PROGRAMAÇÃO ORIENTADA A OBJETOS

Introdução a Classes e Objetos

CONCEITO DE CLASSES E OBJETOS

Classes

"Uma classe é uma **abstração** utilizada para representar um conjunto de objetos com **características** e **comportamentos** idênticos."

Ou em outras palavras:

"**Abstrações** utilizadas para **representar** um conjunto de **objetos** com *características e comportamento idênticos."*

Ou ainda

"Pode ser pensada como um **gabarito**, um **protótipo ou** uma **planta de um objeto**"

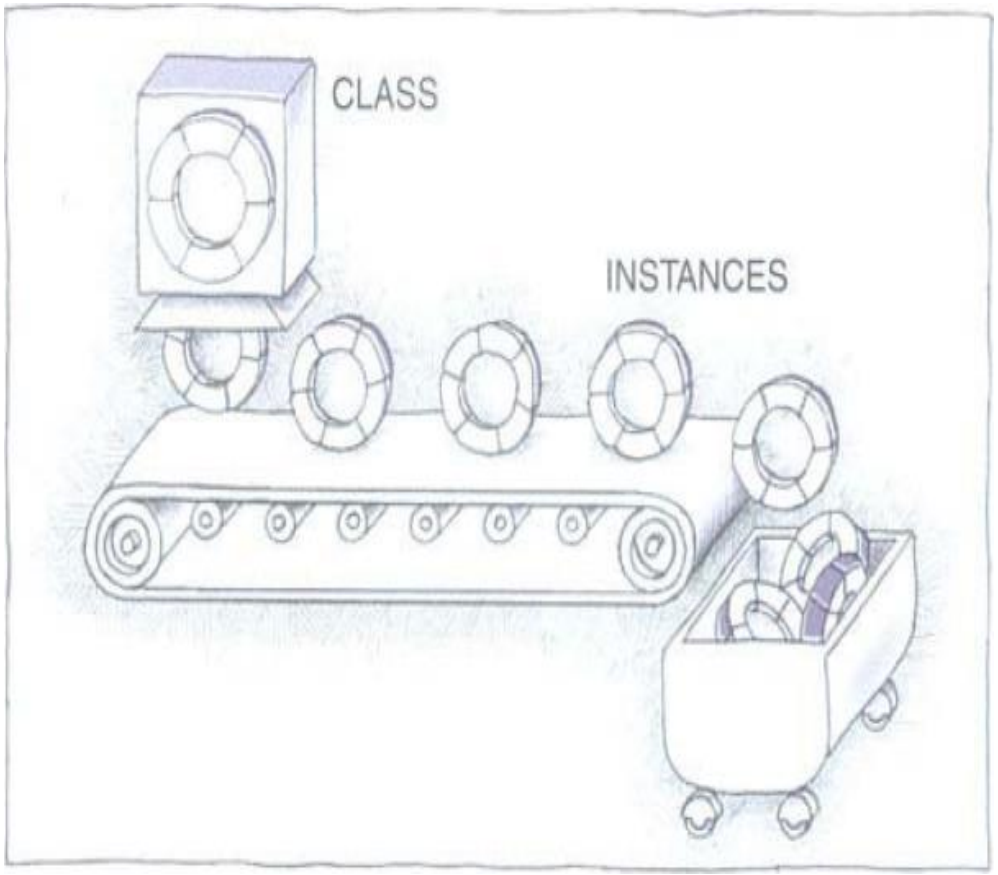
Objetos

Tecnicamente falando, objetos são “**instâncias**” em tempo de execução de uma classe

- Todos os objetos **são instâncias de** alguma **classe**
- Todos os objetos de uma classe são idênticos no que diz respeito a sua interface e implementação (o que difere um objeto de outro é o seu **estado** e sua **identidade**)

Classes e Objetos

Podemos dizer que: “Classes são fábricas de objetos.”



