

Programação Orientada a Objetos

Introdução ao Java

Histórico, características, tipos primitivos, atribuições, estruturas de decisão e repetição, métodos

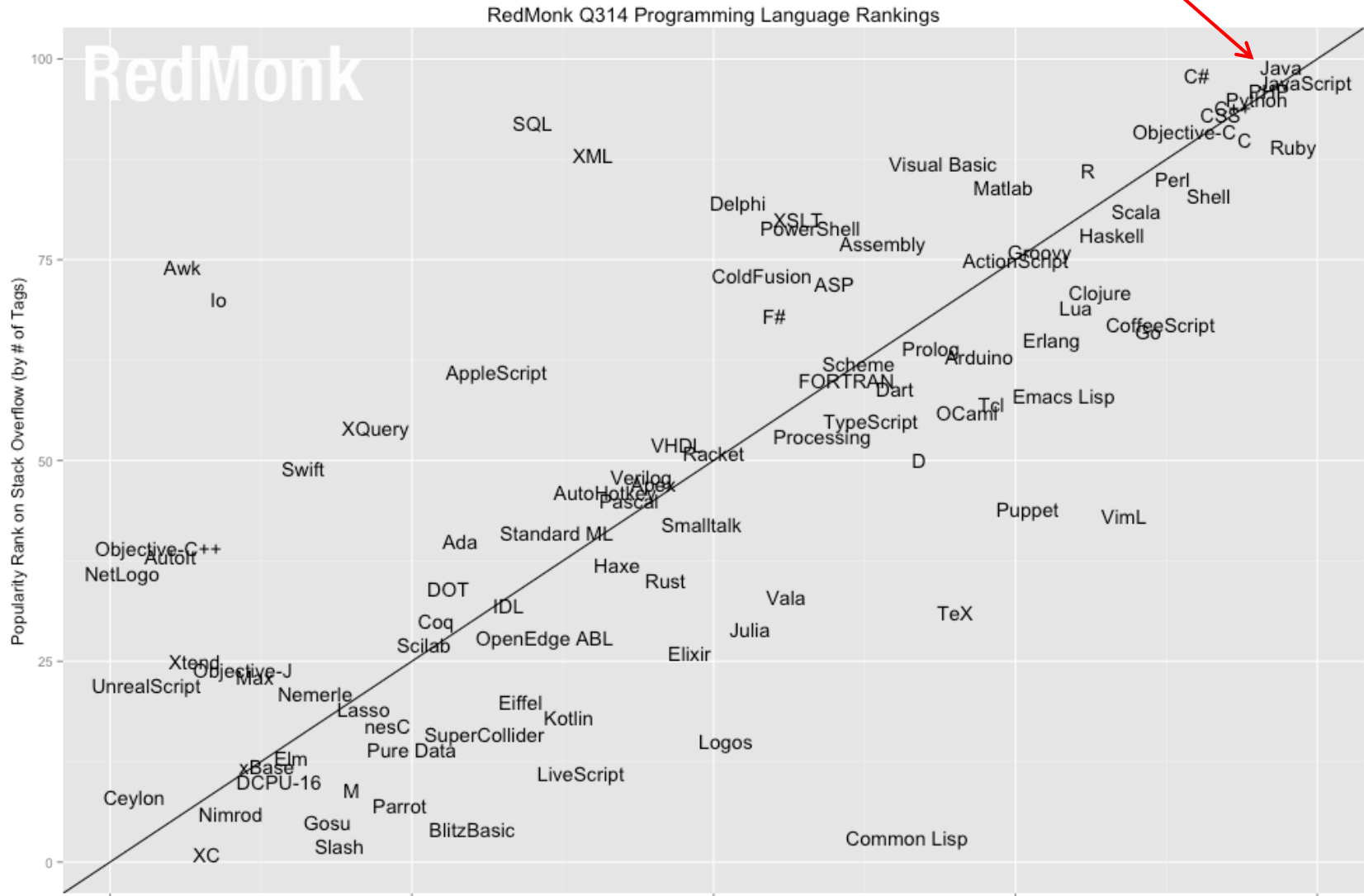
Ely – ely.miranda@ifpi.edu.br

Dúvidas da aula anterior?

Linguagens OO

- Híbridas:
 - Linguagens originalmente projetadas em outro paradigma;
 - Evoluem com recurso de orientação a objetos;
 - Ex: Visual Basic, Object Pascal, C++
- Puras:
 - Projetada desde o início com os conceitos de OO;
 - Ex: Java, Ruby, Python, Smalltalk

Orientação a Objetos e Java



Histórico de Java

- Criada em 1991, tendo como principal projetista, James Gosling da Sun Microsystems;
- A ideia inicial era ter uma linguagem para rodar em pequenos dispositivos de forma portátil;
- C++ não era adequada para o projeto devido à sua complexidade;



Histórico de Java

- Java foi criada como uma versão simples, porém poderosa das ideias da POO;
- Inicialmente chamada de OAK, mas mudou para Java pois o nome original já fora patenteado;
- Inspiração em uma cafeteria local, cujo café vinha de uma ilha da Indonésia chamada Java;
- Hoje a Oracle “gerencia” a plataforma Java após a compra da Sun.



Histórico de Java

- O projeto inicial não vingou;
- O mercado de dispositivos “inteligentes” não estava se desenvolvendo como a Sun previa;
- O crescimento da Internet por volta de 1994 deu um novo impulso ao projeto;
- A Sun voltou seus esforços para acelerar a conclusão da sua linguagem:
 - O objetivo era aproveitar a flexibilidade da linguagem para prover maior dinamismo na Web;
 - Não havia nenhum concorrente na nova era digital que se abria com a Web.

