05

Mover Campo

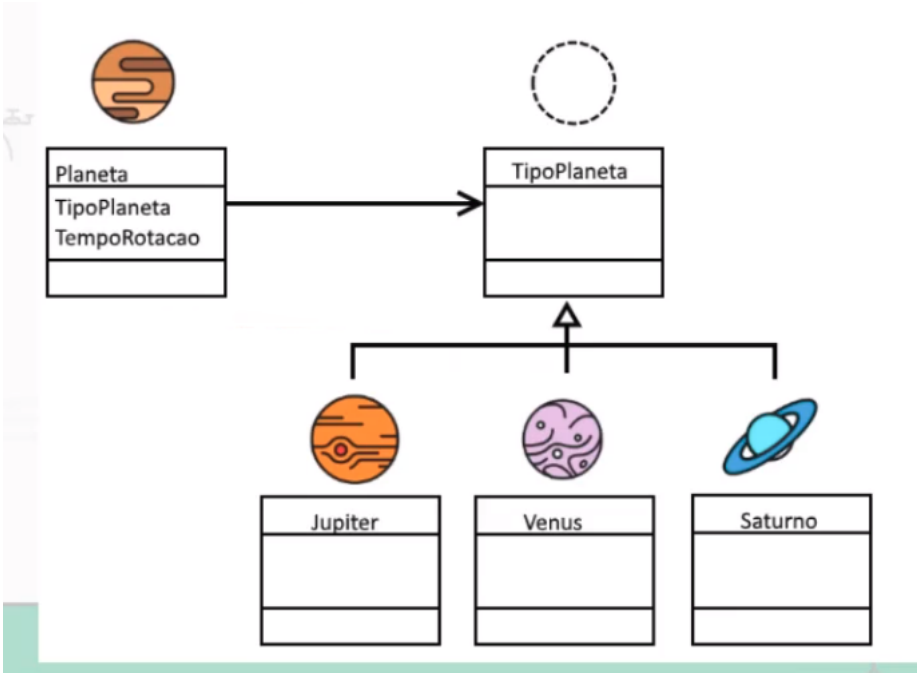
Transcrição

Agora vamos para a técnica **Mover Campo**. Temos uma situação em que uma aplicação representa o *sistema solar*.

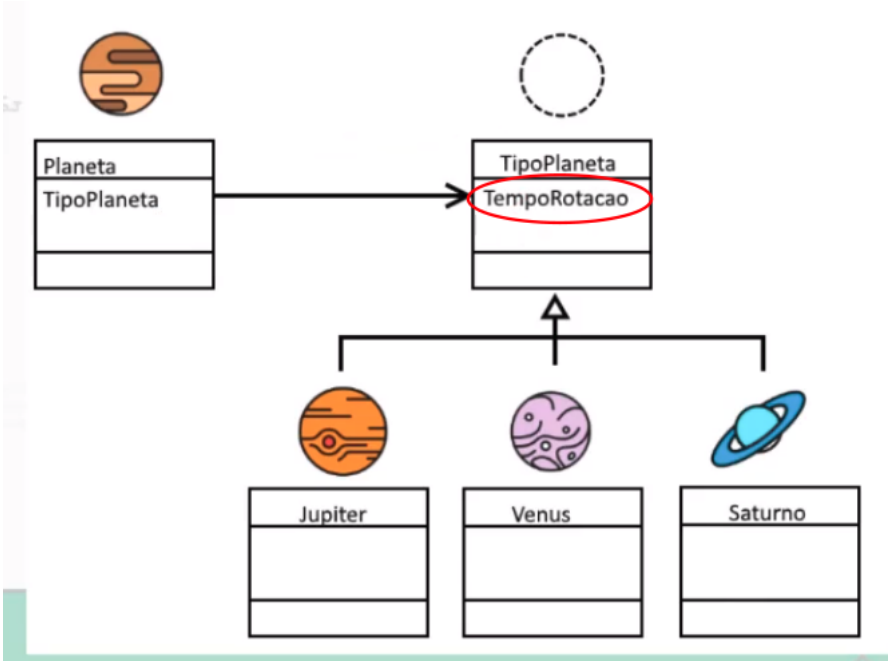
![Imagem de um diagrama de classes, mostrando que a classe `Planeta` possui uma propriedade que está apontando para a classe `TipoPlaneta`](https://s3.ama (https://s3.ama) zonaws.com/caelum-online-public/720-C%23-refatoracao-parte-1/Aula-5/05.02_001_move-field-planet.png)

Temos uma classe `Planeta`, onde a propriedade `TipoPlaneta` está apontando para a classe chamada `TipoPlaneta`. Existe também o `TempoRotacao`, referente ao tempo que o planeta leva para fazer a rotação. O problema é que cada planeta tem uma rotação diferente, e estamos definindo o tempo de rotação dentro da classe `Planeta`.