Classes e Objetos

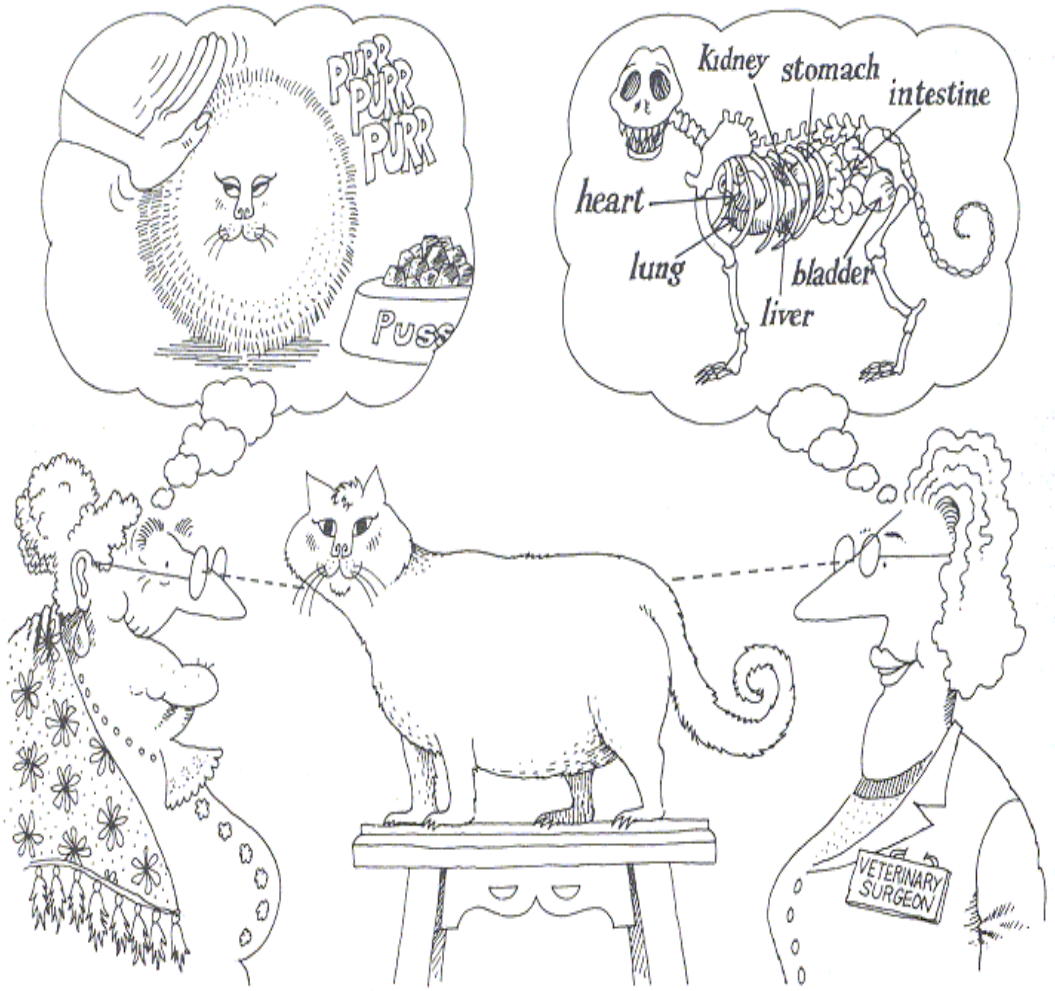
Podemos dizer que: “Classes são fábricas de objetos.”

Abstração

Uma abstração é qualquer modelo que inclui os aspectos relevantes de alguma coisa, ao mesmo tempo em que ignora os menos importantes.

A depender do foco da modelagem, elementos diferentes poderão ser considerados no contexto analisado.

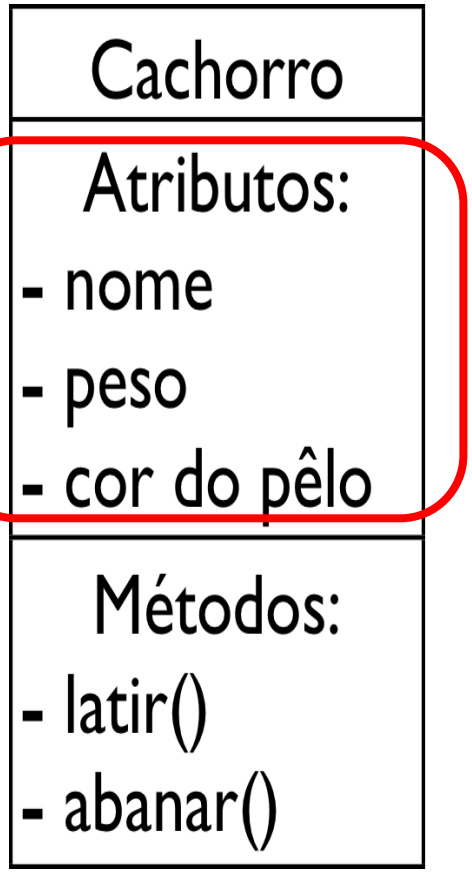
O que uma Senhora e uma Veterinária veem quando olham para um gatinho?



Abstraction focuses upon the essential characteristics of some object, relative to the perspective of the viewer.

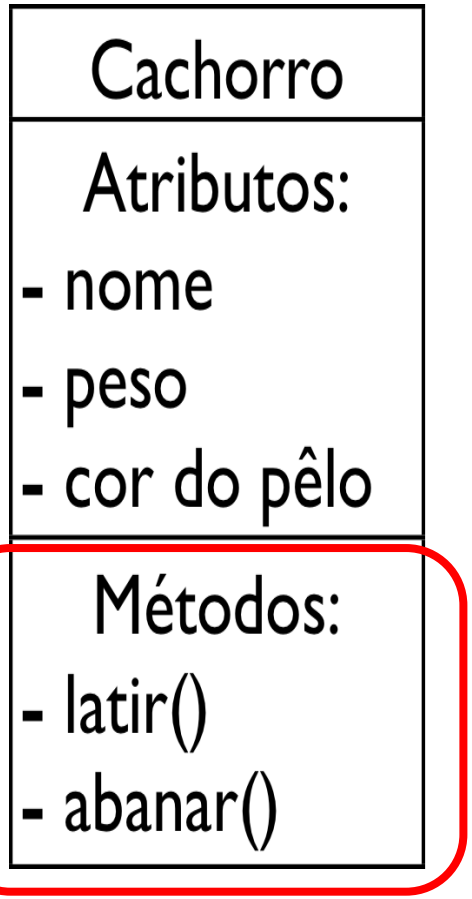
Elementos de uma Classe - Atributos

- Descrevem as características das instâncias de uma classe
- Seus valores definem o estado do objeto
- O estado de um objeto pode mudar ao longo de sua existência
- A identidade de um objeto, contudo, nunca muda