Histórico de Java

- Em 1994, a equipe concebeu um browser
HotJava foi feito para mostrar o poder do Java;
- O HotJava era capaz de executar código Java em
páginas web e foi lançado na SunWorld em 1995;

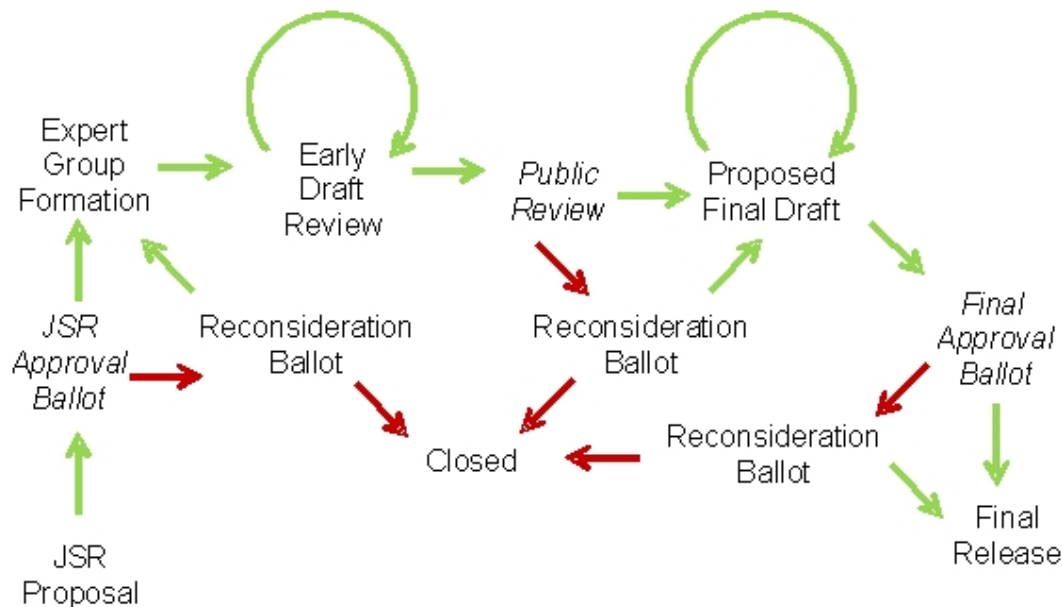


Histórico de Java

- Em 1996, a Netscape decide dar suporte a applets Java ao seu browser Navigator;
- Ainda em 1996 foi lançada a 1ª. Release do Java Development Kit (JDK) 1.0;

Histórico de Java

- 1998 – Java Community Process (JCP)
 - Especificação da tecnologia Java passa a ser conduzida através de processo aberto formado pela Sun e usuários Java;
 - Ciclo para a criação de uma nova *feature* (JSR – Java Specification Request) no Java:

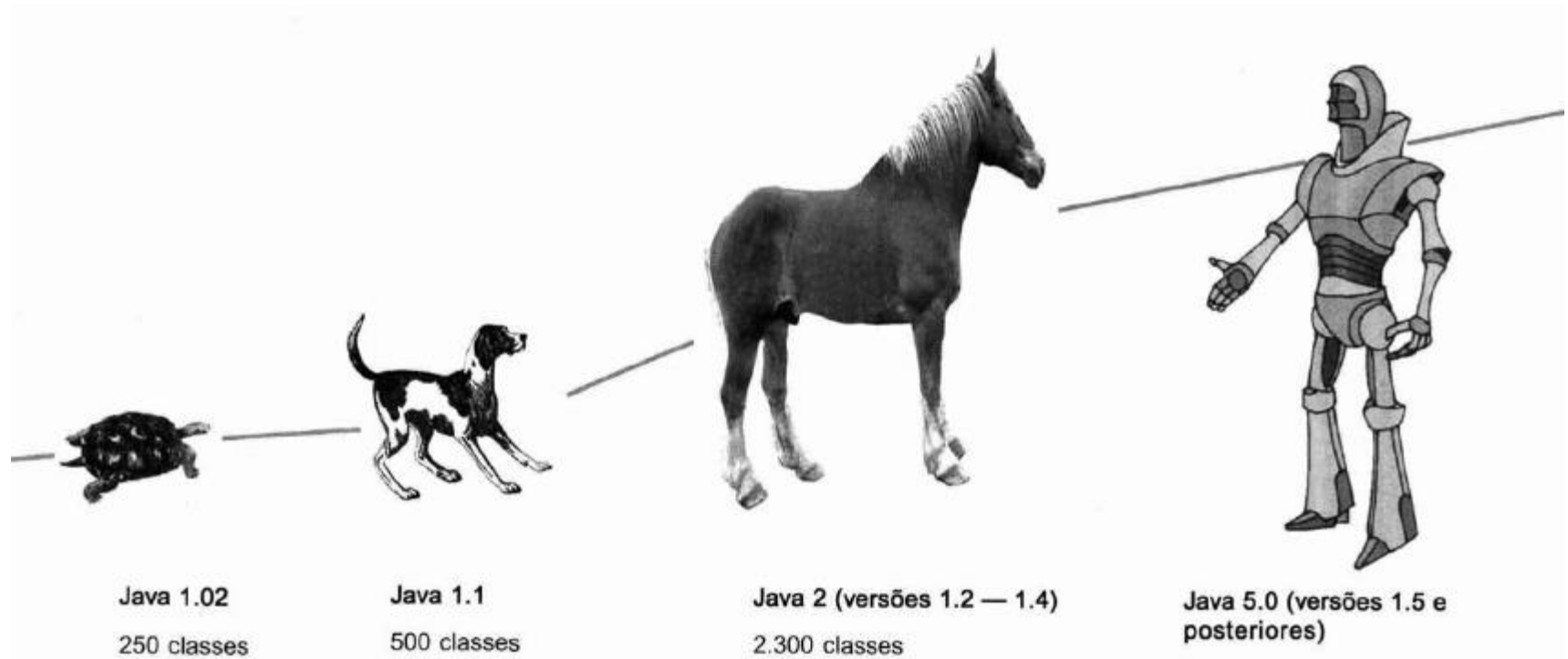


Histórico do Java

- 1999 - Ramificação da plataforma:
 - J2SE (Standard Edition), J2ME (Mobile) e J2EE (Enterprise);
- Atualmente:
 - *JSE 8.0, JEE 6.0*
 - Ambiente de execução aceita várias linguagens e não só Java;

