

# Padrões de Projeto

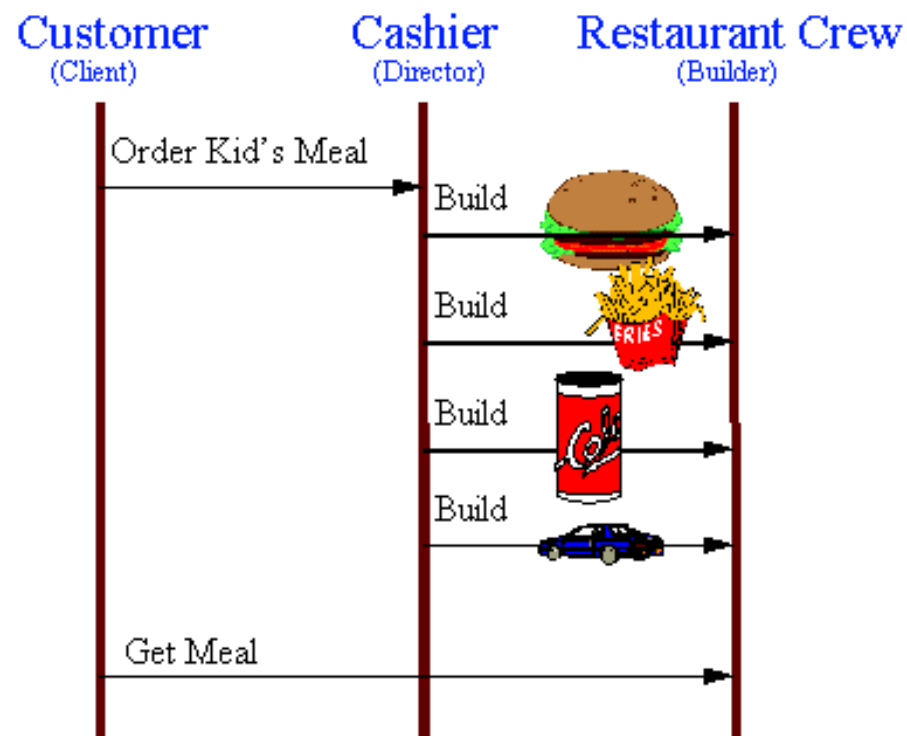
Prof. Adilson Vahldick

Departamento de Engenharia de Software

Udesc Ibirama

# Objetivos da aula

- Conhecer e aplicar o padrão
  - Builder



# Problema (1)

- Temos uma linha de montagem de veículos. Cada veículo tem um conjunto (tipo e quantidade) de peças distintas. Constrói de acordo com a encomenda:

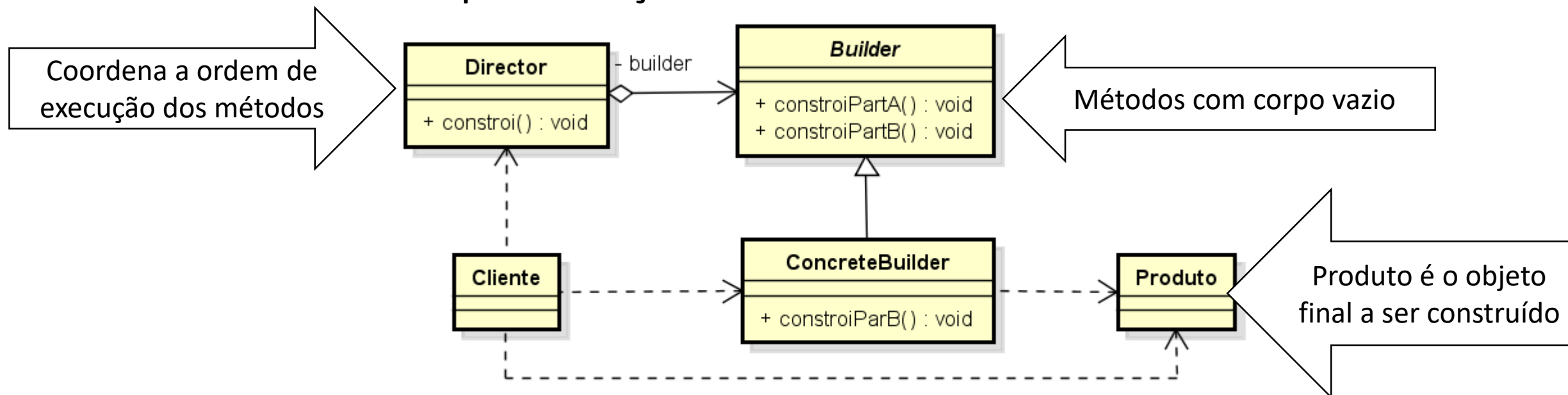
[illegible]

