PROGRAMAÇÃO WEB

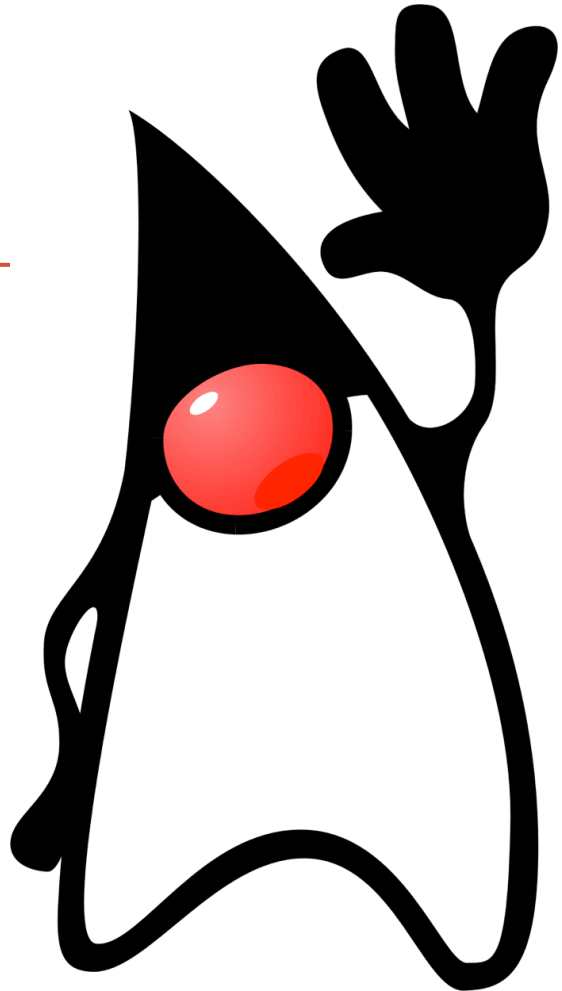
- Programação Orientada a Objetos
- Java e JSP (Java Server Pages) Essencial

Prof. Luiz Carlos Querino Filho

luiz.querino@fatec.sp.gov.br

Fatec Garça – 2017

Parte 03



POO (Programação Orientada a Objetos) Essencial

- Java é uma linguagem de programação orientada a objetos (POO).
- Os elementos de um sistema em POO são modelados como **objetos**.
- Os objetos possuem **características** (também chamados de **atributos**, propriedades ou campos) e **comportamento** (também conhecidos como **métodos** ou ações)
- Os “moldes” dos objetos são as **classes**.
- Um objeto é, portanto, uma “cópia” de uma classe com os seu próprios atributos definidos. Chamamos isso de **instância** de uma classe.

Classes e Objetos

- As classes representam na orientação a objetos os “moldes” dos objetos.
- Este molde traz as características (**atributos**) e comportamentos (**métodos**) que os objetos derivados desta classe possuirão, sem, contudo, possuírem valores para eles.
- Por exemplo, uma classe carro traz já os atributos cor, ano e modelo, mas sem conteúdo.
- As informações de cor, ano e modelo serão “preenchidas” quando do seu instanciamento.
- Os métodos, assim com as funções, podem ou não receber parâmetros, assim como podem retornar ou não um valor.

{
// I
// Code
// Java
}

Classes e Objetos

OBJETO CARRO	
Modelo	Fiat Uno
Cor	Vermelho
QuantidadeDeRodas	4
CapacidadePortaMalas	290 litros
PossuiAirBag	Não

Este objeto já teria os seguintes comportamentos/métodos:

OBJETO CARRO
Transportar()
LigarDesembassador()

De programação estruturada para orientada a objetos: fazendo a transição

- Na programação estruturada (também chamada procedural) as ações de um programa são realizadas por **funções**. Os seus dados ficam armazenados em **variáveis**.
- As funções "atuam" nos dados das variáveis, mas não há necessariamente uma correlação direta entre elas.
- Na **POO** (programação orientada a objetos), os **dados são parte dos objetos, e as funções também**! No caso da POO, as variáveis são os atributos e as funções são os métodos.
- Neste caso, existem sempre uma correlação direta entre os dados e os objetos.