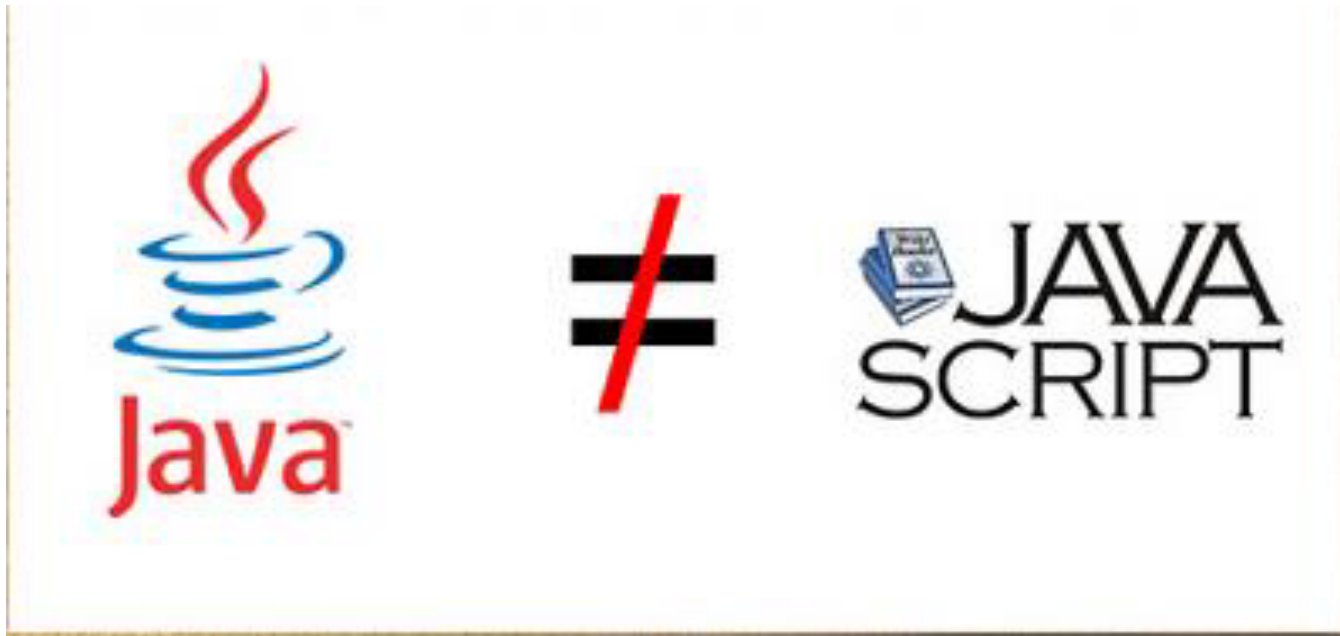
Algumas mitos e curiosidades

- Java é lento
 - Já foi... a distância para C/C++ está diminuindo;
 - Há máquinas virtuais e compiladores que geram código nativo em tempo de execução;



Algumas mitos e curiosidades

- Java e JavaScript NÃO são a mesma coisa:
 - São linguagens bem diferentes;
 - JavaScript é uma linguagem de script para páginas Web e com sintaxe baseada na sintaxe de Java.



Características gerais

- Orientada a Objetos:
 - Implementa os conceitos definidos no paradigma de POO;
 - Foco nos dados e métodos utilizados para manipulá-los;
 - Sintaxe e semântica herdadas de C e C++.
- Simples:
 - Eliminou: arquivos header, variáveis e funções globais, ponteiros, *goto*, *struct*, *union*, número variável de argumentos, tipos fracos, remoção de objetos, classes parametrizadas, sobrecarga de operadores...
 - A forma com que o código é escrito é muito mais clara que C++ e de fácil aprendizado;

Características gerais

- Robusta:
 - A tipagem de dados é forte: os tipos devem ser declarados no código fonte e checados em tempo de compilação;
 - Também faz checagem dinâmica em tempo de execução;
 - Não possui ponteiros, evitando a corrupção de dados em memória;
 - Possui um mecanismo que libera automaticamente a memória que não é mais utilizada;
 - Tem mecanismo para tratamento de exceções, evitando, por exemplo, terminos abruptos da aplicação.

Características gerais

- Independente de arquitetura:
 - Java é executado em qualquer plataforma, o código de bytes gerado é independente de arquitetura de hardware e software;
 - O conjunto de instruções da máquina virtual Java funciona para a maioria das arquiteturas de computadores;
 - Jargão comercial: *compile once, run anywhere*.

Características gerais

- Portável:
 - Tanto a sintaxe quanto a semântica da linguagem são muito bem definidas;
 - Não existem comportamentos específicos da implementação do compilador ou do interpretador.
 - Todas implementações devem seguir exatamente as regras da linguagem;
 - Ter tamanhos fixos para os tipos evita muita dor de cabeça em termos de portabilidade.
 - Tipos inteiros: byte, short, int, long => 8, 16, 32, 64 bits respectivamente
 - Tipos de ponto flutuante: float e double => 32 e 64 bits
 - Seguem o padrão IEEE 754
 - Tipo caractere: char é de 16 bits Unicode
 - Strings usam um formato Unicode padrão.

Características gerais

- Multi-threaded:
 - Threads são uma parte integrante da linguagem Java;
 - Permite a execução concorrente de código, essencial para processamento em segundo plano;
 - Implementação mais simples que em outras linguagens;
 - Torna Java muito atrativa para desenvolvimento no lado servidor.
- Segura:
 - Java foi projetada tendo em mente a transmissão de código através de uma rede;
 - O sistema de execução (runtime) faz as checagens necessárias para garantir a integridade do sistema;
 - Pontos clássicos de vulnerabilidade explorados por vírus e cavalos de tróia não podem ser atingidos.