## Problema (2)

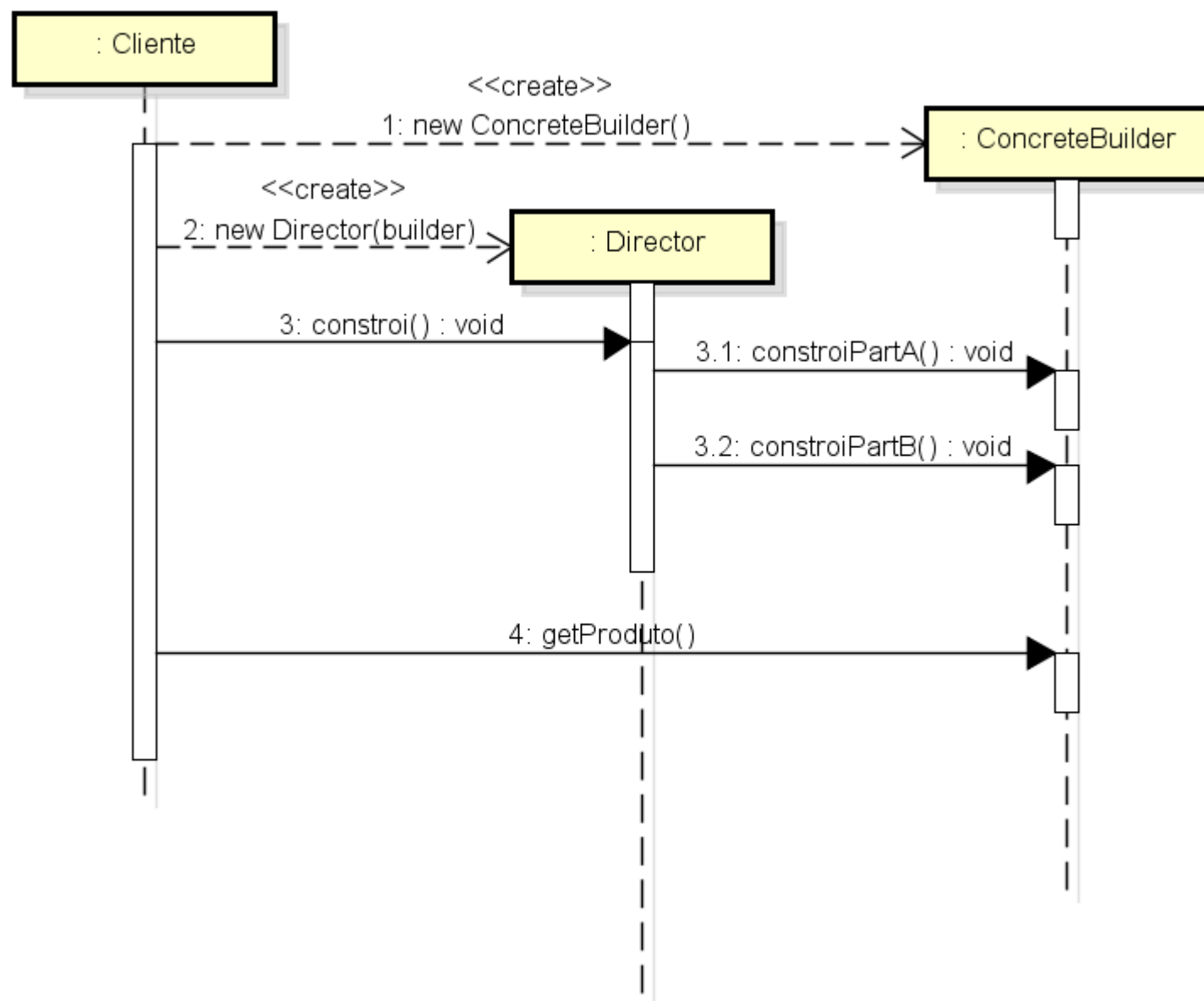
- Qual o problema com essa abordagem ?
  - Novos tipos de carros exigem mudanças no algoritmo de construção
  - Tipos de carros podem ter características (p.e. número de pneus) que modificam a forma de construí-lo

# Solução (1)

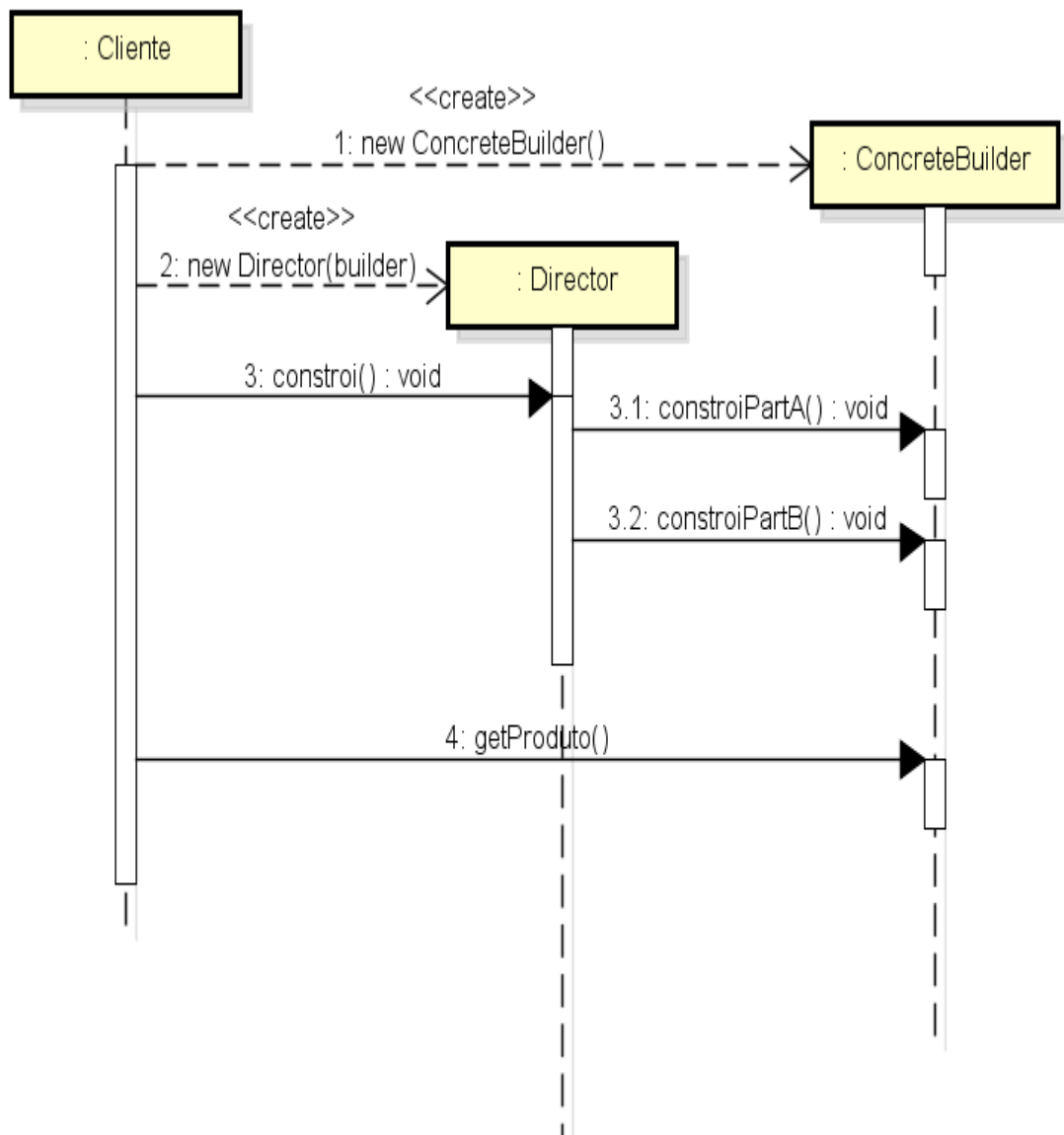
- **Builder:** separar a construção de um objeto complexo da sua representação de modo que o mesmo processo de construção possa criar diferentes representações