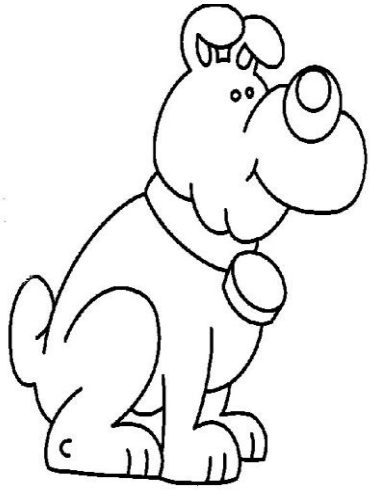
Elementos de uma Classe - Métodos

- Representam o comportamento das instâncias de uma classe
- Correspondem ao protocolo ou ações das instâncias de uma classe
- Em outras palavras, parte do código que representa as ações que o objeto pode executar



Classes e Objetos

Quem você diria que é a classe e quem seria o objeto?



Elementos de uma Classe - Métodos

- Um método é a implementação de uma operação
- Métodos só tem acesso aos dados da classe para a qual foram definidos
- Os dados de uma classe só podem ser manipulados por métodos da classe (pelo princípio do encapsulamento)
- Métodos possuem argumentos, variáveis locais , valor de retorno etc.
- Alguns métodos especiais:
  - Construtores – criam objetos de uma classe
  - Destrutores – destroem objetos de uma classe

Elementos de uma Classe – Atributos e Métodos

- Em resumo:
- **Atributos (variáveis)** são as coisas que a classe conhece, os dados ou informações do objeto, basicamente a estrutura de dados que vai representar a classe.

```
private String nome, endereco, email;
```

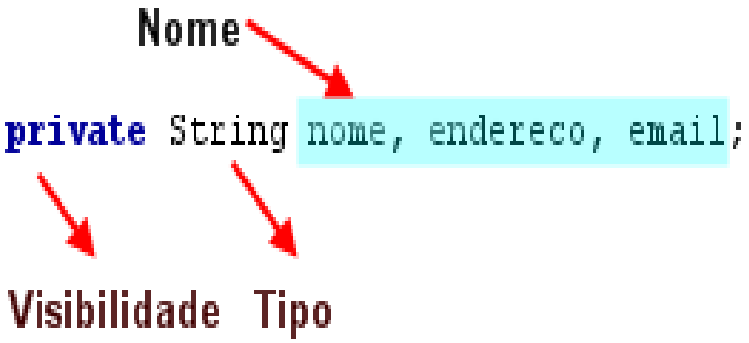
- **Métodos** são as coisas que a classe sabe fazer

```
public void setNome(String nome) {
    this.nome = nome;
}
```

- Falando sobre atributos e métodos, no Java, o **espaço ocupado por um objeto é a soma dos espaços dos seus atributos**. Enquanto os **métodos não ocupam** espaço pois, na verdades, são os mesmos para todas as instâncias

Anatomia de um atributo

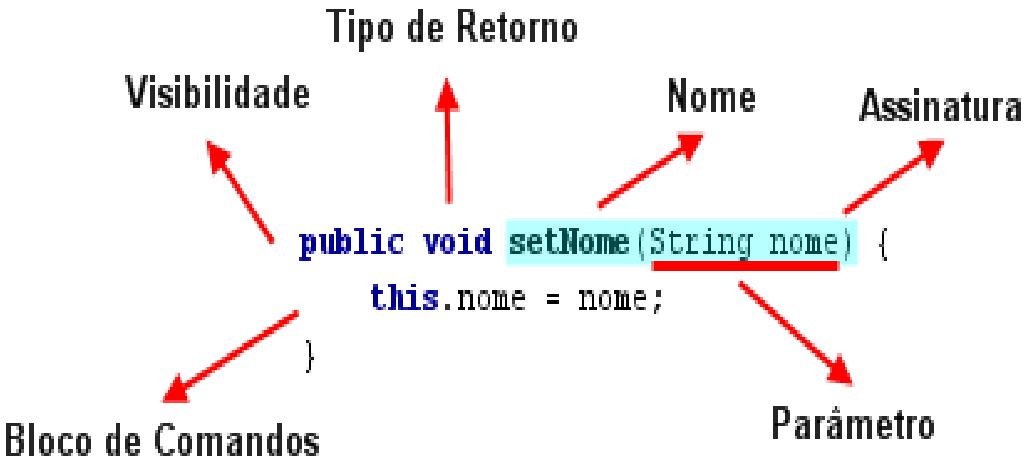
- Os atributos podem ser de classe ou de método:
  - **Atributos de classes** são criados em cada instância da classe, os objetos e devem ser descrito fora dos blocos de comandos dos métodos.
  - Enquanto os **atributos de métodos** só existem dentro do bloco de comando do respectivo método. Os atributos de método não aceitam modificadores de acesso.



Anatomia de um método

- Os métodos encerram um conjunto de declarações de dados e de comandos que são executados mediante a chamada do método por seu nome.
  - O método main é o ponto onde se dá o início da execução do programa, ele é chamado automaticamente pela JVM.

- Existem situações onde um método não precisa retornar nenhuma informação, neste casos é necessário informar o retorno como sendo nulo, com a palavra chave **void**.



