RaiaDrogasil S.A.

Gente,
Saúde e
Bem-estar.

Linguagem Java - Fundamentos

Temas Abordados

- Workshop Java
- Fundamentos
- Estruturas de Controle
- Classes e Métodos
- Arrays e Collections
- Orientação a Objetos
- Tratamento de Erros
- Generics
- Banco de Dados com JDBC

História

História

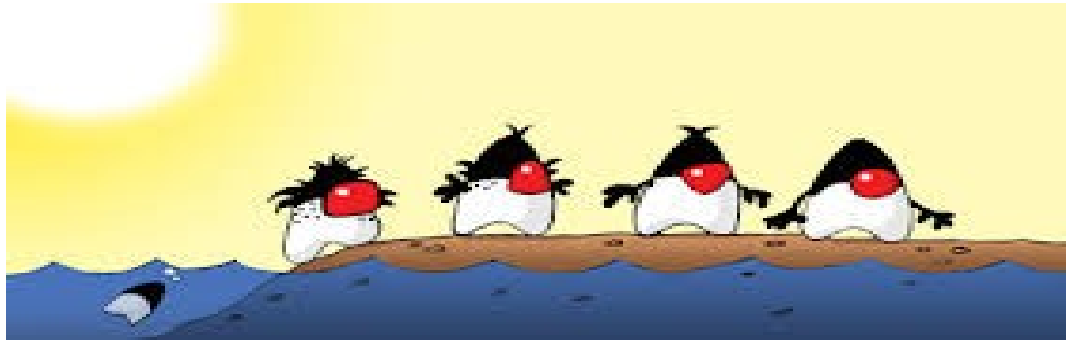
- Historia do Java
  - O que é Java?
  - Como Java evolouiu
  - Como Java está organizado
  - Qual a Filosofia do Java?

História - O que é Java

- Uma linguagem Orientada a Objetos;
- Fortemente tipada;
- Ambiente alvo pequenos dispositivos (videocassetes, televisores e aparelhos de TV a cabo);
- Executada em uma máquina virtual (JVM);
- Criada pelo Green Team da Sun Microsystens;
- Lançada oficialmente em 1996.

História - Como o Java evoluiu

- Com o buuumm da Internet a Sun percebeu que poderia levar o conceito de máquina virtual para possibilitar a execução de aplicações gráficas no navegador.
- Em pouco tempo percebeu-se a robustez da ferramente para execução em grandes servidores e para computação distribuída.
- Hoje esse é o grande mercado do Java.
- Grandes aplicações para servidores, sistemas web e aplicações distribuídas.



História - Como o java está organizado?

Java SE

- JVM = apenas a virtual machine, esse download não existe, ela sempre vem acompanhada;
- JRE = Java Runtime Environment, ambiente de execução Java, formado pela JVM e bibliotecas;
- JDK = Java Development Kit, ferramentas de desenvolvimento (compilador, ferramentas de monitoramento, criptografia, etc.)

História - Como o java está organizado?

Java EE

- *JDBC* (Java Database Connectivity), utilizado no acesso a bancos de dados;
- *Servlets*, são utilizados para o desenvolvimento de aplicações Web. Contém uma API que abstrai e disponibiliza os recursos do servidor Web;
- *JSP* (Java Server Pages), uma especialização do servlet que permite desenvolvimento de páginas web;
- *JTA* (Java Transaction API), é uma API que padroniza o tratamento de transações dentro de uma aplicação Java;
- *EJBs* (Enterprise Java Beans), utilizados no desenvolvimento de componentes de software. Concentram as necessidades do negócio do cliente, deixando questões de infraestrutura, segurança, disponibilidade e escalabilidade são responsabilidade do servidor de aplicações.

História - Como o java está organizado?

Java EE

- *JCA* (Java Connector Architecture), é uma API que padroniza a ligação a aplicações legadas.
- *JPA* (Java Persistence API), é uma API de acesso a banco de dados através de mapeamento Objeto/Relacional dos Enterprise Java Beans.
- *JMS* (Java Message Service), é uma API orientada a mensagens. Através dela é possível realizar a comunicação de forma assíncrona entre duas ou mais aplicações.
- *JSF* (JavaServer Faces), é uma especificação Java para a construção de interfaces de usuário baseadas em componentes para aplicações web.

