
Programação Orientada a Objetos

Introdução ao Java

Profa.: Márcia Sampaio Lima

EST - UEA

Referência: Slides Prof. Flávio José Mendes Coelho

Tecnologia Java

Tecnologia Java

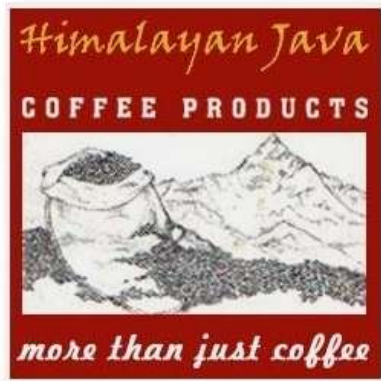
História

- 1991, Sun anuncia formalmente a JAVA.
- James Gosling define a linguagem **Oak** que passou a se chamar **Java**.
- 1995, Sun divulga especificação da **tecnologia Java**.



Tecnologia Java

O que é?



- Além de tipo de café e ilha da Indonésia...

Tecnologia Java

O que é?

- **Plataforma Java** é o nome dado ao ambiente computacional criada pela Sun e vendida para a Oracle depois de alguns anos.
 - Possui grande escalabilidade:
 - Projetos Corporativos de Grande Porte;
 - Sistemas Web;
 - Dispositivos de pequeno porte: celulares, pagers.
 - Vantagem: não está presa a um único sistema operacional ou *hardware*, pois seus programas rodam através de uma máquina virtual.
-

Tecnologia Java

O que é?

- Possui grande escalabilidade:
 - ❑ Projetos Corporativos de Grande Porte;
 - ❑ Sistemas Web;
 - ❑ Dispositivos de pequeno porte: celulares, pagers.
 - Um ambiente de execução presente em browsers, mainframes, celulares, palmtops, cartões inteligentes, eletrodomésticos.
-

Tecnologia Java

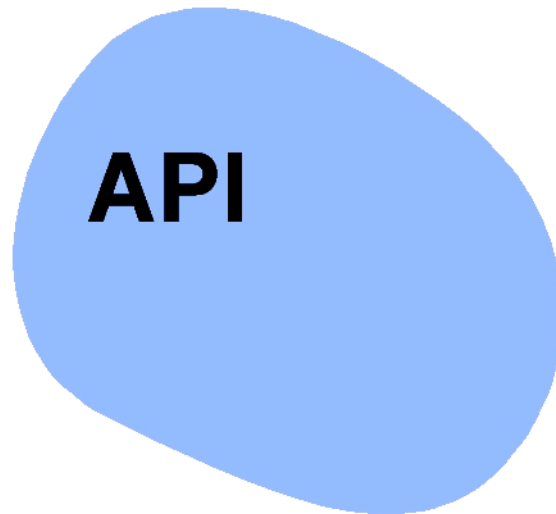
O que é?

- **Java** implementa os conceitos da Orientação Objeto:
 - Classes e objetos
 - Encapsulamento
 - Herança
 - Polimorfismo
-

Tecnologia Java

O que é?

- **Java** possui um conjunto rico de classes pré-definidas que os programadores podem utilizar.
 - “Java já possui tudo pronto!!!”



Tecnologia Java

O que é?

- As classes são agrupadas em coleções de classes, chamadas de pacote e formam a *API (Application Programming Interface)* da linguagem JAVA.
 - Uma coleção de APIs (classes, componentes, frameworks) para o desenvolvimento de aplicações multiplataforma



API

Tecnologia Java

O que é?

- Um programa escrito para a plataforma Java necessita de dois componentes para ser executado:
 - ❑ a máquina virtual Java (JVM) → interpreta os programas escritos em JAVA.
 - ❑ e um conjunto de bibliotecas de classe que disponibilizam um série de serviços para esse programa → API.
 - ❑ O pacote de *software* que contém a máquina virtual e esta biblioteca de classes.

-
- As principais APIs são distribuídas juntamente com os produtos para desenvolvimento de aplicações
 - **Java 2 Standard Edition (J2SE):** ferramentas e APIs essenciais para qualquer aplicação Java (inclusive GUI)
 - **Java 2 Enterprise Edition (J2EE):** ferramentas e APIs para o desenvolvimento de aplicações distribuídas
 - **Java 2 Micro Edition (J2ME):** ferramentas e APIs para o desenvolvimento de aplicações para aparelhos portáteis
 - **Java 2 System Development Kit (J2SDK) :** Coleção de ferramentas de linha de comando para, entre outras tarefas, compilar, executar e depurar aplicações Java.
-

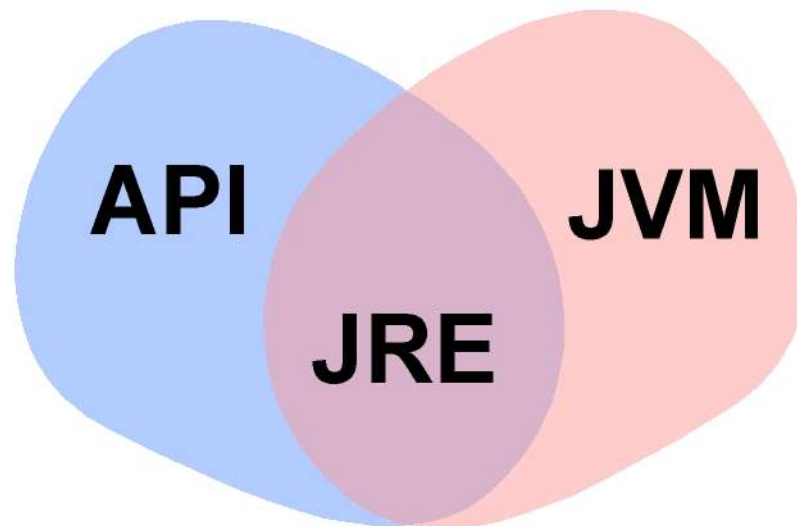
Tecnologia Java

O que é?

- ❑ O pacote de *software* que contém a máquina virtual e esta biblioteca de classes é conhecido como JRE (Java Runtime Environment).
- ❑ Java Runtime Environment (JRE):
 - Tudo o que é necessário para executar aplicações Java .
 - Parte do J2SDK e das principais distribuições Linux, MacOS X, AIX, Solaris, Windows.

Tecnologia Java - JRE

- JRE ou Java Runtime Environment (plataforma ou ambiente Java em tempo de execução) é o mínimo necessário para a execução de programas "executáveis" Java.