# Solução (2)



# Solução (3)



```

public abstract class Builder {

    protected Produto produto=new Produto();

    public Produto getProduto() {
        return produto;
    }

    public void constroiPartA() {
    }

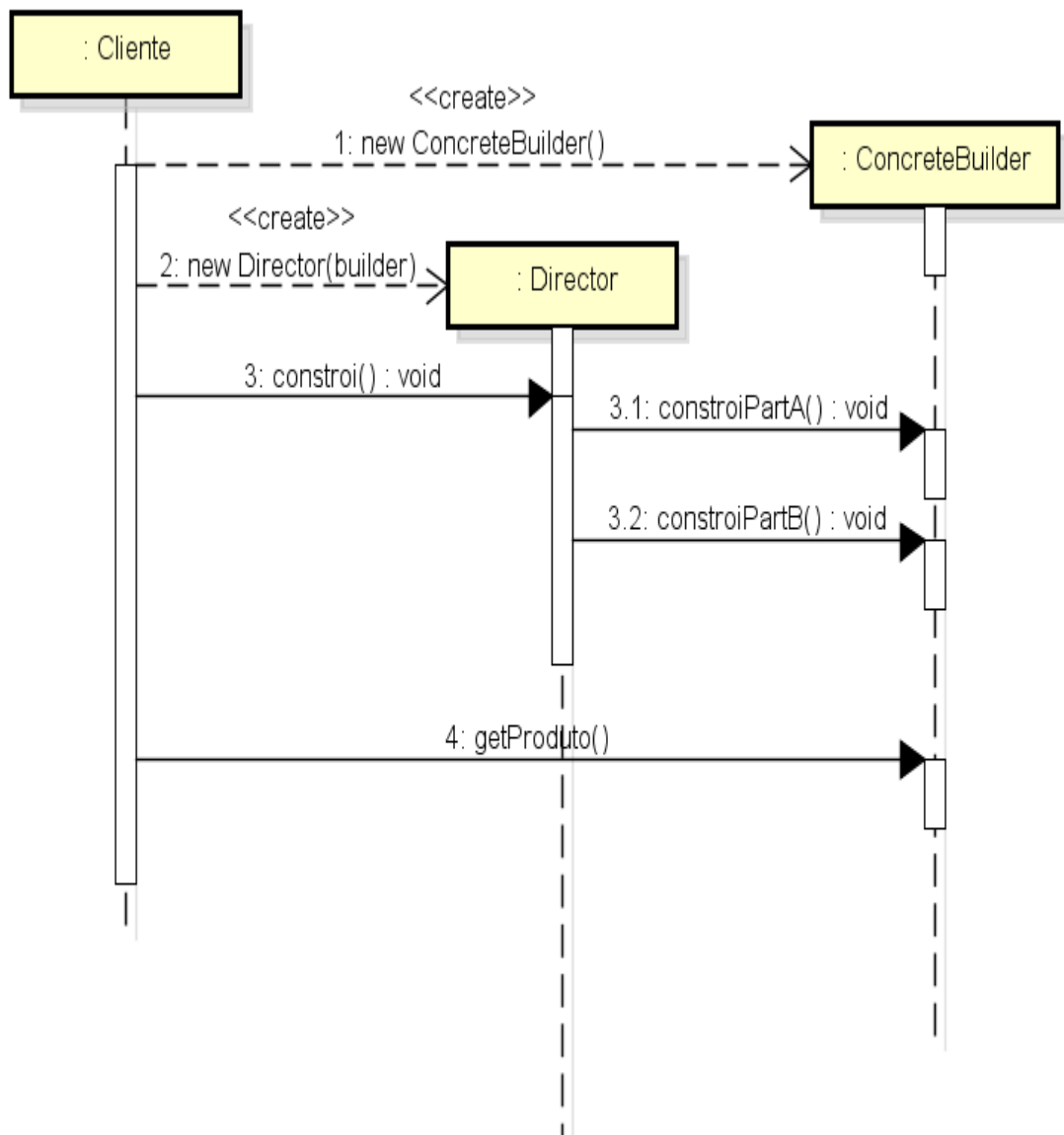
    public void constroiPartB() {
    }

}
    
```

# Solução (4)

```
public class ConcreteBuilder extends
    Builder {
```

```
    public void constroiPartB() {
        produto.setPartB("Muda algo X");
    }
}
```





