Programação Orientada a Objetos

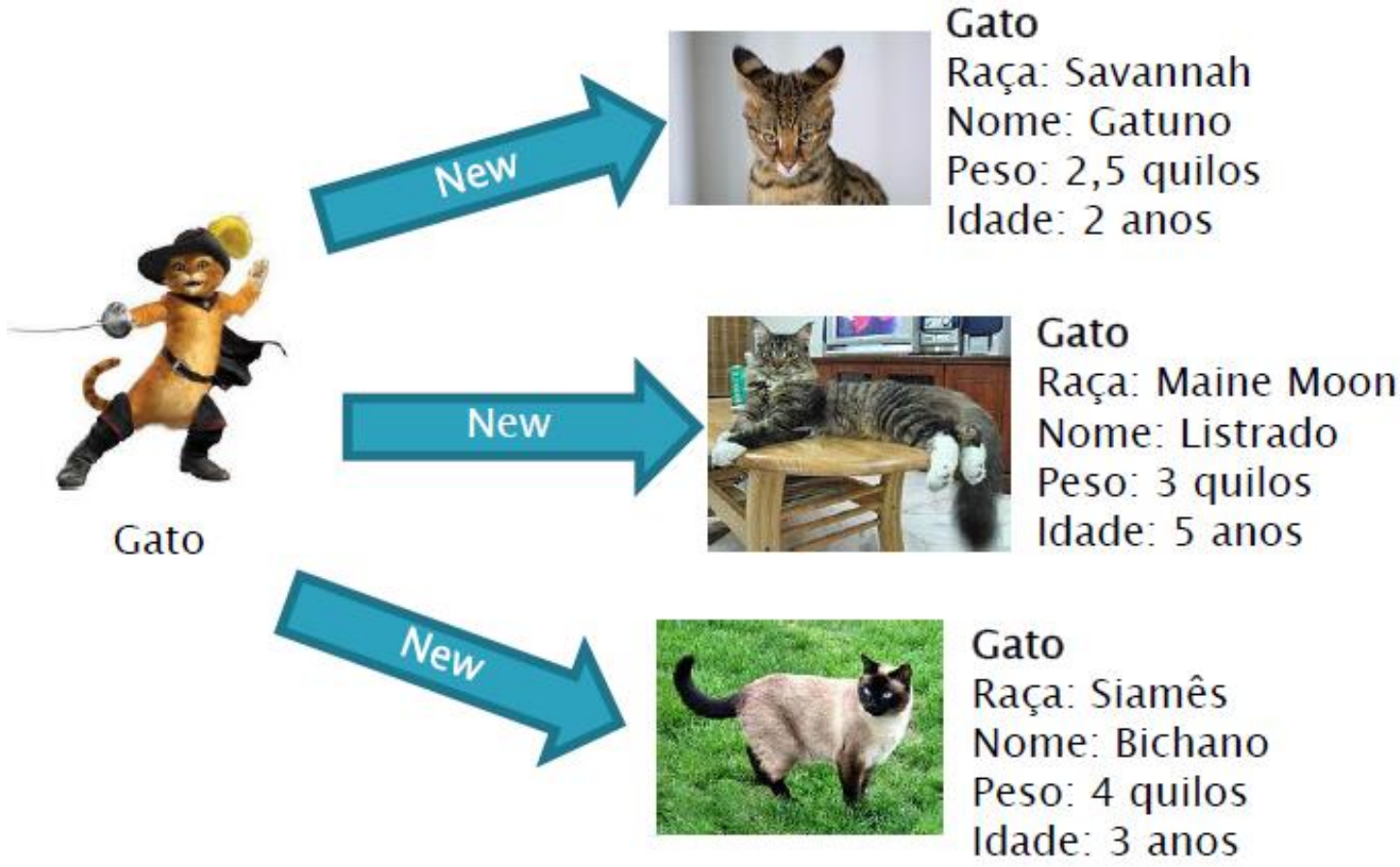
Classes

Professor Me.: Gustavo Siqueira Vinhal

Classes

Classes: Uma descrição que abstrai um conjunto de objetos com características similares.

- São compostas por um conjunto de atributos e métodos, que serão atribuídos aos objetos.



Criando classes

- A definição de uma nova classe se dá pela palavra chave **class**

```
class Nome da classe
        {
                // corpo da classe
        }
```

Criando classes

- Para definir uma subclasse (classe que herdará de uma superclasse) usa-se o comando **extends**

```
class Nome da classe extends Nome da superclasse
        {
                // corpo da classe
        }
```

- Por padrão, toda classe herdam da classe Object.

Criando variáveis de instância e de classe

- As variáveis podem ser classificadas em dois tipos, de instância e de classe:
  - **Variáveis de instância**: atributos exclusivos de cada objetos. A alteração dos valores afeta apenas ao objeto em questão;
  - **Variáveis de classe**: atributos comuns aos objetos de uma mesma classe. A alteração dos valores afeta todos os objetos daquela classe.