Java

- Diferentemente das linguagens convencionais, que são compiladas para código nativo, a linguagem Java é compilada para um bytecode que é executado por uma máquina virtual.
-

Tecnologia Java

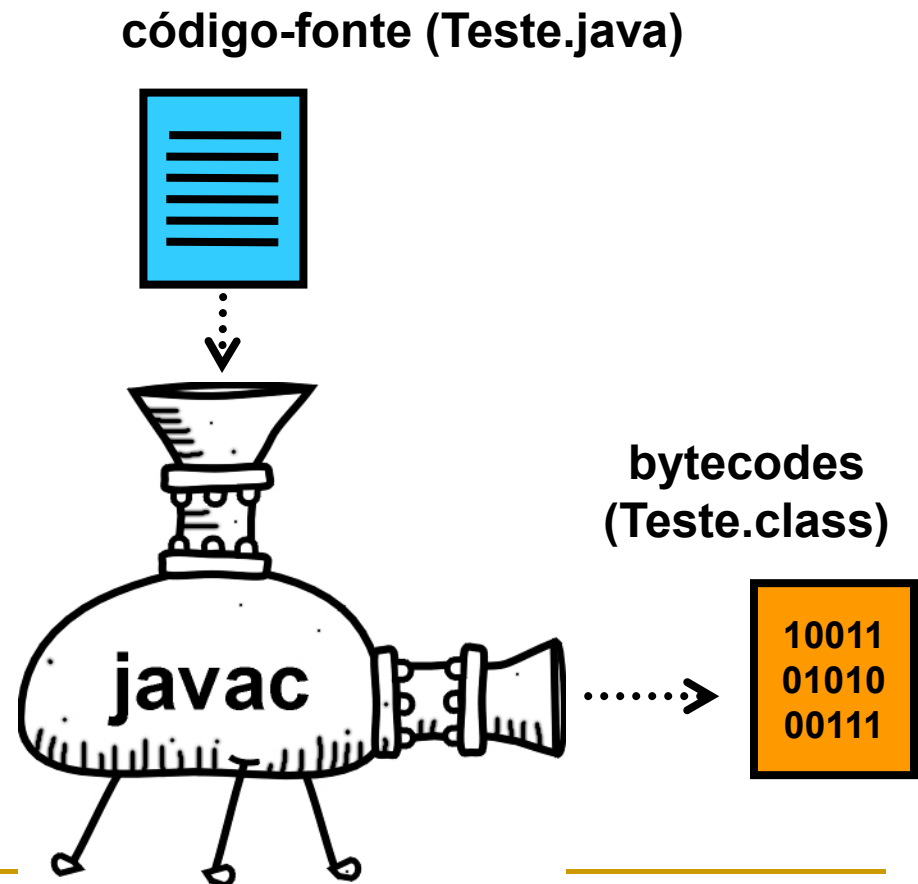
Como a coisa funciona?



Tecnologia Java

Como a coisa funciona?

- Escrevemos o código-fonte na linguagem Java.
- O compilador Java traduz o código fonte em bytecodes.



Tecnologia Java

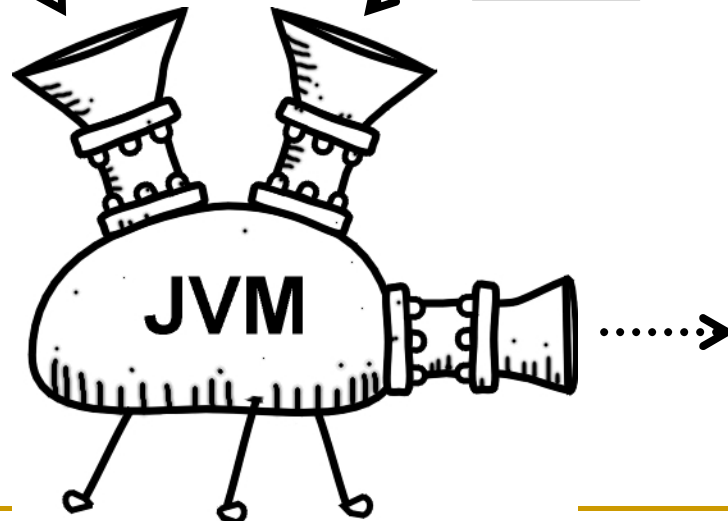
Como a coisa funciona?

bytecodes (Teste.class)

