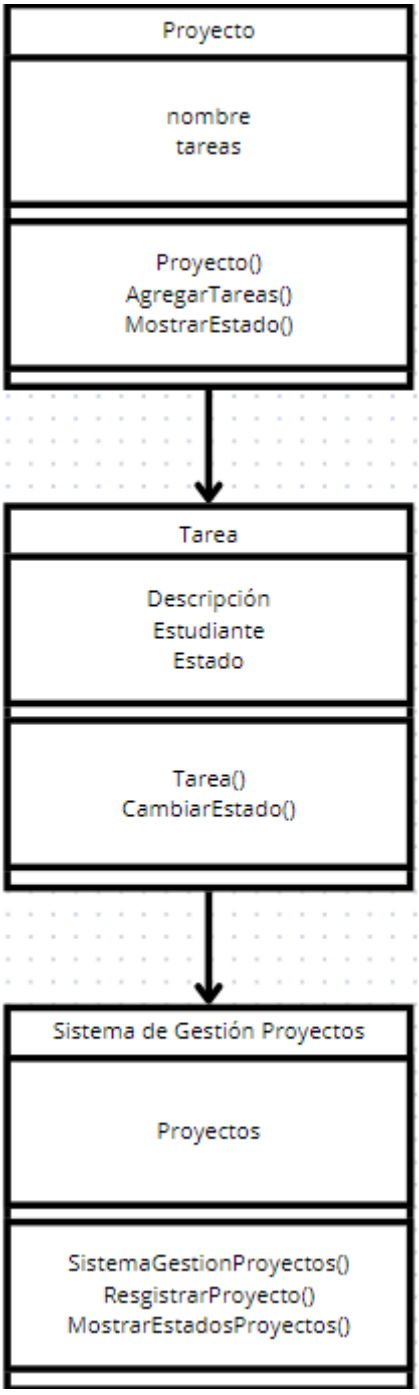
Universidad de las Fuerzas Armadas ESPE

Nombre: Dylan López

NRC: 1322

- **UML:**



- **Código:**

```
1 import java.util.ArrayList;
2 import java.util.List;
3 import java.util.Scanner;
```

- **import java.util.ArrayList**;: Esta clase se utiliza para crear listas dinámicas.
- **import java.util.List;**: Se utiliza para definir una lista. ArrayList es una implementación de esta interfaz.
- **import java.util.Scanner;**: Se usa para leer la entrada del usuario desde el teclado.

```
5 class Proyecto {
6     private String nombre;
7     private List<Tarea> tareas;
```

- **class Proyecto:** Define la clase Proyecto, que representa un proyecto con tareas asignadas.
- **private String nombre;:** Declara una variable de tipo String para almacenar el nombre del proyecto.
- **private List<Tarea> tareas;:** Declara una lista de objetos de tipo Tarea. Cada proyecto puede tener varias tareas.

```
9     public Proyecto(String nombre){
10        this.nombre = nombre;
11        this.tareas = new ArrayList<>();
12    }
```

- **public Proyecto(String nombre):** Es el constructor de la clase Proyecto. Recibe un String como parámetro para iniciar el nombre del proyecto.
- **this.nombre = nombre;**: Asigna el valor del parámetro nombre a la variable de instancia nombre de la clase.
- **this.tareas = new ArrayList<>();:** Inicializa la lista tareas como un nuevo ArrayList, asegurando que cada proyecto comience con una lista vacía de tareas.

```
14      public String getNombre(){
15          return nombre;
16      }
17
18      public void agregarTarea(Tarea tarea){
19          tareas.add(tarea);
20      }
```

- **public String getNombre():** Método getter que devuelve el nombre del proyecto.
- **public void agregarTarea(Tarea tarea):** Agrega una tarea a la lista tareas del proyecto.

```
22      public void mostrarEstado(){
23          System.out.println("Estado del proyecto: "+nombre);
24          if (tareas.isEmpty()) {
25              System.out.println("- Sin tareas asignadas");
26          } else {
27              for (Tarea tarea : tareas) {
28                  System.out.println("- " + tarea);
29              }
30          }
31      }
```

