Modelagem Orientada a Objetos com UML

Cursos para a CTI - IME/USP

Dairton Bassi, Hugo Corbucci e Mariana Bravo
Departamento de Ciência da Computação

www.agilcoop.org.br

AgilCoop

Programa do Curso

- Primeiro dia:
  - Apresentação
  - Conceitos básicos
  - Tipos de relacionamento
  - Modelagem com cartões CRC
  - Diagrama de comunicação
  - Mais conceitos básicos

Programa do Curso

- Segundo dia:
  - Diagrama de casos de uso, de banco de dados, de seqüência, de estados, de pacotes e de implantação
  - Conceitos de modelagem
  - Arquitetura de sistemas

Programa do Curso

- Terceiro dia:
  - Revisão do curso
  - Exercícios

UML (Unified Modeling Language)

UML é uma linguagem usada para representar os elementos de sistemas de software.

Através de UML é possível representar e documentar os elementos de um sistema de forma padronizada, facilitando a comunicação.

Conceitos Básicos

Classe, Tipo:

Uma especificação que determina tipos de dados e comportamentos elementos do sistema.

Exemplo:

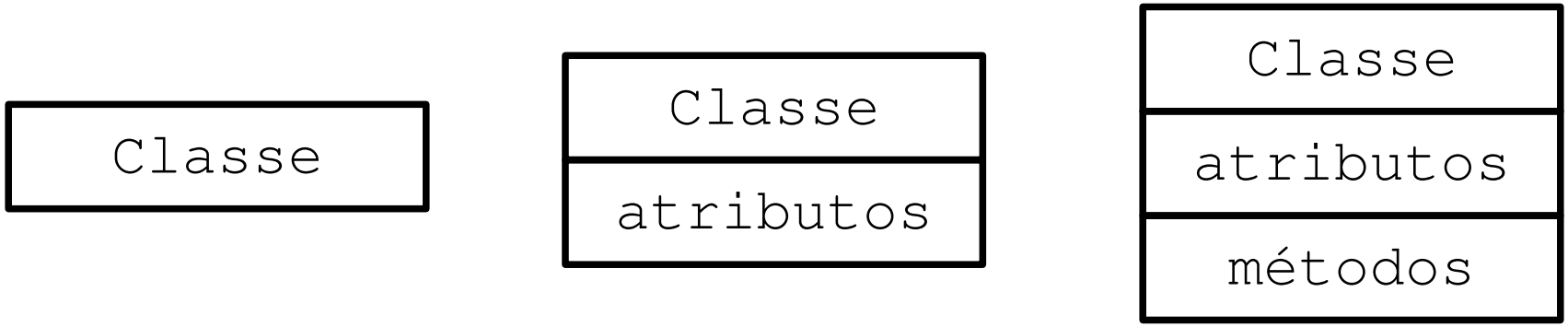
Modelo de carro, receita de bolo

Código:

```
class Jogo { }
```

Classe em UML

- Diagrama



Conceitos Básicos

Instância, Objeto:

Um elemento construído a partir de uma classe, que contém os dados de fato.

Exemplo:

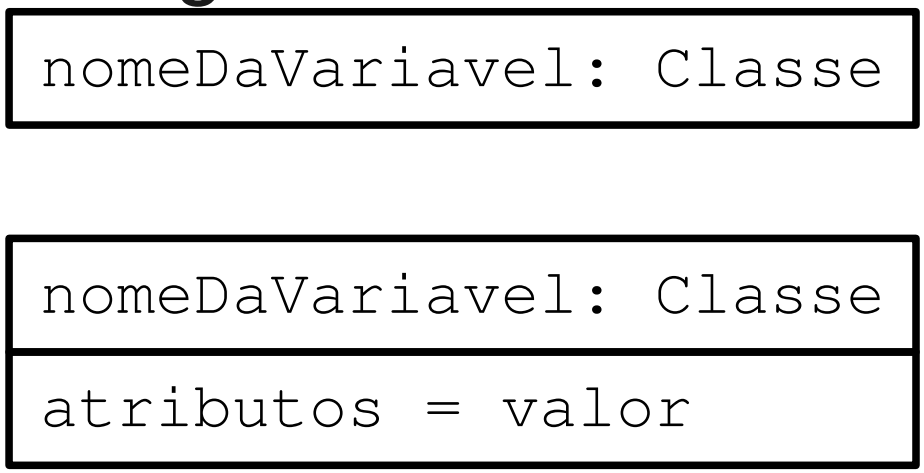
Um carro vermelho, um bolo de chocolate

Código:

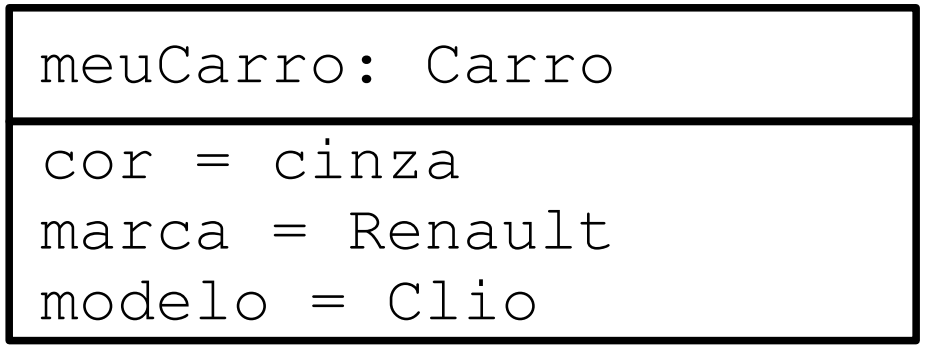
```
new Jogo();
```

Objeto em UML

- Diagrama



- Exemplo:



Exercício

Qual é a diferença entre uma classe e um objeto?

