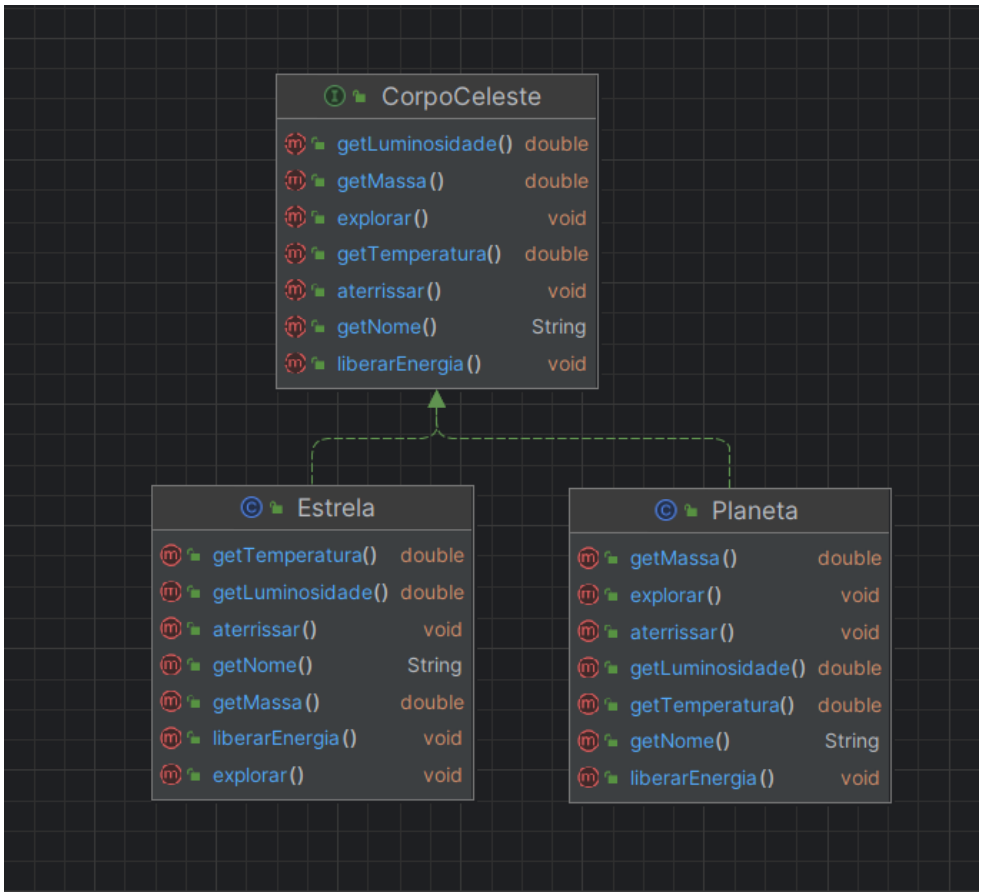


Aqui a gente tem um exemplo de violação do princípio, então a gente tem uma interface corpo celeste que vai ser usada para implementar as classes planeta e estrela, fazendo dessa forma tanto a classe estrela quanto a classe planeta são obrigadas a implementar métodos que não fazem sentido pra sua classe o que vai gerar exceções