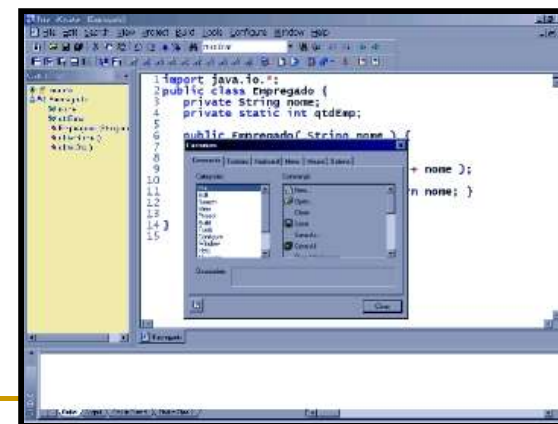
classes API Java

10011
01010
00111

10011
01010
00111

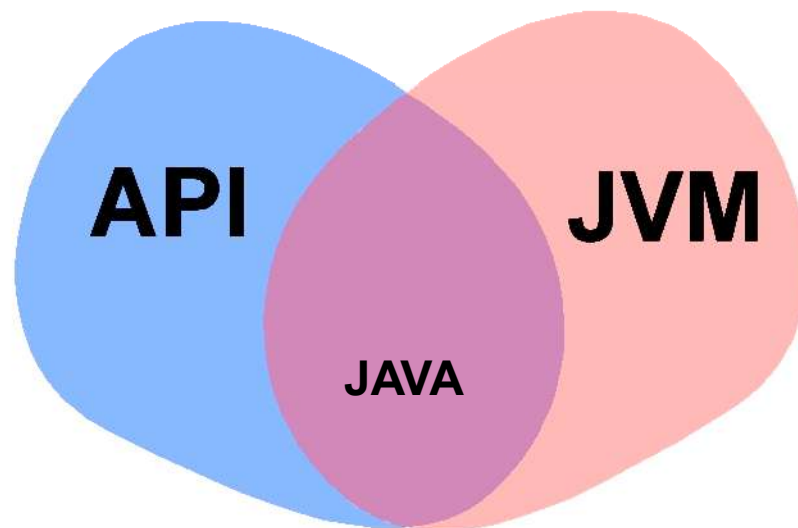


aplicação Teste



Tecnologia Java

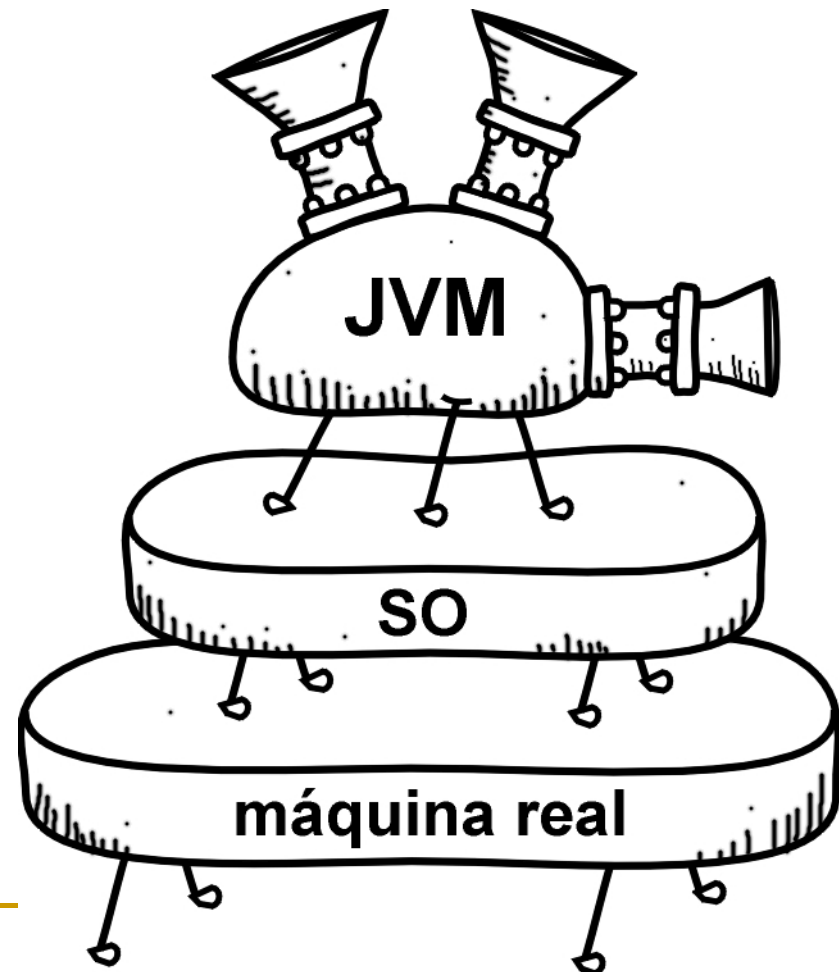
O que é?



Tecnologia Java

Como a coisa funciona?

- **A Máquina Virtual**
Java é um programa que roda sobre um SO.



Tecnologia Java

Características da Linguagem

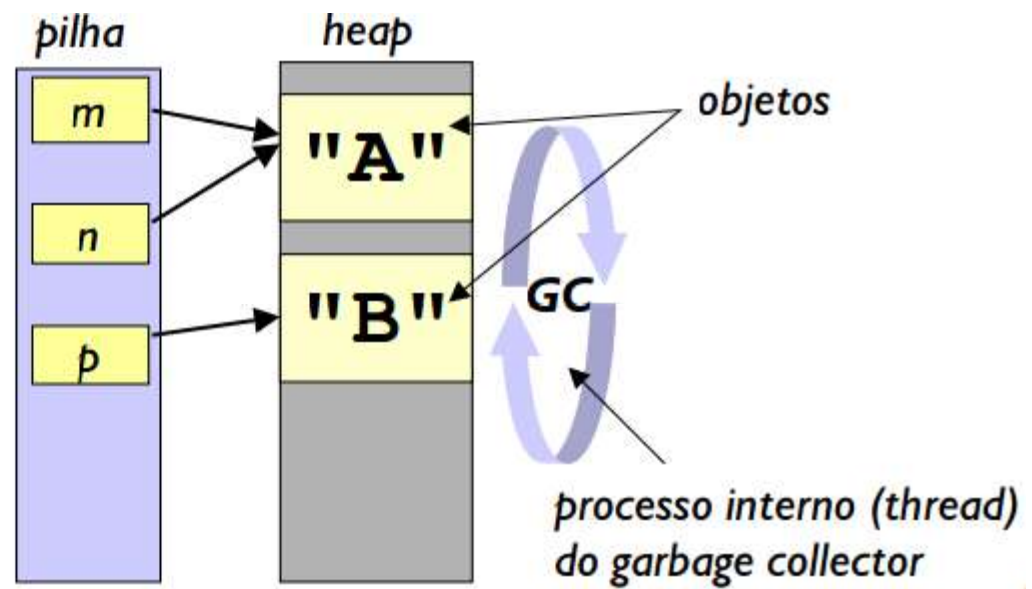
- **Orientada a Objetos:** classes/objetos, composição, encapsulamento, herança e polimorfismo.
- **Interpretada:** programas rodam em JVMs.
- **Portável:** “Write Once, Run Anywhere.”
- **Tratamento de exceções.**
- **Multithreading:** computação concorrente.
- Programação de aplicações GUI e applets.
- Coleta de Lixo

Tecnologia Java

Coletor de Lixo

- A memória alocada em Java não é liberada pelo programador
 - Ou seja, objetos criados não são destruídos pelo programador
 - A criação de objetos em Java consiste de
 - 1. Alocar memória no heap para armazenar os dados do objeto
 - 2. Inicializar o objeto (via construtor)
 - 3. Atribuir endereço de memória a uma variável (referência).
 - Mais de uma referência pode apontar para o mesmo objeto.

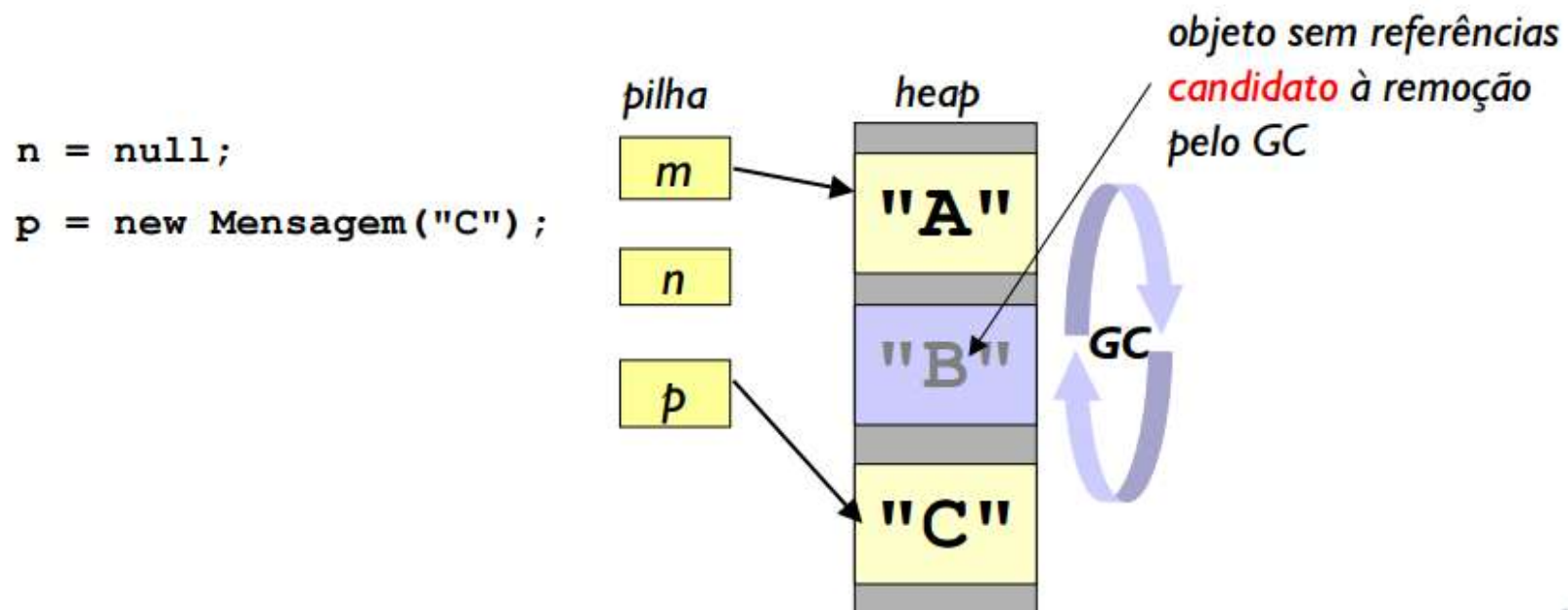
```
Mensagem m, n, p;  
m = new Mensagem("A");  
n = m;  
p = new Mensagem("B");
```