Criando variáveis de instância e de classe

- Para definir uma variável de instância:

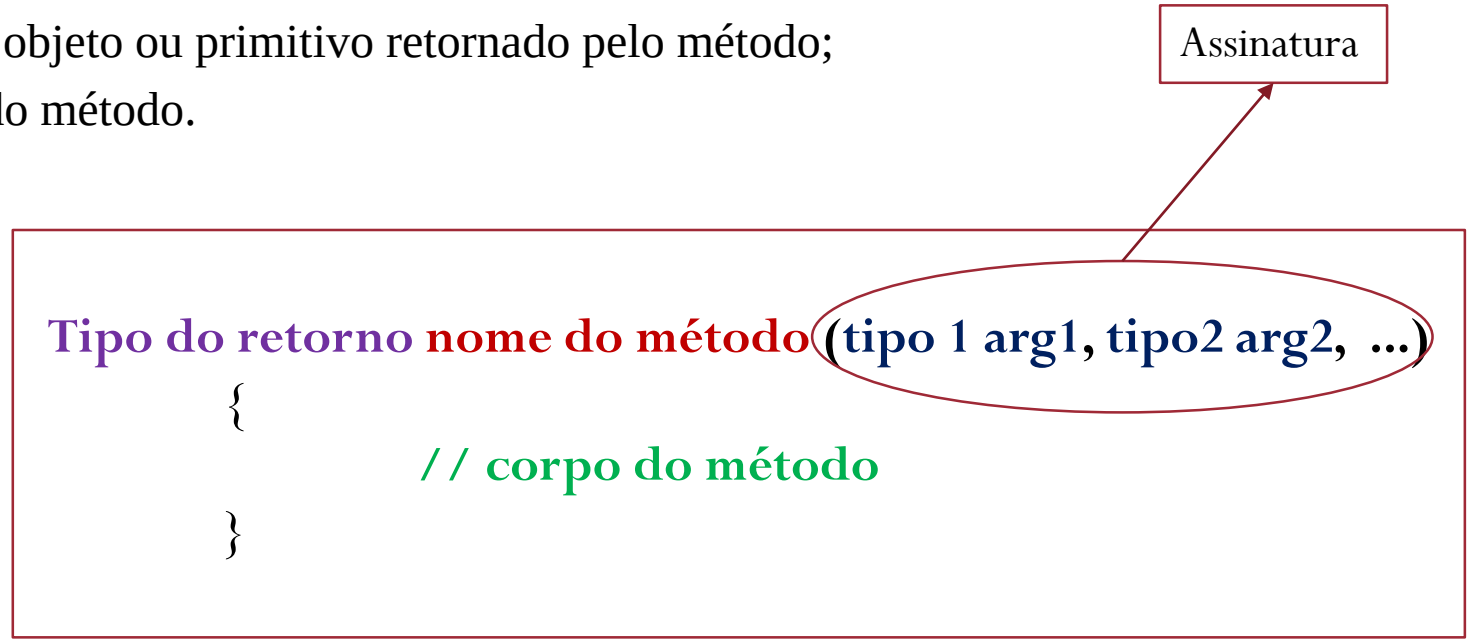
```
tipo nome da variável;
```

- Para definir uma variável de classe deve-se usar a diretiva **static**.

```
static tipo nome da variável;
```

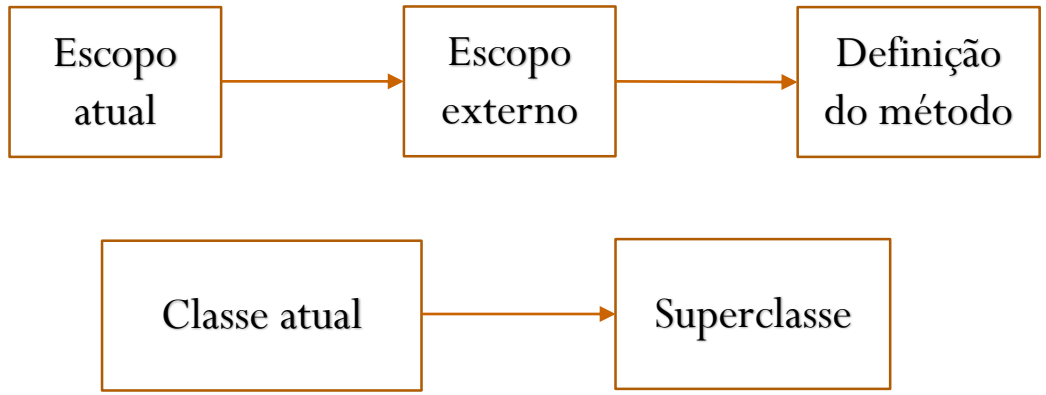
Criando métodos

- Todo método, em Java, possui quatro partes básicas:
  - O nome do método;
  - Uma lista de parâmetros;
  - O tipo de objeto ou primitivo retornado pelo método;
  - O corpo do método.