Características gerais

- Com coleta de lixo automática:
 - A coleta de lixo permite que a memória alocada para objetos seja recuperada.
 - Java faz isso automaticamente, aumentando a produtividade e reduzindo o número de *bugs*.



Características gerais

- Distribuída:
 - Possui um vasto conjunto de classes para o gerenciamento de comunicação de rede de baixo nível;
 - Permite o acesso a objetos remotos através da rede como se fossem objetos locais.
 - Possui implementações completas para Programação para a WEB
- Rica:
 - Possui um vasto conjunto de APIs padronizadas para as mais diversas funções.
 - Acesso a bancos de dados;
 - Interface gráfica;
 - Acesso ao sistema de arquivos;
 - Compressão, E-mail, ...
 - Dispositivos móveis;

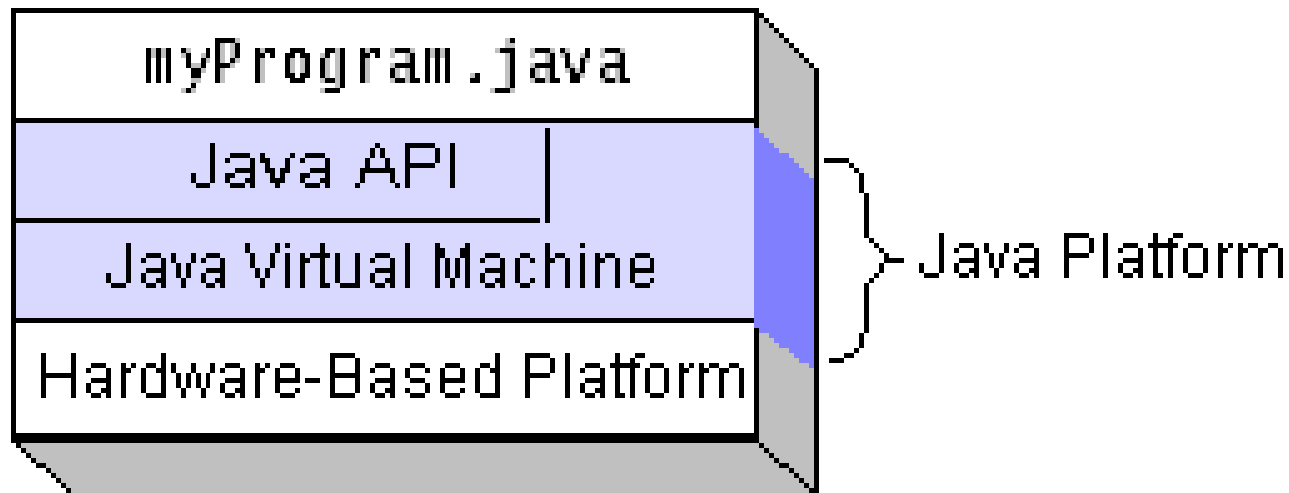
Plataformas

- Plataforma
 - Ambiente de hardware e/ou software no qual um programa é executado
- A plataforma Java é somente de software
 - Esta plataforma de execução funciona sobre outras plataformas de hardware e software
- Plataformas Java
 - JME – Java Platform, Micro Edition
 - Plataforma de desenvolvimento para dispositivos com suporte à Java como Palms, celulares, pagers, etc.
 - Perdeu espaço com a criação do Android
 - JSE – Java Platform, Standard Edition
 - Contém serviços padrão para aplicações Desktop e Applets.
 - JEE – Java Platform, Enterprise Edition
 - Plataforma de desenvolvimento completa para aplicações distribuídas e para a WEB.