De programação estruturada para orientada a objetos: fazendo a transição

- **Na programação estruturada**, o fluxo de execução de um programa é determinado pela **sequência de execução de funções** – uma chamando a outra.
- **Já na POO**, o fluxo de execução de um programa é determinado pela **interação entre os objetos**, de uma forma mais semelhante ao mundo real.
- Imagine um programa feito em C (uma linguagem estruturada) para gerenciar um cadastro de clientes.
- Você poderia criar uma `struct` C para armazenar todos os dados do cliente…
- Mas na hora de programar as ações relativas ao cliente, você criaria funções que atuariam nesta `struct`. Porém, a princípio, as funções ficariam “dispersas”, sem relação direta com a variável.

De programação estruturada para orientada a objetos: fazendo a transição

- Suponha que você pudesse, além dos dados, agregar à `struct` todas as funções relacionadas a um cliente.
- Você teria, então, uma "unidade ativa" no seu sistema: esta `struct` teria características e comportamento…seria, então, um **objeto**!
- Mas isso não é possível em uma linguagem estruturada como C…
- Porém, é totalmente viável (e desejável) quando se usa uma linguagem de programação orientada a objetos, como **Java**.

USANDO JAVA PARA CRIAR APLICATIVOS WEB

Unindo as vantagens da Programação Orientada a Objetos com a Web

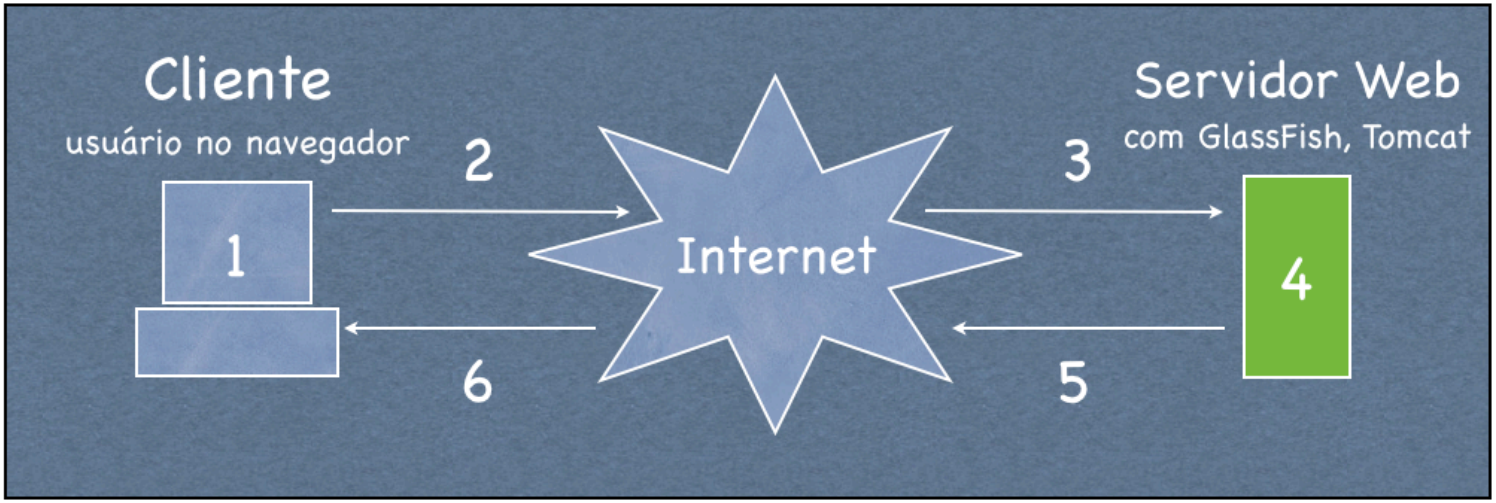
Conteúdo estático x Conteúdo dinâmico

- PÁGINAS COM CONTEÚDO ESTÁTICO
  - Não interagem com o usuário
  - Seu conteúdo não muda automaticamente
  - Escritas essencialmente em HTML (HyperText Markup Language), que não é uma linguagem de programação
  - Criadas e atualizadas manualmente em programas editores
  - NINGUÉM volta a uma página se seu conteúdo não for atualizado!!!
- PÁGINAS COM CONTEÚDO DINÂMICO
  - Possibilitam interação direta do usuário
  - Seu conteúdo pode ser modificado de forma automatizada
  - Escritas em HTML, mas com uso de linguagens de programação
  - São geradas basicamente pelo uso conjunto do HTML e de linguagens de programação.