Escopo de variável e definições de método

- Escopo é a parte de um programa em que uma variável ou outro tipo de informação existe.
- Variáveis com escopo local existem apenas dentro do bloco que foi definida.
- Variáveis de instância possuem um escopo que se estende à classe inteira.
- A referência a variáveis segue uma hierarquia:



Escopo de variável e definições de método

```
class ScopeTest{
    int test = 10;

    void printTest(){
        int test = 20;
        System.out.println("Test: " + test);
    }

    public static void main(String args[]){
        ScopeTest st = new ScopeTest();
        st.printTest();
    }
}
```

- A variável test vai aparecer o valor 20;
- Para evitar isso, troca-se o nome ou usa-se o this

Acessando e definindo valores

- No corpo de uma definição de um método é possível se referir ao objeto atual.
- Isso pode ser feito com o objetivo de acessar as variáveis de instância ou para passar o objeto atual como argumento para outro método.
- Para se referir ao objeto atual, utiliza-se a palavra chave **this**
- Métodos de classe não podem usar **this**

Exemplos:

```
t = this.x; // a variável de instância x para esse objeto
this.resetData(this); // chama o método resetData, definido nesta classe, passando o objeto atual
return this; // retorna o objeto atual
```

Acessando e definindo valores

- Em alguns casos, o uso do **this** é implícito:
  - Variáveis de instância (exceto quando há variáveis com mesmo nome no escopo local);
  - Chamadas de métodos

Exemplos:

```
t = this.x; // a variável de instância x para esse objeto
this.resetData(this); // chama o método resetData, definido nesta classe, passando o objeto atual
return this; // retorna o objeto atual
```

```
t = x; // a variável de instância x para esse objeto
resetData(this); // chama o método resetData, definido nesta classe, passando o objeto atual
return this; // retorna o objeto atual
```

Passando argumentos aos métodos

- Quando se passa um objeto como argumento para um método, o mesmo é passado por referência:
  - Todas as modificações realizadas persistem fora do método (o original é afetado).
- Quando se passa um tipo primitivo como argumento para um método, o mesmo é passado por valor:
  - Todas as modificações são realizadas localmente.

Tipos de métodos

- Existem dois tipos de métodos:
  - Instância: são aqueles que operam sobre uma instância da classe. Para ser utilizado deve ser criado um objeto e o mesmo invocar o método.
  - Classe: são aqueles que não necessitam de uma instância para invoca-lo.

Métodos de classe

- Os métodos de classe são métodos que ficam disponíveis a qualquer instância da própria classe e outras classes.
- Não exige instância para utilizá-los.
- Para criá-los, basta usar o comando **static.**

```
static Tipo do retorno nome do método (tipo 1 arg1, tipo2 arg2,  ...)
        {
                // corpo do método
        }
```

Métodos de classe

- Um método que opera ou afeta determinado objeto, deve ser definido como método de instância;

- Um método que oferece uma capacidade geral, mas não afetam diretamente uma instância de classe, deve ser definido como método de classe.

Sobrecarga

- É possível que dois métodos possuam o mesmo nome, diferenciando-se apenas pela sua assinatura:
  - Número de argumentos que eles utilizam;
  - Tipo de dados ou objetos de cada argumento.
- O uso de vários métodos com mesmo nome mas com assinaturas diferentes é chamado de sobrecarga.
- A sobrecarga permite que métodos se comportem de modo diferente, dependendo dos argumentos que recebem.
- **Atenção:** Java não considera o tipo de retorno para diferenciar métodos sobrecarregados.

Redefinindo métodos

- Quando um objeto chama um método, o Java procura a definição do método na classe do objeto. Caso ele não encontre, ele procura na classe superior.

- Redefinir um método significa criar um método na subclasse com mesma assinatura de um na superclasse.

- Se dois métodos possuírem a mesma assinatura, ele executa o método mais inferior a hierarquia de classes.

- Se deseja executar um método em uma superclasse com mesma assinatura de outro na subclasse, utiliza-se o comando **super.NomeDoMétodo()**. Essa chamada é realizada dentro do método da subclasse.

Métodos construtores

- Um método construtor é um método chamado sobre um objeto quando ele é criado.
- Quando o comando **new** é chamado ocorrem os passos:
  - Aloca memória para o objeto;
  - Inicializa as variáveis de instância desse objeto, seja para valores iniciais ou para um padrão;
  - Chama o método construtor da classe.
- Regras de um método construtor:
  - Tem o mesmo nome da classe;
  - Não tem tipo de retorno;
  - Não podem retornar nenhum valor usando **return**.
- Métodos construtores também podem ser sobrecarregados.

Métodos finalizadores

- Opostos aos métodos construtores.
- Servem para destruir objetos e liberar memória.

```
protected void finalize() throws Throwable{
    super.finalize();
}
```

Prática:

1 – Crie um programa que realize operações entre matrizes:

- Adição;
- Subtração;
- Multiplicação por um escalar;
- Multiplicação entre matrizes;
- Transposta.


Obrigado!