Nota: toda documentação que tiver J2... Está desatualizada

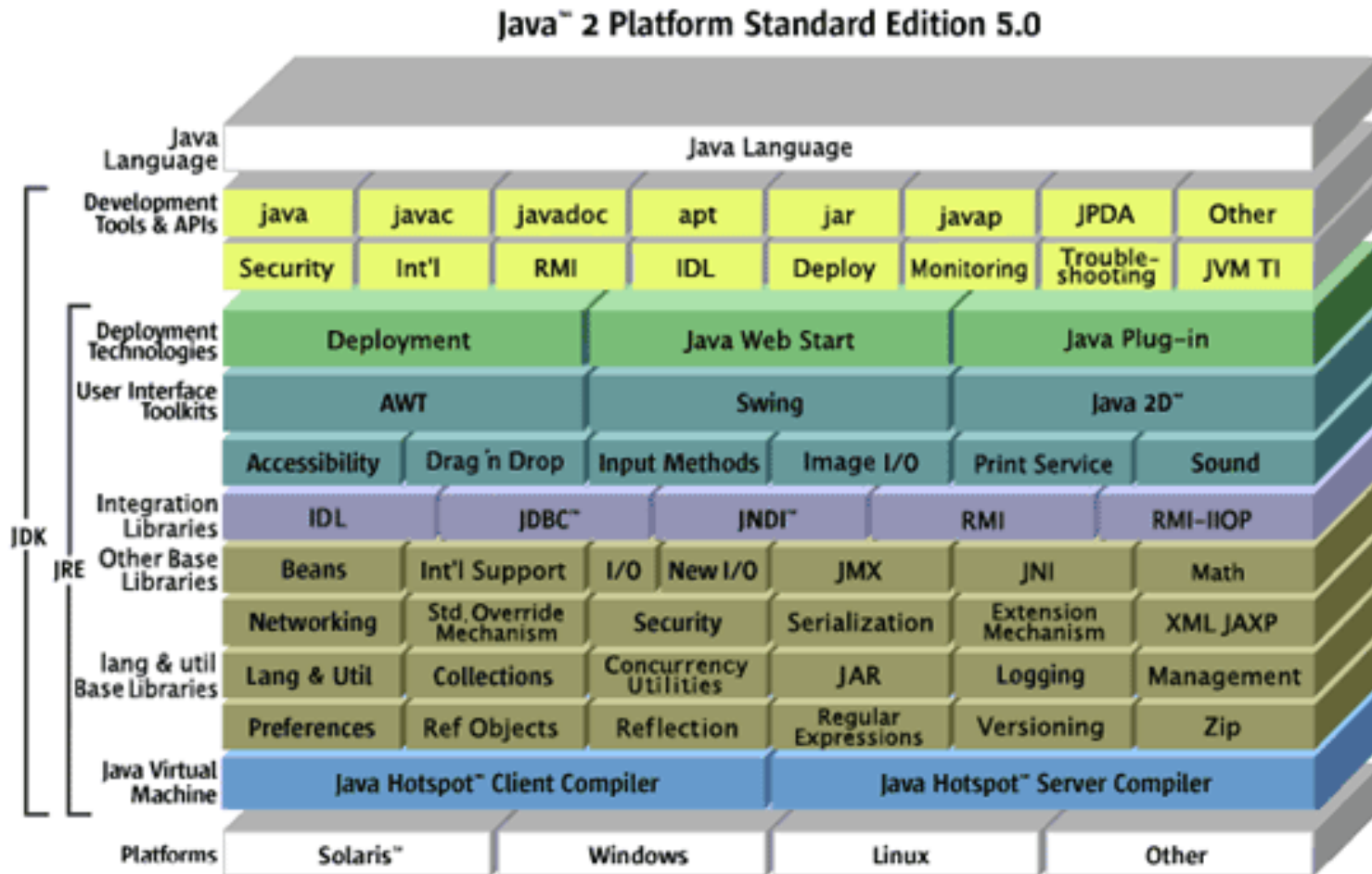
Componentes da plataforma Java

- Java Virtual Machine (JVM)
 - Provê suporte para independência de plataforma
 - Máquina virtual que interpreta código Java compilado
 - Hoje a JVM já suporta outras linguagens que não seja Java
- Java Application Programming Interface (Java API)
 - Provê suporte para programação de aplicações em geral
 - Vasta gama de componentes de software prontos para uso



Componentes da plataforma Java

- Outra visão:



JRE x JDK

- JRE – Java Runtime Environment:
 - Pacote básico para a **execução de aplicativos** Java;
 - Necessário também para execução de Applets em navegadores;
- JDK – Java Development Kit:
 - Pacote para a criação, compilação e execução de aplicativos Java
 - Inclui o JRE
- Ambos contêm uma JVM

SDK – Software Development Kit: conjunto básico de qualquer linguagem: compilador, ferramentas de debugging e etc

Instalando o JDK

- <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Overview

Downloads

Documentation

Community

Technologies

Training



Java SE Downloads



Latest Release



Next Release (Early Access)

Embedded Use

Previous Releases


Download 
Java Platform (JDK) 7


Download 
JavaFX 2.0 Beta
















Download 
JDK 7 + N

Here are the Java SE downloads in detail:

Java SE Development Kit 7

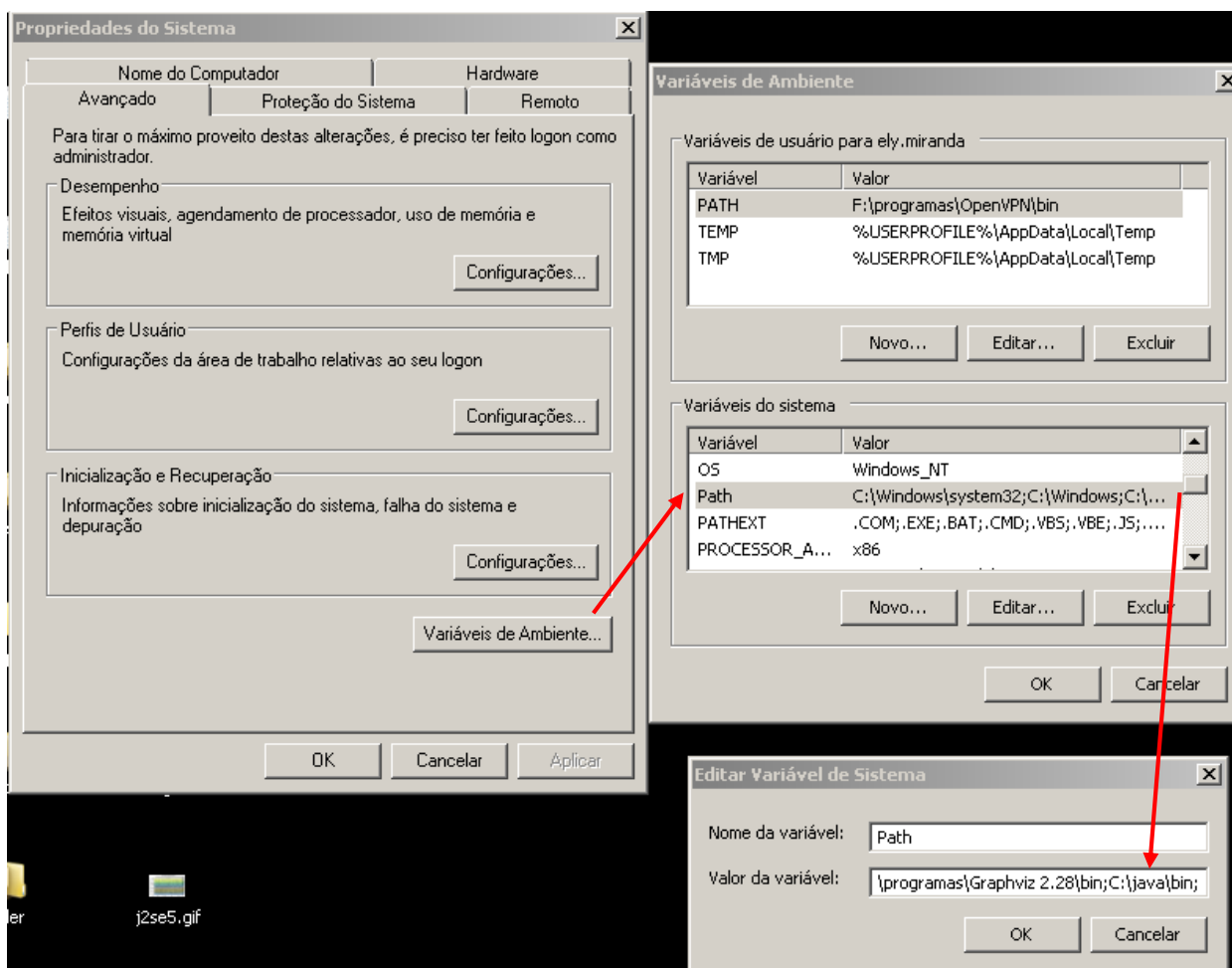
You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