História - Qual a Filosofia do Java?

Java foi criada com a filosofia de escreva uma vez e execute em qualquer lugar.

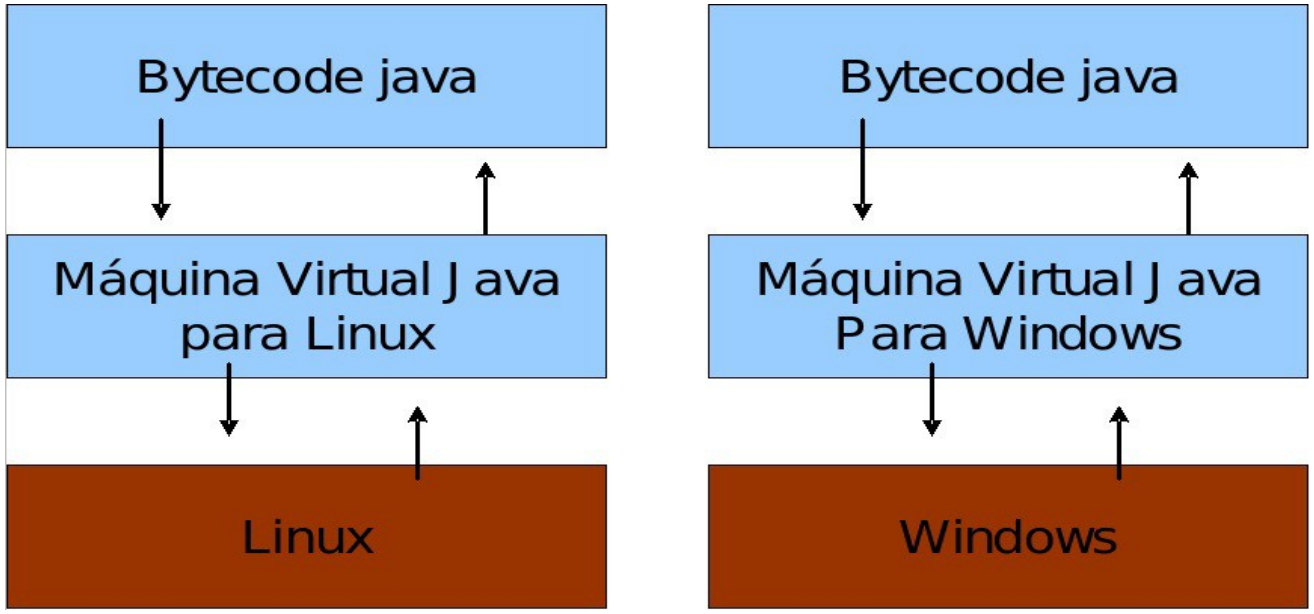
Fazendo uma comparação com as ferramentas de desenvolvimento “tradicionais” concluímos que isso se dá da seguinte forma:

Modelo Tradicional:

A compilação é feita diretamente do código fonte para o binário otimizado para a plataforma. Se for alterado o SO, o código precisa ser no mínimo recompilado, em algumas situações, modificado e recompilado.

História - Qual a Filosofia do Java?

