# **Plataforma Java**

Java é uma linguagem de programação **orientada a objetos** desenvolvida na década de 90 por uma equipe de programadores chefiada por James Gosling, na empresa Sun Microsystems

## Sintaxe Java

Quase tudo em Java é baseado em objetos. Basicamente, apenas os tipos nativos são considerados de forma diferente, mas para cada tipo nativo existe uma ou mais classes **Wrapper**.

As linguagens Java e JavaScript descendem da linguagem C; por esse motivo apresentam muitas semelhanças, mas também reservam algumas diferenças.

Vamos observar, a seguir, a estrutura geral de um programa Java.

public class Exemplo001 {

public static void main(String args[ ]) {

System.out.println("Alo Mundo");

}

}

As estruturas de decisão e repetição também são equivalentes àquelas encontradas no JavaScript, com algumas diferenças no uso do for, pelo fato de o Java ser fortemente tipado.

Vamos observar um exemplo de uso do for a seguir.

public class Exemplo002 {

public static void main(String[] args) {

// Calculo do valor médio da sequencia y = f(x) = x \* x

// Media = Somatorio dos valores / quantidade

// Limites 1 a 5

double soma = 0.0;

for(int x=1; x<=5; x++)

soma += Math.pow(x, 2);

// eleva x a potência 2 e acumula

System.out.println(soma/5);

}

}

Na orientação a objetos, as classes definem estruturas fechadas, nas quais o acesso a seus atributos e métodos deve ser controlado, e para tal iremos contar com três níveis de acesso:

### **Público (public)**

### Permite que qualquer um acesse o atributo ou método;

### **Privado (private)**

### Não permite acessos externos, sendo utilizado apenas na programação interna da classe;

### **Protegido (protected)**

### Permite a utilização na classe e nos descendentes, mas não permite acessos externos.

Aqui, quando falamos de descendentes, estamos nos referindo a uma característica importante da orientação a objetos, que é a herança, onde utilizamos a palavra reservada extends, como pode ser observado a seguir.

public class Profissional extends Pessoa{

public String profissao;

@Override

public void exibir() {

super.exibir();

// Chama o exibir de Pessoa, imprimindo nome e telefone

System.out.println("\tTrabalha como "+profissao);

// Complementa a informação acerca da profissão

}

}

# **Ambiente Servidor**

Aprendemos a criar toda a interface com tecnologias cliente, inclusive a criação de formulários, possibilitando o envio de dados pelos métodos GET ou POST do HTTP.

### **Mas para quem enviamos os dados?**

Os dados são enviados para alguma tecnologia servidora, através de uma requisição HTTP, a qual irá iniciar algum processamento no servidor e este, ao final, retornará uma **resposta**, normalmente em **HTML** ou **XML**.

No ambiente Java, um objeto da classe **HttpServletRequest** irá:

* **Tratar os dados da requisição HTTP -** normalmente denominado **request**, permitindo efetuar qualquer tipo de processamento com estes dados, como persistência e validações de segurança.
* **Controlar Os dados de retorno para o usuário -** costuma ser denominado response, o qual permitirá escrever o conteúdo HTML ou XML de saída, além de outros elementos, como cookies

## Web Server Tomca

O servidor **Tomcat** é um projeto da Apache Software Foundation para a implementação com código aberto de tecnologias como Java Server Pages (JSP), Java Servlet, Java WebSocket e Java Expression Languag

É possível alterar várias características do Tomcat editando o arquivo **server.xml**.

#### Exemplo

Por exemplo, o servidor Tomcat executa, por padrão, na porta **8080**, mas podemos modificar esta porta procurando e alterando a ocorrência deste valor no arquivo.

# **Servlet**

A tecnologia **Servlet** foi criada com o intuito de se tornar uma solução genérica para a criação de aplicativos hospedados em servidores

Tudo que precisamos fazer é criar um descendente da classe **HttpServlet**, herdando toda a integração com o ambiente já existente, e alterar os métodos **doGet** e **doPost** para personalizar as respostas.

Response

response.setContentType("text/html;charset=UTF-8");

try (PrintWriter out = response.getWriter()) {

Request

int a = new Integer(request.getParameter("a")), b = a;

# **Java Server Pages**

A criação de Servlets seria suficiente para prover as necessidades de todo e qualquer aplicativo Web, porém a construção de páginas através de código direto pode se tornar desconfortável para a maioria dos designers.

Uma solução para isso foi a definição de um novo modelo de programação, no qual códigos Java são escritos dentro do conteúdo HTML ou XML, permitindo a edição visual através de ferramentas como o DreamWeaver da Adobe.