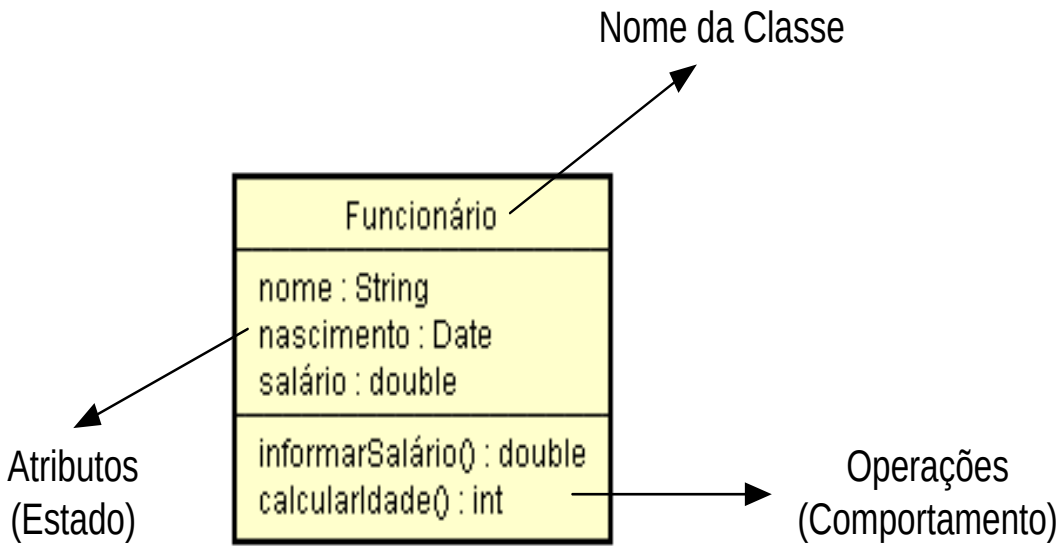
Modificadores de visibilidade

- **public**
  - Permite que o atributo ou método seja acessado livremente por qualquer outra classe, que importe a classe em questão.
- **private**
  - Restringe o acesso ao atributo ou ao método à própria classe.
- **protected**
  - Restringe o acesso externo aos métodos e atributos. Ou seja, o acesso só é permitido a própria classe, a suas subclasses (mesmo se em pacotes diferentes) e ao pacote onde estes se encontram.
- **default (sem declaração)**
  - Restringe o acesso externo aos métodos e atributos. Apenas classes do mesmo pacote possuem acesso

Com relação à visibilidade dos atributos, a visibilidade pública para atributos é considerada prática ruim de programação, pois expõe o estado do objeto para manipulação externa, permitindo portanto que o mesmo seja colocado em um estado inconsistente

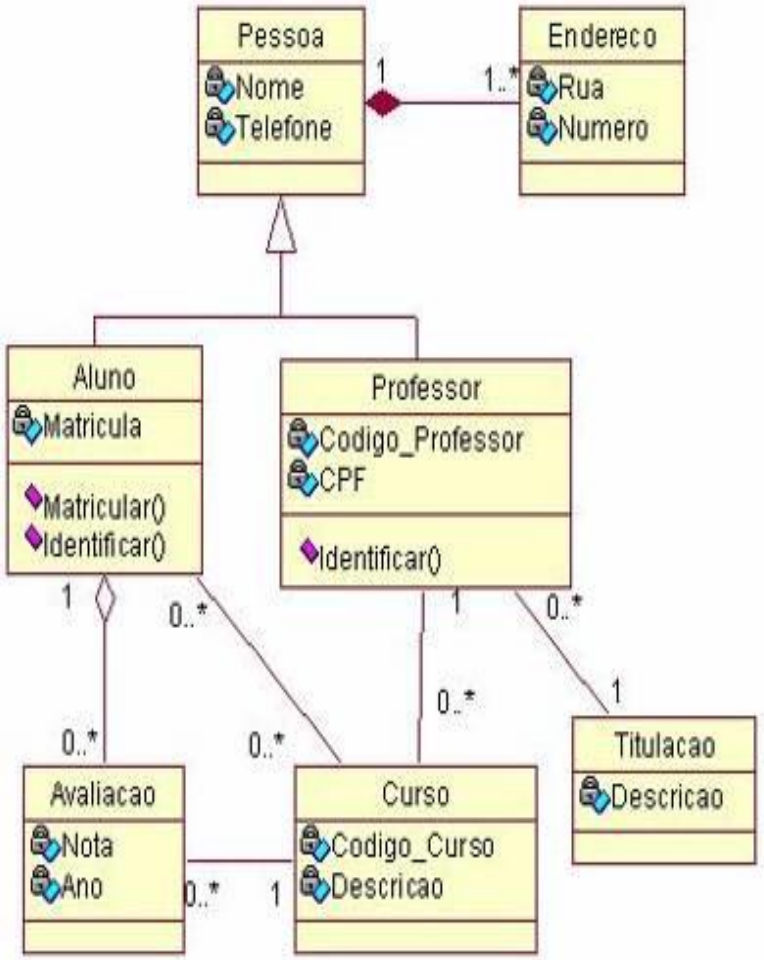
Representação da Classe em UML

- Existem diversas ferramentas para modelagem em UML
- Exemplo de representação de uma classe em UML



Representação da Classe em UML

- Diagrama de Classes da UML representa as classes de um escopo de sistema e seus relacionamentos



