Patrón de fabricación abstracta

Abstract Factory es un patrón de diseño creacional que nos permite producir familias de objetos relacionados sin especificar sus clases concretas.
El patrón Abstract Factory es útil cuando un objeto cliente desea crear un conjunto de instancias de clases relacionadas y dependientes, sin tener que conocer cuales clases específicas y concretas son instanciadas, manteniendo las restricciones propias de la familia de objetos.

Interfaz Abstracta para la Fábrica

- Se define la interfaz FactoryControlador, la cual obliga a las fábricas concretas a implementar el método crearControlador().
- Retorna un Object.

```
package factory;
import controlador.*;

public interface FactoryControlador {
    Object crearControlador();
}
```

Fábrica concreta para ControladorCliente

- Implementa la interfaz FactoryControlador y su método crearControlador().
- Usa un patrón Builder para configurar la instancia de ControladorCliente, activando logs y mensajes de error.

```
package factory;

import controlador.ControladorCliente;

public class FactoryCliente implements FactoryControlador {
    @Override
    public ControladorCliente crearControlador() {
        return new ControladorCliente.Builder()
                .habilitarLogs(true)
                .mostrarMensajesError(true)
                .build();
    }
}
```

Fábrica concreta para ControladorProducto

- Similar a FactoryCliente, pero para ControladorProducto.
- Usa un Builder para configurar la instancia de ControladorProducto.

```
package factory;

import controlador.ControladorProducto;

public class FactoryProducto implements FactoryControlador {
    @Override
    public ControladorProducto crearControlador() {
        return new ControladorProducto.Builder()
                .habilitarLogs(true)
                .mostrarMensajesError(true)
                .build();
    }
}
```

Fábrica concreta para ControladorReporte

- Implementa FactoryControlador y retorna una instancia de ControladorReporte.
- A diferencia de los anteriores, ControladorReporte no usa el patrón Builder.

```
package factory;

import controlador.ControladorReporte;

public class FactoryReporte implements FactoryControlador {
    @Override
    public ControladorReporte crearControlador() {
        return new ControladorReporte();
    }
}
```

Uso en un formulario (FormClientes)

- FormClientes es un formulario que necesita un ControladorCliente.
- Se instancia una fábrica concreta (FactoryCliente) y se usa para crear el controlador.
- Se hace un casting explícito de Object a ControladorCliente porque FactoryControlador devuelve un Object.

```
package formulario;
import factory.*;
import controlador.*;

public class FormClientes {
    private ControladorCliente controlador;

    public FormClientes() {
        FactoryControlador factory = new FactoryCliente();
                        this.controlador   =   (ControladorCliente)
factory.crearControlador();
    }
}
```

Diagrama UML