Material descargable

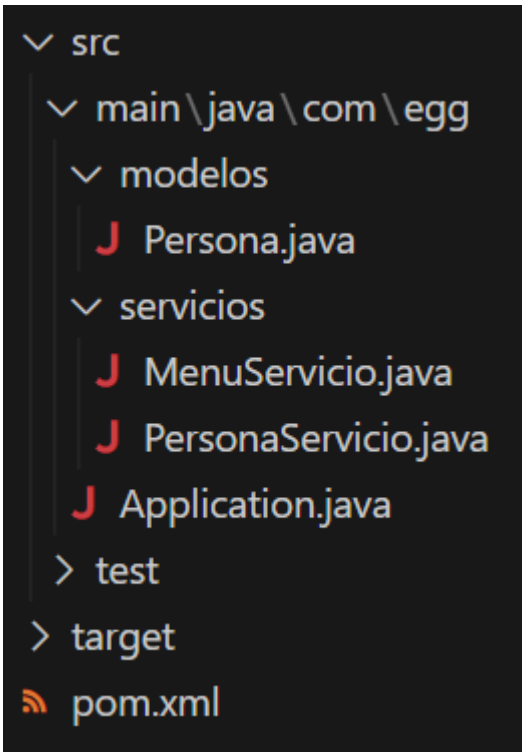
Patrón experto y las clases de servicio

egg

El Patrón Experto, uno de los principios de GRASP (General Responsibility Assignment Software Patterns), orienta el diseño de software hacia objetos responsables. Este principio sugiere que **la responsabilidad de una tarea específica debe asignarse al módulo o clase que posea la información necesaria para ejecutarla de manera óptima**.

Por ejemplo, el método **imprimirPropiedades()** de la clase **Persona** debería devolver un **String** en lugar de realizar directamente una impresión con **System.out.println()**. La tarea de impresión debería delegarse a otra clase, mientras que el método debería renombrarse a **obtenerPropiedades()** para reflejar su función más precisa.

Esta clase responsable de la impresión se conoce comúnmente como "Servicio", por lo que en este caso sería "PersonaServicio" y debería ubicarse en un paquete "servicios" al mismo nivel que el directorio de "modelos".



La clase **PersonaServicio** debe encapsular toda la lógica relacionada con la manipulación de objetos **Persona**, incluyendo la creación, validación y manipulación de datos. También es visible cómo existe un **MenuServicio**, el cual gestiona la lógica del menú y la navegación dentro de la aplicación.

Una posible mejora en la clase **PersonaServicio** sería mantener un arreglo dinámico o una estructura de datos más flexible para almacenar las personas, en lugar de un arreglo estático de tamaño fijo.

```
public class PersonaServicio {

    private Scanner scanner = new Scanner(System.in);
    private List<Persona> personas = new ArrayList<>();

    public Persona crearPersona() {
        System.out.println("Ingrese un nombre:");
        String nombre = scanner.nextLine();
        System.out.println("Ingrese una edad:");
        int edad = scanner.nextInt();
         scanner.nextLine(); // Consumir la nueva línea pendiente después de
nextInt()
        Persona persona = new Persona(nombre, edad);
        almacenarPersona(persona);
        return persona;
    }

    public void almacenarPersona(Persona persona) {
        personas.add(persona);
    }
}
```

La clase **MenuServicio** debe tener una referencia a **PersonaServicio** para interactuar con los métodos relacionados.

```
public void generarMenu() {
        System.out.println("Menú:");
        System.out.println("1 - Crear persona");
        System.out.println("Seleccione una opción:");
        int opcion = scanner.nextInt();
         scanner.nextLine(); // Consumir la nueva línea pendiente después de
nextInt()
        switch (opcion) {
            case 1:
                opcionCrearPersona();
                break;
            default:
                System.out.println("Opción no válida.");
        }
    }
```

```
    public void opcionCrearPersona() {
        personaServicio.crearPersona();
    }
}
```

Finalmente, la clase Application debe utilizar MenuServicio para iniciar el programa.

```
public class Application {

    public static void main(String[] args) {
        PersonaServicio personaServicio = new PersonaServicio();
        MenuServicio menuServicio = new MenuServicio(personaServicio);
        menuServicio.generarMenu();
    }
}
```

La aplicación organizada de esta manera promueve un código más claro, modular y mantenible.