Programação Orientada a Objetos

Educação a Distância
Cruzeiro do Sul Educacional
Campus Virtual

Material Teórico

Classes x Objetos

Responsável pelo Conteúdo:
Prof. Esp. Alexander Gobbato Albuquerque

Revisão Textual:
Prof. Ms Rosemary Toffoli

UNIDADE Classes x Objetos

- Classes x Objetos
- Principais Conceitos
- Tipos de dados em Java
- Set e Get

Objetivo de APRENDIZADO

Aprenderemos o que é classe e o que é objeto. Veremos como trazer o problema do mundo real para o mundo da orientação a objetos.

Aprenderemos o conceito de atributos, “métodos de acesso”, “parâmetros” e outros.

Lembramos a você da importância de realizar todas as atividades propostas dentro do prazo estabelecido para cada Unidade, dessa forma, você evitará que o conteúdo se acumule e que você tenha problemas ao final do semestre.

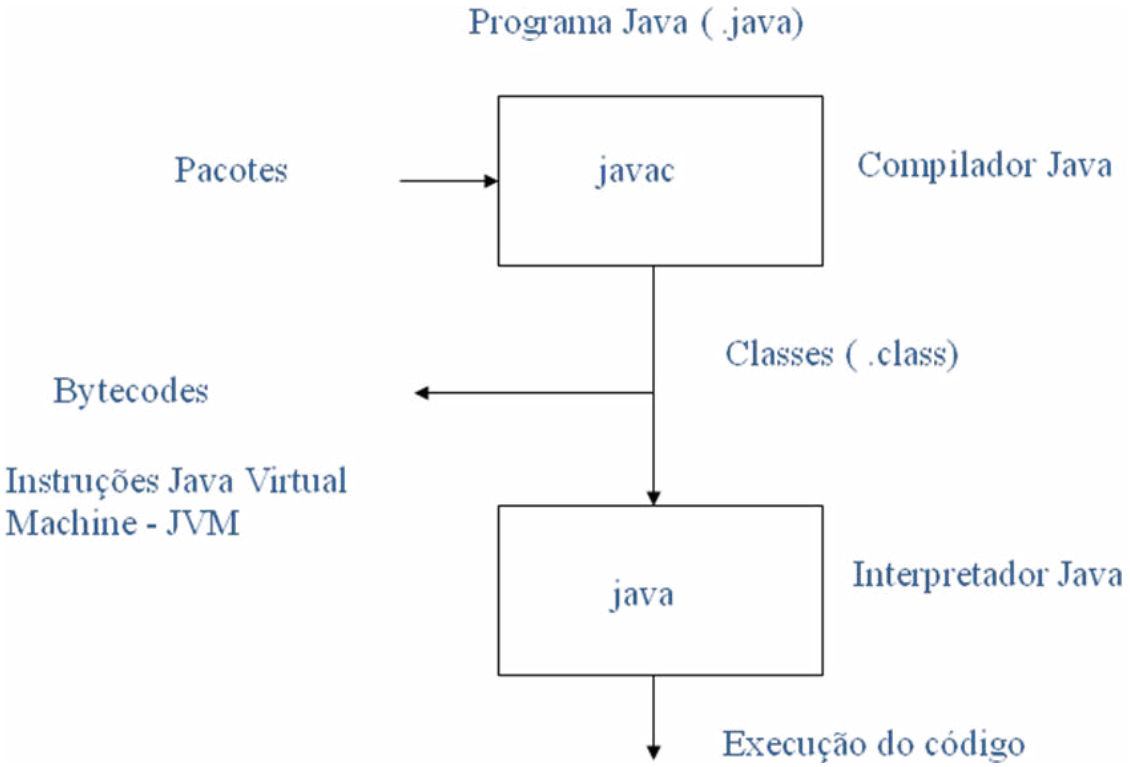
Caso tenha problemas para acessar algum item da disciplina, ou dúvidas com relação ao conteúdo, não deixe de entrar em contato com seu professor tutor através do botão mensagens.

Contextualização

Vimos na unidade anterior como é a estrutura de um programa orientado a objetos e de um programa estruturado. Agora vamos apresentar como criar os métodos, usar encapsulamento e chamar os métodos criados através de objetos.

Classes x Objetos

Mecanismo de Trabalho



A JVM interpreta os bytecodes gerados pelo compilador.

