Aula sobre Injeção de Dependência (DI) em Java

1. O que é Injeção de Dependência?

Injeção de dependência é uma técnica usada para desacoplar a criação de objetos e suas dependências. Ao invés de uma classe criar suas próprias dependências, elas são passadas (ou 'injetadas') de fora. Isso é feito de forma que a classe não precise saber como suas dependências são criadas, promovendo flexibilidade e manutenibilidade.

2. Por que usar DI?

O principal benefício de usar DI é desacoplar os componentes do sistema, permitindo que as classes e objetos não precisem se preocupar com a criação de suas dependências. Isso proporciona vários benefícios:

- Flexibilidade: Facilita a troca de implementações.
- Testabilidade: Facilita o uso de mocks ou stubs para testar unidades de código.
- Manutenibilidade: Ao desacoplar a criação de objetos, é mais fácil modificar a implementação de uma dependência sem afetar a classe que a utiliza.

3. Tipos de Injeção de Dependência

Existem três formas principais de injeção de dependência:

3.1. Injeção via Construtor

A dependência é fornecida no momento da criação do objeto. Esse é o tipo mais comum e amplamente usado.

Exemplo:

```java
class Carro {
    private Motor motor;

    public Carro(Motor motor) {
        this.motor = motor;
    }
}
```

3.2. Injeção via Setter

A dependência é fornecida por um método setter após a criação do objeto.

Exemplo:

```java
class Carro {
    private Motor motor;

    public void setMotor(Motor motor) {
        this.motor = motor;
    }
}
```

3.3. Injeção via Interface

A dependência é fornecida através de uma interface, permitindo que a classe utilize a implementação da dependência sem conhecer sua classe concreta.

Exemplo:

```java
interface Motor {
    void ligar();
}

class Carro {
    private Motor motor;

    public Carro(Motor motor) {
        this.motor = motor;
    }
}
```

4. Como a Injeção de Dependência é aplicada em frameworks como o Spring?

Frameworks como o Spring automatizam a injeção de dependências, gerenciando o ciclo de vida dos objetos e suas dependências.

4.1. Spring IoC Container

O Spring tem um container de Inversão de Controle (IoC) que gerencia os objetos e suas dependências. Você configura as dependências, e o Spring as injeta automaticamente.

Exemplo com Spring:

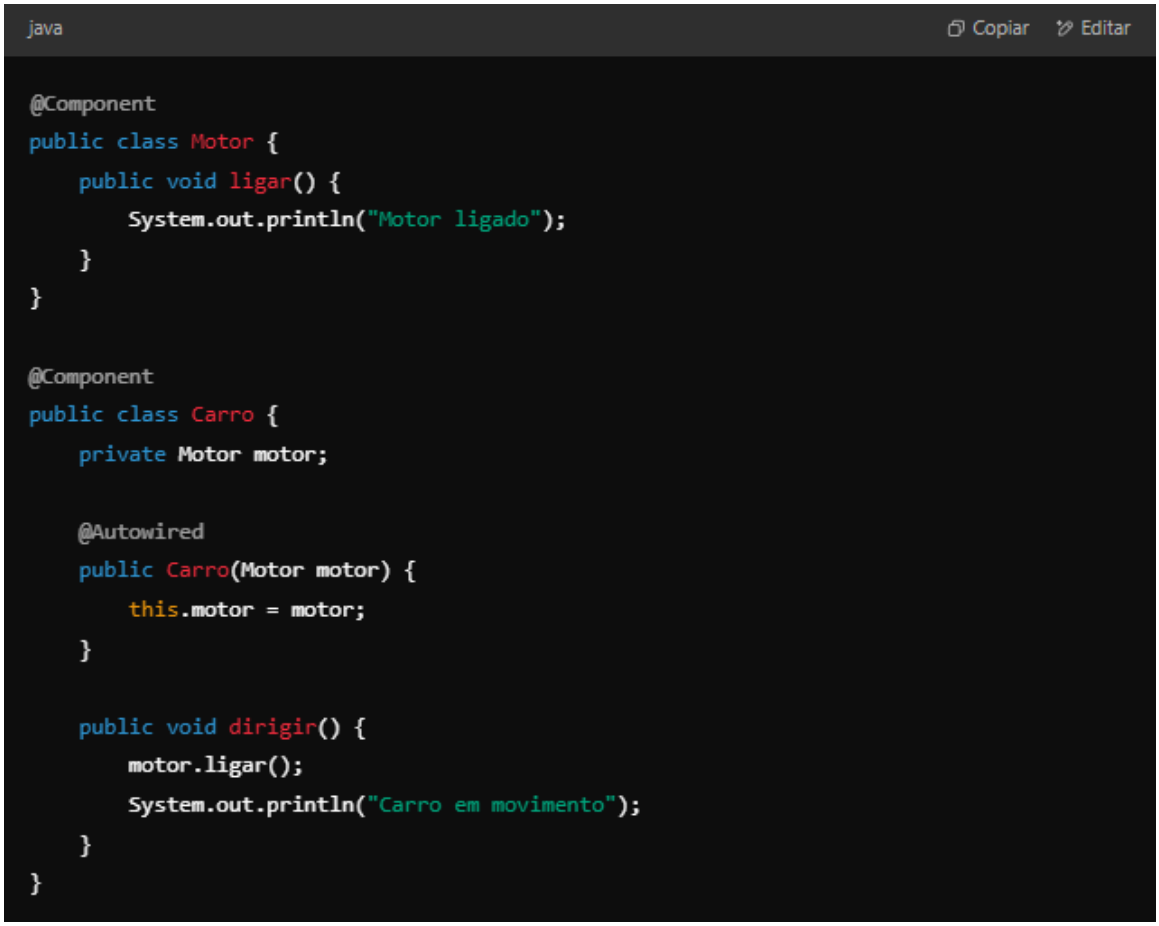
```java
@Component
public class Carro {
    private Motor motor;

    @Autowired
    public Carro(Motor motor) {
        this.motor = motor;
    }
}
```