Em Java

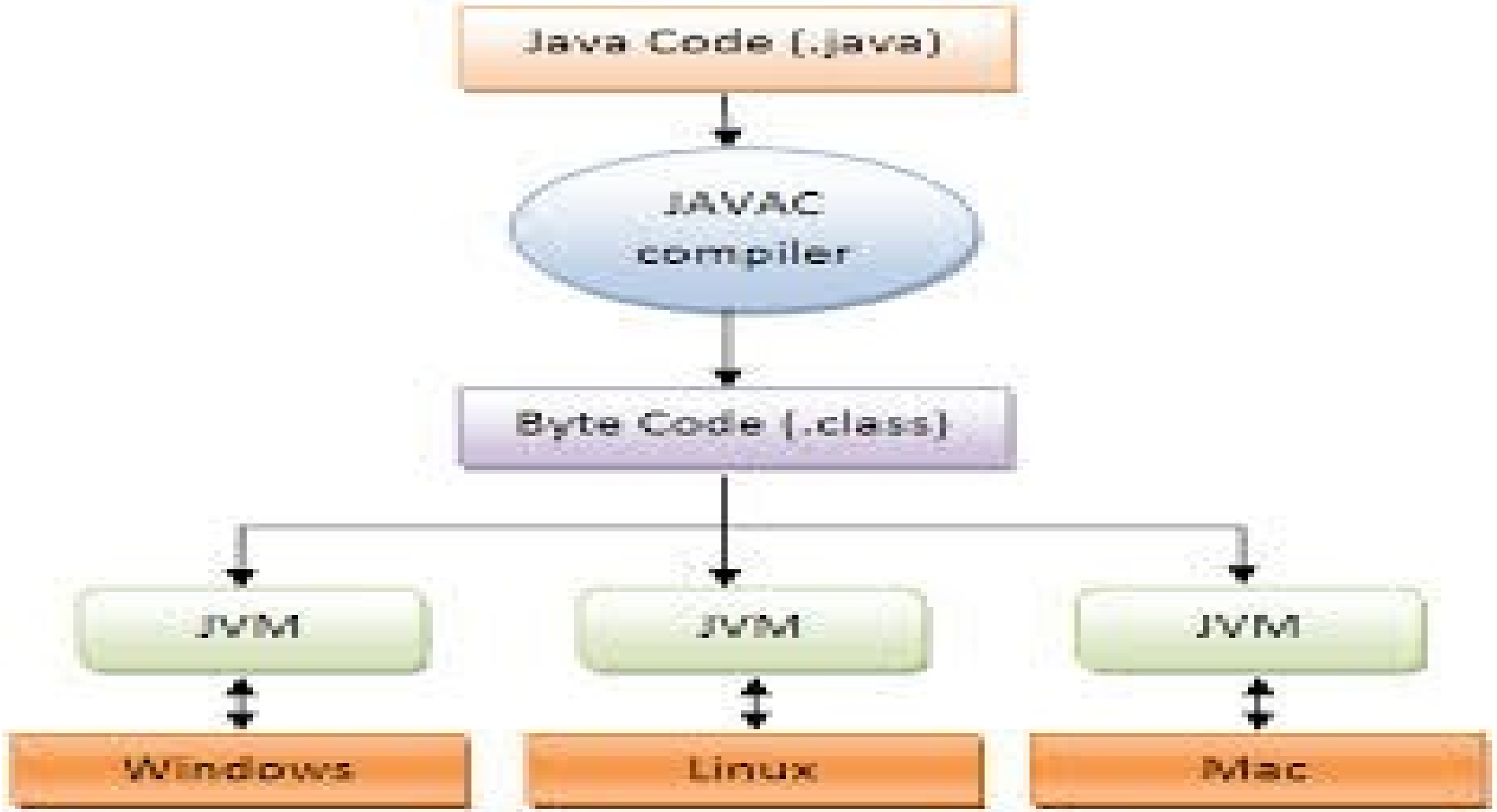


O código fonte não é compilado diretamente para o código binário. Ao invés disso, é gerado um código intermediário : o bytecode.

A JVM interpreta esse bytecode e gera o código binário, em tempo real, que será executado pelo SO.

História - Qual a Filosofia do Java?

Máquina Virtual (JVM)



História - Qual a Filosofia do Java?

Máquina Virtual (JVM)

- A máquina virtual é o grande segredo do Java.
- Ela é, de uma forma simplificada, um computador montado em software. É ela, e não mais o SO, que é responsável pela execução, mas não só por essa função, da aplicação Java.
- Ela é responsável pelo gerenciamento de memória, pilhas, threads, segurança, etc.
- Dessa forma a aplicação é executada sem comunicação direta com o hardware ou com o SO. Dessa forma, se a JVM “cair, somente as aplicações que estavam em execução na mesma serão finalizadas. Os demais recursos do SO continuam em execução.

História - Qual a Filosofia do Java?

Maaaaasssss .....

Não seria melhor compilar todo o código de uma vez e executar na JVM? Nem sempre.

O .exe gerado em C, Delphi, C++ é otimizado no momento da compilação o que pode não representar o melhor cenário.