# Solução (4)

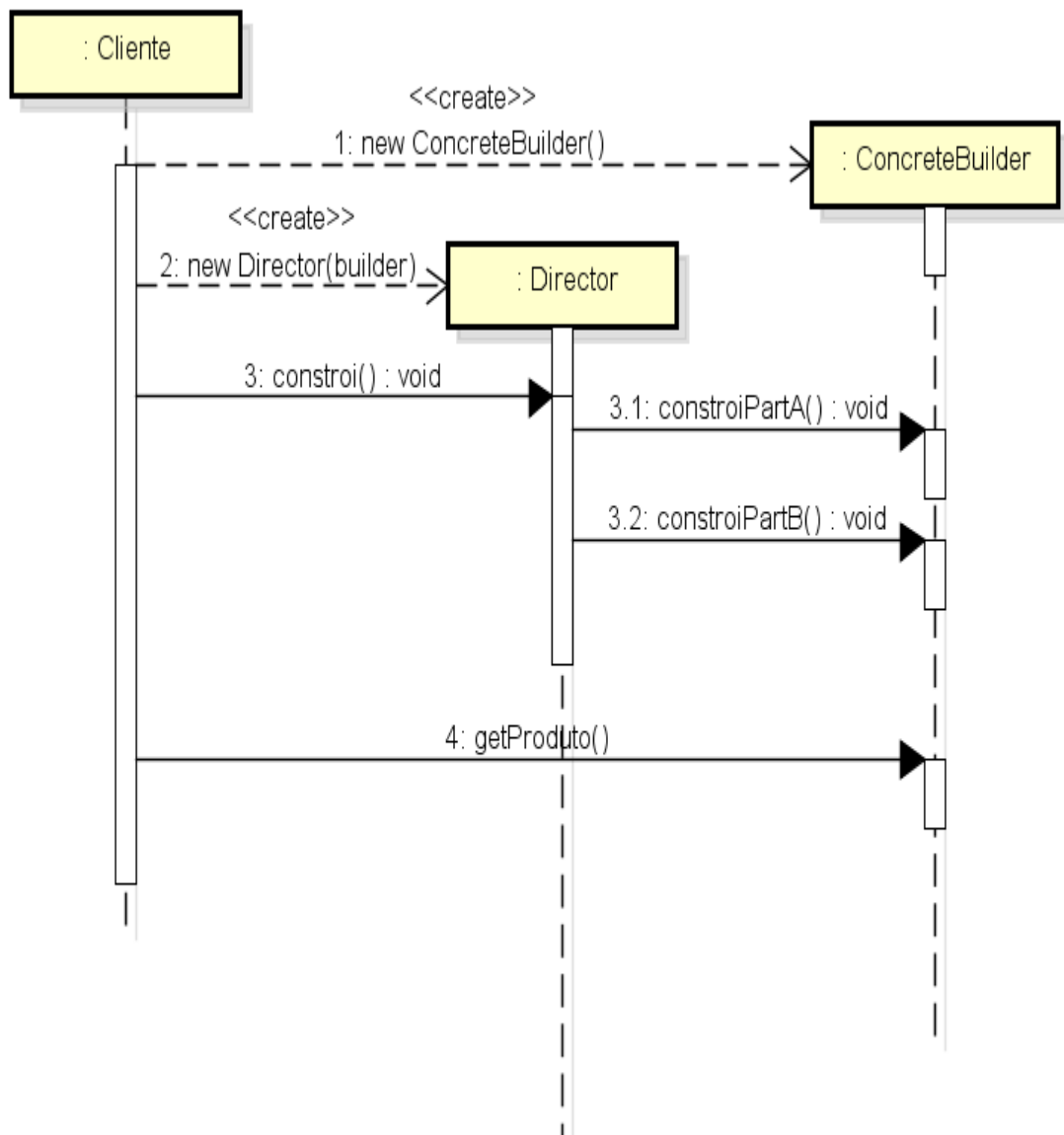
```
public class Director {
```

```
    private Builder builder;
```

```
    public Director(Builder builder) {
        this.builder = builder;
    }
```

```
    public void constroi() {
        builder.constroiPartA();
        builder.constroiPartB();
    }
```

```
}
```