4.2. Anotação @Autowired

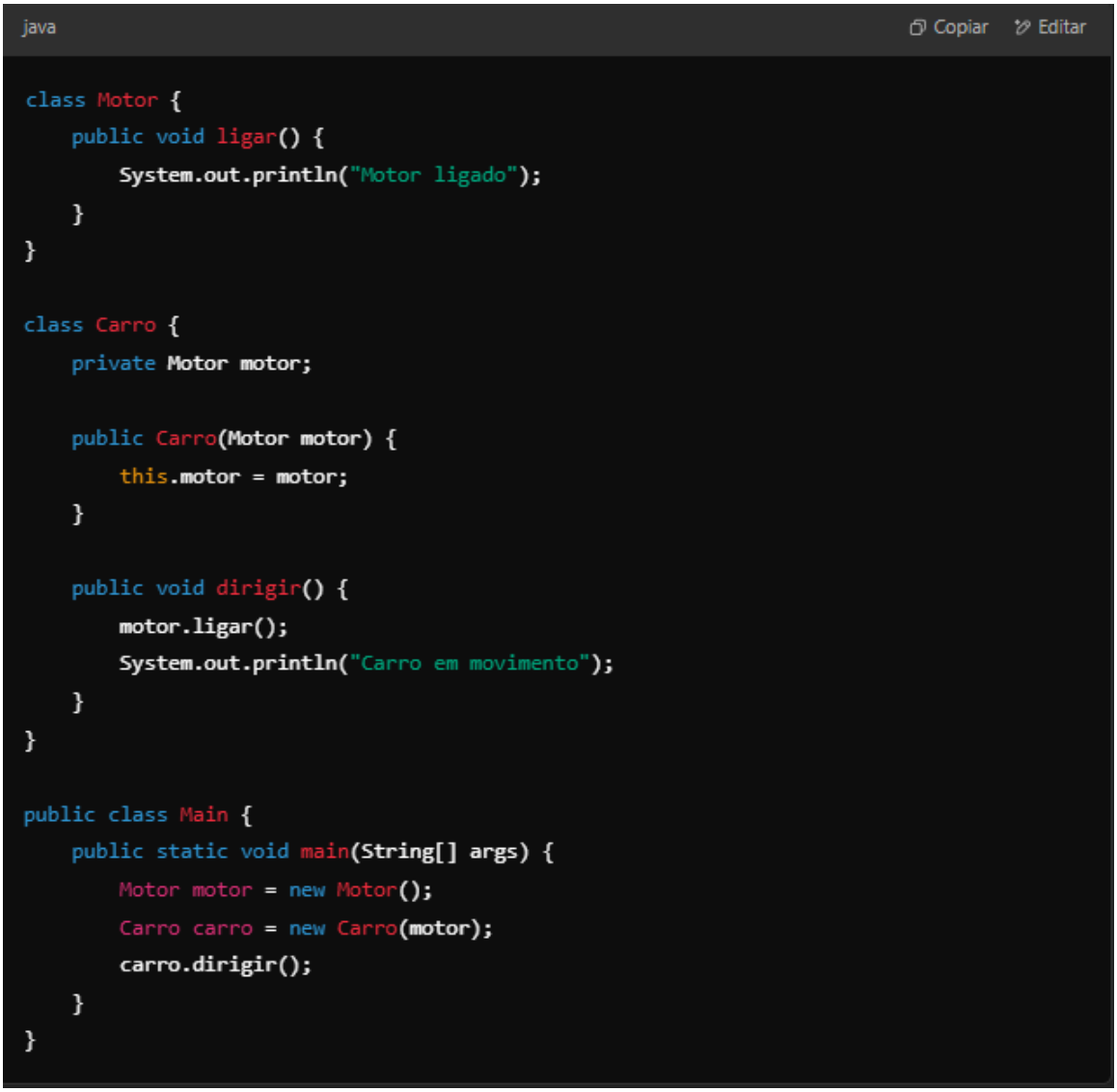
A anotação @Autowired indica ao Spring que a dependência deve ser injetada automaticamente pelo container do Spring.

Exemplo com Spring:



5. Exemplos de Injeção de Dependência

5.1. Exemplo com Injeção via Construtor



}
}

5.2. Exemplo com Injeção via Setter

java Copiar Editar

```java
class Motor {
    public void ligar() {
        System.out.println("Motor ligado");
    }
}

class Carro {
    private Motor motor;

    public void setMotor(Motor motor) {
        this.motor = motor;
    }

    public void dirigir() {
        motor.ligar();
        System.out.println("Carro em movimento");
    }
}

public class Main {
    public static void main(String[] args) {
        Motor motor = new Motor();
        Carro carro = new Carro();
        carro.setMotor(motor);
        carro.dirigir();
    }
}
```

5.3. Exemplo com Injeção via Interface

```java
interface Motor {
    void ligar();
}

class MotorEletrico implements Motor {
    public void ligar() {
        System.out.println("Motor elétrico ligado");
    }
}

class MotorCombustao implements Motor {
    public void ligar() {
        System.out.println("Motor a combustão ligado");
    }
}

class Carro {
    private Motor motor;

    public Carro(Motor motor) {
        this.motor = motor;
    }

    public void dirigir() {
        motor.ligar();
        System.out.println("Carro em movimento");
    }
}

public class Main {
    public static void main(String[] args) {
        Motor motorEletrico = new MotorEletrico();
        Carro carro = new Carro(motorEletrico);
        carro.dirigir();
    }
}
```

6. Conclusão

A Injeção de Dependência (DI) é uma técnica poderosa e essencial para o desenvolvimento de sistemas desacoplados, escaláveis e fáceis de testar. Ela é

amplamente adotada em frameworks como Spring, e sua compreensão e aplicação correta são fundamentais para desenvolvedores que desejam escrever código modular e flexível.