Em Java, como a compilação é em tempo real, a otimização acontece no momento da execução baseada nas métricas da JVM de execuções anteriores.

Lembrando que esse ganho de performance refere-se ao ambiente de aplicações distribuídas, principalmente web. Não vale o comparativo em ambientes onde os recursos impedem a execução ideal desse ambiente (firmwares, computação embarcada, computação em tempo real, etc).

História do Java

James Gosling

- Cientista da Computação e projetista chefe do projeto que deu origem ao Java.

História - Um exemplo mais concreto de tudo isso

Fundamentos

Fundamentos – Eclipse Configuração do Ambiente

- Workspace
- Perspectivas
- Views
- Criação do Projeto Java
- Criação de pacotes
  - Organização de pacotes e classes de acordo com o contexto

Fundamentos – Eclipse Atalhos

Ctrl + n ==> Criação de novo artefato

Ctrl + 1 ==> Aciona o quick fixes com sugestões para correção de erros.

Ctrl + Espaço ==> Completa códigos

Ctrl + 3 ==> Aciona modo de descoberta de menu. Experimente digitar Ctrl+3 e depois digitar ggas e enter. Ou então de Ctrl + 3 e digite new class.

Ctrl + F11 ==> roda a última classe que você rodou. É o mesmo que clicar no ícone verde que parece um botão de play na barra de ferramentas.

Ctrl + PgUp e Ctrl + PgDown ==> Navega nas abas abertas. Útil quando estiver editando vários arquivos ao mesmo tempo.

Ctrl + Shift + F ==> Formata o código segundo as convenções do Java

Ctrl + M ==> Expande a View atual para a tela toda (mesmo efeito de dar dois cliques no título da View)

Ctrl + Shift + L ==> Exibe todos os atalhos possíveis.

Ctrl + O ==> Exibe um outline para rápida navegação

Alt + Shift + X e depois J ==> Roda o main da classe atual. Péssimo para pressionar! Mais fácil você digitar Control+3 e depois digitar Run!. Abuse desde já do Control+3

Fundamentos – Operador Ponto

É um operador binário (possui dois operandos, um do lado esquerdo e outro do lado direito). Normalmente chamado dot operator.

O mais comum é que do lado esquerdo fica o objeto ao qual está se referenciando. Do lado direito está a mensagem que está passando para o objeto. O mais comum é ser a invocação de um método, mas também pode ser o acesso direto à uma variável do objeto. Então ele é o operador de acesso a membros de um objeto. Existe caso que o lado esquerdo não ser o objeto mas algum membro que dá acesso ao um objeto.

Há casos em que o acesso não é feito no objeto e sim diretamente na classe.

Também pode usar como separador de nomes dos pacotes.

Fundamentos – Imports

A instrução import da linguagem Java tem como objetivo disponibilizar em uma classe, de um determinado pacote, o acesso a demais classes que estejam em pacotes diferentes. Há duas formas de realizar a importação de uma classe usando a instrução import, a forma explícita e a forma implícita.

A palavra import é uma das muitas palavras reservadas da linguagem Java e não poderá ser usada como nome de variável. Para importar uma classe deve-se usar a instrução import logo após a instrução package, caso exista, e antes da declaração da classe. A instrução será seguida pelo caminho do pacote, delimitado por pontos, e terminará com o nome de uma classe ou um caractere do tipo asterisco, encerrando a instrução com um ponto e vírgula.

É sempre uma boa pratica o uso da instrução import de forma explícita (java.util.List) ao invés do uso da forma implícita (java.util.*). Essa é uma boa pratica porque favorece ao programador que ele determine rapidamente quais classes foram importadas.

Fundamentos – Console

Console é o local onde podemos exibir ou capturar informações em Java.

Pode ser uma área específica utilizada pela a própria IDE que se utiliza para desenvolvimento ou o shell do sistema operacional (COMMAND ou PowerShell no Windows e Bash, Korn, etc, no Linux / Unix).

Fundamentos – Tipo String
Definição

String é um tipo de dado/estrutura/objeto que é utilizada para armazenar texto dentro de um programa sendo composta por uma série de caracteres, mas como estamos falando de Java vamos nos ater a String como um objeto, na verdade um objeto muito singular, pois ela se comporta um pouco diferente.

