PROGETTO SPRING BOOT PATTERN

✅ 1. Creazione del progetto su Spring Initializr

Vai su 👉 https://start.spring.io e inserisci:

Campo	Valore
Group	`it.epicode`
Artifact	`patterns`
Name	`patterns`
Packaging	`Jar`
Language	`Java`
Version	ultima stabile
Dependencies	✅ `Lombok`

➡️ Clicca su **Generate** e importa lo zip generato nel tuo IDE (IntelliJ o Eclipse).

📁 2. Struttura del progetto

```
src/
└── main/
    └── java/
        └── it/
            └── epicode/
                └── patterns/
                    ├── DesignPatternApp.java
                    ├── adapter/
                    │   ├── AdapterRunner.java
                    │   ├── UtenteGenerico.java
                    │   ├── UtenteSistemaA.java
                    │   ├── UtenteSistemaB.java
                    │   └── UtenteBAdapter.java
                    ├── centrocosto/
                    │   ├── CentroDiCosto.java
                    │   ├── Dipendente.java
                    │   ├── Dipartimento.java
                    │   └── CentroDiCostoRunner.java
```

```
└── chain/
    ├── Supporto.java
    ├── Junior.java
    ├── Senior.java
    ├── Manager.java
    └── ChainRunner.java
```

🚀 3. Avvio applicazione Spring

DesignPatternApp.java

```java
package it.epicode.patterns;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DesignPatternApp {
    public static void main(String[] args) {
        SpringApplication.run(DesignPatternApp.class, args);
    }
}
```

🧩 4. Adapter – Due sistemi utente diversi

adapter/UtenteGenerico.java

```java
package it.epicode.patterns.adapter;

public interface UtenteGenerico {
    String getNomeCompleto();
    int getEta();
}
```

adapter/UtenteSistemaA.java

```java
package it.epicode.patterns.adapter;

import lombok.AllArgsConstructor;
import lombok.Getter;

@Getter
```

```java
@AllArgsConstructor
public class UtenteSistemaA implements UtenteGenerico {
    private String nomeCompleto;
    private int eta;
}
```

adapter/UtenteSistemaB.java

```java
package it.epicode.patterns.adapter;

import lombok.AllArgsConstructor;
import lombok.Getter;

import java.time.LocalDate;

@Getter
@AllArgsConstructor
public class UtenteSistemaB {
    private String nome;
    private String cognome;
    private LocalDate dataNascita;
}
```

adapter/UtenteBAdapter.java

```java
package it.epicode.patterns.adapter;

import java.time.LocalDate;
import java.time.Period;

public class UtenteBAdapter implements UtenteGenerico {
    private final UtenteSistemaB utente;

    public UtenteBAdapter(UtenteSistemaB utente) {
        this.utente = utente;
    }

    @Override
    public String getNomeCompleto() {
        return utente.getNome() + " " + utente.getCognome();
    }

    @Override
    public int getEta() {
        return Period.between(utente.getDataNascita(), LocalDate.now()).getYears();
    }
}
```

adapter/AdapterRunner.java

```java
package it.epicode.patterns.adapter;

import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

import java.time.LocalDate;

@Component
public class AdapterRunner implements CommandLineRunner {
    @Override
    public void run(String... args) {
        UtenteGenerico u1 = new UtenteSistemaA("Mario Rossi", 40);
        UtenteGenerico u2 = new UtenteBAdapter(new UtenteSistemaB("Luca", "Bianchi", L

        System.out.println("Adapter:");
        System.out.println(u1.getNomeCompleto() + " - Età: " + u1.getEta());
        System.out.println(u2.getNomeCompleto() + " - Età: " + u2.getEta());
        System.out.println();
    }
}
```

📚 5. Composite – Centro di Costo

centrocosto/CentroDiCosto.java

```java
package it.epicode.patterns.centrocosto;

public interface CentroDiCosto {
    String getNome();
    int getCosto();
}
```

centrocosto/Dipendente.java

```java
package it.epicode.patterns.centrocosto;

import lombok.AllArgsConstructor;
import lombok.Getter;

@Getter
@AllArgsConstructor
public class Dipendente implements CentroDiCosto {
    private String nome;
```

```
    private int costo;
}
```

centrocosto/Dipartimento.java

```
package it.epicode.patterns.centrocosto;

import java.util.ArrayList;
import java.util.List;

public class Dipartimento implements CentroDiCosto {
    private final String nome;
    private final List<CentroDiCosto> elementi = new ArrayList<>();

    public Dipartimento(String nome) {
        this.nome = nome;
    }

    public void aggiungi(CentroDiCosto cdc) {
        elementi.add(cdc);
    }

    @Override
    public String getNome() {
        return nome;
    }

    @Override
    public int getCosto() {
        return elementi.stream().mapToInt(CentroDiCosto::getCosto).sum();
    }
}
```

centrocosto/CentroDiCostoRunner.java

```
package it.epicode.patterns.centrocosto;

import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

@Component
public class CentroDiCostoRunner implements CommandLineRunner {
    @Override
    public void run(String... args) {
        Dipartimento azienda = new Dipartimento("Azienda");
        Dipartimento it = new Dipartimento("IT");

        it.aggiungi(new Dipendente("Anna", 2000));
        it.aggiungi(new Dipendente("Luca", 2500));
```

```
        azienda.aggiungi(it);
        azienda.aggiungi(new Dipendente("CEO", 5000));

        System.out.println("Composite (Centro di Costo):");
        System.out.println("Costo totale: " + azienda.getCosto());
        System.out.println();
    }
}
```

🪖 6. Chain of Responsibility – Supporto

chain/Supporto.java

```
package it.epicode.patterns.chain;

public abstract class Supporto {
    protected Supporto superiore;

    public void setSuperiore(Supporto superiore) {
        this.superiore = superiore;
    }

    public abstract void gestisci(int livello);
}
```

chain/Junior.java

```
package it.epicode.patterns.chain;

public class Junior extends Supporto {
    @Override
    public void gestisci(int livello) {
        if (livello <= 1)
            System.out.println("Gestita da Junior");
        else if (superiore != null)
            superiore.gestisci(livello);
    }
}
```

chain/Senior.java

```
package it.epicode.patterns.chain;
```

```
public class Senior extends Supporto {
    @Override
    public void gestisci(int livello) {
        if (livello <= 2)
            System.out.println("Gestita da Senior");
        else if (superiore != null)
            superiore.gestisci(livello);
    }
}
```

chain/Manager.java

```
package it.epicode.patterns.chain;

public class Manager extends Supporto {
    @Override
    public void gestisci(int livello) {
        if (livello <= 3) System.out.println("Gestita da Manager");
        else  System.out.println("Richiesta troppo complessa");
    }
}
```

chain/ChainRunner.java

```
package it.epicode.patterns.chain;

import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

@Component
public class ChainRunner implements CommandLineRunner {
    @Override
    public void run(String... args) {
        Supporto junior = new Junior();
        Supporto senior = new Senior();
        Supporto manager = new Manager();
        junior.setSuperiore(senior);
        senior.setSuperiore(manager);
        System.out.println("Chain of Responsibility:");
        junior.gestisci(1);
        junior.gestisci(2);
        junior.gestisci(3);
        junior.gestisci(4);
        System.out.println();
    }
}
```