Tempo de vida

O tempo de vida de uma classe é maior do que o de um objeto.

Uma classe existe a partir do momento em que o código é compilado e ela não muda de uma execução do programa para outra.

O objeto é criado durante a execução do programa e pode durar no máximo até o fim daquela execução.

Conceitos Básicos

Método:

Uma operação de um objeto que pode ser executada por ele próprio ou por outros objetos.

Exemplo:

Acelerar o carro, cortar o bolo, falar

Código:

```
int calculaDano() { }
```

Método em UML

```
nomeDoMetodo() : tipoDoRetorno
```

Exemplo:

```
ligar() : boolean
```

```
nomeDoMetodo(parametro1: tipo,
parametro2: tipo) : tipoDoRetorno
```

Exemplo:

```
adicionarCobertura(cobertura:
Chocolate) : void
```

Conceitos Básicos

Atributo:

Variável que armazena valores inerentes ao objeto.

Exemplo:

Cor do carro, sabor do recheio do bolo, data de nascimento do Paulo

Código:

```
class Jogo {
    int numeroDeRodadas; ... }
```

Atributo em UML

```
nomeDoAtributo : tipo
```

Exemplo:

```
combustivel : int
```

```
nomeDoAtributo : tipo = valorPadrao
```

Exemplo:

```
ano : int = 2008
```

Conceitos Básicos

Modificadores de acesso:
Métodos e atributos podem ter acesso público, protegido, de pacote ou privado.

Serve para garantir que apenas o objeto tenha acesso as algumas de suas informações. Isso auxilia na criação de sistemas modularizados.

Conceitos Básicos

As permissões determinam quais classes têm acesso a cada método e atributo.

Público: Todas as classes têm acesso

De pacote: Só as classes no mesmo pacote têm acesso

Protegido: Só classes filhas têm acesso

Privado: Só a própria classe tem acesso

Acesso em UML

Público: Adicione '+' antes do elemento

Privado: Adicione '-' antes do elemento

Protegido: Adicione '#' antes do elemento

De pacote: Sem marcação

Do código para o UML

```
public class Carro {
    protected String modelo;
    protected String marca;
    int ano;
    public double velocidade;
    private int combustivel;

    public void acelera() { ... }
    public void freia(double intensidade) { ... }
    private int consomeCombustivel() { ... }
}
```

Solução

Carro
#modelo : String #marca : String ano : int +velocidade : double -combustivel : int
+acelera() : void +freia(intensidade : int) : void -consomeCombustivel() : int

Do problema para o UML

- Seu sistema precisa de uma classe para representar uma conta bancária. Ela será usada para controlar a quantidade de dinheiro na conta, e deve permitir saque e deposito de qualquer quantia até um certo limite.

  Quais os atributos dessa classe? Quais são seus possíveis métodos? Qual é seu nome?

Relação entre classes: Associação

Relação simples e estática que permite a comunicação entre os objetos.

Pode ser representada com uma flecha e o nome da associação ou com os papéis que cada classe representa na associação.

Exemplos:

Carro e Estrada

Relação entre classes: Agregação

Associação de define a relação de parte-todo entre o agregado (todo) e o componente (parte).

É representada por uma ligação com um losango do lado do agregado.

Exemplos:
Carro e Pneus ou Empresa e pessoas

Relação entre classes: Composição

Uma relação semelhante à agregação mas com dependência da parte para o todo.

É representada por uma linha com um losango preenchido do lado do objeto principal.

Exemplos:
Pessoa e Cabeça

Relação entre Classes

Resumo:

Associação, composição e agregação são formas específicas de apresentar relações do tipo “tem um”.

Exercício 2

Defina o tipo de relacionamento das classes a seguir e faça o UML para elas:

a)

```
public class Barco {
    private Motor motor1; private Motor motor2;
    public Barco(Motor m1, Motor m2) {motor1=m1;motor2=m2;}
    public void acelera() {motor1.acelera();motor2.acelera();}
}

class Motor {
    private int potencia = 0;
    public void acelera() { potencia++; }
}
```

Exercício 2

Defina o tipo de relacionamento das classes a seguir e faça o UML para elas:

b)

```
public class Pedido {
    private List<Item> items;
    public Pedido() { this.items = new LinkedList<Item>(); }
    public void adicionaItem(int id, double valor) {
        this.items.add(new Item(id, valor));}
}
class Item {
    public int id; public double valor;
    public Item(int id, double valor) {
        this.id=id;this.valor=valor; }
}
```

Exercício 3

Defina o tipo de relacionamento das classes a seguir e faça o UML para elas:

c)

```
public class Carro {
    private List<Pessoa> passageiros;
    public Carro() {
        this.passageiros = new LinkedList<Pessoa>();
    }
    public acolhe(Pessoa passageiro) {
        this.passageiros.add(passageiro);
    }
}

class Pessoa {
}
```