Injeção de dependências

Padrão de projeto para diminuir o acoplamento entre classes.

Injeção de dependências é o ato de remover a dependência fixa da classe, que também é uma classe, e ter a possibilidade de atribuir outros objetos caso necessário sem que seja necessário modificar a classe principal.

Código sem aplicar o uso do padrão injeção de dependências:
Na linha “3” podemos identificar que a variável “engine” está sendo inicializada com um objeto do tipo “Engine”, mais precisamente o objeto “EngineV2”.

Vehicle.java

```java
public abstract class Vehicle {
    private String model;
    private Engine engine = new EngineV2();

    public Vehicle(String model) {
        this.model = model;
    }

    public String getModel() {
        return this.model;
    }

    public Engine getEngine() {
        return this.engine;
    }

//    public void setEngine(Engine engine) {
//        this.engine = engine;
//    }

    public void powerOn() {
        if (this.engine == null) {
            System.out.printf("O veículo %s não possuí um motor", this.model);
        } else {
            this.engine.engine(this);
        }
    }
}
```

Arquivo Main.java

```java
public class Main {
    public static void main(String[] args) {
        Vehicle mt09  = new MT09();
        Vehicle teslaS = new TeslaModelS();

        mt09.setEngine(new EngineV1());
        mt09.powerOn();

        teslaS.setEngine(new EngineV2());
        teslaS.powerOn();
    }
}
```

Ao usarmos a classe abstrata “Vehicle“ para criar dois veículos diferentes, inicializar os veículos em nosso arquivo “Main” e finalmente “darmos a partida” em ambos, será exibida a seguinte mensagem:

```
O motor do veículo Yamaha MT-09 agora está ligado e ele possui 230.0 cavalos de potência.
O motor do veículo Tesla Model S agora está ligado e ele possui 230.0 cavalos de potência.

Process finished with exit code 0
```

A mensagem foi exibida corretamente, porém uma moto não tem o mesmo motor de um carro. Independente de quantos veículos criarmos usando a classe abstrata “Vehicle”, todos possuirão o mesmo motor.

Para corrigir este problema, podemos usar o padrão “Injeção de Dependência” na classe “Vehicle”:

```java
public abstract class Vehicle {
    private String model;
    private Engine engine;

    public Vehicle(String model) {
        this.model = model;
    }

    public String getModel() {
        return this.model;
    }

    public Engine getEngine() {
        return this.engine;
    }

    public void setEngine(Engine engine) {
        this.engine = engine;
    }

    public void powerOn() {
        if (this.engine == null) {
            System.out.printf("O veículo %s não possuí um motor", this.model);
        } else {
            this.engine.engine(this);
        }
    }
}
```

Inicializando a variável “engine” na linha “3”, mas sem atribuir um valor, estaremos atribuindo um novo motor através do método “setter” na linha “17”, dando início ao padrão de projeto injeção de dependência.

Este formato de desenvolvimento permite que o código possua baixo acoplamento entre as classes e possua um alto nível de flexibilidade, sendo possível criar diversos motores e podendo passar um motor para cada veículo sem que seja necessário modificar a classe Vehicle.

Aplicando o uso da injeção de dependência:

```java
public class Main {
    public static void main(String[] args) {
        Vehicle mt09  = new MT09();
        Vehicle teslaS = new TeslaModelS();

        mt09.setEngine(new EngineV1());
        mt09.powerOn();

        teslaS.setEngine(new EngineV2());
        teslaS.powerOn();
    }
}
```

Nas linhas “6” e “9” estamos atribuindo os motores “EngineV1” e “EngineV2” aos veículos, sendo possível obtermos o seguinte resultado:

```
O motor do veículo Yamaha MT-09 agora está ligado e ele possui 100.0 cavalos de potência.
O motor do veículo Tesla Model S agora está ligado e ele possui 230.0 cavalos de potência.

Process finished with exit code 0
```

Desta vez os “cavalos” dos motores não são iguais, representando os valores de uma forma mais próxima a realidade.

Pesquisando pela dica “**Injeção de dependência com Spring**”, passada pelo professor, foi possível localizar algumas informações sobre o assunto.

O framework Spring injeta automaticamente a dependência nos códigos.

Mas não compreendi como o Spring sabe qual dependência desejo usar em cada caso. Como o Spring poderia saber que desejo usar o motor “EngineV1” na MT09 ao invés do motor “EngineV2”?