- **public void mostrarEstado():** Imprime el estado del proyecto y de sus tareas.
- **System.out.println("Estado del proyecto: "+nombre);:** Muestra el nombre del proyecto.
- **if (tareas.isEmpty()) {:** Si la lista de tareas está vacía, imprime un mensaje indicando que no hay tareas asignadas.
- **else { for (Tarea tarea : tareas) { :** Si el proyecto tiene tareas, imprime cada tarea en la lista utilizando el método toString() de la clase Tarea.

```
32  public void cambiarEstadoTareas() {
33          Scanner scanner = new Scanner(System.in);
34          for (Tarea tarea : tareas) {
35              System.out.print("¿Desea cambiar el estado de la tarea \"" + tarea.getDescripcion() + "\" a 'Finalizado'? (s/n): ");
36              String respuesta = scanner.nextLine();
37              if (respuesta.equalsIgnoreCase("s")) {
38                  tarea.cambiarEstado("Finalizado");
39                  System.out.println("Tarea \"" + tarea.getDescripcion() + "\" marcada como 'Finalizado'.");
40              }
41          }
42      }
43  }
```

- **public void cambiarEstadoTareas():** Este método permite al usuario cambiar el estado de las tareas a "Finalizado".
- **Scanner scanner = new Scanner(System.in);:** Crea un objeto Scanner para leer la entrada del usuario desde el teclado.
- **System.out.print("¿Desea cambiar el estado de la tarea ");:** Muestra un mensaje solicitando al usuario si desea cambiar el estado de la tarea a "Finalizado".
- **String respuesta = scanner.nextLine();:** Lee la respuesta del usuario.

- **if (respuesta.equalsIgnoreCase("s")) {:** Si la respuesta es "s" (sí), cambia el estado de la tarea a "Finalizado" y muestra un mensaje confirmando el cambio.

```
45 class Tarea {
46     private String descripcion;
47     private String responsable;
48     private String estado;
```

- **class Tarea:** Define la clase Tarea, que representa una tarea asociada a un proyecto.
- **private String descripcion**;: Almacena la descripción de la tarea.
- **private String responsable;:** Almacena el nombre del responsable de la tarea.
- **private String estado;:** Almacena el estado de la tarea "Pendiente" o "Finalizado").

```
50     public Tarea(String descripcion, String responsable){
51         this.descripcion = descripcion;
52         this.responsable = responsable;
53         this.estado = "Pendiente";
54     }
```

- **public Tarea(String descripcion, String responsable):** Constructor de la clase Tarea, que recibe una descripción y un responsable para crear una nueva tarea. El estado se inicializa como "Pendiente".

```
56     public String getDescripcion() {
57         return descripcion;
58     }
59
60     public void cambiarEstado(String nuevoEstado) {
61         this.estado = nuevoEstado;
62     }
63
64     public String toString(){
65         return descripcion + " (Estudiante: " + responsable + ", Estado: " + estado + ")";
66     }
67 }
```

- **public String getDescripcion():** Método getter para obtener la descripción de la tarea.
- public void cambiarEstado(String nuevoEstado): Método que cambia el estado de la tarea al valor pasado como argumento.
- **public String toString():** Sobrescribe el método toString() para devolver una representación en formato de texto de la tarea, incluyendo la descripción, el responsable y el estado.

```java
public class SistemaGestionProyectos {
    private List<Proyecto> proyectos;

    public SistemaGestionProyectos() {
        proyectos = new ArrayList<>();
    }

    public void registrarProyecto(String nombre) {
        proyectos.add(new Proyecto(nombre));
    }

    public Proyecto obtenerProyectoPorIndice(int indice) {
        if (indice>=0 && indice<proyectos.size()) {
            return proyectos.get(indice);
        }
        return null;
    }
```