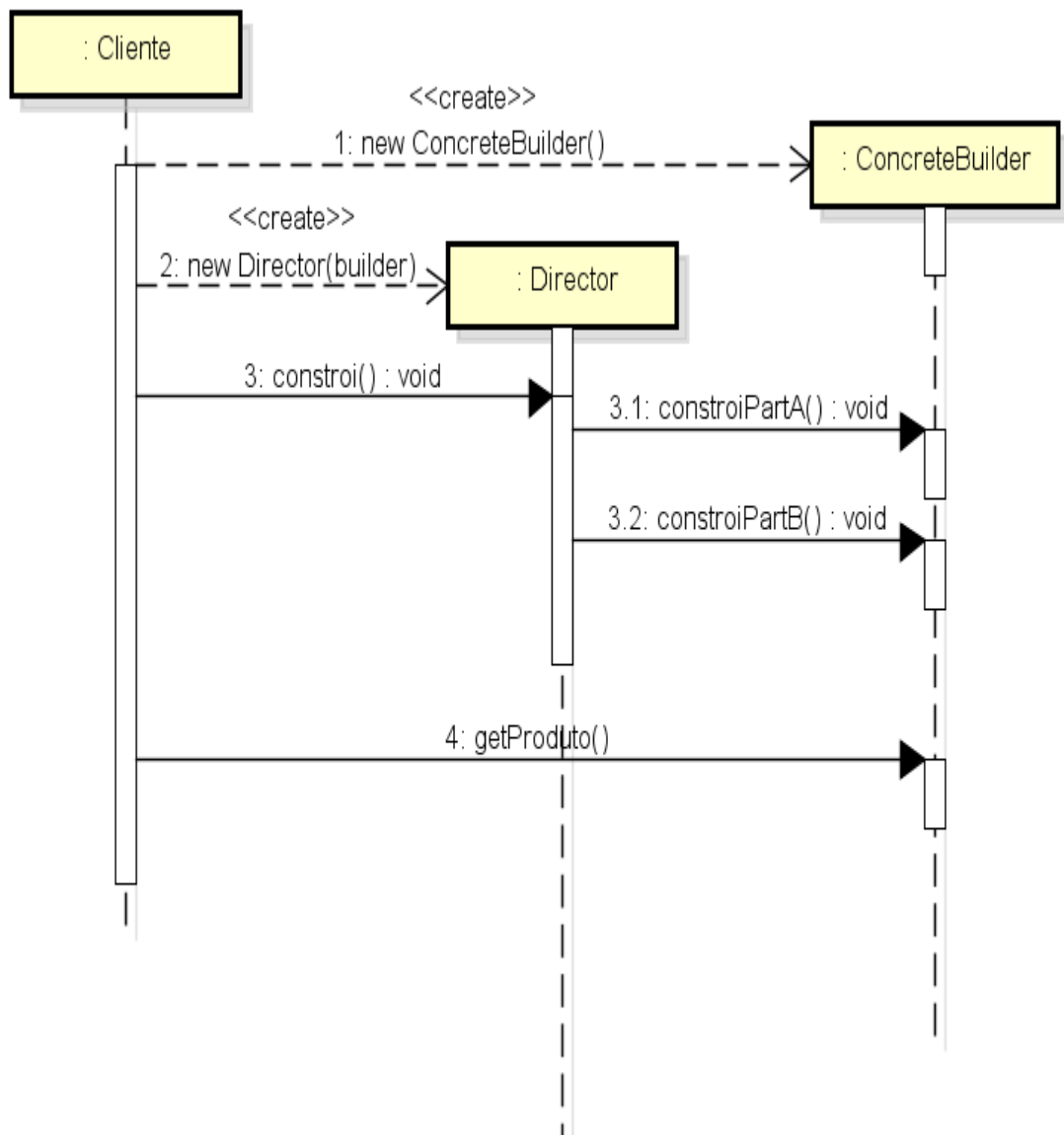


# Solução (5)

```
public class Cliente {
```

```
    método main {
        ConcreteBuilder cb =
            new ConcreteBuilder();
        Director d = new Director(cb);
        d.constroi();
        Produto p = cb.getProduto();
    }
}
```

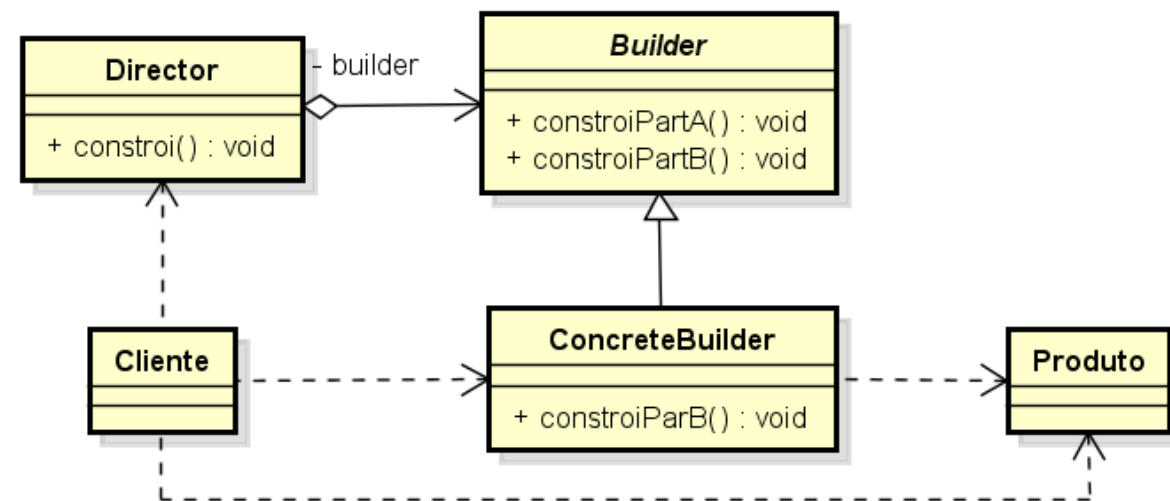
```
}
```



# Problema (3) – builder1

```

Carro carro = new Carro();
if (tipo == "esportivo") {
    carro.setMotor(new MotorV8());
    carro.setPneus(new Pneu[]{new Pneu19(), new Pneu19(),
                                new Pneu19(), new Pneu19()});
} else {
    carro.setMotor(new Motor1_0());
    if (tipo == "anfíbio")
        carro.setPneus(new Pneu[]{new Pneu14(), new Pneu14(), new Pneu14(),
                                    new Pneu14(), new Pneu14(), new Pneu14()});
    else
        carro.setPneus(new Pneu[]{new Pneu14(),
                                    new Pneu14(), ew Pneu14(), new Pneu14()});
}
    
```



# Problema (4) – builder1

- O que é o carro ?

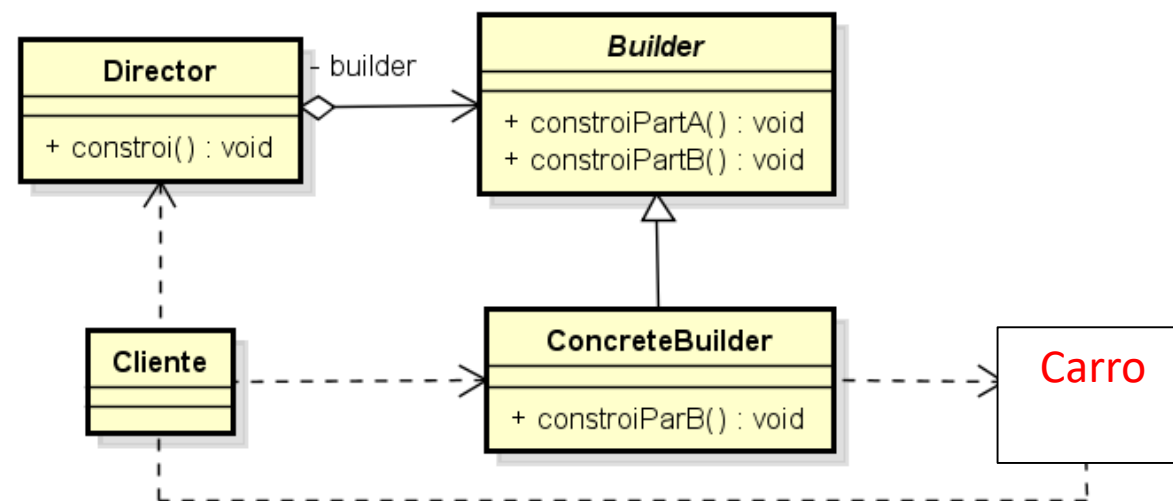
```
public class Carro {
```

```
    private Motor motor;
    private Pneu[] pneus;
```