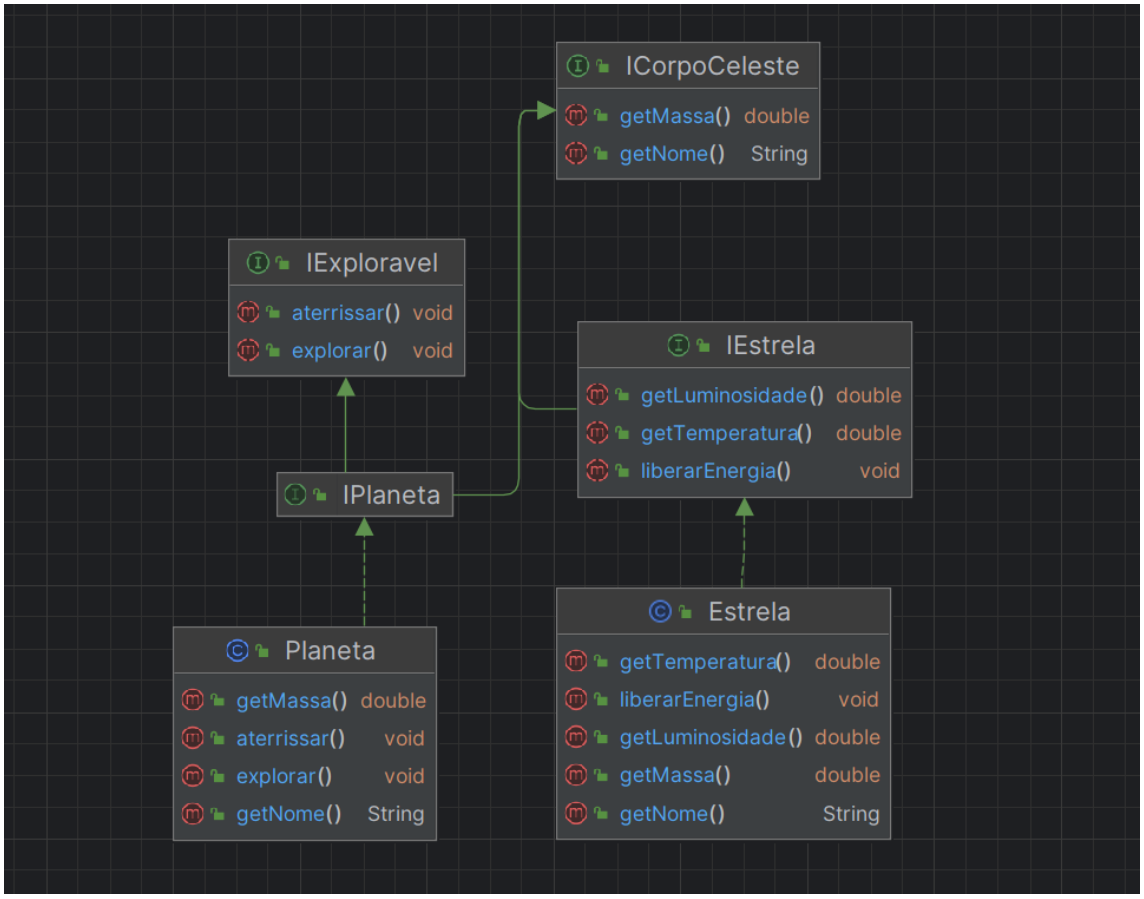
```
    no usages
37  public void aterrissar() {
38      throw new UnsupportedOperationException("Método 'aterrissar' não implementado.");
39  }
40
    no usages
41  @Override
42  public void explorar() {
43      throw new UnsupportedOperationException("Método 'explorar' não implementado.");
44  }
45 }
```

```
    no usages
31  @Override
32  public double getLuminosidade() {
33      throw new UnsupportedOperationException("Método não implementado.");
34  }
    no usages
35  public void liberarEnergia(){
36      throw new UnsupportedOperationException("Método não implementado.");
37  }
38 }
39
```

Aqui por exemplo na classe planeta, a gente alguns métodos em comum como o get massa get nome e get temperatura, e os métodos getluminosidade e liberar energia não tem implementações o que gera uma exceção com a mensagem método não implementado.