O objetivo da JVM é permitir que qualquer sistema operacional possa executar uma aplicação Java

Classe Java

```
public class BemVindo {
    public static void main(String args[]) {
        System.out.println("Bem vindo a Java.");
    }
}
```

Atenção

Lembre-se o Java é case sensitive (há diferença entre maiúsculas e minúsculas)

Programa = Classe

```
public class BemVindo {
    public static void main(String args[]){
      System.out.println(”Bem vindo”); /* esta linha impirme uma mensagem na tela*/
    }
}
```

Atenção

Os arquivos Java têm que ter o nome da classe principal.

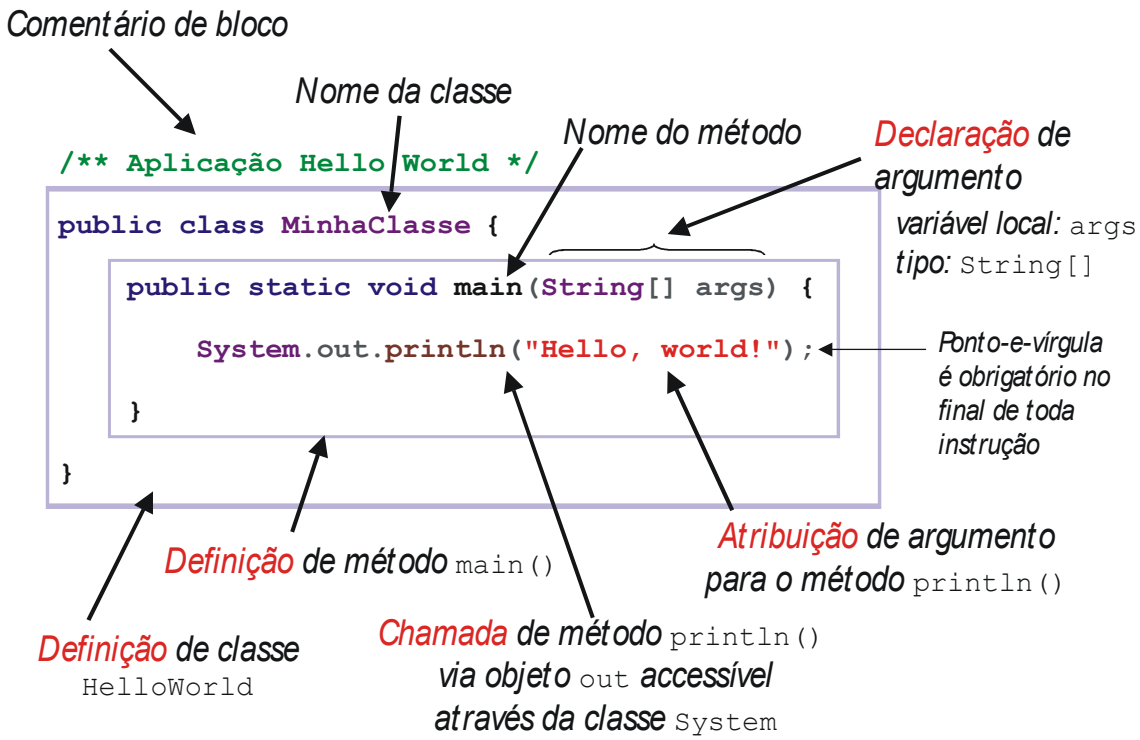
No exemplo acima, o nome do arquivo seria Bemvindo.java e depois de compilado ficaria Benvindo.class

Nome da classe = Nome dos Arquivos

```
public class BemVindo {
    public static void main(String[ ] args){
      System.out.println(”Benvindo”);
    }
}
```

Método main

```
public class BemVindo {
    public static void main(String[ ] args){
      System.out.println(”Bemvindo”);
    }
}
```



Principais Conceitos

O que são Objetos?

São quaisquer coisas na natureza que possuam propriedades (características) e comportamentos (operações).

Exemplos de Objetos: um bolo, um cachorro, um livro, etc...

Orientação a Objetos:

O termo orientação a objetos significa organizar o mundo real como uma coleção de objetos que incorporam estrutura de dados e um conjunto de operações que manipulam estes dados, como exemplo podemos montar os seguintes objetos pessoas e pássaros.