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

```
    public void setPneus(Pneu... pneus) {
        this.pneus = pneus;
    }
```

```
    public String toString() {
        return motor + " " + Arrays.toString(pneus);
    }
```



powered by Astah

# Problema (5) – builder1

- O que o builder precisa construir ?

```
public abstract class Robo {

    protected Carro carro;

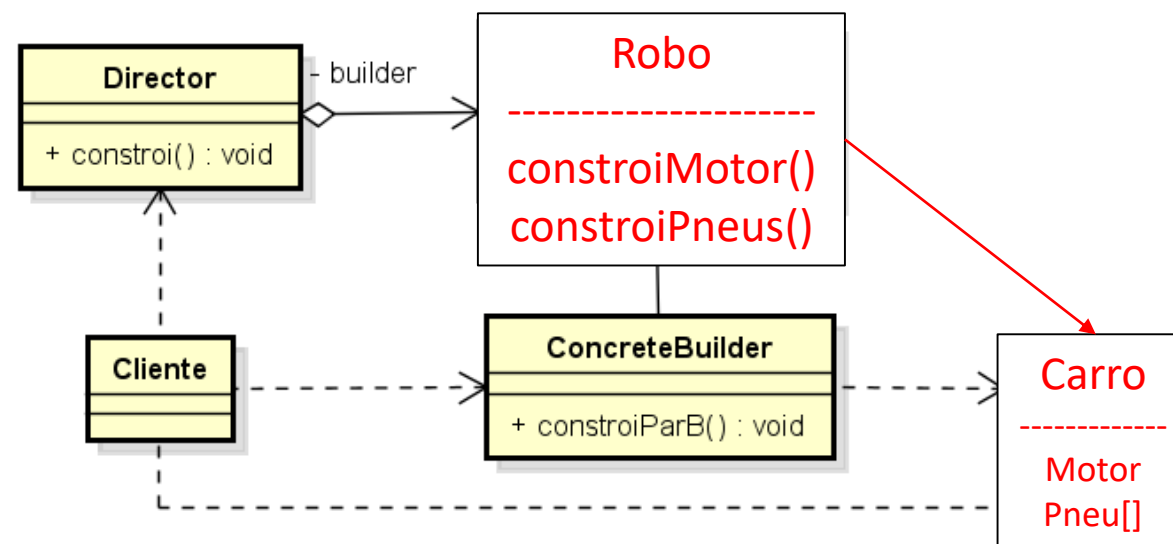
    public Robo() {
        this.carro = new Carro();
    }

    public Carro getCarro() { return carro; }

    public void constroiMotor() {}

    public void constroiPneus() {}

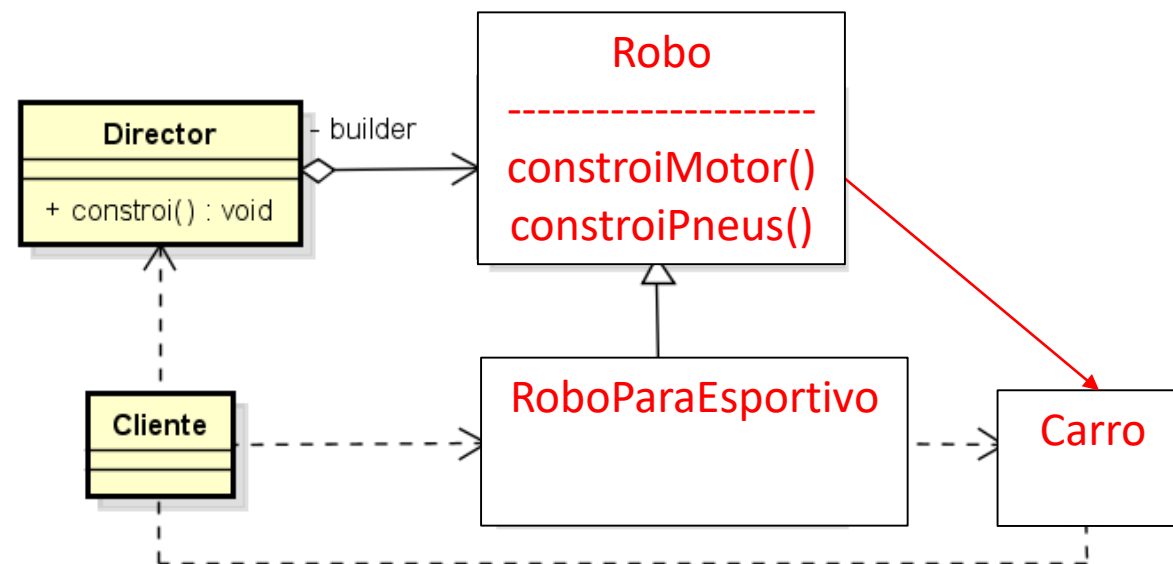
}
```



powered by Astah

# Problema (6) – builder1

- Quem são os builders concretos ?
- Esportivo
- Popular
- PopularAnfibio



powered by Astah

# Problema (7) – builder1

```
public class RoboParaEsportivo extends Robo {  
  
    public void constroiMotor() {  
        carro.setMotor(new MotorV8());  
    }  
  
    public void constroiPneus() {  
        carro.setPneus(new Pneu19(), new Pneu19(),  
                       new Pneu19(), new Pneu19());  
    }  
}
```

```
public class RoboParaPopular extends Robo {  
  
    public void constroiMotor() {  
        carro.setMotor(new Motor1_0());  
    }  
  
    public void constroiPneus() {  
        carro.setPneus(new Pneu14(),  
                       new Pneu14(), new Pneu14(),  
                       new Pneu14());  
    }  
}
```

```
public class RoboParaPopularAnfibio extends RoboParaPopular {  
  
    public void constroiPneus() {  
        carro.setPneus(new Pneu14(), new Pneu14(), new Pneu14(),  
                       new Pneu14(), new Pneu14(), new Pneu14());  
    }  
}
```