Tecnologia Java

Coletor de Lixo

- Quando um objeto não tem mais referências apontando para ele, seus dados não mais podem ser usados, e a memória deve ser liberada.
- O coletor de lixo irá liberar a memória na primeira oportunidade



Tecnologia Java – Não Suporta

- Java não suporta herança múltipla de implementação
 - Herança múltipla é característica comum a várias linguagens OO, e permite reuso de código de várias classes em outra classe
 - Tem vantagens porém aumenta a complexidade
 - Java oferece uma solução que preserva as principais vantagens da herança múltipla e evita os problemas

Tecnologia Java – Não Suporta

- Java não suporta aritmética de ponteiros
 - Ponteiros, ou referências, são usados em várias linguagens, inclusive Java, para manipular eficientemente grandes quantidades de informação na memória
 - Com ponteiros, em vez de copiar uma informação de um lugar para outro, copia-se apenas o seu endereço. Em linguagens como C, o programador pode manipular o endereço (que é dependente de plataforma) diretamente
 - Isto aumenta a complexidade e diminui a portabilidade

Java - Primeiro Programa

HelloWorld.java

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

HelloWorld.java

Comentário de bloco

Nome da classe

Nome do método

Declaração de argumento

variável local: args
tipo: String[]

Ponto-e-vírgula
é obrigatório no
final de toda
instrução

```
/** Aplicação Hello World */
```

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

Definição de método main()

Atribuição de argumento
para o método println()

Definição de classe
HelloWorld

Chamada de método println()
via objeto out acessível
através da classe System

Java - Primeiro Programa

■ Tipos de Comentários:

- // → comentário de uma linha
- /* Comenta um bloco */
- É possível a inserção de comentários no meio da linha

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

Java - Primeiro Programa

- Tipos de Comentários:
 - Comentários no estilo javadoc:
 - `/** Aplicação Hello World */`
 - Tipo de comentário que pode ser utilizado posteriormente pelo utilitário javadoc para geração de documentação automática.

Java - Primeiro Programa

■ Declaração da classe

- { } delimitadores do escopo da classe
- Pode conter: atributos e métodos
- Java é case sensitive
 - HelloWorld != helloworld

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

HelloWorld.java

Java - Primeiro Programa

- Método main():
 - public static void main(String[] args)
 - Inicia a execução da aplicação java
 - Todo aplicativo java deve conter um método main()
 - public static void main(String[] args)

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

HelloWorld.java

Java - Primeiro Programa

- `System.out.println()`
 - Comando de saída
 - Toda instrução deve ser finalizada com ;

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

HelloWorld.java

Java - Primeiro Programa

- Para executar:
 - >javac HelloWorld.java
 - >java HelloWorld
 - >Hello, world!

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

HelloWorld.java

Tecnologia Java

O Exemplo mais Simples

1. Escreva o código-fonte em um editor.

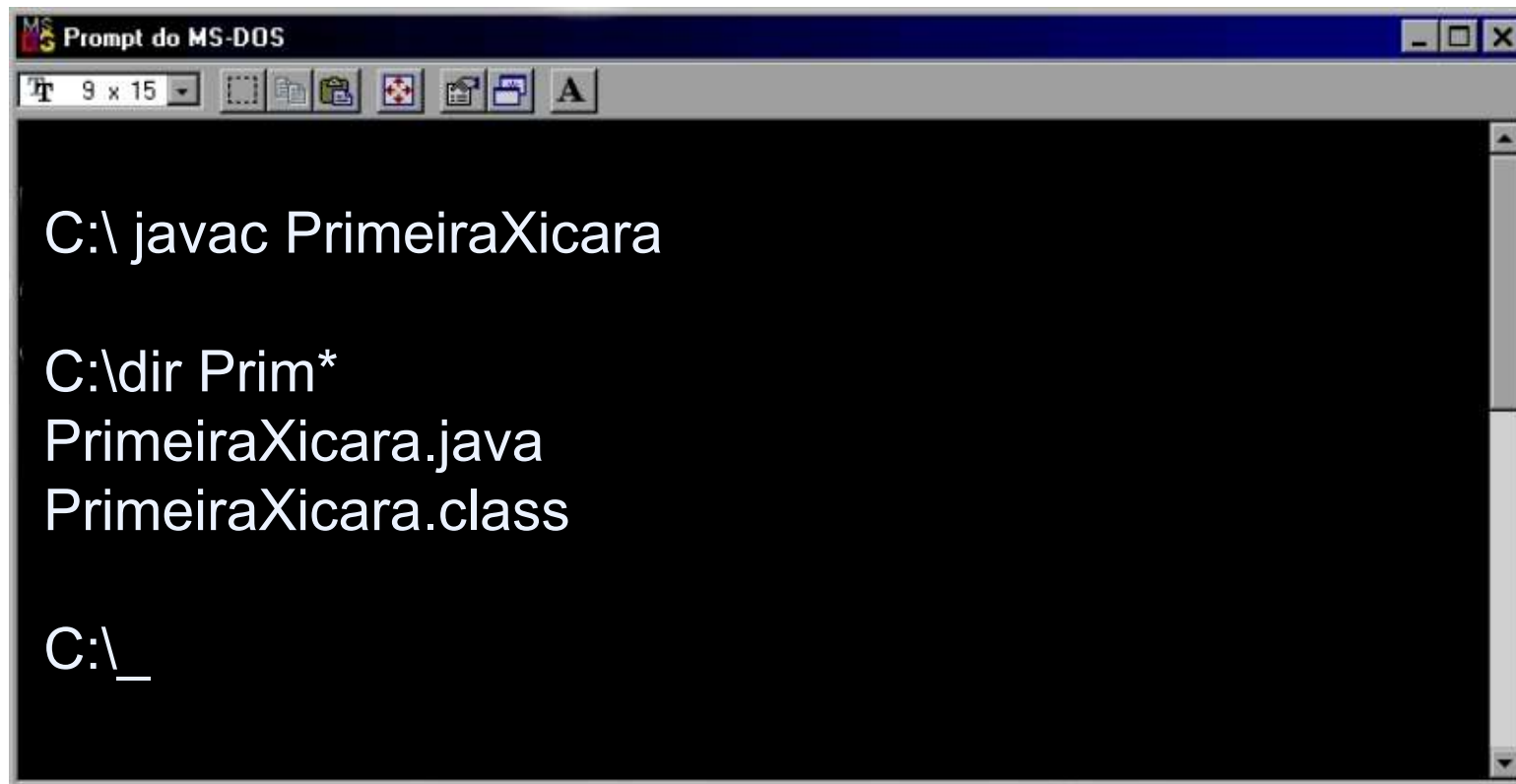
PrimeiraXicara.java

```
public class PrimeiraXicara {  
    public static void main (String[ ] args) {  
        System.out.println("A Terra é azul!");  
    }  
} // fim classe
```