☐ Accept License Agreement ☒ Decline License Agreement

Product / File Description	File Size	Download
Linux x86 - RPM Installer	77.28 MB	 jdk-7-linux-i586.rpm
Linux x86 - Compressed Binary	92.17 MB	 jdk-7-linux-i586.tar.gz
Linux x64 - RPM Installer	77.91 MB	 jdk-7-linux-x64.rpm
Linux x64 - Compressed Binary	90.57 MB	 jdk-7-linux-x64.tar.gz
Solaris x86 - Compressed Packages	154.74 MB	 jdk-7-solaris-i586.tar.Z
Solaris x86 - Compressed Binary	94.75 MB	 jdk-7-solaris-i586.tar.gz
Solaris SPARC - Compressed Packages	157.81 MB	 jdk-7-solaris-sparc.tar.Z
Solaris SPARC - Compressed Binary	99.48 MB	 jdk-7-solaris-sparc.tar.gz
Solaris SPARC 64-bit - Compressed Packages	16.28 MB	 jdk-7-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit - Compressed Binary	12.38 MB	 jdk-7-solaris-sparcv9.tar.gz
Solaris x64 - Compressed Packages	14.66 MB	 jdk-7-solaris-x64.tar.Z
Solaris x64 - Compressed Binary	9.39 MB	 jdk-7-solaris-x64.tar.gz
Windows x86	79.48 MB	 jdk-7-windows-i586.exe
Windows x64	80.25 MB	 jdk-7-windows-x64.exe

Configurando o Path

- Faz com que os aplicativos java e javac sejam "enxergados" no prompt de comando
- Adicione o caminho para a pasta **bin** de onde foi instalado o JDK:



Alguns links

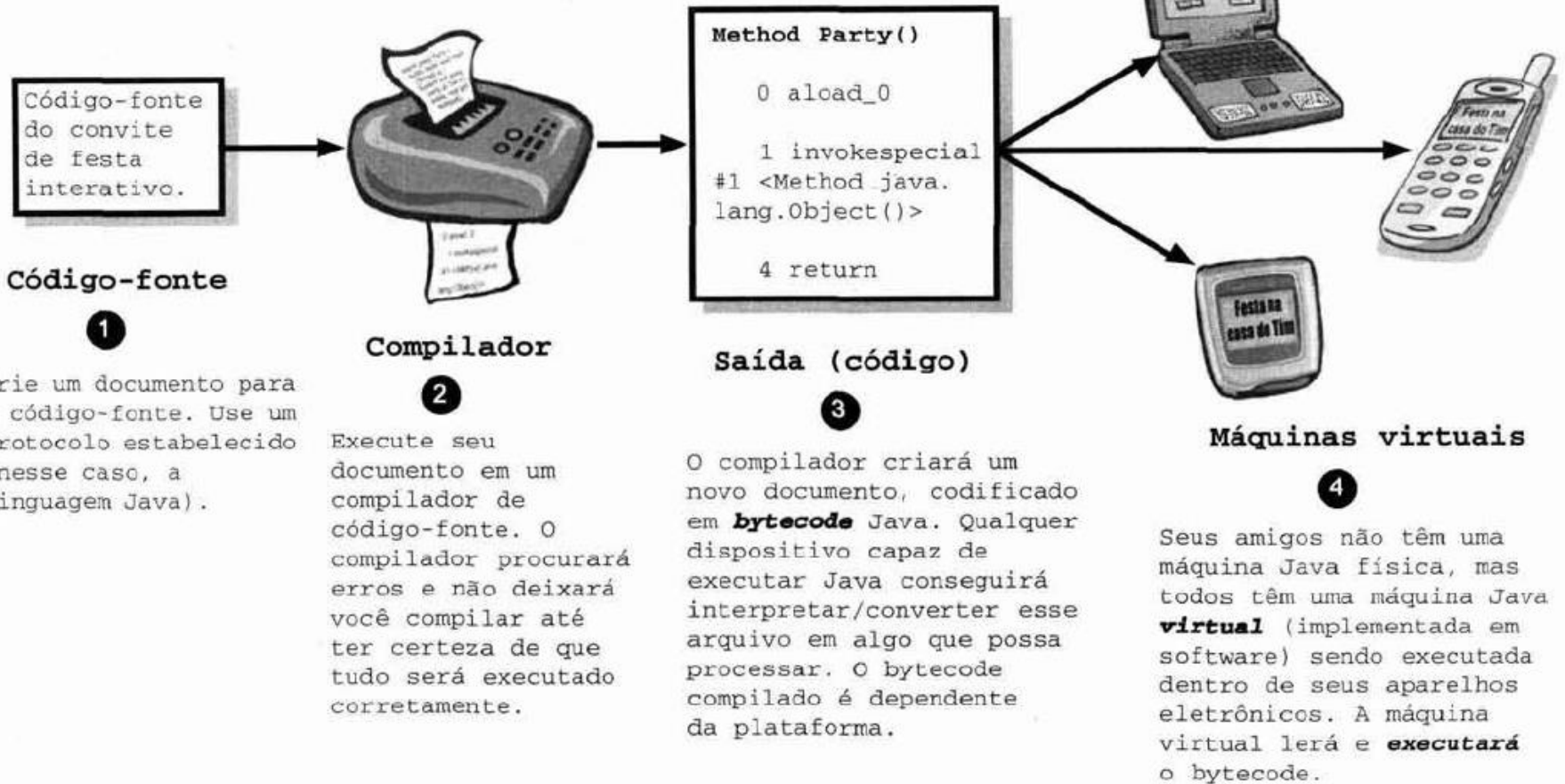
- Página oficial, downloads e ferramentas:
 - <http://www.oracle.com/technetwork/java/>
- Exemplos de codificação:
 - <http://www.java2s.com/>
- Cursos e treinamentos:
 - www.dfjug.org/DFJUG/jedi/
 - <http://www.coreservlets.com/>
 - <http://www.caelum.com.br/>
 - <http://www.k19.com.br/>

