CURSO JAVA 1

Programación Orientada a Objetos

Clase Servicio

OBJETIVOS DE LA GUÍA

En esta guía aprenderemos a:

- Utilizar arquitectura en capas
- Utilizar una clase servicio con métodos que siguen reglas del negocio

egg | Argentina programa 4.0

Clase servicio

Vamos a continuar con el paradigma orientado a objetos pero ahora vamos a ver algo que no es parte realmente del paradigma orientado a objetos, sino que es una clase que nos va a ayudar mucho a la realización del paradigma orientado a objetos y es la clase servicio.

¿Qué es la clase servicio?

La clase servicio (service) o control, va a ser una clase auxiliar que nos va a ayudar con el manejo de las clases y los objetos de esas clases, pero para poder explicar esto, primero vamos a tener que ver **los patrones generales de software GRASP**. Aunque se considera que más que **patrones** propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software.

Patrones grasp

Un proceso de desarrollo sirve para normalizar quién hace qué cosa en cada momento y cómo debe realizarse esta cosa.

GRASP es el acrónimo de General **Responsibility Assignment Software Patterns**. Una de las cosas más complicadas en Orientación a Objeto consiste en elegir las clases adecuadas y decidir cómo estas clases deben interactuar.

Patron Experto

Dentro de los patrones GRASP, vamos a utilizar el patrón experto. El GRASP de experto en información es el principio básico de asignación de responsabilidades. Nos indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo obtendremos un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento).

Problema

¿Cuál es el principio general para asignar responsabilidades a los objetos?

Solución

Asignar una responsabilidad al experto en información.

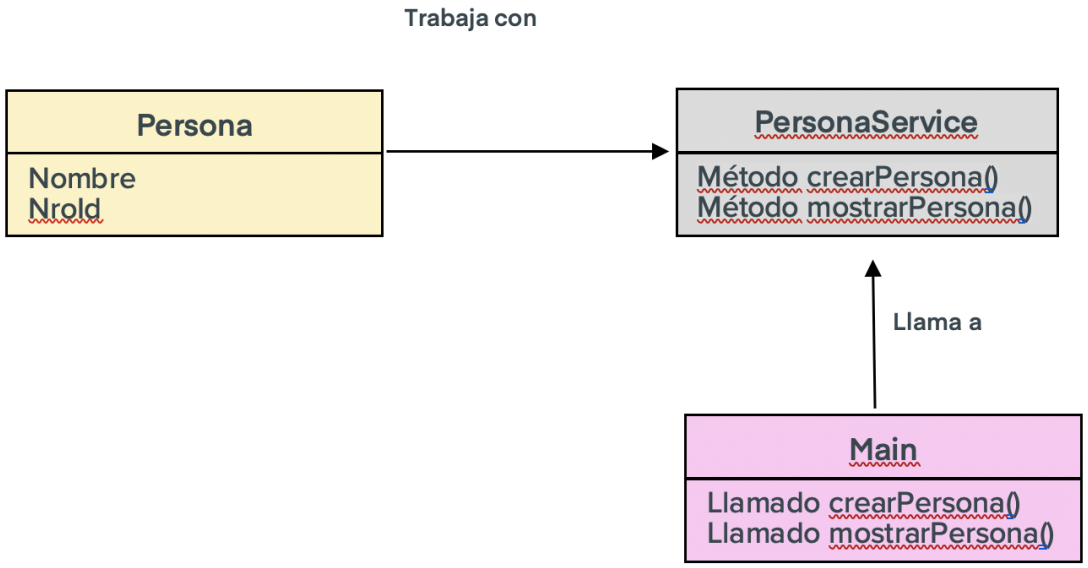
Beneficios

Se mantiene el encapsulamiento, los objetos utilizan su propia información para llevar a cabo sus tareas. Se distribuye el comportamiento entre las clases que contienen la información requerida. Son más fáciles de entender y mantener.

Clase servicio

Esta clase del patrón experto va a ser la clase servicio. Es una clase común y corriente pero que se va a encargar de crear los objetos y va a tener todos los métodos necesarios para la utilización de ese objeto. Supongamos que necesitamos un método que le sume un valor x a un atributo del objeto, este método estará en la clase control.