Tecnologia Java

O Exemplo mais Simples

2. Compile o seu código.



```
MS-DOS Prompt
9 x 15
C:\ javac PrimeiraXicara

C:\dir Prim*
PrimeiraXicara.java
PrimeiraXicara.class

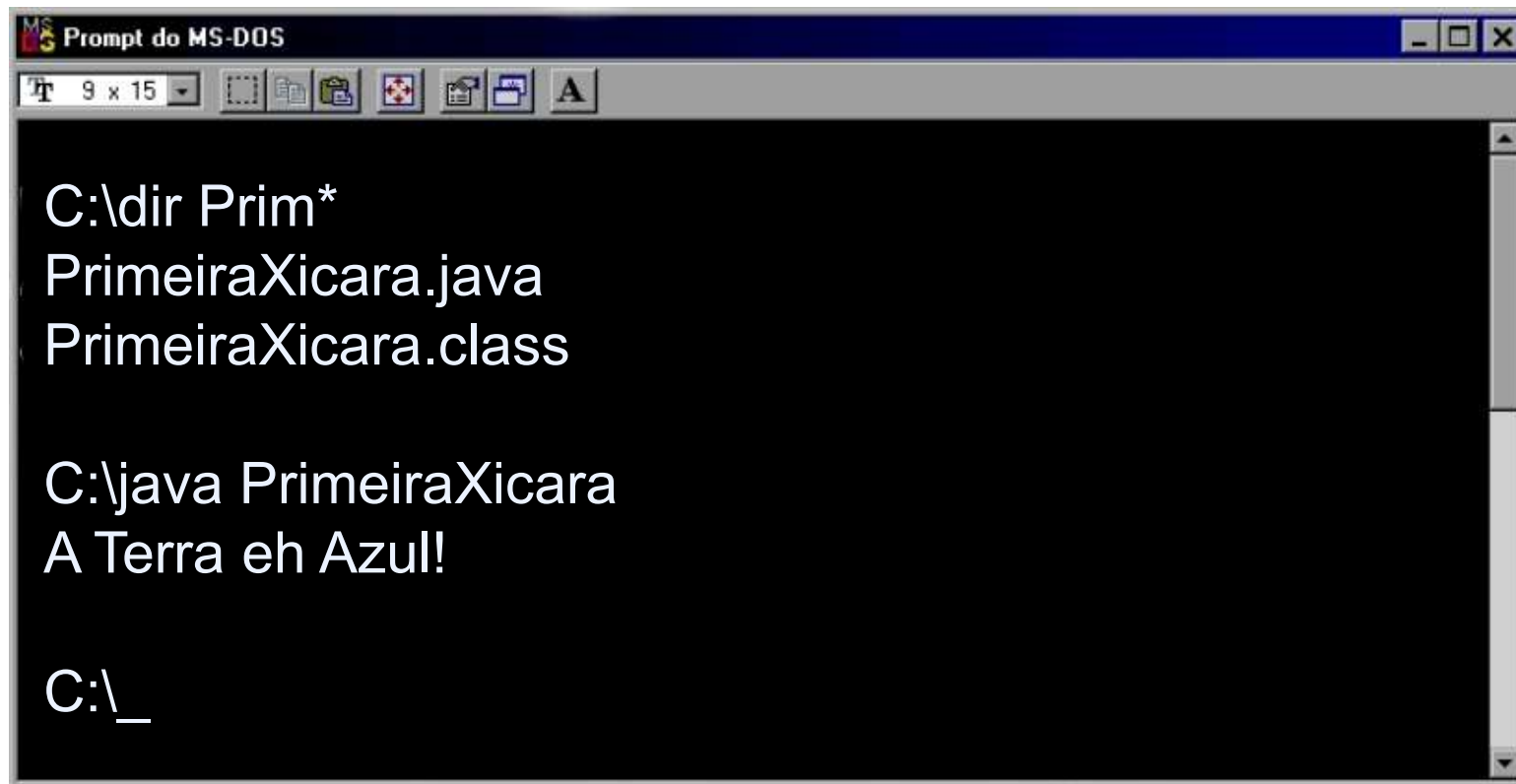
C:\_
```

The image shows a screenshot of a Windows MS-DOS Prompt window. The title bar reads "MS-DOS Prompt". The window contains the following text: "C:\ javac PrimeiraXicara", "C:\dir Prim*", "PrimeiraXicara.java", "PrimeiraXicara.class", and "C:_". The window has a standard Windows interface with a menu bar, a toolbar, and a scroll bar on the right.