O mesmo aqui pra classe estrela que não implementa os métodos aterrissar explorar

Então seguindo o princípio da segregação de interfaces a implementação correta seria essa, em que separamos as interfaces para que cada classe tenha que seguir apenas os métodos que vai utilizar:



Aqui a gente tem uma interface planeta que estende as interfaces corpo celeste e explorável e uma interface estrela que estende a interface corpo celeste

Main.java | ICorpoCeleste.java | IExploravel.java

```
2 usages  4 implementations
public interface ICorpoCeleste {
    no usages  2 implementations
    String getNome();
    no usages  2 implementations
    double getMassa();
}
```

Aqui na interface corpo celeste a gente tem métodos getNome e getMassa que servem para todos os corpos celestes

Main.java | IExploravel.java | ICorpoCeleste.java

```
1 usage  2 implementations
public interface IExploravel {
    no usages  1 implementation
    void aterrissar();
    no usages  1 implementation
    void explorar();
}
```

Na interface explorável a gente tem os métodos aterrissar e explorar

Main.java | IPlaneta.java | IExploravel.java | ICorpoCeleste.java

```
1 usage  1 implementation
public interface IPlaneta extends ICorpoCeleste, IExploravel{
}
```

E a interface planeta estende as interfaces corpo celeste e explorável

Main.java | IEstrela.java | IPlaneta.java | IExploravel.java

```
1 usage  1 implementation
public interface IEstrela extends ICorpoCeleste{
    no usages  1 implementation
    double getTemperatura();
    no usages  1 implementation
    double getLuminosidade();
    no usages  1 implementation
    void liberarEnergia();
}
```

A interface estrela estende a interface corpo celeste e tem mais alguns métodos próprios

Main.java | Planeta.java × | IEstrela.java | IPlaneta.java | IExploravel.java | ICorpoCeleste.java

```java
no usages
public class Planeta implements IPlaneta{
    4 usages
    private String nome;
    2 usages
    private double massa;

    no usages
    public Planeta(String nome, double massa) {
        this.nome = nome;
        this.massa = massa;
    }

    no usages
    @Override
    public String getNome() {
        return nome;
    }

    no usages
    @Override
    public double getMassa() { return massa; }

    no usages
    public void aterrissar() { System.out.println("Aterrissando no planeta " + nome); }

    no usages
    @Override
    public void explorar() { System.out.println("Explorando o planeta " + nome); }
}
```

A classe planeta implementa a interface planeta e tem a implementação de todos os métodos que é obrigada a implementar pela interface

Main.java | Estrela.java × | Planeta.java | IEstrela.java | IPlaneta.java | IExploravel.java

```java
no usages
public class Estrela implements IEstrela {
    2 usages
    private String nome;
    2 usages
    private double massa;
    2 usages
    private double temperatura;
    2 usages
    private double luminosidade;

    no usages
    public Estrela(String nome, double massa, double temperatura, double luminosidade) {
        this.nome = nome;
        this.massa = massa;
        this.temperatura = temperatura;
        this.luminosidade = luminosidade;
    }
```

```
    no usages
    @Override
    public String getNome() { return nome; }

    no usages
    @Override
    public double getMassa() { return massa; }

    no usages
    @Override
    public double getTemperatura() { return temperatura; }

    no usages
    @Override
    public double getLuminosidade() { return luminosidade; }

    no usages
    public void liberarEnergia() { System.out.println("Liberando energia estelar."); }
}
```

O mesmo pra classe estrela.

Main.java | ICorpoCeleste.java | IExploravel.java | IPlaneta.java | IEstrela.java | Planeta.java

```
public class Main {
    public static void main(String[] args) {
        Planeta terra = new Planeta( nome: "terra", massa: 59722*10^24);
        Estrela sol = new Estrela( nome: "sol", massa: 1989*10^30, temperatura: 5772, luminosidade: 1);

        terra.aterrissar();
        terra.explorar();

        sol.liberarEnergia();
    }
}
```

Aqui no main eu criei um objeto terra do tipo planeta e um objeto sol do tipo estrela e cada um esta chamando seus métodos a terra aterrissar e explorar e o sol liberar energia

Run | Main

```
C:\Users\nitro\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.2\lib\idea_rt.jar=57558:C:\Program Files\JetBrains\IntelliJ IDEA
Aterrissando no planeta terra
Explorando o planeta terra
Liberando energia estelar.

Process finished with exit code 0
```

E esse é o resultado da execução