- **public class SistemaGestionProyectos:** Define la clase principal que gestiona múltiples proyectos.
- **private List<Proyecto> proyectos;:** Declara una lista de proyectos, que se utilizará para almacenar todos los proyectos registrados.
- **public SistemaGestionProyectos():** Constructor de la clase, que inicializa la lista de proyectos como un ArrayList vacío.
- **public void registrarProyecto(String nombre):** Método que crea un nuevo proyecto con el nombre proporcionado y lo agrega a la lista de proyectos.
- **public Proyecto obtenerProyectoPorIndice(int indice):** Método que devuelve un proyecto específico de la lista de proyectos según su índice. Si el índice no es válido, devuelve null.

```java
    public void mostrarEstadosProyectos(){
        if (proyectos.isEmpty()) {
            System.out.println("No hay proyectos registrados.");
        } else {
            for (Proyecto proyecto : proyectos){
                proyecto.mostrarEstado();
                System.out.println();
            }
        }
    }
```

- **public void mostrarEstadosProyectos():** Método que muestra el estado de todos los proyectos registrados. Si no hay proyectos, muestra un mensaje indicándolo.

```
98      public static void main(String[] args) {
99          SistemaGestionProyectos sistema = new SistemaGestionProyectos();
100         Scanner scanner = new Scanner(System.in);
```

- **public static void main(String[] args):** Método principal que ejecuta el programa.
- **SistemaGestionProyectos sistema = new SistemaGestionProyectos();:** Crea una instancia del sistema de gestión de proyectos.
- **Scanner scanner = new Scanner(System.in);:** Crea un objeto Scanner para leer la entrada del usuario desde el teclado.

```
101         System.out.print("Ingrese el nombre del primer proyecto: ");
102         String nombreProyecto1 = scanner.nextLine();
103         sistema.registrarProyecto(nombreProyecto1);
104         System.out.print("Ingrese el nombre del segundo proyecto: ");
105         String nombreProyecto2 = scanner.nextLine();
106         sistema.registrarProyecto(nombreProyecto2);
```

- Solicita al usuario el nombre de un proyecto y lo registra en el sistema.
- Solicita el nombre del segundo proyecto y lo registra.

```
108     for (int i=0;i<2;i++) {
109         Proyecto proyecto = sistema.obtenerProyectoPorIndice(i);
110         if (proyecto != null) {
111             System.out.print("\n");
112             System.out.print("Ingrese la descripción de la tarea para el proyecto " + proyecto.getNombre() + ": ");
113             String descripcion = scanner.nextLine();
114             System.out.print("Ingrese el estudiante de la tarea: ");
115             String responsable = scanner.nextLine();
116             proyecto.agregarTarea(new Tarea(descripcion, responsable));
117             System.out.println("Tarea asignada al proyecto: " + proyecto.getNombre());
118         }
119     }
```

- Pide al usuario que ingrese una descripción y responsable para las tareas y las asigna a los proyectos.

```
120         System.out.print("\n");
121         System.out.println(" ----Estado de los proyectos---- ");
122         sistema.mostrarEstadosProyectos();
```

- Muestra el estado inicial de los proyectos.

```
124         for (Proyecto proyecto : sistema.proyectos) {
125             proyecto.cambiarEstadoTareas();
126         }
```

- Llama a cambiarEstadoTareas() para cada proyecto, permitiendo al usuario cambiar el estado de las tareas.

```
        System.out.print("\n");
        System.out.println(" ----Estado final de los proyectos---- ");
        sistema.mostrarEstadosProyectos();
        scanner.close();
    }
}
```

- Muestra el estado final de los proyectos después de haber cambiado el estado de las tareas. Luego cierra el scanner para liberar los recursos.