Fonte: Devmedia – www.devmedia.com.br

CODIFICANDO CLASSES

Estrutura da Classe e seus Atributos

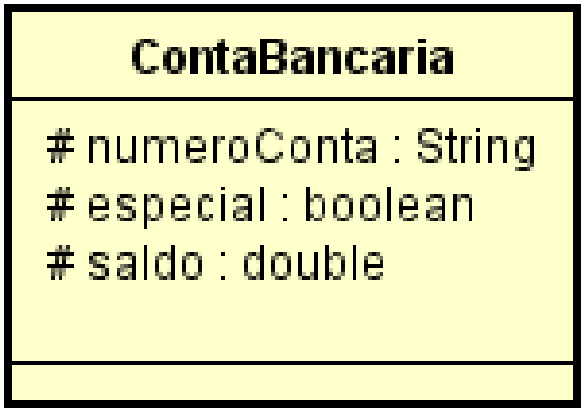
Vendo na prática, ou melhor, no código

- Vamos pensar no exemplo de uma conta bancária.

Quais atributos poderíamos escolher para descrever uma conta bancária?

Vendo na prática, ou melhor, no código

- Inúmeros atributos podem ser escolhidos, vai depender do escopo de nossa aplicação e abstração do modelo.




```java
package br.senac.contabancaria;

public class ContaBancaria {

    //Atributos da classe Conta
    String numeroConta;
    boolean especial;
    double saldo;

}
```

Classe do diagrama UML

Classe em código Java

Classe Conta

```
package br.senac.contabancaria;

public class ContaBancaria {

    //Atributos da classe Conta
    String numeroConta;
    boolean especial;
    double saldo;

}
```

A Classe Conta pode ser executada?

- Sem o método main, a classe não poderá ser executada, ou seja, não será possível criar um objeto com base nela.
- Também não cabe colocarmos o método main na classe Conta, essa classe terá seus próprios métodos. Vamos deixar isso para uma classe específica para a execução do programa

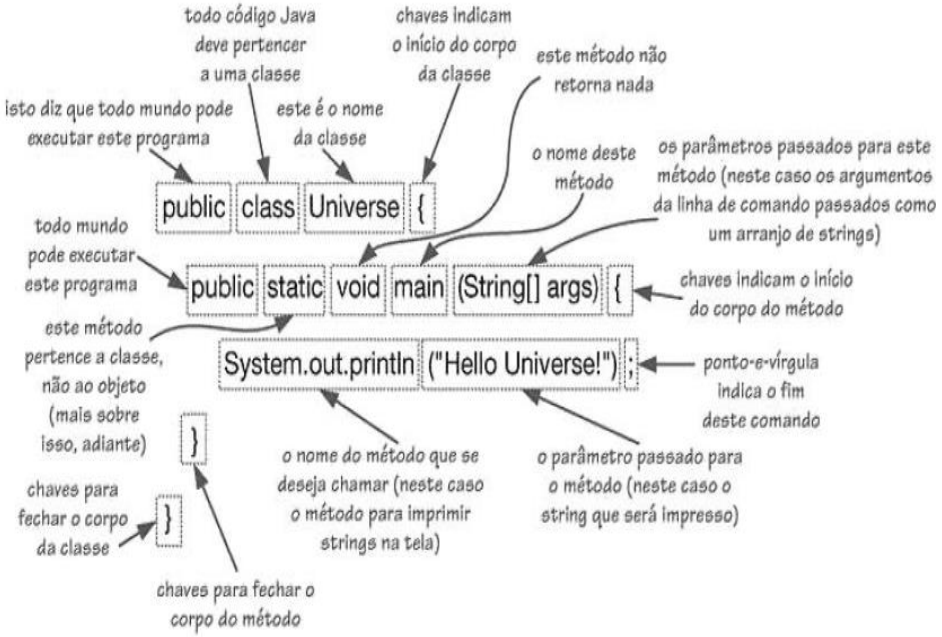
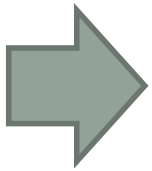


Figura 1.1 O programa "Hello Universe!"

Criando a Classe Aplicação

- Será a Classe com o método main, fazendo uso de objetos derivados da Classe ContaBancaria

```java
package br.senac.contabancaria;

public class ContaBancaria {

    //Atributos da classe Conta
    String numeroConta;
    boolean especial;
    double saldo;

}
```

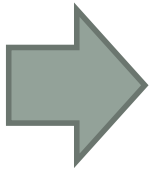


```java
package br.senac.contabancaria;

public class AplicacaoContaBancaria {

    public static void main(String[] args) {

        //Instânciando um objeto da Classe ContaBancaria
        ContaBancaria novaConta;
        novaConta = new ContaBancaria();

    }
}
```

Acessando os atributos de Conta

- Precisaremos atribuir valores nos atributos de Conta. Para isso usaremos a navegação da classe através do ponto “ . ”

```
package br.senac.contabancaria;

public class ContaBancaria {

    //Atributos da classe Conta
    String numeroConta;
    boolean especial;
    double saldo;

}
```



