

Organização de programas em Java

Leonardo Gresta Paulino Murta
leomurta@ic.uff.br

Vamos programar em Java! Mas...

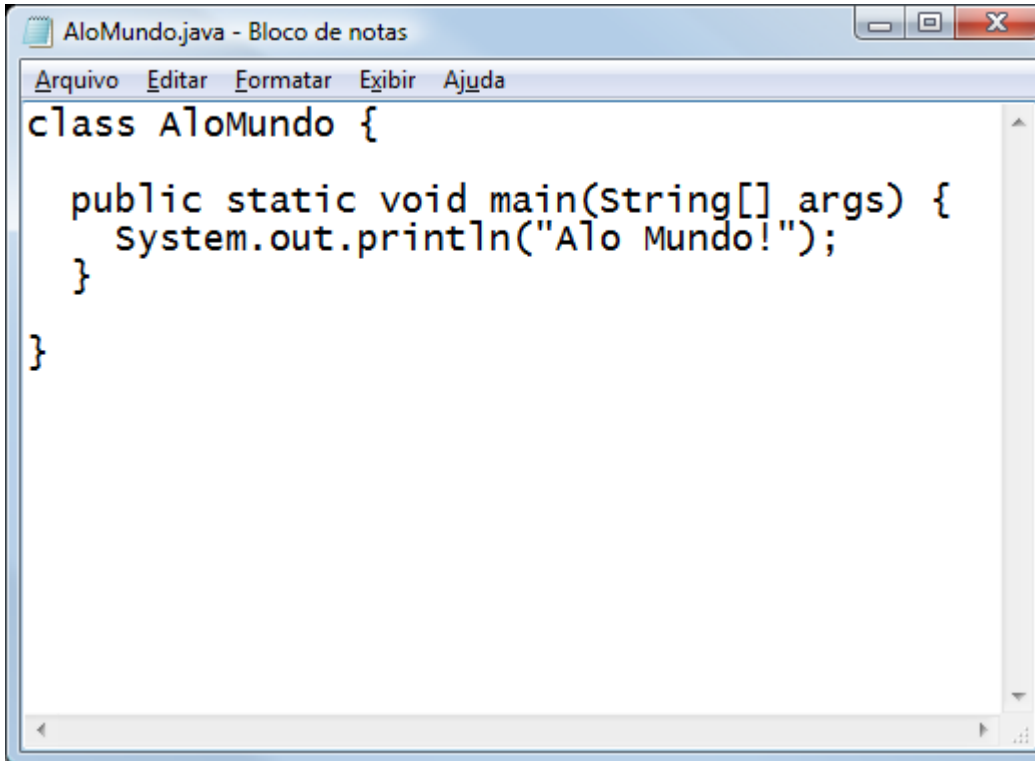
- Como um programa é organizado?
- Quais são os tipos de dados disponíveis?
- Como variáveis podem ser declaradas?
- Como atribuir valores às variáveis?
- Como entrada e saída básica de dados podem ser feitas?

Instalação do JDK

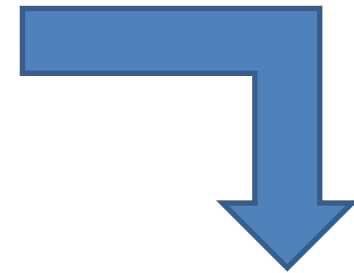
- Download do JDK
 - <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
- Versão mais recente para plataforma Java SE
- Programas principais
 - javac (compilador)
 - java (máquina virtual)



Primeiro passo: escrever o programa!