Fundamentos – Variáveis e Constantes

Em Java, para representação dos tipos de dados, temos a opção de definir os tipos de dados baseados em Classes as chamadas variáveis de referência (atributos), como o de utilizar os tipos primitivos.

Tipos primitivos, como o próprio nome dá a entender, são tipos de dados mais simples, eles não representam classes e são utilizados para o trabalho com dados mais básicos.

Essa característica da linguagem é o que faz com que ela não seja totalmente Orientada a Objetos.

Fundamentos – Tipos Primitivos

Tipo inteiro:

- int x = 10;
- int y = 25;

Tipo ponto flutuante:

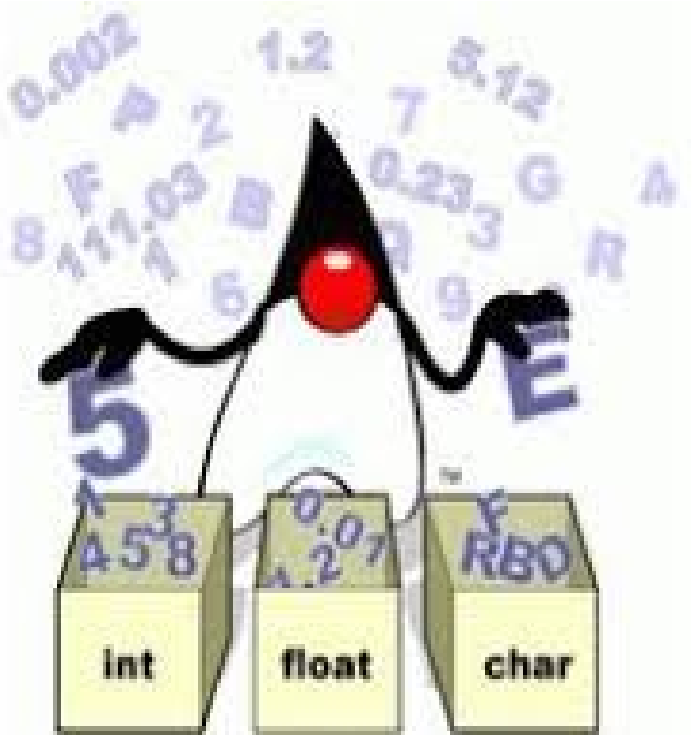
- double n = 2.45;
- double nn = 100;

Tipo Booleano:

- boolean sucesso = true;
- boolean teste = false;

Tipo caractere:

- char variavel = ‘a’;
- char yes = ‘y’;



Fundamentos – Tipos Primitivos
Faixas de Valores

		Valores possíveis				
Tipos	**Primitivo**	**Menor**	**Maior**	**Valor Padrão**	**Tamanho**	**Exemplo**
Inteiro	byte	-128	127	0	8 bits	byte ex1 = (byte)1;
	short	-32768	32767	0	16 bits	short ex2 = (short)1;
	int	-2.147.483.648	2.147.483.647	0	32 bits	int ex3 = 1;
	long	-9.223.372.036.854.770.000	9.223.372.036.854.770.000	0	64 bits	long ex4 = 1l;
Ponto Flutuante	float	-1,4024E-37	3.40282347E + 38	0	32 bits	float ex5 = 5.50f;

	double	-4,94E-307	1.79769313486231570E + 308	0	64 bits	double ex6 = 10.20d;
						ou
						double ex6 = 10.20;

Caractere	char	0	65535	\0	16 bits	char ex7 = 194; ou char ex8 = 'a';
Booleano	boolean	false	true	false	1 bit	boolean ex9 = true;