Conteúdo Dinâmico

- Client-side: são linguagens executadas no lado do cliente (usuário no navegador)
  - Geralmente usadas para validações e efeitos
  - Exemplo: JavaScript
  - Não possui acesso direto ao servidor
- Server-side: são linguagens executadas no servidor Web
  - Geram o conteúdo da página previamente e enviam ao cliente
  - Possibilitam o acesso a recursos do servidor, como banco de dados
  - Exemplo: **Java**, php, ASP

Como funciona uma linguagem Server-Side



- (1) O usuário digita o endereço do servidor ou página desejada
- (2 e 3) A requisição é enviada pela Internet até o servidor Web
- (4) O servidor Web recebe a requisição e processa (executa) os comando no código Java
- (5 e 6) O resultado gerado (em HTML) é enviado de volta ao usuário pela internet.
- NO NAVEGADOR DO USUÁRIO É EXIBIDA A RESPOSTA EM HTML. O USUÁRIO NÃO PODE, PORTANTO, VISUALIZAR O CÓDIGO JAVA.

Java

- Linguagem de programação orientada a objetos
- Multiplataforma
- Criada pela Sun Microsystems, atualmente parte da Oracle
- É a linguagem de programação mais usada no mundo (TIOBE Index – Agosto de 2017)

TIOBE Index

Aug 2017	Aug 2016	Change	Programming Language	Ratings	Change
1	1		Java	12.961%	-6.05%
2	2		C	6.477%	-4.83%
3	3		C++	5.550%	-0.25%
4	4		C#	4.195%	-0.71%
5	5		Python	3.692%	-0.71%
6	8	▲	Visual Basic .NET	2.569%	+0.05%
7	6	▼	PHP	2.293%	-0.88%
8	7	▼	JavaScript	2.098%	-0.61%
9	9		Perl	1.995%	-0.52%
10	12	▲	Ruby	1.965%	-0.31%

Java

- A plataforma Java executa em diversos dispositivos, de computadores a dispositivos móveis, passando por servidores web e *set-top-boxes*.
- Graças a JVM (Java Virtual Machine, máquina virtual Java) um aplicativo Java pode executar com pouca (ou nenhuma) modificação em vários aparelhos distintos.
- Existem duas divisões no “Java tradicional”:
  - Java SE (Standard Edition): voltada principalmente à criação de aplicativos desktop tradicionais
  - Java EE (Enterprise Edition): destinada ao desenvolvimento de aplicativos empresariais e para a Web. É um complemento da Java SE.

Java EE (Enterprise Edition)

- Atualmente na versão 7.
- Abrange uma série de especificações e tecnologias para a criação de aplicações Java empresariais:
  - Persistência de dados
  - Transações
  - Segurança
  - Web Services (JAX-WS e JAX-RS)
  - Aplicações Web usando as tecnologias:
    - Servlets
    - Java Server Pages (JSP)
    - Java Server Faces (JSF)
    - Context and Dependency Injection
    - etc. etc. etc.

O Código “produzido” por um servidor de aplicações é HTML!

- Por isso, é necessário pelo menos um conhecimento básico desta linguagem.
- HTML (HyperText Markup Language): Linguagem de Marcação de Hipertexto
- HTML não é uma linguagem de programação
- HTML é uma linguagem de estruturação. Ela determina como deve ser a estrutura de um documento
- Além do arquivo HTML, uma página também pode possuir imagens, sons, animações e vídeos, todos associados ao HTML.
- Folhas de estilo (CSS: Cascading Style Sheets) são utilizadas na definição de padrões de formatação usados em arquivos HTML.

O que Java herdou das linguagens estruturadas (C, especificamente)

- **CASE SENSITIVE**! Diferencia letras maiúsculas de minúsculas em **TUDO**!
- Tipos primitivos (ou básicos):
  - `int` (padrão para números inteiros)
  - `float`
  - `double` (padrão para números decimais)
  - `char`
  - `void`
- Estruturas de repetição (sintaxe idêntica à C):
  - `for`
  - `while`
  - `do...while`

O que Java herdou das linguagens estruturadas (C, especificamente)

- Operadores lógicos e booleanos:
  - `=` atribuição
  - `==` igualdade
  - `!=` diferença
  - `>` maior que
  - `<` menor que
  - `>=` maior ou igual
  - `<=` menor ou igual
  - `&&` and
  - `||` or
  - `!` not

O que Java herdou das linguagens estruturadas (C, especificamente)