Interpretação e execução

- Semi Interpretada:
 - Escreve-se o código Java em qualquer editor
 - O código fonte é **compilado** com o aplicativo **javac**
 - O código gerado são instruções mais simples chamados de *bytecodes*
 - Os bytecodes são **interpretados** por uma JVM específica
 - Os bytecodes são independentes de plataforma, pois há implementações há implementações específicas do ambiente (Windows, Linux...)
- Performance:
 - Mais rápida que interpretação;
 - Um pouco mais lenta que a compilação de C nativo;

Compilação x Execução

qualquer dispositivo que seus amigos tiverem.



Esqueleto de uma classe Java

- Uma classe Java é um módulo onde definimos:
 - Atributos: dados ou características;
 - Métodos: comportamentos
 - Métodos comuns;
 - Métodos construtores (vistos mais à frente).

visibilidade → **public class** HelloWorld {

Nome da classe = Nome do arquivo →

Corpo da classe ←

```
// Atributos
// Construtores
// Métodos
}
```

Entendendo uma classe em Java

```
class HelloWorld {  
    public static void main (String args[]) {  
        System.out.println("Hello World!");  
    }  
}
```

- Salve o arquivo em um diretório qualquer com o mesmo nome da classe: HelloWorld.java

Compilação:

- javac NomedaClasse.java
- Geração de um arquivo .class
- Execução:
 - java NomedaClasse

Método main()

- Toda classe precisa ser criada e carregada na memória através de uma instanciação;
- Uma classe que possui um método main() é considerada uma classe “executável”;
- O método main() é executado quando a classe é instanciada;
- Dentro dele podemos programar (de forma procedural, desaconselhável) ou criar novos objetos ;
- O modificador static faz com que o método seja executado sem que a classe seja instanciada

Torna a classe "executável"

```
public class HelloWorld {  
    public static void main (String args[]) {  
        System.out.println("Hello World!");  
    }  
}
```

Saída padrão em terminal/console

Tipos de dados

- Primitivos:
 - São os tipos mais elementares da linguagem;
 - int, float, double, boolean...
- Objetos:
 - São os tipos complexos da linguagem:
 - Podem ser definidos pelo próprio Java ou definidos pelo programador;
 - Exemplo: Object, String, Retangulo .

Tipos primitivos

- São 8 tipos primitivos:

- Inteiros:

- byte
 - short
 - int
 - long

- Tipo lógico

- boolean



*Aceitam apenas
true ou false*

- Caractere

- char

- Ponto flutuante (decimais)

- float
 - double

Tipo	Tamanho em Bytes
byte	1
short	2
int	4
long	8
float	4
double	8
char	2
boolean	1

Java possui um tipo String, mas na verdade é uma classe

ely.miranda@ifpi.edu.br

Declaração

- Usam a mesma notação do C/C++:

<Tipo> <identificador>;

- Declarando variáveis:

- Uma por linha:

```
float peso;
```

```
float altura;
```

- Várias por linha:

```
float peso, altura, salario;
```

Declaração e atribuição

- Podemos declarar a variável e ao mesmo tempo inicializá-la através de uma atribuição:

<tipo> <identificador> = <expressão/valor>;

```
int x = 1, y = 100;
```

```
long big = 12341234L;
```

```
double peso = 60.5;
```

```
boolean b = true;
```

- Atribuindo valores:

<identificador> = <expressão/valor>;

```
int x, y;
```

```
x = 10;
```

```
y = 20;
```

Declaração e inicialização

- Notas:
 - Podemos declarar variáveis em qualquer trecho do código e não só no início
 - Declarar uma variável, não inicializá-la e tentar fazer uso dela, gera um erro de compilação
 - Ex:

```
public class TestaInicializacao {  
    public static void main(String args[]) {  
        int i;  
        i = i+1; //erro em tempo de compilação  
        System.out.println(i);  
    }  
}
```

Problemas

```
int i;  
double d, d2;  
String s = 5; // não compila  
d = 3.1415;  
i = d; // não compila  
i = 3.14; // não compila  
d = 5; // ok, double pode conter um inteiro  
i = d; // não compila  
// Quando há possibilidade de perda, a  
// conversão  
// não pode ser feita.  
i = 5; // ok, pois não há perda de dados.  
d2 = i; // ok
```

Coerção

- Um número pode sempre ser atribuído a outro de maior precisão:

```
int a = 274;
```

```
long b = a;
```

- A operação inversa requer **coerção** explícita ("*casting*"):

```
long a = 274;
```

```
int b = (int) a;
```

```
int a = 18.7; // o que acontece?
```

```
int a = (int)18.7; //valor armazenado?
```

O tipo String

- Utilizamos a classe String (não é um tipo primitivo) para manipular cadeias de caracteres;
- Strings são cadeias de caracteres envolvidas por áspas duplas;
- Ex:

```
String nome = "Ely";
```

```
String sobrenome = "Miranda";
```

```
String nomeCompleto = nome + " " +  
    sobrenome;
```

- Strings serão estudadas mais adiante.