Tecnologia Java

O Exemplo Mais Simples

3. Execute o executável (.class) gerado.



```
Prompt do MS-DOS

C:\dir Prim*
PrimeiraXicara.java
PrimeiraXicara.class

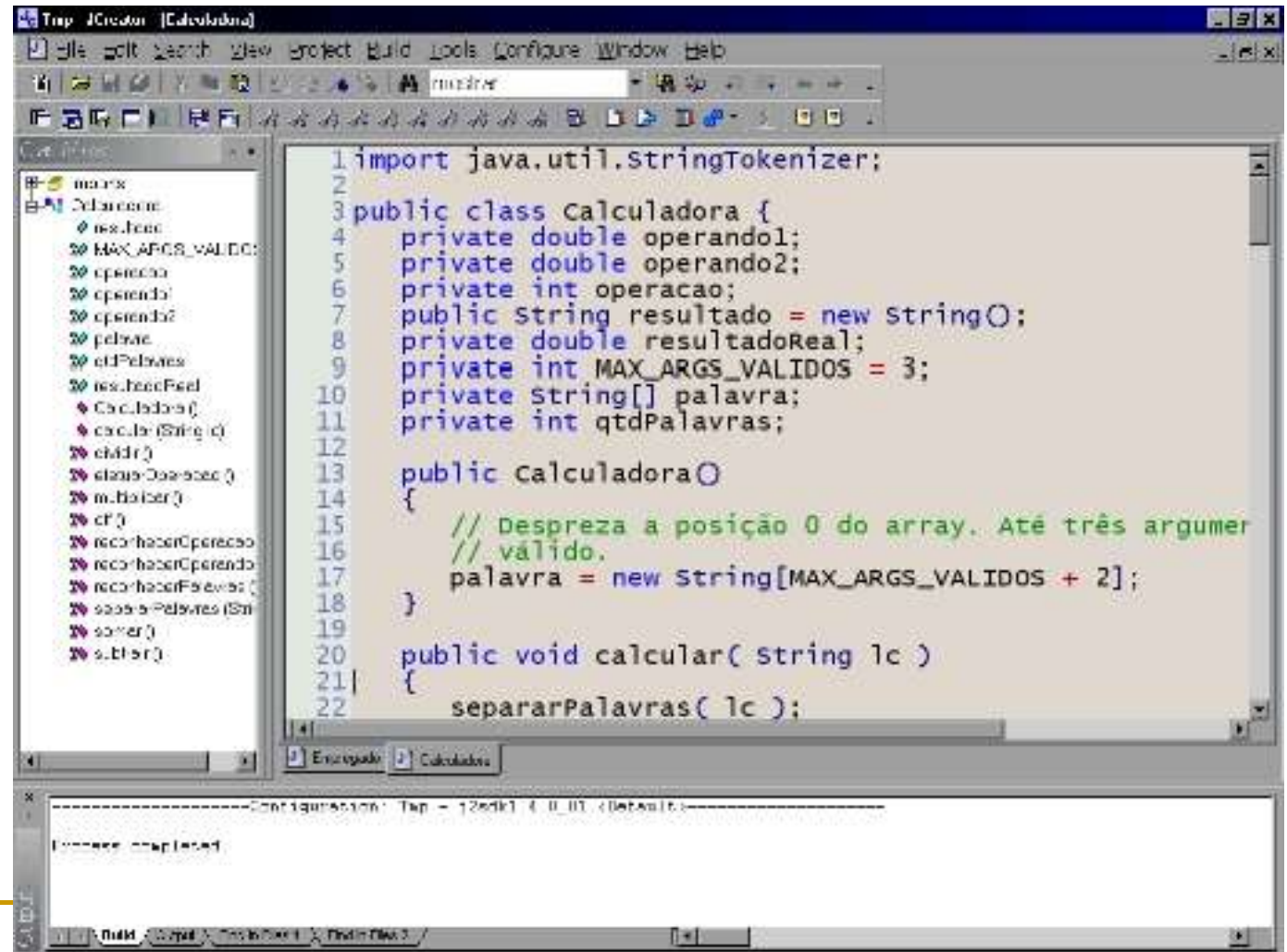
C:\java PrimeiraXicara
A Terra eh Azul!

C:\_
```