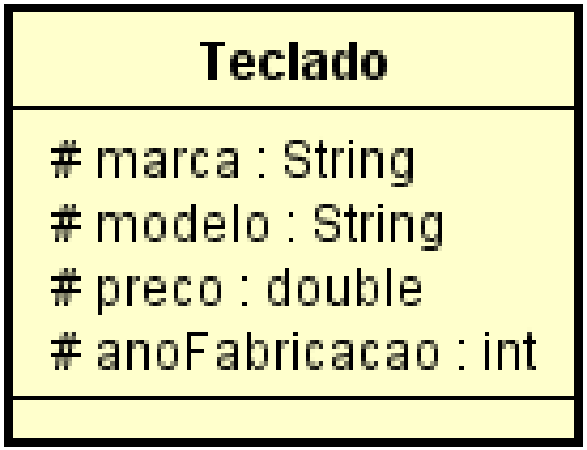
```
package br.senac.contabancaria;

public class AplicacaoContaBancaria {

    public static void main(String[] args) {

        //Instânciando um objeto da Classe ContaBancaria
        ContaBancaria novaConta;
        novaConta = new ContaBancaria();

        novaConta.numeroConta = "25998-7";
        novaConta.especial = true;
        novaConta.saldo = 9500.00;
    }
}
```

Classe Aplicação completa

```
package br.senac.contabancaria;

public class AplicacaoContaBancaria {

    public static void main(String[] args) {

        //Instânciando um objeto da Classe ContaBancaria
        ContaBancaria novaConta;
        novaConta = new ContaBancaria();

        novaConta.numeroConta = "25998-7";
        novaConta.especial = true;
        novaConta.saldo = 9500.00;
    }
}
```

Exercício

- Desenvolva uma classe que representa produtos do tipo teclados para computador a venda numa loja. Abaixo segue parte do diagrama de classes contando a especificação da classe.
- Crie também uma Classe de aplicação para instanciar objetos da Classe Teclado.



Exercício - Solução

```
package br.senac.teclado;

public class Teclado {

	//Atributos
	String marca, modelo;
	double preco;
	int anoFabricacao;

}
```

Exercício – Solução (continuação)

```
package br.senac.teclado;

public class AplicacaoTeclado {

    public static void main(String[] args) {

        Teclado novoTeclado = new Teclado();

        novoTeclado.marca = "Logitech";
        novoTeclado.modelo = "K400";
        novoTeclado.preco = 135.90;
        novoTeclado.anoFabricacao = 2015;

        System.out.println("Informações do produto");
        System.out.println("----------------------");
        System.out.println("Marca: " +novoTeclado.marca);
        System.out.println("Modelo: " +novoTeclado.modelo);
        System.out.println("Preço: R$" +novoTeclado.preco);
        System.out.println("Fabricação: " +novoTeclado.anoFabricacao);
        System.out.println("----------------------");
    }
}
```

CODIFICANDO CLASSES

Métodos da Classe

Métodos

- Como visto anteriormente, os Métodos são ações ou comportamentos da Classe. São operações que podem utilizar como insumo os valores dos atributos da Classe.
- Vamos escrever o código ao lado →
- Crie uma Classe aplicação (main) para testar!

```
package br.senac.classesimples;

public class ExemploMetodos {

    //Atributos
    int numero1;
    int numero2;

    //Métodos
    public int somar(){
        return numero1 + numero2;
    }

    public int subtrair(){
        return numero1 - numero2;
    }

    public int calcularMod(){
        return numero1 % numero2;
    }
}
```

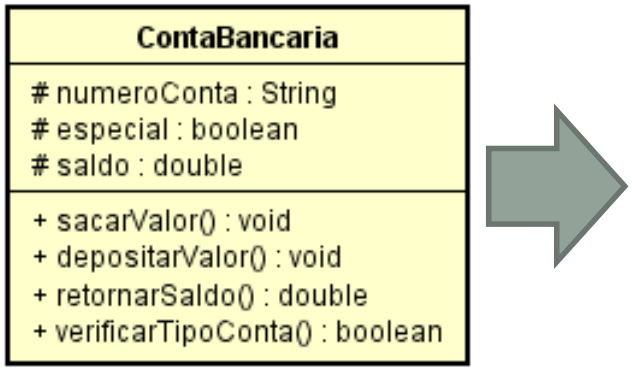
Adicionando os métodos da Classe Conta

- No exemplo anterior a classe Conta possuía apenas os atributos, sem os métodos.

- E se quisermos adicionar algumas ações que uma conta pode realizar utilizando os seus atributos, por exemplo:
  - Sacar um valor da conta
  - Depositar um valor na conta
  - Exibir o saldo da conta
  - Exibir se é conta especial