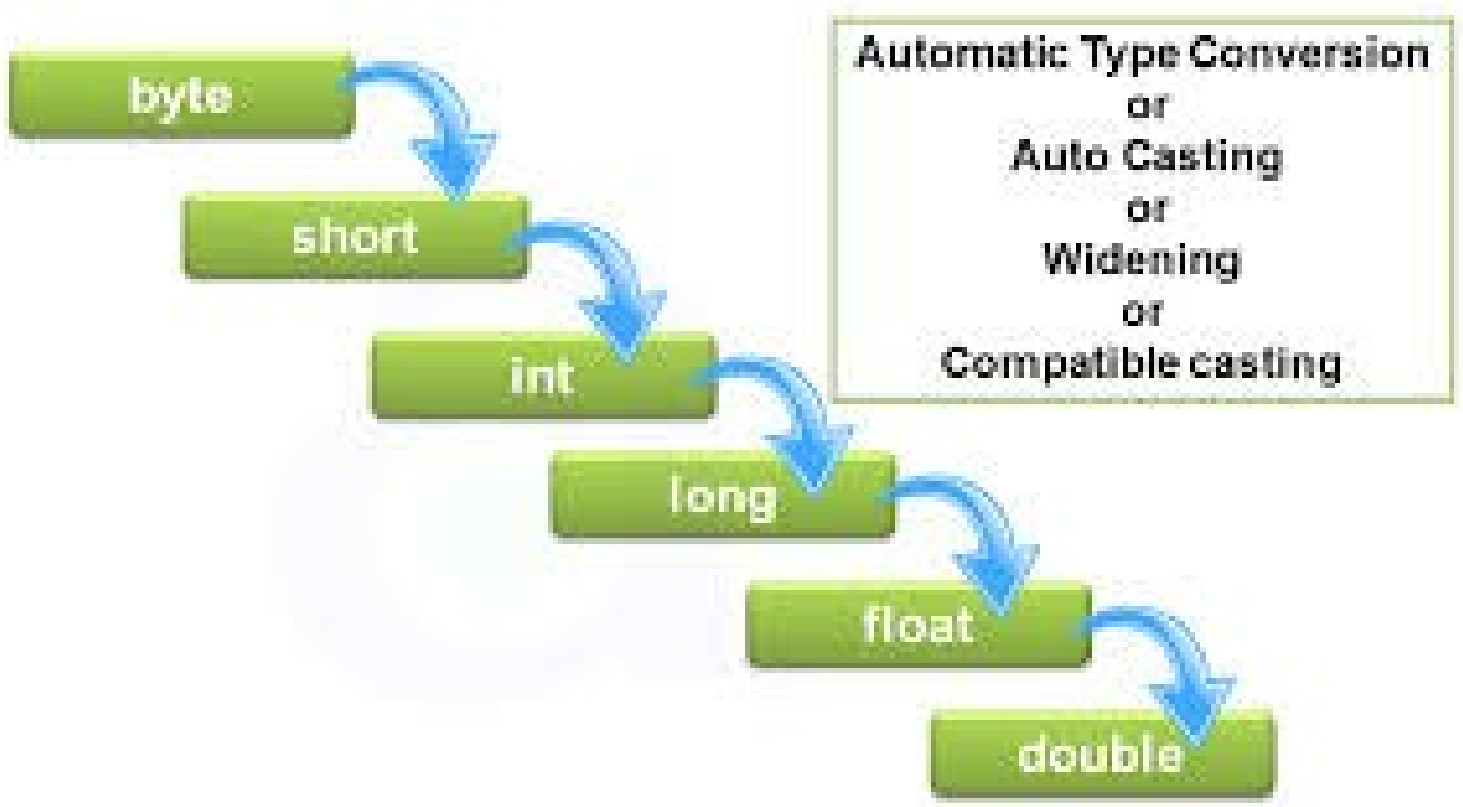
Fundamentos – Tipos Primitivos
Casos “Especiais”

byte → É o tipo de dado de mais baixo nível em Java. Muito utilizado no tratamento de imagens, manipulação de arquivos, acesso à disco, processamento de streams.
short → Número inteiro de 16 bits, utilizado para manter compatibilidade com arquiteturas mais antigas ou para aplicativos embarcados que trabalham nessa arquitetura.
long → Número inteiro de 64 bits, possui uma faixa maior para alocação dos valores. Ideal para geração de números de controle, identificadores em bancos de dados.
float → Número de ponto flutuante de 32 bits. Possuí uma precisão menor que o tipo double e assim como o tipo short, é utilizado para desenvolvimento em arquiteturas que trabalham com essa alocação de informação e para compatibilidade com sistemas mais antigos.