Podemos imaginar que a classe `TipoPlaneta` é uma **classe base** para outras classes que implementam `Planeta`, assim como mostra a imagem a seguir:



Júpter, Vênus e Saturno possuem rotações diferentes, e com isso vimos que a propriedade `TempoRotacao` não está em um lugar apropriado para ele. Nossa tarefa é aplicar o conceito de refatoração *move field* na propriedade `TempoRotacao`.



Vamos exemplificar em código. Dentro do projeto `refatoracao`, temos a pasta "Aula05 > R11.MoveField > depois" que contém o arquivo `ContaDeLuz.cs`.

Esse arquivo contém quatro classes:

```
class ContaDeLuz{...}

abstract class TipoDeConta{...}

class ContaResidencial{...}

class ContaComercial{...}
```

A `ContaDeLuz` representa uma conta qualquer e possui campos chamados de `TipoDeConta`, que apontam para a classe abstrata `TipoDeConta`. As classes `ContaResidencial` e `ContaComercial` herdam de `TipoDeConta`.

Além dos campos `TipoDeConta` na classe `ContaDeLuz`, também temos o `jurosAoMes`.

```
class ContaDeLuz
{
    private TipoDeConta tipoDeConta;
    public TipoDeConta TipoDeConta {...}

    private decimal jurosAoMes;
    public decimal JurosAoMes
    {
        get { return jurosAoMes; }
        set { jurosAoMes = value; }
    }
}
```

Claro que cada conta tem a sua taxa de juros, mas nós percebemos no *construtor* da `ContaDeLuz` que o `jurosAoMes` é definido de acordo com o `tipoDeConta`.

```
    public ContaDeLuz(TipoDeConta tipoDeConta)
    {
        this.tipoDeConta = tipoDeConta;

        if (tipoDeConta is ContaResidencial)
        {
            this.jurosAoMes = .030M;
        }
        else
        {
            this.jurosAoMes = .060M;
        }
    }
```

Se a conta é `ContaResidencial`, o `jurosAoMes` será de `.030M`. Caso seja diferente de uma `ContaResidencial`, então o `jurosAoMes` será de `.060M`. O `jurosAoMes` é um campo que também está deslocado, um local que não é apropriado para ele. Vamos copiá-lo para a classe `TipoDeConta`, pois dependendo do tipo de conta, teremos um juros diferente.

```
abstract class TipoDeConta
{
    private decimal jurosAoMes;
    public decimal JurosAoMes
    {
        get { return jurosAoMes; }
        set { jurosAoMes = value; }
    }
}
```

Depois, vamos transformar o `if` que determina o juros no construtor, em um código mais simples definindo o juros ao mês no construtor de cada uma das classes `TipoDeConta`. Na classe `ContaResidencial`, faremos o seguinte:

```
class ContaResidencial : TipoDeConta
{
    public ContaResidencial()
    {
        this.jurosAoMes = .030M;
    }
}
```

Faremos o mesmo com a classe `ContaComercial`.

```
class ContaComercial : TipoDeConta
{
    public ContaComercial()
    {
        this.jurosAoMes = .060M;
    }
}
```

A variável `jurosAoMes` não compilou, pois estamos sem acesso à classe base `TipoDeConta`. Para resolver esse problema, definiremos que a variável não será mais `private`, e sim `protected`. Agora o código vai parar de dar erro. Vamos

também remover o `setter` da propriedade `JurosAoMes`, pois não queremos que esse juro seja manipulado de fora das classes `TipoDeConta`.

```
abstract class TipoDeConta
{
    private decimal jurosAoMes;
    public decimal JurosAoMes
    {
        get { return jurosAoMes; }
    }
}
```

Removeremos a condição inteira do construtor da classe `ContaDeLuz`, pois não precisamos mais.

```
    public ContaDeLuz(TipoDeConta tipoDeConta)
    {
        this.tipoDeConta = tipoDeConta;
    }
```

E por último, removeremos o campo `jurosAoMes`, pois já o movemos para a classe base `TipoDeConta`. Quando fazemos isso, a propriedade `JurosAoMes` do método `CalcularValorDosJuros()` não é mais encontrada, pois foi movida para outra classe. Acessaremos essa propriedade por meio do `tipoDeConta`:

```
public decimal CalcularValorDosJuros(decimal principal, int diasAtraso)
{
    decimal jurosAoDia = tipoDeConta.JurosAoMes / 30.0M;
    decimal juros = jurosAoDia * diasAtraso;
    return juros * principal;
}
```

Vamos compilar por meio do atalho "Ctrl + Shift + B", e então conseguimos aplicar essa refatoração **Mover Campo** com sucesso.