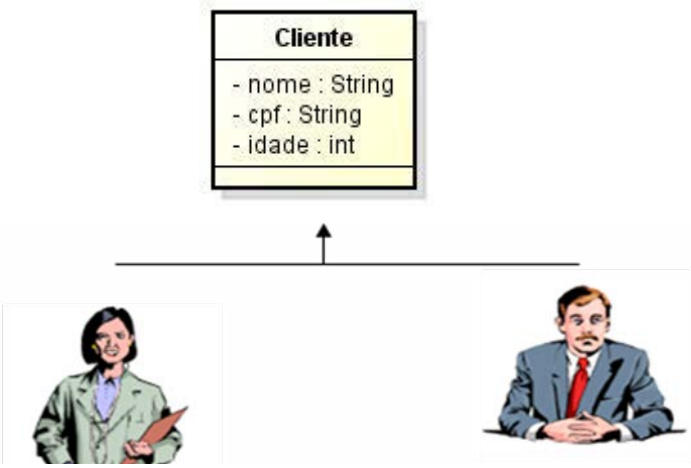
Classe: Pessoa	
Propriedades	**Comportamento**
Nome	Andar
Profissão	Correr
Data de Nascimento	Trabalhar
Altura	Chorar
Peso	Dançar

Classe: Pássaro	
Propriedades	**Comportamento**
Espécie	Andar
Cor das penas	Correr
Tamanho	Voar
Peso	Pousar

Estrutura de um objeto

Um objeto tem identificação, dados e comportamento, atributos e métodos.

Podemos dizer que objeto é espécie da classe, ou seja, uma instância da classe.



Cliente: clientemulher
nome = Fulana
cpf = 022.200.708-12
idade = 27

Cliente: clientehomem
nome = Ciclano
cpf = 022.200.708-55
idade = 32

Para finalizarmos as comparações entre Classe e Objetos temos os seguintes exemplos:

Classes	**Objetos**
Funcionário	Os funcionarios Pitoco Aguiar e José da Silva
Carro de passeio	O astra, vinho, placa CIL 8445
Nota Fiscal	A nota fiscal #29470 referente a um computador notbook

Em resumo, o objeto é definido pelo conjunto de valores dos seus atributos em determinado instante. O comportamento é definido pelo conjunto de métodos. Ao criar uma classe, ela passa a ser um tipo abstrato de dados que pode instanciar objetos. Para instanciar objetos usamos o operador new.

Uma coleção de carros pode ser representada por uma classe chamada Carro. Cada carro desta coleção é a "instância" (objeto) da classe Carro.

Atributos

Os atributos são as características dos objetos, o estado do objeto equivale aos valores de todos os atributos. No caso do exemplo acima os atributos são representado pelo motor, cor e marca.

Construindo a classe em Java

Um arquivo em Java precisa de uma classe pública com o mesmo nome do arquivo, abaixo um exemplo de como criar o arquivo.

```
public class <nome_da_classe> {
    <lista de atributos>
     <lista de métodos>
}
```

Atenção

Por padrão as classes e atributo em java devem seguir algumas regras:

- ✓ Somente letras, números e underline
- ✓ Não pode começar por números
- ✓ Não pode ter espaços em branco
- ✓ Não pode ser palavra reservada

- **Classes:** Possuem as iniciais maiúsculas e sem espaços ou caracteres especiais
- **Atributos:** São sempre em letras minúsculas

Tipos de dados em Java

Java possui 8 tipos primitivos que podem ser usados como tipos de atributos e que são eles:

- byte - 8 bits
- short - 16 bits
- int - 32 bits
- long - 64 bits
- float - ponto flutuante de 32 bits
- double - ponto flutuante de 64 bits
- char - Unicode de 16 bits
- boolean – true / false
- String - classe

Vejamos um exemplo de criação da classe carro e a instância da classe.


```
public class Carro {
    String motor;
    String cor;
    String marca;
}
```

```
public class TestarCarro{
    public static void main(String args[]){
        Carro c1 = new Carro();
        Carro c2 = new Carro();

        c1.motor = '2.0';
        c1.cor = 'Prata'
        c1.marca = 'Citroen';

        c1.motor = '1.6';
        c1.cor = 'Vermelho'
        c1.marca = 'Ford';
    }
}
```

Classes: Atributos (Propriedades) + Métodos (Comportamento)

Método é a implementação de uma operação. Os métodos expressam os comportamentos que todos os objetos dessa classe possuem.

tipo nome (parâmetros) {

instruções;

<return resposta>;

}

Set e Get

Servem como métodos de leitura/escrita aos atributos de classes.

Um método de leitura para um atributo deve ser chamado de getXxx (onde Xxx é o nome do atributo). Este método não recebe nada como parâmetro, e retorna o mesmo tipo do atributo.

Já um método de gravação deve ser chamado setXxx, não retorna nada (geralmente), e recebe como parâmetro o valor que deve ser armazenado no atributo.