Siempre se crea una clase control, por cada clase que tengamos, si tenemos las clases Persona y Sueldo, crearemos una clase control para Persona y otra para Sueldo. La idea es que una clase servicio, se encargue de solo una clase.



Como podemos ver en el diagrama, PersonaService llama y trabaja con la clase Persona y crea instancias de ese objeto a través de un método, también, tiene un método para mostrar la persona.

Pero, el encargado de llamar a esos métodos de la clase PersonaService va a ser la clase Main, que vendría a ser el usuario llamando a las cosas que quiere realizar con la Persona.

Para trabajar con clase de servicio o control debemos crear un nuevo paquete en nuestro proyecto que se llame Servicios.

Métodos clase servicio

Como pudimos ver en el diagrama anterior la clase PersonaService, tenía sus propios métodos, que iban a trabajar con la clase Persona, pero no que no se encuentran en la clase Persona, sino en PersonaService. Es importante que a la hora de trabajar con métodos dentro de la clase servicio o métodos externos, tengamos en cuenta las siguientes cosas:

Parámetros y argumentos

Los parámetros de un método definen la cantidad y el tipo de dato de los valores que recibe un método para su ejecución. Los argumentos son los valores que se pasan a un método durante su invocación. El método recibe los argumentos correspondientes a los parámetros con los que ha sido declarado. Un método puede tener tantos parámetros como sea necesario. La lista de parámetros de la cabecera de un método se define con la siguiente sintaxis:

```
tipo nombre [tipo nombre,]
```

Durante la invocación de un método es necesario que el número y el tipo de argumentos coincidan con el número y el tipo de parámetros declarados en la cabecera del método. Durante el proceso de compilación se comprueba que durante la invocación de un método se pasan tantos argumentos como parámetros tiene declarados y que además coinciden los tipos. Esta es una característica de los lenguajes que se denominan “strongly tiped” o “fuertemente tipado”

Paso de parámetros

Cuando se invoca un método se hace una copia de los valores de los argumentos en los parámetros. Esto quiere decir que, si el método modifica el valor de un parámetro, nunca se modifica el valor original del argumento.

Cuando se pasa una referencia a un objeto se crea un nuevo alias sobre el objeto, de manera que esta nueva referencia utiliza el mismo espacio de memoria del objeto original y esto permite acceder al objeto original.

El valor de retorno

Un método puede devolver un valor. Los métodos que no devuelven un valor se declaran void, mientras que los métodos que devuelven un valor indican el tipo que devuelven: int, double, char, String o un tipo de objeto.

Sólo utiliza métodos void para mostrar información. Si realizarás cambios en los atributos del objeto, envía ese objeto por parámetro y retornalo con los cambios realizados

Sobrecarga de métodos

La sobrecarga de métodos es útil para que el mismo método opere con parámetros de distinto tipo o que un mismo método reciba una lista de parámetros diferente. Esto quiere decir que puede haber dos métodos con el mismo nombre que realicen dos funciones distintas. La diferencia entre los métodos sobrecargados está en su declaración, y más específicamente, en la cantidad y tipos de datos que reciben.

¿NECESITAS UN **EJEMPLO**?

```java
public Persona crearPersona(){
    //Instanciamos un objeto persona con sus atributos vacíos
    Persona personaCompleta = new Persona();

    //Pedimos al usuario que ingrese la informacion
    //que se alojará en el atributo por consola
    System.out.println("Ingrese el nombre de la persona");

    //Utilizamos el objeto para invocar al método SET
    //Y con el Scanner recibimos la información
    personaCompleta.setNombre(leer.next());


    //este método retorna un objeto persona son sus atributos
    //llenos de información
    return personaCompleta;
}

```

En el Main:

```java
        //Debemos instanciar un objeto del tipo Servicio
        //para acceder a sus métodos
        PersonaService persServicio = new PersonaService();

        //Alojamos el retorno del método en un objeto tipo Persona
        Persona terceraPersona = persServicio.crearPersona();

    }
```

¡MANOS A LA OBRA!

Ejercicio void

Crea un método void que reciba un objeto tipo persona como parámetro y utilice el get para mostrar sus atributos. Llama ese método desde el main.