#### Comentário

1. Qual tipo de classe permite manter valores entre chamadas sucessivas no ambiente Java Web?

a) HttpServletRequest

b) HttpServletResponse

c) HttpServlet

d) HttpListener

e) HttpSession

2. Implemente o método processRequest, de um Servlet criado a partir do NetBeans para receber dois números via HTTP e retornar a soma entre eles.

response.setContentType("text/html;charset=UTF-8");

try (PrintWriter out = response.getWriter()) {

int a = new Integer(request.getParameter("a"));

int b = new Integer(request.getParameter("b"));

out.println("");

out.println("A soma de "+a+" e "+b+" será "+(a+b));

out.println("</body></html>");

}

3. Crie uma página JSP para apresentar os números ímpares entre 1 e 100 no formato de lista HTML.

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<body>

<u>

<%

for(int i=1; i<=100; i++)

if(i%2!=0)

out.println("<li>"+i+"</li>");

%>

</ull>

</body>

</html>

|  |  |  |
| --- | --- | --- |
| |  | | --- | | Qual será a saída obtida pelo seguinte trecho com sintaxe Java?  int a = 1, b=100; while( ++a < 5 )   b = b / a;  System.out.println(b); | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| Certo |  | 4 |
|  |  | 12,5 |
|  |  | 96 |
|  |  | 4,17 |
|  |  | 25 |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ é a tecnologia que gerencia a memória alocada para um programa, a fim de liberar objetos que não estão sendo utilizados. Complete a questão escolhendo um dos itens a sequir: | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| Certo |  | Garbage Collector |
|  |  | Java Collector |
|  |  | Java Virtual Machine |
|  |  | Garbage Drunk |
|  |  | Java Clean |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | "Uma (I) nada mais é do que uma mensagem que aparece na barra de status Android. Seu objetivo é notificar ao usuário sobre algum evento, seja através de um ícone na barra de status, através de vibração, som ou acender de luzes do dispositivo. É muito comum seu uso quando temos uma aplicação que roda em segundo plano, como, por exemplo, um aplicativo de mensagem instantânea, e não podemos ser interrompidos em uma aplicação que roda em primeiro plano, como, por exemplo, uma ligação telefônica."  (I) na sentença acima explica o conceito de: (assinale a alternativa correta) | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| Certo |  | Notificação Android |
|  |  | Text Box |
|  |  | Classe |
|  |  | Caixa de texto |
|  |  | Chamada Android |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | No que tange ao conceito da Classe Toast, leia as asserções abaixo e, a seguir, assinale a alternativa correta:  I. A classe Toast (android.widget.Toast) é muito utilizada quando desejamos exibir mensagens de alertas para o usuário.  II. A classe Toast (android.widget.Toast) exibe uma pequena tela, que pode ser personalizada, sem caráter permanente, ou seja, desaparecendo sem qualquer tipo de intervenção do usuário.  III. A classe Toast (android.widget.Toast) exibe um pop-up de caráter permanente e que necessita de interação do usuário para ser finalizada (fechada).   IV. Para implementarmos este tipo de mensagem, é necessário obter uma instância de objeto Toast através do método estático, desta mesma classe, makeText(), que possui três parâmetros: Toast.makeText (contexto, texto, duração) | | |
|  | | |

|  |  |  |
| --- | --- | --- |
|  |  | Somente as asserções II, III e IV estão corretas |
|  |  | Somente as asserções III e IV estão corretas |
| Certo |  | Somente as asserções I, II e IV estão corretas |
|  |  | Somente as asserções I, III e IV estão corretas |
|  |  | Somente as asserções I, II e III estão corretas |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | Em termos de herança, em determinadas situações é necessário acessar um método do pai a partir do método de mesmo nome do filho, quando tratamos de uma operação sobrescrita no processo de polimorfismo. Qual palavra reservada do Java permite este acesso? | | |
|  | | |

|  |  |  |
| --- | --- | --- |
|  |  | new |
|  |  | implements |
|  |  | this |
| Certo |  | super |
|  |  | extends |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | A Classe AlertDialog tem como objetivo exibir uma caixa de dialogo (Android *Dialog*) para o usuário no Android. Ela possui vários métodos, dentre os quais poderemos observar através das alternativas subsequentes, exceto (ou seja, assinale a alternativa incorreta): | | |
|  | | |

|  |  |  |
| --- | --- | --- |
|  |  | setMessage: define a mensagem de nossa AlertDialog |
|  |  | setPositiveButton: define um botão para aceitar e conntinuar uma ação (Exemplo: OK) |
| Certo |  | setNegativeButton: exibe a AlertDialog |
|  |  | setTitle: define o título de nossa AlertDialog |
|  |  | setNeutralButton: define um botão que nem cancela e nem define uma ação (Exemplo: Notifique-me mais tarde) |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | A classe Toast(android.widget.Toast) é muito utilizada quando desejamos exibir mensagens de alertas para o usuário. Ela exibe uma pequena tela, que pode ser personalizada, sem caráter permanente, ou seja, desaparecendo sem qualquer tipo de intervenção do usuário. Para implementarmos este tipo de mensagem, é necessário obter uma instância de objeto Toast através do método estático, desta mesma classe, makeText(), que possui três parâmetros, demonstrados a seguir:  **Toast.makeText (contexto, texto, duração)**    Dada a explicação supracitada, analise a explicação de cada um dos parâmetros citados e, após, assinale a alternativa correta:  I. No parâmetro **contexto**, definimos os botões do aplicativo a serem exibidos na mensagem.  II. No parâmetro **texto**, definimos o texto a ser exibido na mensagem.  III. No parâmetro **duração**, podemos definir o tempo de duração que a mensagem será exibida. A própria classe Toast nos oferece constantes para definirmos este tempo de duração, sendo elas Toast.LENGTH\_SHORT e Toast.LENGTH\_LONG. | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| Certo |  | Somente as afirmações II e III estão corretas |
|  |  | Somente a afirmação II está correta |
|  |  | Somente as afirmações I e III estão corretas |
|  |  | Somente a afirmação I está correta |
|  |  | Somente as afirmações I e II estão corretas |