

### Tipos

Son plantillas que identifican problemas en el sistema y proporcionan soluciones apropiadas a problemas generales

- Creacionales
- Estructurales
- Comportamiento

### Patron Creacional

Factory Method	Provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created.	Can be used to extend internal components
Abstract Factory	Produce families of related objects <i>without</i> specifying their concrete classes	An object that has multiple Factory Methods
Builder Pattern	Lets you construct complex objects step by step. The pattern allows you to produce different types and representations of an object using the same construction code.	
Singleton	Ensures that a class has just a single instance and provide global access point to that instance	Used to have a stricter control over global variables
Prototype	Lets you copy existing objects without making your code dependent on their classes.	

Proporcionan diversos mecanismos de **creación de objetos**, lo cual aumenta la **flexibilidad** y la **reutilización** del código existente.

### Ejemplo



### Patrones Estructurales

Adapter	Se utiliza para vincular 2 interfases que no son compatibles
Bridge	Una interfaz conecta 2 o mas clases las cuales se desarrollan de manera independiente
Composite	Se usa para agrupar objetos como un solo objeto y luego trabajar con estas como si fueran objetos individuales
Facade	Proporciona una interfaz simplificada para un conjunto complejo de clases
Flyweight	Reduce el uso de memoria y mejora el rendimiento al reducir la creación de objetos
Proxy	Se utiliza para crear objetos que puedan representar objetos de otras clases u objetos y la interfaz se utiliza para acceder a estas funcionalidades

Nos especifican como los objetos y las clases se relacionan entre ellos para formar estructuras más complejas



By **FreddyAlmeida**

Not published yet.

Last updated 27th July, 2022.

Page 2 of 2.

Sponsored by **ApolloPad.com**

Everyone has a novel in them. Finish

Yours!

<https://apollopad.com>

### Patrones Comportamiento

- Chain of Responsibility** - Evita acoplar el emisor de una petición a su receptor; Da a más de un objeto la posibilidad de responder a una petición
- Command** - Sugiere encapsular la lógica de ciertas funcionalidades en objetos para luego ser abstraídos y usados en un orden específico de manera independientes

Se ocupa de la **comunicación** entre objetos de clase

### Patron - Singleton

```
public class Singleton {
    private String nombre;
    private static Singleton singleton;

    // El constructor es privado, no permite que se genere un constructor por defecto.
    private Singleton(String nombre) {
        this.nombre = nombre;
        System.out.println("Mi nombre es: " + this.nombre);
    }

    public static Singleton getInstance(String nombre) {
        if (singleton == null) {
            singleton = new Singleton(nombre);
        }
        else {
            System.out.println("No se puede crear el objeto " + nombre + " porque ya existe un objeto de la clase Singleton");
        }
        return singleton;
    }
    // metodos getter y setter
}
```

**Beneficios** - Siempre se accedera al mismo pool

**Desventajas** - No se debe usar en ambientes con multihilos; difícil control



By **FreddyAlmeida**

[cheatography.com/freddyalmeida/](https://cheatography.com/freddyalmeida/)

Not published yet.

Last updated 27th July, 2022.

Page 3 of 2.

Sponsored by **ApolloPad.com**

Everyone has a novel in them. Finish Yours!

<https://apollopap.com>