Atualizando a Classe ContaBancaria

- Adicionando os métodos




```java
package br.senac.contabancaria;

public class ContaBancaria {

    //Atributos da Classe Conta
    String numeroConta;
    boolean especial;
    double saldo;

    //Métodos da Classe
    public void sacarValor(double saque){
        saldo = saldo - saque;
    }

    public void depositarValor(double deposito){
        saldo = saldo + deposito;
    }

    public double retornarSaldo(){
        return saldo;
    }

    public boolean verificarTipoConta(){
        return especial;
    }
}
```

```java
package br.senac.contabancaria;

public class ContaBancaria {

    //Atributos da Classe Conta
    String numeroConta;
    boolean especial;
    double saldo;

    //Métodos da Classe
    public void sacarValor(double saque){
        saldo = saldo - saque;
    }

    public void depositarValor(double deposito){
        saldo = saldo + deposito;
    }

    public double retornarSaldo(){
        return saldo;
    }

    public boolean verificarTipoConta(){
        return especial;
    }
}
```

```java
import javax.swing.JOptionPane;

public class AplicacaoContaBancaria {

    public static void main(String[] args) {

        //Instânciando um objeto da Classe ContaBancaria
        ContaBancaria novaConta;
        novaConta = new ContaBancaria();

        novaConta.numeroConta = "25998-7";
        novaConta.especial = true;
        novaConta.saldo = 9500.00;


        //Realizando operação de saque
        String entradaDados = JOptionPane.showInputDialog("Digite o valor para saque");
        double valorTransacao = Double.parseDouble(entradaDados);
        novaConta.sacarValor(valorTransacao);
        JOptionPane.showMessageDialog(null, "Saldo da conta R$" +novaConta.retornarSaldo());

        //Realizando operação de depósito
        entradaDados = JOptionPane.showInputDialog("Digite o valor para deposito");
        valorTransacao = Double.parseDouble(entradaDados);
        novaConta.depositarValor(valorTransacao);
        JOptionPane.showMessageDialog(null, "Saldo da conta R$" +novaConta.retornarSaldo());

        //Verificando o tipo de conta
        boolean tipoConta = novaConta.verificarTipoConta();
        if (tipoConta == true){
            JOptionPane.showMessageDialog(null, "Conta do tipo Especial");
        }else{
            JOptionPane.showMessageDialog(null, "Conta do tipo Comum");
        }
    }
}
```

Métodos construtores

- É um tipo especial de “método” que é chamado no momento em que o Objeto é instanciado
- Responsável pela alocação de recursos de memória para o uso do Objeto
- Quando não declarado explicitamente, o Java cria e executa um construtor default (padrão)
- O comando “new” que cria o Objeto, chama o construtor
- Se não forem repassados valores de parâmetro para o construtor, o Objeto é criado com valores nulos (null), ou zero se numéricos. Em outras linguagens pode ser necessário zerar os atributos na criação para não pegar “lixo” da memória

Métodos construtores

```java
//Classe que vai montar a estrutura de armazenamento de dados sobre alunos
public class RegAluno {

    //Declaração dos atributos de alunos
    String nomeAluno;
    int matriculaAluno;
    int anoMatricula;
    double crAluno; //Coeficiente de rendimento do aluno

    //Método construtor do registro do aluno
    RegAluno (String nome, int matricula, int ano, double cr) {
        nomeAluno = nome;
        matriculaAluno = matricula;
        anoMatricula = ano;
        crAluno = cr;
    } //Fim do construtor do registro
} //Fim da classe aluno
```

- Note que o nome do construtor é o nome da Classe!
- É possível com o construtor, criar o Objeto já com valores para os seus atributos

Métodos construtores

```
import javax.swing.JOptionPane;
public class ProgramaAlunoTeste {

    public static void main (String [] args){

        //Criando um objeto do tipo RegAluno
        RegAluno novoAluno = new RegAluno("Henrique", 1230, 2014, 8.3);

        JOptionPane.showMessageDialog(null,"Nome do aluno: " +novoAluno.nomeAluno +"\n"
                                           +"Matrícula do aluno: " +novoAluno.matriculaAluno +"\n"
                                           +"Ano da matrícula: " +novoAluno.anoMatricula +"\n"
                                           +"Coeficiente de rendimento: " +novoAluno.crAluno);
    }
}
```

- O código acima faz uso da classe RegAluno
- Veja que os atributos do novo aluno foram informados como parâmetro na chamada do construtor.

OBJETOS E REFERÊNCIA NA MEMÓRIA

Objetos são acessados por referências

- Ao fazer um:

```
Conta minhaConta = new Conta();
```

- Não estamos colocando o objeto na variável minhaConta
- Estamos colocando a referência da memória (endereço na memória) na variável
- Em Java uma variável **nunca é um objeto!**

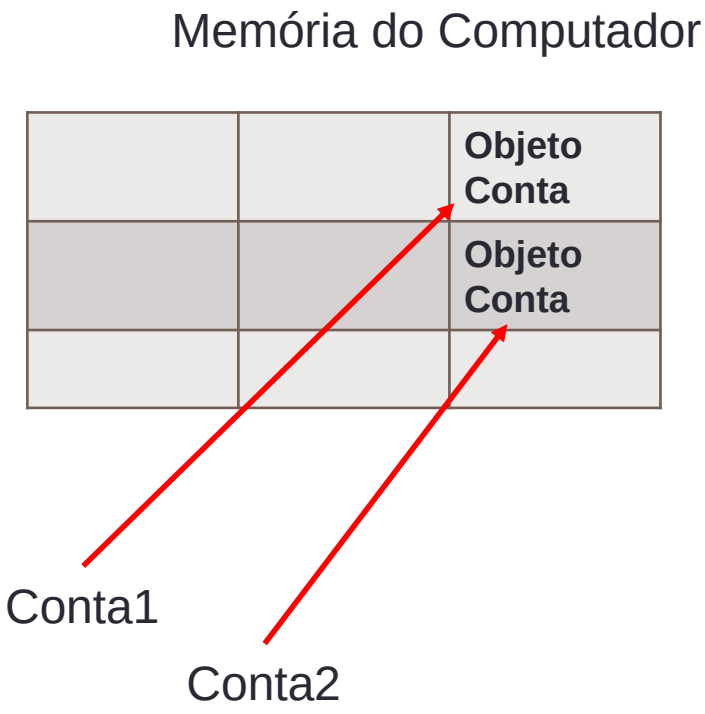
Objetos são acessados por referências

- Exemplo:

```
Conta conta1 = new Conta();
Conta conta2 = new Conta();
```

As variáveis apontam para o endereço de memória onde os objetos estão armazenados