Na programação orientada a objetos, os atributos da classe quase nunca estão visíveis para os usuários, é necessário criar um método público para que a aplicação possa acessá-lo. Temos basicamente 3 modificadores de acesso:

- **Private (-):** é o mais restritivo de todos, atributos e métodos com esse modificador são visíveis somente dentro da definição da própria classe, acessando-o diretamente ou através de uma instância da mesma classe.
- **Protected (#):** define que atributos e métodos somente podem ser acessados por subclasses da classe onde está sendo definido.
- **Public (+):** é o mais abrangente de todos os tipos de acesso, declara que elementos que o utilizam são acessíveis de qualquer classe Java.

Vejamos a implementação do código:

```
public class Carro {
    private String motor;
    private String cor;
    private String marca;

    public String getMotor(){
        return motor;
    }

    public void setMotor(String m){
        motor = m;
    }

    public String getCor(){
        return cor;
    }

    public void setCor(String c){
        cor = c;
    }

    public String getMarca(){
        return marca;
    }

    public void setMarca(String mc){
        marca = mc;
    }
}
```

Agora os atributos da classe Carro são privados, não é mais possível atribuir informações diretamente:

```
public class TestarCarro{
    public static void main(String args[]){
        Carro c1 = new Carro();
        Carro c2 = new Carro();

        c1.motor = '2.0';
        c1.cor = 'Prata'
        c1.marca = 'Citroen';

        c1.motor = '1.6';
        c1.cor = 'Vermelho'
        c1.marca = 'Ford';
    }
}
```

```
public class TestarCarro{
    public static void main(String args[]){
        Carro c1 = new Carro();
        Carro c2 = new Carro();

        c1.setMotor('2.0')
        c1.setCor('Prata')
        c1.setMarca('Citroen');

        c1.setMotor('1.6')
        c1.setCor('Vermelho')
        c1.setMarca('Ford');
    }
}
```

Os métodos também possuem padrão de nomenclatura. Os métodos possuem sempre a primeira inicial minúscula e as outras iniciais maiúsculas.

pegarInformacao(), executarComandoInicial()

Os métodos geralmente são ações que podem ser efetuadas entre os atributos do objeto.

Métodos que não retornam valores, apenas executam ações, são do tipo void.

Por exemplo, na classe Carro, podemos criar um método que imprima seus atributos na tela, ou como vimos anteriormente, mostre o estado do objeto.

```
public class Carro {
    private String motor;
    private String cor;
    private String marca;

    public String getMotor()..

    public void setMotor(String m)..

    public String getCor()..

    public void setCor(String c)..

    public String getMarca()..

    public void setMarca(String mc)..

    public void imprimeDados(){
        System.out.println("Motor: " + motor);
        System.out.println("Cor: " + cor);
        System.out.println("Marca: " + marca);
    }
}
```

Métodos - Procedimentos

Os métodos podem ou não assumir tipos de dados, caso não assumam, são chamados de procedimentos, pois executam um conjunto de instruções sem devolverem valor algum a quem os chamou. Um método sem tipo recebe em sua definição a palavra-chave void no lugar do tipo.

```
//Procedimento sem parâmetro
void frase() {
    System.out.println("Procedimento sem parâmetros");
}
```

Métodos - Funções

Quando os métodos assumem algum tipo, eles são chamados de funções e precisam do comando return para devolver o valor resultante da execução de suas instruções internas.

```
//Função sem parâmetro
String frase() {
    String mensagem = "Função sem parâmetro";
    return mensagem;
}
```

Métodos - Parâmetros

Os métodos podem receber dados para serem utilizados internamente, os quais são chamados de parâmetros ou de argumentos.

Quando os parâmetros são passados para os métodos, é criada uma cópia dos valores.

Podemos passar vários parâmetros para os métodos, inclusive de tipos diferentes.

```
//Procedimento com parâmetro
void numero (int n) {
     int resposta;
     resposta = n * 5;
    System.out.println(resposta);
}
```

```
//Função com parâmetro
int soma(int num1, int num2) {
    int resul;
    resul = n + m;
    return resul ;
}
```

Material Complementar

Livros Disponíveis na Biblioteca Virtual Universitária da Pearson Education

- Aprenda Programação Orientada a Objetos em 21 dias (Anthony Sintes)
- Java Como Programar, 8 edição (Paul Deitel e Harvey Deitel)
- Core Java – Volume 1 (Cay Horstman e Gary Cornell)

Conceitos de POO em inglês, site da SUN:

http://download-llnw.oracle.com/javase/tutorial/java/concepts/index.html

Referências

SINTES, Tony. (2002) **Aprenda Programação Orientada a Objetos em 21 dias**. 1 ed. São Paulo: Pearson Education do Brasil, 2002, v. 1.