Tecnologia Java

Ambientes de Desenvolvimento

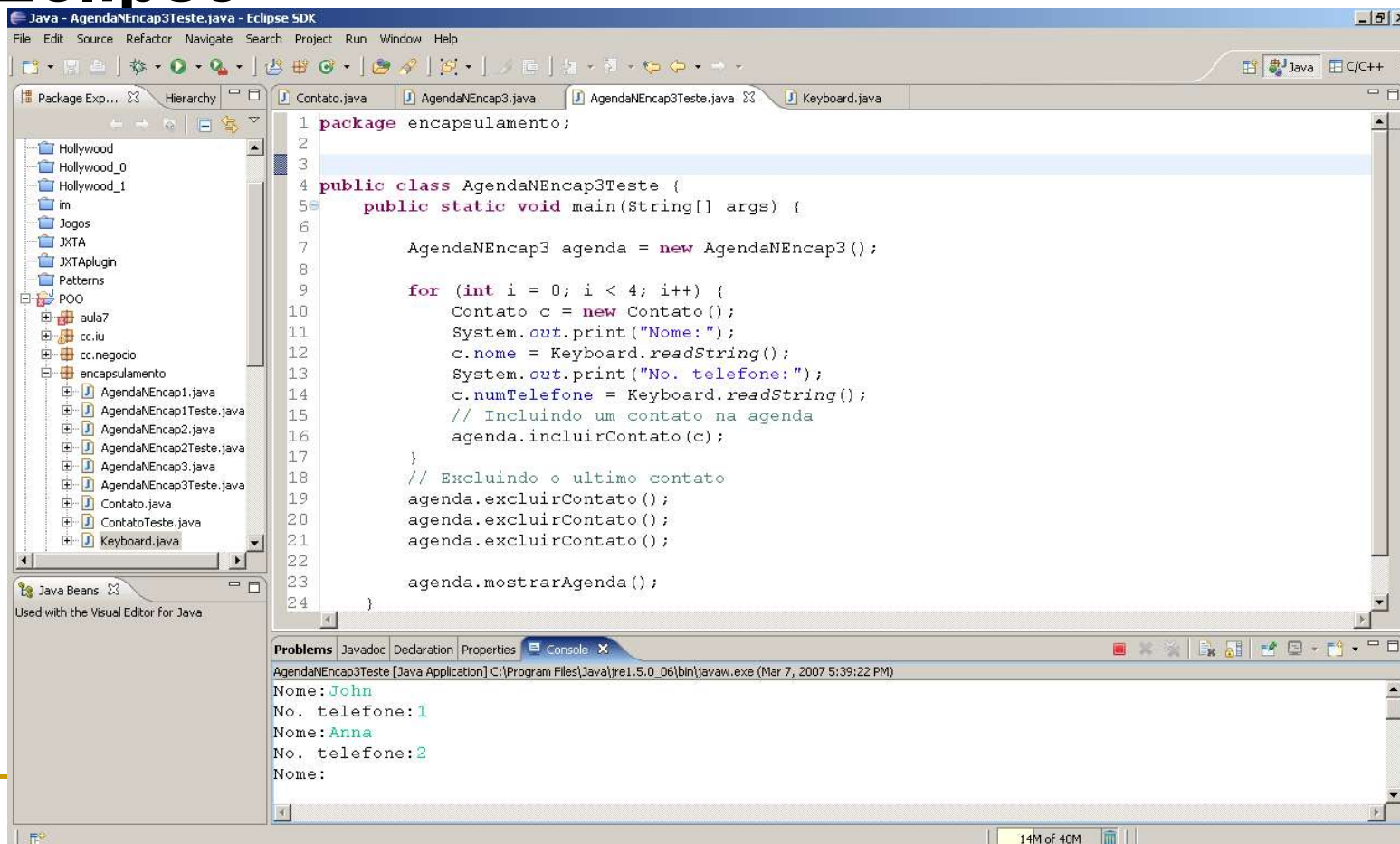
■ JCreator



Tecnologia Java

Ambientes de Desenvolvimento

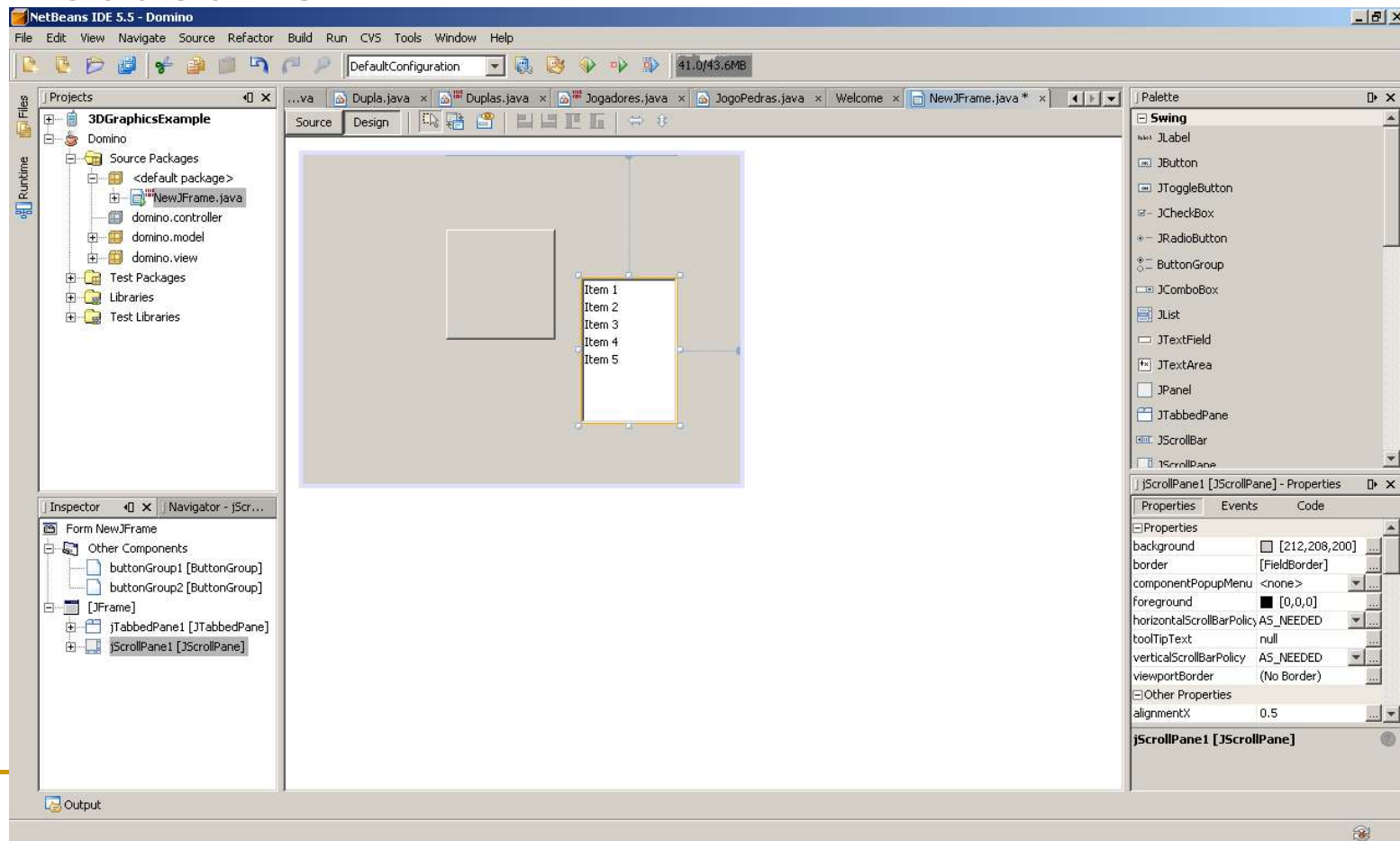
■ Eclipse



Tecnologia Java

Ambientes de Desenvolvimento

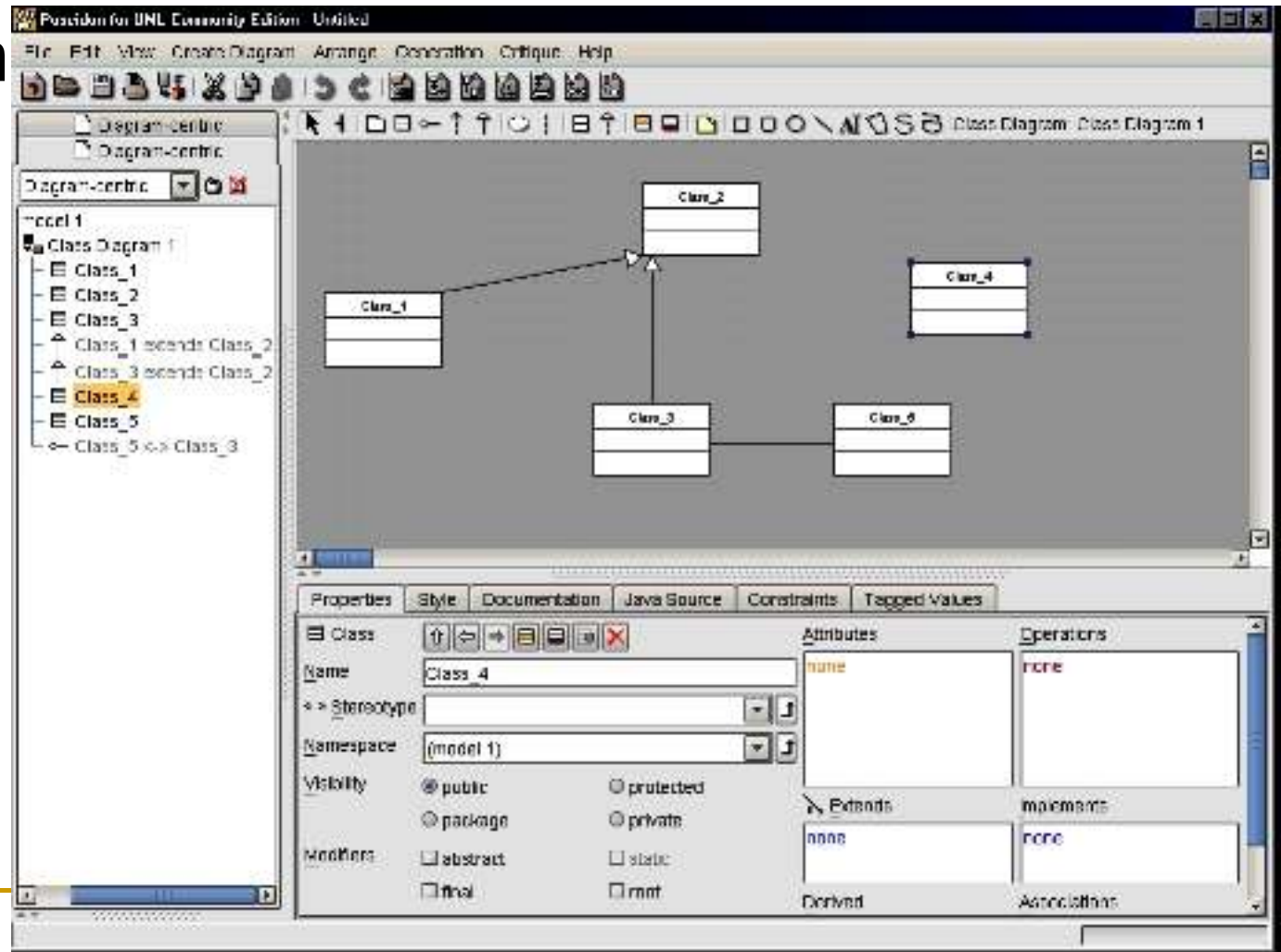
■ Netbeans IDE



Tecnologia Java

Ambientes de Desenvolvimento

■ Poseidon for UML



Tecnologia Java

Ambientes de Desenvolvimento

- E muitas outras ferramentas (de modelagem) tais como Jude, Enterprise Architect, Argo UML, ...

Elementos básicos do Java

Elementos Básicos do Java

Tipos de Dados Primitivos

- Por portabilidade, a faixa de valores dos tipos primitivos é o mesmo para diferentes plataformas.

Tipo Primitivo	Bits	Faixa de Valores		Classe
boolean	8	false	true	Boolean
char	16	Unicode 0	Unicode $2^{16} - 1$	Character
byte	8	-128	127	Byte
short	16	$-2^{15} - 1$	$+2^{15} - 1$	Short
int	32	$-2^{31} - 1$	$+2^{31} - 1$	Integer
long	64	$-2^{63} - 1$	$+2^{63} - 1$	Long
float	32	-3.40292347E+38	+3.40292347E+38	Float
double	64	-1.79769313486231570E+308	+1.79769313486231570E+308	Double
void	-	-	-	Void

Operadores Java

Categoria	Operadores	Exemplos
atribuição	=	a = 3; b = a; c = 'x';
incremento e decremento	++ (incremento), -- (decremento)	a++; ++a; a--; --a;
atribuição e ...	+=, -=, *=, /=, %=	a += 3; b *= w; c -= 0.5f;
matemáticos	+ (soma), - (subtração), * (produto), / (divisão), % (módulo)	s = a+b; x = a+(b-c)*4.7f; media = soma / qtd; resto = p % q;
igualdade	== (igual), != (diferente)	a == b; 3 != 4; x.obterValor() == y.obterValor();
relacionais	> (maior que), >= (maior ou igual), < (menor), <= (menor ou igual)	2 > 3; x <= 4; x >= 'a'; p.valor < q.valor;
lógicos	&& (e), (ou), ! (não)	a == 3 && b==4; 3 > 5 2 <= 3; !(x >= 1.0f)
unário + unário -	+, -	x = y * (-b);

Elementos Básicos do Java

Operadores Java

- Há também dois operadores especiais:
 - o operador **new** usado para a instanciação e objetos.
 - o operador **instanceof** verifica se um objeto é instância de uma classe.
- Os operadores relacionais e lógicos obtêm sempre, como resultado, um valor booleano (true ou false).
- Os operadores == e != trabalham com todos os tipos primitivos.
- Os operadores relacionais não comparam tipos booleanos.

Elementos Básicos do Java

Estruturas de Controle