- **Código Completo:**

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

class Proyecto {
    private String nombre;
    private List<Tarea> tareas;

    public Proyecto(String nombre){
        this.nombre = nombre;
        this.tareas = new ArrayList<>();
    }

    public String getNombre(){
        return nombre;
    }

    public void agregarTarea(Tarea tarea){
        tareas.add(tarea);
    }

    public void mostrarEstado(){
        System.out.println("Estado del proyecto: "+nombre);
        if (tareas.isEmpty()) {
            System.out.println("- Sin tareas asignadas");
        } else {
            for (Tarea tarea : tareas) {
                System.out.println("- " + tarea);
            }
        }
    }
public void cambiarEstadoTareas() {
        Scanner scanner = new Scanner(System.in);
        for (Tarea tarea : tareas) {
            System.out.print("¿Desea cambiar el estado de la tarea \"" + tarea.getDescripcion() + "\" a 'Finalizado'? (s/n): ");
            String respuesta = scanner.nextLine();
            if (respuesta.equalsIgnoreCase("s")) {
                tarea.cambiarEstado("Finalizado");
                System.out.println("Tarea \"" + tarea.getDescripcion() + "\" marcada como 'Finalizado'.");
            }
        }
    }
}

class Tarea {
    private String descripcion;
    private String responsable;
    private String estado;

    public Tarea(String descripcion, String responsable){
        this.descripcion = descripcion;
        this.responsable = responsable;
        this.estado = "Pendiente";
```

```java
    }

    public String getDescripcion() {
        return descripcion;
    }

    public void cambiarEstado(String nuevoEstado) {
        this.estado = nuevoEstado;
    }

    public String toString(){
        return descripcion + " (Estudiante: " + responsable + ", Estado: " + estado + ")";
    }
}

public class SistemaGestionProyectos {
    private List<Proyecto> proyectos;

    public SistemaGestionProyectos() {
        proyectos = new ArrayList<>();
    }

    public void registrarProyecto(String nombre) {
        proyectos.add(new Proyecto(nombre));
    }

    public Proyecto obtenerProyectoPorIndice(int indice) {
        if (indice>=0 && indice<proyectos.size()) {
            return proyectos.get(indice);
        }
        return null;
    }

    public void mostrarEstadosProyectos(){
        if (proyectos.isEmpty()) {
            System.out.println("No hay proyectos registrados.");
        } else {
            for (Proyecto proyecto : proyectos){
                proyecto.mostrarEstado();
                System.out.println();
            }
        }
    }

    public static void main(String[] args) {
        SistemaGestionProyectos sistema = new SistemaGestionProyectos();
        Scanner scanner = new Scanner(System.in);
        System.out.print("Ingrese el nombre del primer proyecto: ");
        String nombreProyecto1 = scanner.nextLine();
        sistema.registrarProyecto(nombreProyecto1);
        System.out.print("Ingrese el nombre del segundo proyecto: ");
        String nombreProyecto2 = scanner.nextLine();
        sistema.registrarProyecto(nombreProyecto2);
```

```
107
108         for (int i=0;i<2;i++) {
109             Proyecto proyecto = sistema.obtenerProyectoPorIndice(i);
110             if (proyecto != null) {
111                 System.out.print("\n");
112                 System.out.print("Ingrese la descripción de la tarea para el proyecto " + proyecto.getNombre() + ": ");
113                 String descripcion = scanner.nextLine();
114                 System.out.print("Ingrese el estudiante de la tarea: ");
115                 String responsable = scanner.nextLine();
116                 proyecto.agregarTarea(new Tarea(descripcion, responsable));
117                 System.out.println("Tarea asignada al proyecto: " + proyecto.getNombre());
118             }
119         }
120         System.out.print("\n");
121         System.out.println(" ----Estado de los proyectos---- ");
122         sistema.mostrarEstadosProyectos();
123 
124         for (Proyecto proyecto : sistema.proyectos) {
125             proyecto.cambiarEstadoTareas();
126         }
127 
128         System.out.print("\n");
129         System.out.println(" ----Estado final de los proyectos---- ");
130         sistema.mostrarEstadosProyectos();
131         scanner.close();
132     }
133 }
```