Identificadores e comentários

- Comentário de linha: `// ...`
- Comentário de bloco: `(/* ... */)`
- Identificadores são quaisquer nomes que estejam fora da sintaxe da linguagem:
 - Devem começar com caractere, `_` ou `$`;
 - Não podem conter espaços;
 - Maiúsculas diferentes de minúsculas;
- Constuma-se padronizar a escrita de métodos, atributos e Classes com o estilo Camel Case

Padronização de nomes

- Nomes de variáveis e métodos:
 - Iniciam com minúsculas e as palavras subsequentes devem iniciar com maiúsculas;
 - Ex: nome, nomeComposto, nota, total, totalDePontos(), calculaMedia() ...
- Nomes de classes:
 - iniciam com maiúsculas e as palavras subsequentes também;
 - Ex: Classe, ClasseComposta, Aluno, CarrinhoDeCompras, ContaCorrente ...
- Nomes de constantes são escritos em caixa alta e os nomes compostos separados pelo caracter '_'.
 - Ex:

```
final float PI = 3.1415;  
final String OUTRA_CONSTANTE = "JAVA"
```

Operadores e atribuições

- Operadores “compostos” e simplificados também podem ser utilizados em atribuições:

– Ex:

```
int b = 5;
```

```
b += 5; // é o mesmo que b = b + 5;
```

Operador	Significado
+	adição
-	subtração
*	multiplicação
/	divisão
%	resto da divisão (módulo)

Operador	Exemplo	Expressão equivalente
+=	x += y	x = x + y
--	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

Operadores de Incremento

Operador	Exemplo	Significado
++	++a	Soma mais 1 e depois atribui Pré-Incremento
	a++	Atribui depois soma mais 1 Pós-Incremento
--	--a	Subtrai 1 e depois atribui Pré-Decremento
	a--	Atribui e depois subtrai 1 Pós-Decremento

Operadores de comparação

- Retornam **true** ou **false**

Operador	Significado
==	Igual a
!=	Diferente de
<	Menor que
>	Maior que
<=	Menor ou Igual a
>=	Maior ou Igual a

Operadores lógicos

Operador	Significado
&&	E (<i>"logical AND"</i>)
&	E (<i>"boolean logical AND"</i>)
 	OU (<i>"logical OR"</i>)
 	OU (<i>"boolean logical inclusive OR"</i>)
^	OU EXCLUSIVO (<i>"boolean logical exclusive OR"</i>)
!	NÃO (<i>"logical NOT"</i>)

Curto-circuito

- Java usa operações curto circuito:
 - Uma enésima comparação pode não ser avaliada caso as primeiras já satisfaçam os operadores, por exemplo:
- Ex 1:

```
public class CurtoCircuito {  
    public static void main(String args[]) {  
        int a = 10;  
        int b = 0;  
        if (a <= 10 || (++b)==0) {  
            System.out.println(b);  
        }  
    }  
}
```



Qual o valor de b?

Curto-circuito

- Ex 2:

```
public class CurtoCircuito2 {  
    public static void main(String[] args) {  
        int a = 10;  
        int b = 0;  
        if (a > 10 && (++b) == 0) {  
            System.out.println(b);  
        } else {  
            System.out.println(b);  
        }  
    }  
}
```

Qual o valor de impresso?

Estruturas de decisão

- if e else

```
if <condicional>  
    <Comando>;
```

```
if <condicional> {  
    <Comando>;  
    <Comando>;  
}
```

```
if <condicional> {  
    <Comando>;  
}else if <condicional> {  
    <Comando>;  
}
```

Estruturas de decisão

- Switch

```
switch (<expressão>) {  
    case valor1:  
        <instruções>;  
        break;  
    case valor2:  
        <instruções>;  
        break;  
    default:  
        <instruções>;  
}
```

Estruturas de repetição

- While

- Com teste no início:

- Só entra no enquanto se a condição for verdadeira
 - Após entrar, sai somente quando a condição for falsa

```
while <condição> {  
    <comandos>  
}
```

- Com teste no final:

- Executa pelo menos uma vez as instruções

```
do {  
    <comandos>  
}while <condição>;
```

Estruturas de repetição

- For:

```
for (<inicializações> ; <expressão> ; <passo> ) {  
    <comandos>;  
}
```

- Ex:

```
public class RepeticaoFor {  
    public static void main(String[] args) {  
        int quantidade = 1000;  
        for (int i = 1; i <= quantidade; i++) {  
            if (i % 2 == 0)  
                System.out.println(i + " : é Par;");  
            else  
                System.out.println(i + " : é Impar;");  
        }  
    }  
}
```

Métodos

- Grandes programas são criados a partir de pequenos módulos chamados métodos;
- Métodos podem ser individualmente desenvolvidos, testados e reutilizados;
- O usuário do método não precisa saber como ele funciona, apenas como utilizá-lo;
- Modularizar em métodos também é considerada uma forma de abstração.

Métodos

- Métodos são compostos por um identificador, parâmetros de entrada e tipo de retorno;
- Entre as chaves temos instruções;
- Métodos que não retornam void, usam a instrução return

```
public class Conta {  
    double saldo;  
    public double calcularDesconto(double taxa) {  
        double desconto = saldo * taxa/100;  
        return desconto;  
    }  
}
```

Tipo de retorno → **double** (before `calcularDesconto`)

Identificador (nome do método) → **calcularDesconto**

parâmetro → **double taxa**


Instrução de retorno → **return**

elvy.miranda@ifpi.edu.br

Métodos

- Havendo mais de um parâmetro, devem ser separados por vírgula:


```
public class Conta {  
    double saldo;  
    //...  
    public double calcularJurosSimples(double taxa, int tempo) {  
        double juros = saldo * taxa/100 * tempo;  
        return juros;  
    }  
}
```



Testando a classe Conta

```
public class TestaConta {  
    public static void main(String[] args) {  
        Conta c = new Conta();  
        c.saldo = 100;  
  
        System.out.println(c.calcularDesconto(10));  
        System.out.println(c.calcularJurosSimples(10, 2));  
    }  
}
```

Chamando os métodos



Programação Orientada a Objetos

Introdução ao Java

histórico, características, tipos primitivos, atribuições, estruturas de decisão e repetição, métodos

Ely – ely.miranda@ifpi.edu.br