O comando new sempre cria um objeto na memória

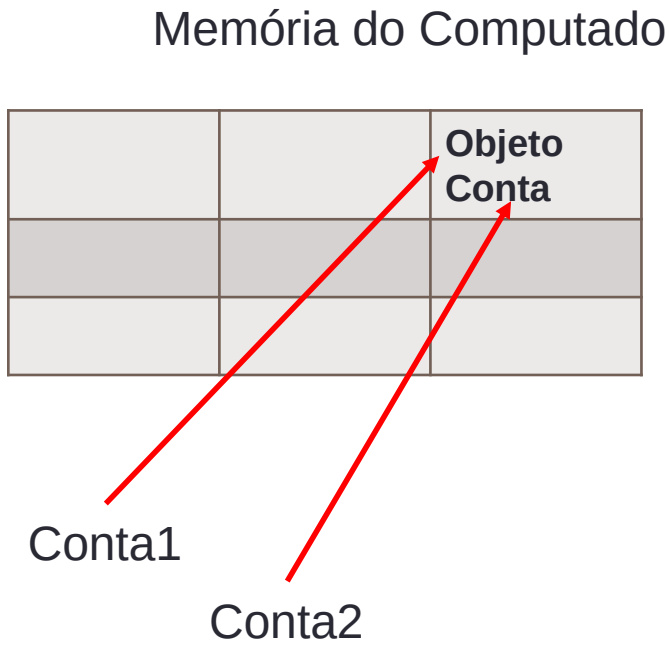


Objetos são acessados por referências

- Exemplo:

```
Conta conta1 = new Conta();
Conta conta2 = conta1;
```

Ambas as variáveis apontarão para o mesmo objeto. Ou seja uma modificação em conta2 surtirá efeito em conta1, já que referenciam o mesmo objeto



Objetos são acessados por referências

- Exemplo:

Nesse caso c1 e c2 apontam para o mesmo objeto

Vamos supor que o saldo seja 1000

c1 faz um depósito de 100 e agora a conta tem 1100

c2 faz um depósito de 200, mas como aponta para o mesmo objeto o saldo não será mais 1100 e sim 1300

```
class TestaReferencias {
    public static void main(String args[]) {
        Conta c1 = new Conta();
        c1.deposita(100);

        Conta c2 = c1;  // linha importante!
        c2.deposita(200);

        System.out.println(c1.saldo);
         System.out.println(c2.saldo);
    }
}
```

Objetos são acessados por referências

- Exemplo:

Aqui duas contas são criadas: c1 e c2

Ambas tem new, ou seja, dois objetos foram criados e referenciados separadamente

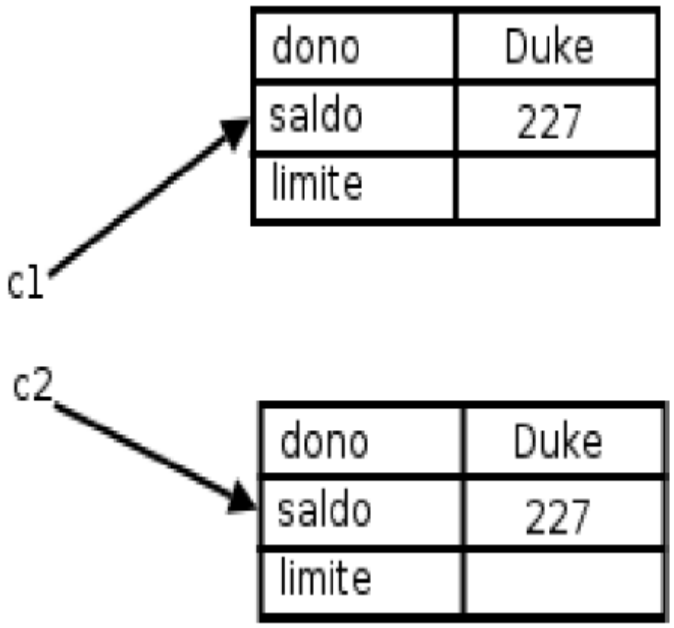
Apesar dos valores dos atributos serem os mesmos, os objetos são diferentes

O programa não compara o valor de atributos e sim o valor do endereço de memória referenciado nas variáveis c1 e c2

```
public static void main(String args[]) {
    Conta c1 = new Conta();
    c1.dono = "Duke";
    c1.saldo = 227;

    Conta c2 = new Conta();
    c2.dono = "Duke";
    c2.saldo = 227;

    if (c1 == c2) {
        System.out.println("Contas iguais");
    }
}
```

Objetos são acessados por referências

```
public static void main(String args[]) {
    Conta c1 = new Conta();
    c1.dono = "Duke";
    c1.saldo = 227;

    Conta c2 = new Conta();
    c2.dono = "Duke";
    c2.saldo = 227;

    if (c1 == c2) {
        System.out.println("Contas iguais");
    }
}
```



KEEP CALM AND HAVE A BREAK