- Operadores matemáticos:
  - `+` adição
  - `-` subtração
  - `/` divisão
  - `*` multiplicação
  - `%` módulo (resto da divisão inteira)
  - `++` incremento de 1
  - `--` decremento de 1

O que Java herdou das linguagens estruturadas (C, especificamente)

- Estruturas de decisão (sintaxe idêntica à C):
  - `if…else`
  - `switch....case`
  - Operador ternário: `Condição ? Se verdadeiro : Se falso`
- Índice de vetores:
  - Assim como em C, são especificados entre colchetes logo após o nome da variável vetor.
  - Iniciam em 0 e terminam em **n** – 1 (onde **n** é o tamanho do vetor)
  - Para definir um valor ao terceiro elemento do vetor de inteiros denominado `numeros`:
    - `numeros[2] = 22;`

O que Java traz de novidade

- Strings:
  - Em Java, não é necessário usar vetores de char. Podemos declarar objetos da classe **String** e usá-los como variáveis:
  - `String minhaString = "Isso não era possível em C…";`
  - Como se trata de um objeto da classe **String**, `minhaString` possui uma série de métodos que podem ser usados colocando um ponto após seu nome:
  - `String tudoMaiusculo = minhaString.toUpperCase();`
  - **As Strings podem ser concatenadas usando o sinal +**
- Um novo tipo primitivo
  - **`boolean`**: Pode possuir apenas valores true ou false

O que Java traz de novidade

- `switch....case`
  - Na versão 7 do JDK, Strings podem ser usados nos testes de um switch…case.
- Declaração de vetores
  - `int[] numeros = new int[100];`
  - A linha acima declara um vetor chamado numeros que vai armazenar 100 números inteiros.
  - Seu primeiro elemento pode ser acessado em `numeros[0]`.
  - O último elemento será acessado em `numeros[99]`.

Java Essencial: Conversão de Tipos

- Para converter valores String para seus equivalentes numéricos:

```
String numTexto = “22”;
int numeroInt = Integer.parseInt(numTexto);
String altura = “1.83”;
double altNum = Double.parseDouble(altura);
```

- Para converter um número **`int`** ou **`double`** para String:

```
double saldo = 456.75;
String saldoString = String.valueOf(saldo);
```

Obtendo valores de campos de formulários no arquivo JSP

- O objeto **`request`** existe no Java Server Pages para que possamos obter facilmente os dados passados por um formulário.
- Para isso, usamos o seu método **`getParameter()`**. Passamos como argumento ao **`getParameter()`** uma **`String`** contendo o nome do campo (atributo **name** do `<input…>`).
- Este método retorna o valor colocado naquelr campo como uma **`String`**. Por exemplo, para obter o valor colocado em um campo chamado "**cep**" existente em um formulário, e guardá-lo em uma variável **`String`**, usamos:
- **`String cep = request.getParameter("cep");`**
- Valores que serão usados numericamente deverão ser convertidos, usando **Integer.parseInt** ou **`Double.parseDouble`**.

AVISO!!!

- Por enquanto, nos nossos programas Java Web, não criaremos classes.
- Usaremos apenas objetos e classes **JÁ EXISTENTES NO JAVA**.
- À medida que você evolui na disciplina de POO, passaremos a começar a ver exemplos em Programação Web onde criaremos e usaremos as nossas próprias classes.
- Exemplo de classes úteis existentes em Java:
  - **String**: para armazenar cadeias de caracteres
  - **Random**: para gerar números aleatórios
  - **Integer** e **Double**: classes para os tipos básicos `int` e `double`, respectivamente. Um dos usos para estas classes é conversões de valores (de **String** para `int` ou `double`, por exemplo)

Exemplo de uso de uma classe em Java

- A classe Java **Random** é usada para sortear números aleatórios.
- Para usá-la, precisamos instanciá-la, criando um **objeto** a partir dela.
- A obtenção de um objeto em Java envolve duas etapas:
  - **Declarar um objeto**: especificar seu nome e de qual classe ele é. Isso é feito da mesma forma que declaramos variáveis.
  - **Inicializar este objeto**: fazemos isso usando um comando do Java chamado **new** e executando o método **construtor** da classe.
- O método **construtor** é um comando especial, que pode ou não receber parâmetros, e que possui o mesmo nome da classe.