# Problema (8) – builder1

- E agora precisamos de um diretor

```

public class Montadora {

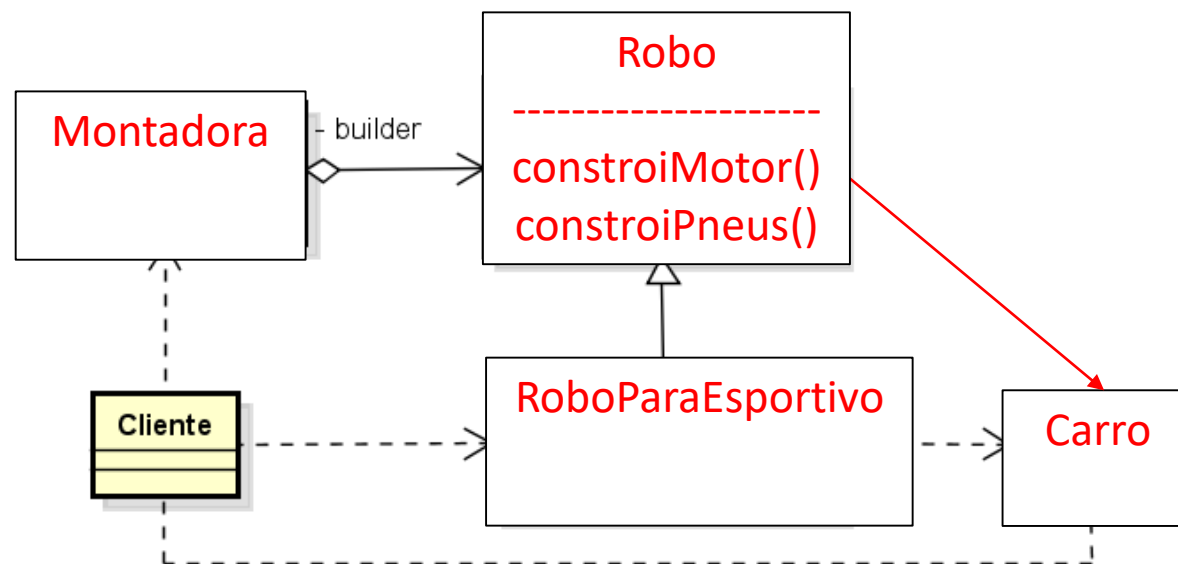
    private Robo robo;

    public Montadora(Robo robo) {
        this.robo = robo;
    }

    public void construir() {
        robo.constroiMotor();
        robo.constroiPneus();
    }

    public Carro getCarro() { return robo.getCarro(); }
}

```

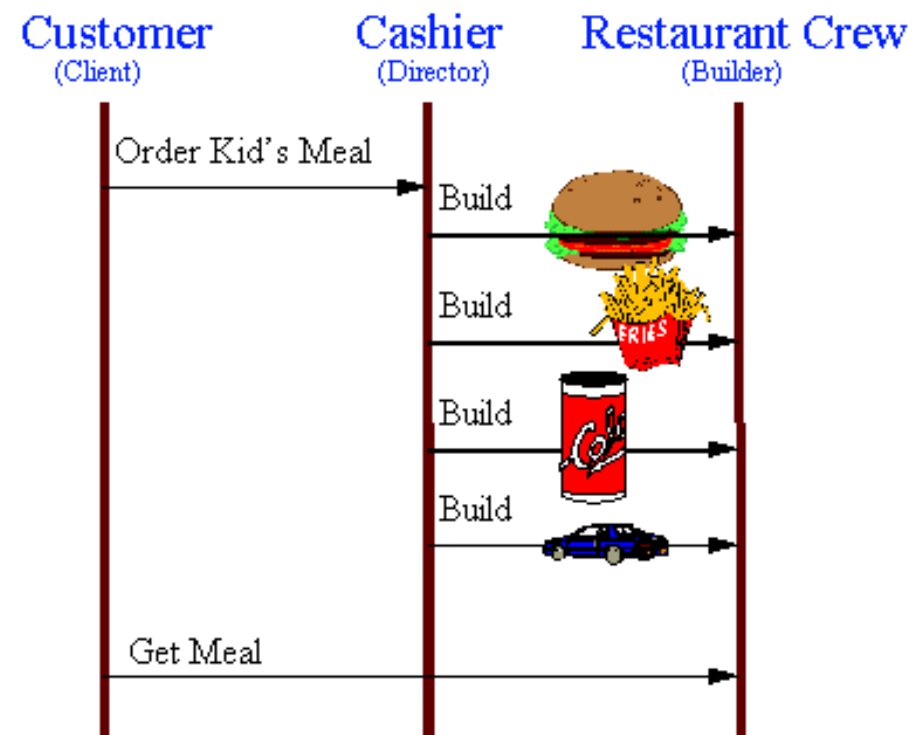
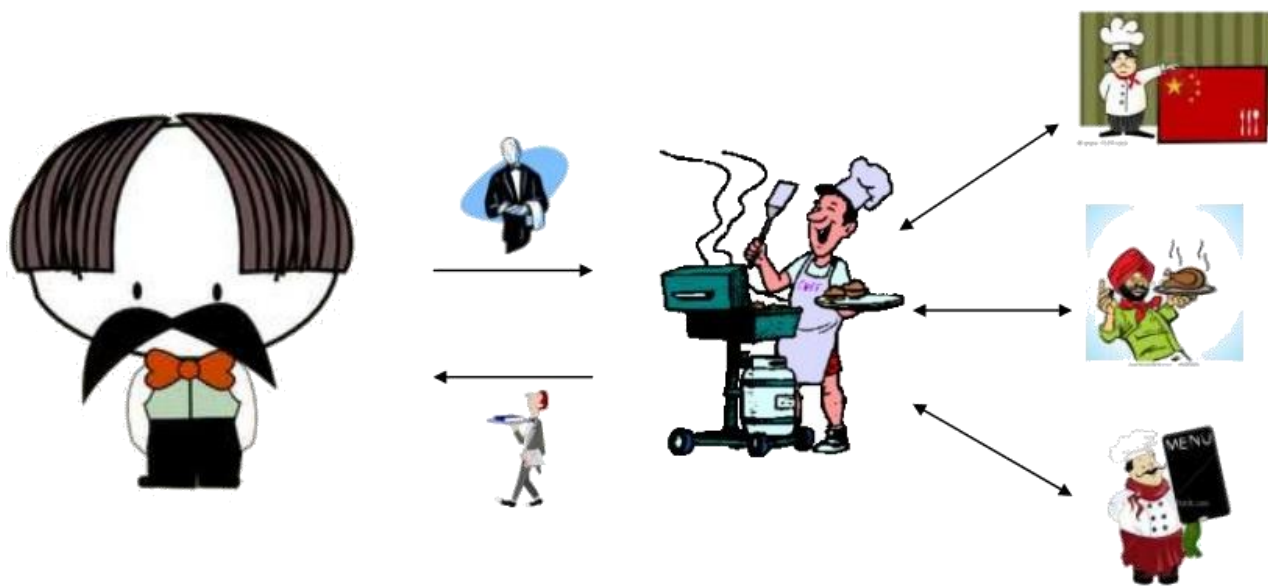