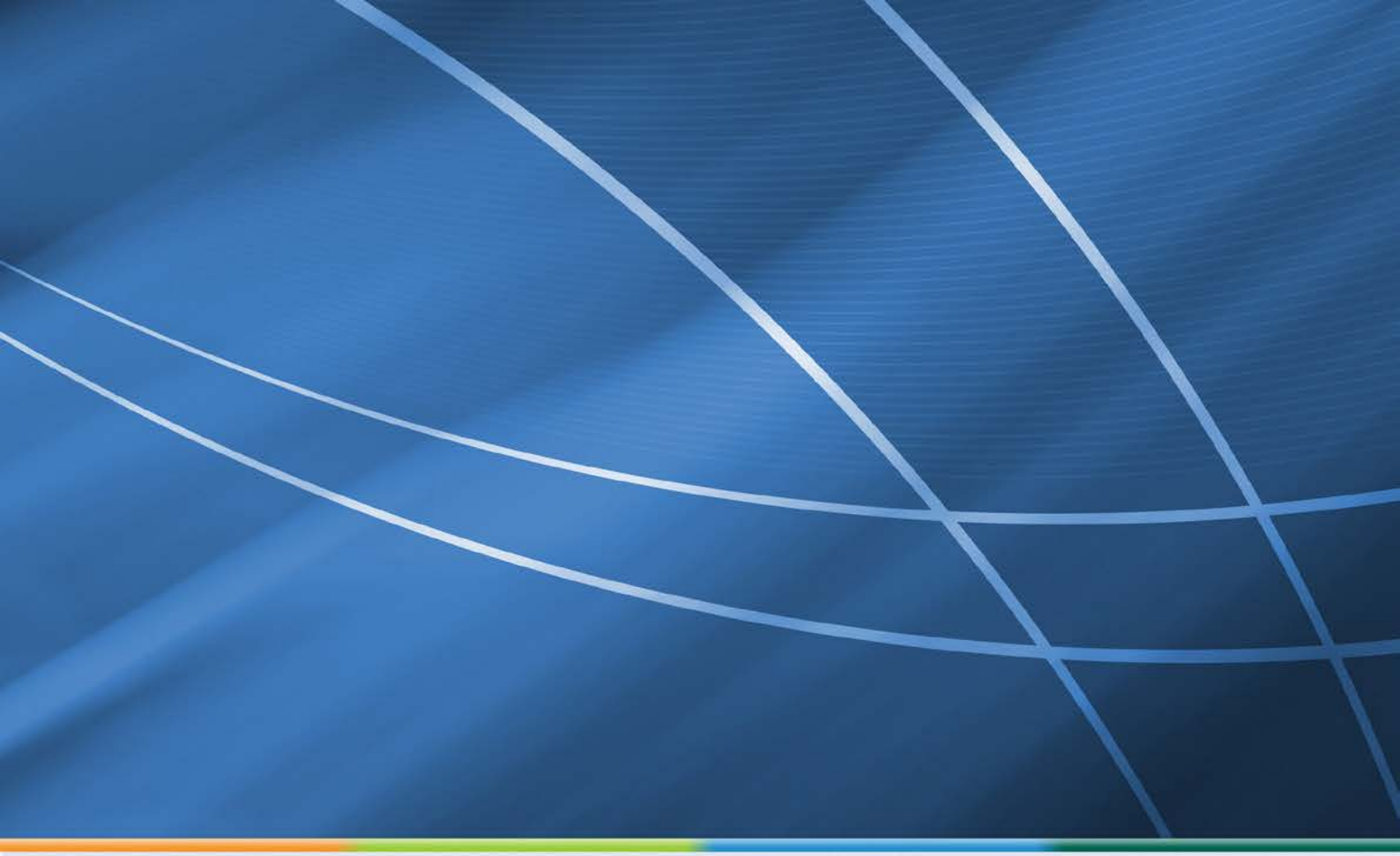
DEITEL, P.; DEITEL, H. (2010) **Java Como Programar,** 8 ed. São Paulo: Pearson Education do Brasil, 2010.

HORSTMANN, C.S.; CORNELL, G. (2010) **Core Java**. 8 ed. São Paulo: Pearson Education do Brasil, 2010, v. 1.

Anotações

Educação a Distância
Cruzeiro do Sul Educacional
Campus Virtual

www.cruzeirodosulvirtual.com.br
Campus Liberdade
Rua Galvão Bueno, 868
CEP 01506-000
São Paulo SP Brasil
Tel: (55 11) 3385-3000



Universidade Cruzeiro do Sul

UNICID Universidade Cidade de S. Paulo

UNIFRAN Universidade de Franca

UDF Centro Universitário

Módulo Centro Universitário