÁFICAS

- MUÑOZ D., JAVIER F., GUTIERREZ L., PIMENTEL S., (2007). *Programación orientada a objetos con Java*. Ediciones Paraninfo, S.A.
- López, L. (2013). *Metodología de la programación orientada a objetos*. Alpha Editorial.