powered by Astah




# Abstract Factory X Builder

## Qual a diferença entre esses padrões ?



# Exercício 1 (1) – builder2

 <b>Bradesco</b>		<b>237-2</b>		<b>23791.11103 60000.000103 01000.222206 1 48622000000000</b>	
Local de pagamento <b>PAGÁVEL PREFERENCIALMENTE NAS AGÊNCIAS DO BRADESCO</b>				Vencimento <b>29/01/2011</b>	
Cedente <b>NF-e Associacao NF-e</b>				Agência / Código cedente <b>1111-8/0002222-5</b>	
Data do documento <b>25/01/2011</b>	Nº documento <b>NF 1 1/1</b>	Espécie doc.	Aceite <b>N</b>	<b>25/01/2011</b>	Carteira / Nosso número <b>1111-8/0002222-5</b>
Uso do banco	Carteira <b>06</b>	Espécie <b>R\$</b>	Quantidade	(x) Valor	(=) Valor documento <b>R\$ 20,000,000.00</b>
Instruções (Texto de responsabilidade do cedente) <b>Não receber após o vencimento.</b> <b>Boleto 1 de 1 referente a NF 1 de 06/05/2008 com chave</b> <b>3508-0599-9990-9091-0270-5500-1000-0000-0151-8005-1273</b>				(-) Desconto / Abatimentos	
				(-) Outras deduções	
				(+/-) Mora / Multa	
				(+/-) Outros acréscimos	
				(-) Valor cobrado	
Sacado <b>DISTRIBUIDORA DE AGUAS MINERAIS CNPJ: 00.000.000/0001-91</b> <b>AV DAS FONTES 1777 10 ANDAR</b> <b>PARQUE FONTES - Sao Paulo/SP - CEP: 13950-000</b>				Cód. baixa	
Sacador / Avalista				Autenticação mecânica - Ficha de Compensação	

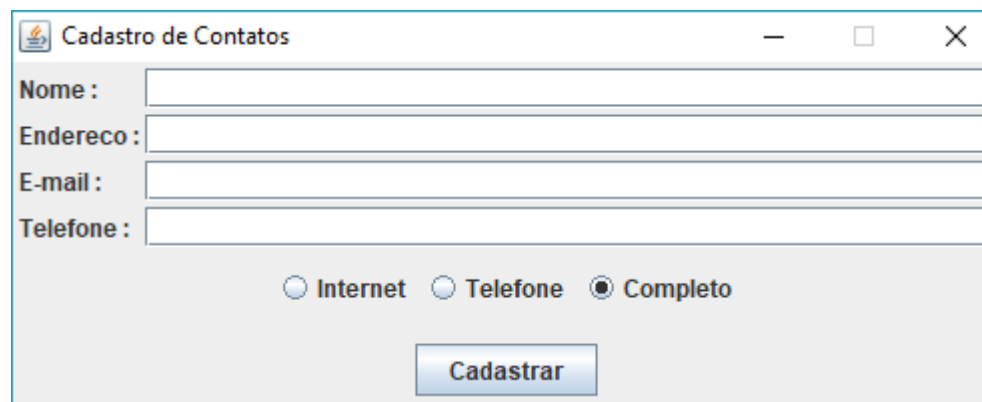


# Exercício 1 (2) – builder2

- Cada banco tem uma forma diferente de imprimir o boleto e calcular o número digitável. Todos os boletos tem os mesmos dados.
- Considerando apenas os campos indicados com setas, aplique o padrão Builder para desenvolver boletos para os bancos Caixa e Bradesco.
  - Considere todas as informações como String (não estamos interessados na lógica do boleto !!!)
  - Caixa: o nome do cedente é armazenado em maiúsculas (toUpperCase())
  - Bradesco: a carteira sempre será 06
- Faltam os ConcreteBuilders
- Atualizar a classe Cliente

## Exercício 2 – builder3

- Contatos são criados de acordo com um dos três tipos:
  - Internet: Nome e E-mail
  - Telefone: Nome e Telefone
  - Completo: Todos os campos



Cadastro de Contatos

Nome :

Endereco :

E-mail :

Telefone :

☐ Internet ☐ Telefone ☒ Completo

Cadastrar

- Somente o cliente foi criado, porém existem instruções para guiá-lo no evento do botão Cadastrar

## Exercício 3 – builder4

- Na cadeia de fast-food existe um padrão para montagem de lanches de crianças. O sanduíche (hambúrguer ou cheeseburger), batata (pequena, média e grande) e o brinquedo (carrinho ou bonequinha) são colocados dentro da caixa e a bebida (coca ou suco) é entregue fora da caixa. Desenvolva uma aplicação em console (Scanner e System.out.println), que solicite as opções, e monte o lanche da criança.
- Cada item do lanche deve ser uma classe distinta
- Imprimir a bandeja no final