- Observação:
  - A declaração do tipo float não pode ser realizada na forma:
    - float = 10.5 e sim na forma float = 10.5f.
  - Isso porque toda as declarações literais de ponto flutuante em Java são double. Com o prefixo “f” (que pode ser minúsculo ou maiúsculo), o compilador sabe que vai alocar um valor de precisão menor não gerando estouro de tipo.
  - Da mesma forma, o tipo long deve sr declarado na forma long = 10L para a declaração de um número interio de 64 bits.

Fundamentos – Tipos Primitivos
Conversões

Consiste na conversão, sempre que possível, de um tipo de dados primitivo para outro com ou sem perda de informação.

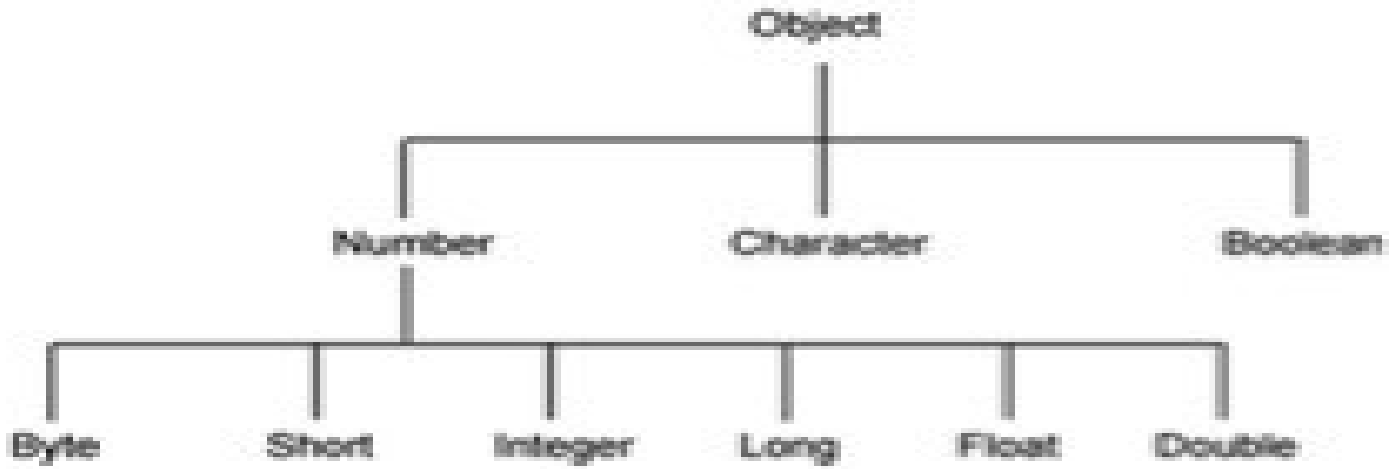


Fundamentos – Tipos Primitivos
Conversões - Observações

- Por que é necessário fazer Casting?
  - Porque o Java realiza os cálculos e aloca o espaço na memória sempre se orientando pelo maior tipo de dado da expressão, nesse caso o double.
- Outro ponto importante, o tipo boolean não aceita castring uma vez que só trabalha com os valores true e false.

Fundamentos – Tipos Primitivos Wrappers

Os Wrapper são conhecidos na linguagem Java como classes especiais que possuem métodos capazes de fazer conversões em variáveis primitivas e também de encapsular tipos primitivos para serem trabalhados como objetos, ou seja, é feita um embrulho de streams que são fluxo de dados através de canais.
Sendo assim, existe uma classe Wrapper para cada tipo primitivo identificado pelo mesmo nome do tipo que possui e tendo a primeira letra maiúscula. Essa regra de declaração é aplicada a todos os tipos, exceto aos que são char classificados como Character e boolean como Boolean.



Fundamentos – Operadores Aritméticos

São funções básicas de somar, subtrair, multiplicar, dividir, etc, além do operador para retorno do módulo (resto) da divisão.
São utilizados na forma:

<operando> <operador> <operando>

Nos casos em que os operandos são números(INTEGER e FLOAT) podem ser usados normalmente, se forem de tipos diferentes, seus valores serão convertidos antes da realização da operação.