- Estruturas de repetição:
 - **for, while, do-while**
- Estruturas de seleção:
 - **if, if-else e switch**
- Estrutura de seqüência:
 - **{ instr 1; instr 2; ...; instr *n*; }**

Elementos Básicos do Java

Estruturas de Controle

■ Estruturas de repetição: **for**

```
int y = 0;
for (int x = 0; x <= 10; x++) {
    y = x*x;
    System.out.print( "y=" + y );
    System.out.println();
}
```

Elementos Básicos do Java

Estruturas de Controle

■ Estruturas de repetição: **while**

```
int y = 0;
int x = 0;
while ( x <= 10 ) {
    y = x*x;
    System.out.print( "y=" + y );
    System.out.println();
    x++;
}
```

Elementos Básicos do Java

Estruturas de Controle

■ Estruturas de repetição: **do-while**

```
int y, x;  
x = y = 0;  
do {  
    y = x*x;  
    System.out.print( "y=" + y );  
    System.out.println();  
    x++;  
} while ( x <= 10 );
```

Elementos Básicos do Java

Estruturas de Controle

- A instrução ***break*** nas estruturas **while**, **do-while**, e **for**, fazem o fluxo de execução sair da estrutura.
- A instrução ***continue*** nas estruturas **while**, **do-while**, e **for**, fazem o fluxo de execução saltar para a próxima iteração.
- **Cuidado:** ***continue*** combinado com **while** e **do-while** pode resultar em loop infinito.

Elementos Básicos do Java

Estruturas de Controle

■ Estruturas de seleção: **if**

```
if ( a > b) {  
    System.out.println( a + " é mais  
                          pesado que " + b);  
    b++;  
}
```

Elementos Básicos do Java

Estruturas de Controle

■ Estruturas de seleção: **if-else**

```
if ( a > b ) {  
    System.out.println( a + " é mais  
                          pesado que " + b );  
}  
else {  
    System.out.println(b + " é mais  
                        pesado que " + a );  
}
```

Elementos Básicos do Java

Estruturas de Controle

■ Estruturas de seleção: **switch**

```
switch (x) {  
    case 1:  
        x++;  
        System.out.println( "somei 1" );  
    break;  
    case 2:  
        x--;  
    break;  
    default: x = 0; // opcional  
}
```

Bibliografia

- Fowler, Martin. UML Essencial. 2o. ed. – Porto Alegre: Bookman, 2000.
- Deitel, Harvey. Java - Como Programar. 6a. ed. Porto Alegre: Bookman, 2006.
- Eckel, Bruce. Thinking in Java. 3a. edição. Prentice-Hall, 2001.