Exemplo de uso de uma classe em Java

- No exemplo seguinte, estamos instanciando a classe **Random** em um objeto chamado `random` e executando um de seus métodos (`nextInt`), que retorna um número inteiro aleatório entre 0 e 100:

```
Random random = new Random();
int numero = random.nextInt(101);
out.println("O número sorteado foi: " +
numero);
```

Exercícios

- **Calculadora básica**: soma, subtração, multiplicação, divisão, raiz quadrada e potência. Pesquise na Internet como calcular a raiz quadrada e a potência em Java.
- **MegaSena**: gere seis números aleatórios (não repetidos e em ordem) entre 1 e 60 e apresente ao usuário.
- **MegaSenaPlus**: permita que o usuário escolha uma quantidade de apostas (conjuntos de seis números). Gere e exiba os números para as postas na tela. IMPEÇA que o usuário solicite MAIS DE 10 CONJUNTOS DE NÚMEROS.

Exercícios

- **CalculadoraPlus**: para que o usuário possa escolher qual operação deseja, utilize o campo de formulário **<select>**:

```
<select name="escolha">
      <option>Opção 1</option>
      <option>Opção 2</option>
      …
</select>
```

A opção escolhida vai ao JSP como valor do campo, em formato **String**!

DICA: comparando Strings Java

- Use um **`if`** no código Java para saber qual opção o usuário escolheu e qual resultado deverá ser exibido.
- Para comparar se uma **String** é igual a outra, use o método **`equals()`**, existente em todos os objetos dessa classe:

```
String formaDeEnvio = request.getParameter("envio");
if (formaDeEnvio.equals("sedex")) {
        // cálculo de envio da encomenda por Sedex
} else if (formaDeEnvio.equals("pac")) {
        // cálculo de envio da encomenda por PAC
}
```

Exercício - HISTOGRAMA

- Crie um aplicativo Java Web que realize 100 sorteios de números entre 1 e 10.
- Armazene em um vetor de dez posições quantas vezes cada um dos números foi sorteado.
- Por exemplo, se o número sorteado for 2, acrescente 1 na segunda posição do vetor.
- Após realizar todos os sorteios, apresente o resultado como um “gráfico” da seguinte forma:

```
1: ###       (neste caso, o número 1 foi sorteado três vezes)
2: #####     (o número 2 foi sorteado cinco vezes)
3: #         (1 vez)
...
10: ####### (7 vezes)
```

Exercício - HISTOGRAMA

- DICA 1: Use uma fonte de largura fixa para exibir o texto do histograma. Ex.:
- `<div style="font-family: 'Consolas'"></div>`
- DICA 2: Tenha pena da máquina virtual Java e do GlassFish…**não instancie 100 objetos desnecessários**!

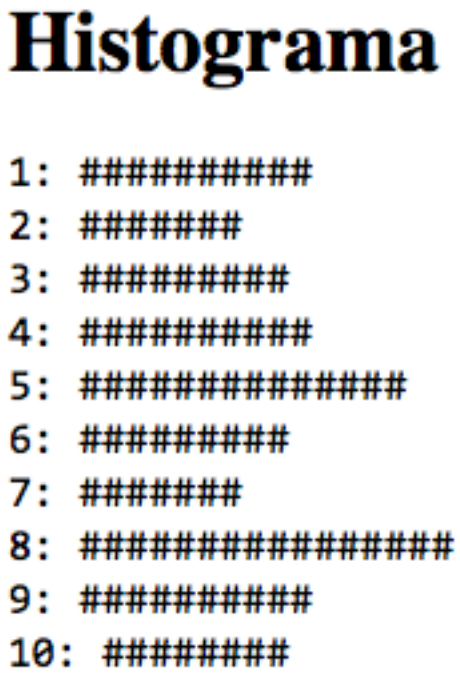
```
<%
    for (int i = 0; i < 100; i++) {
        Random random = new Random();
        random.nextInt(10);
    }
%>
```

```
<%
    Random random = new Random();

    for (int i = 0; i < 100; i++) {
        random.nextInt(10);
    }
%>
```

Exercício - HISTOGRAMA

- Resultado:



BIBLIOGRAFIA

- HEFFELFINGER, D. W. Java EE 6 Development with NetBeans 7. Birmingham: Packt Publishing, 2011.
- PANDA, D.; RAHMAN, R.; LANE, D. EJB 3 in Action. Greenwich: Manning Publications, 2007.
- BASHAM, Bryan. Use A Cabeça! Servlets e JSP. Alta Books, 2008.
- KURNIAWAN, B. Java para Web com Servlets, JSP e EJB. São Paulo: Ciência Moderna, 2002.