Operador	Descrição
+	Operador de Adição – Pode ser utilizado para concatenar Strings
–	Operador de Subtração
*	Operador de Multiplicação
/	Operador de Divisão
%	Operador de Resto de Divisão

Fundamentos – Operadores Lógicos

Operador lógico obtém o valor lógico de uma expressão. O resultado da resolução da expressão é chamado de valor lógico, que em Java, também chamamos de um valor booleano. Em Java temo disponível 3 operadores lógicos, são eles:

- conjunção e
- disjunção ou
- negação not

Uma proposição só pode assumir uma condição lógica, ou seja, ou a proposição é verdadeira, ou não é. Assim, o tipo boolean é um tipo especial utilizado para o armazenamento dessa informação.

Os operadores lógicos são utilizados para efetuarmos operações em condições lógicas.

Por exemplo: podemos verificar se a idade de uma pessoa é maior do que 18 anos, para isso, basta nós utilizarmos os operadores relacionais que então será retornado, a relação entre a idade da pessoa e o número 18. Porém, se nós precisarmos verificar a idade e o sexo de determinada pessoa? Ou seja, se nós precisarmos obter a relação entre a idade informada e o número 18, e o sexo informado com os gêneros possíveis, como que nós faremos?

Fundamentos – Operadores Lógicos

É nessas situações que entram os operadores lógicos. Eles fazem a conexão entre dois valores lógicos e retornam a relação existente. Assim, vamos supor a situação onde num time desportivo as mulheres precisam ter no mínimo 10 anos para participar, enquanto que os homens, precisam ter no mínimo 11 anos. Uma das condições possíveis seria:

> gênero é igual a homem **E** idade é maior ou igual a 11 - condição para que os homens possam participar do time desportivo. A condição que verificaria se uma mulher pode participar, pode ser feita da seguinte forma: gênero é igual a mulher **E** idade é maior ou igual a 10.

Os operadores lógicos são fundamentais para que possamos construir expressões lógicas elaboradas. Do contrário, nós até conseguimos implementar as condições lógicas através do aninhamento de tomadas de decisão, porém, o código fica muito mais extenso e a leitura e entendimento se tornam muito mais difícil.

Fundamentos – Operadores Relacionais

Operadores Relacionais são utilizados em expressões lógicas para se testar as relações entre dois valores do mesmo tipo.

São eles:

=, <, >, <=, >=, <>, ==, !=.

Fundamentos – Operadores Atribuição

Em Java, temos um único operador de atribuição simples e 5 operadores compostos. O operador simples é representado por um único sinal de igual, enquanto que os operadores compostos são constituídos pela junção do operador de atribuição simples junto com um operador matemático.

Todos os operadores que estudamos em aulas passadas, agora, possuíram um operador correspondente de atribuição. A seguir, podemos ver a lista completa:

+=, -=, /=, *=, %=

Agora, se nós utilizássemos expressões, nós teríamos algo semelhante a listagem que se segue:

x -= 1 (é igual) x = x - 1;
x += 1 (é igual) x = x + 1;
x *= 1 (é igual) x = x * 1;
x /= 1 (é igual) x = x / 1;
x %= 1 (é igual) x = x % 1;

Fundamentos – Operadores Unários

Operadores unários é uma "instrução" composta por dois símbolos, que não precisam estar necessariamente entre dois termos (variáveis, etc).

Esses operadores são aplicados especificamente sobre um operador. Eles realizam alguns trabalhos básicos como incremental, decremental e inversão de valores numéricos e booleanos:

Operador	Descrição
+	Operador unário de valor positivo – números são positivos sem esse operador explicitamente.
–	Operador unário de valor negativo – nega um número ou expressão aritmética.
++	Operador unário de incremento de valor – incrementa o valor em 1 unidade.
—	Operador unário de decremento de valor – decrementa o valor em 1 unidade.
!	Operador unário lógico de negação – nega o valor de uma expressão booleana.

Fundamentos – Operadores Ternários

O operador ternário é um recurso para tomada de decisões codificado em apenas uma linha.

Sintaxe do operador ternário:

(expressão booleana) ? valorSeVerdadeiro : valorSeFalso;

FIM DO PRIMEIRO EPISÓDIO