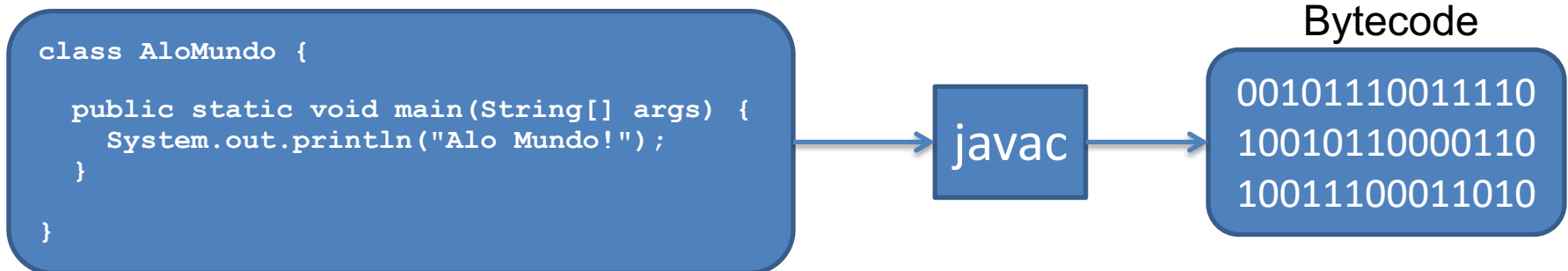
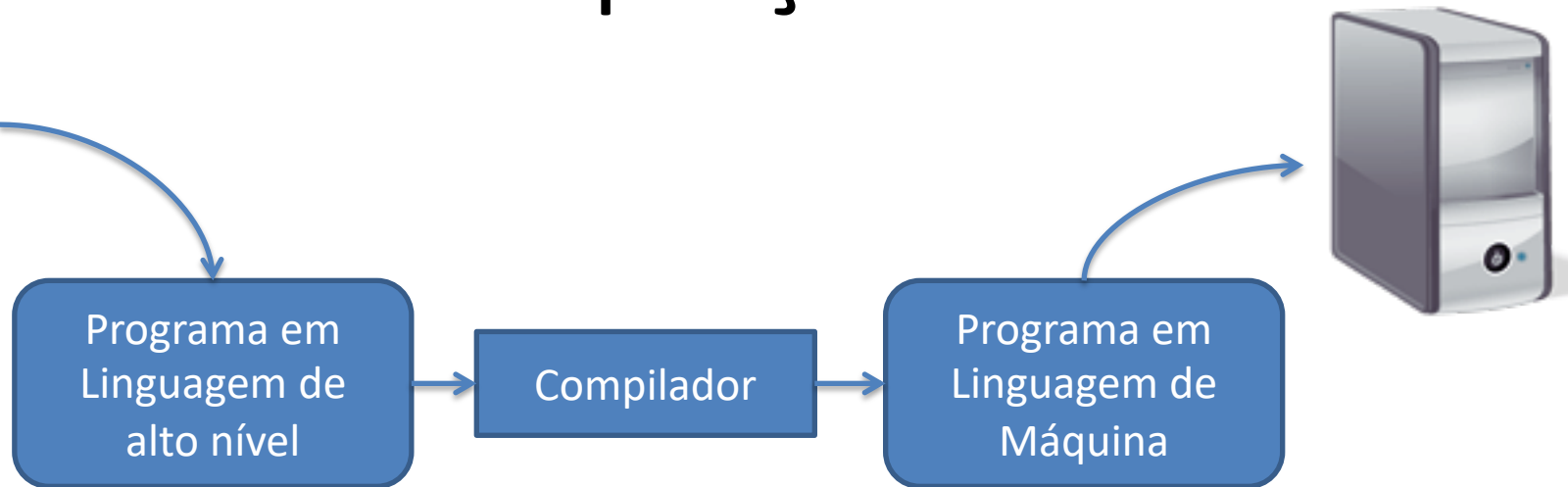


```
class AloMundo {  
    public static void main(String[] args) {  
        System.out.println("Alo Mundo!");  
    }  
}
```



AloMundo.java

Compilação



Compilação

```

C:\Windows\system32\cmd.exe

c:\Users\leomurta\prog1>dir
O volume na unidade C é OS
O Número de Série do Volume é 3A6F-C6C3

Pasta de c:\Users\leomurta\prog1
26/03/2012  10:54    <DIR>          .
26/03/2012  10:54    <DIR>          ..
26/03/2012  10:36                111 A!oMundo.java
                1 arquivo(s)                111 bytes
                2 pasta(s)  153.176.731.648 bytes disponíveis

c:\Users\leomurta\prog1>javac A!oMundo.java

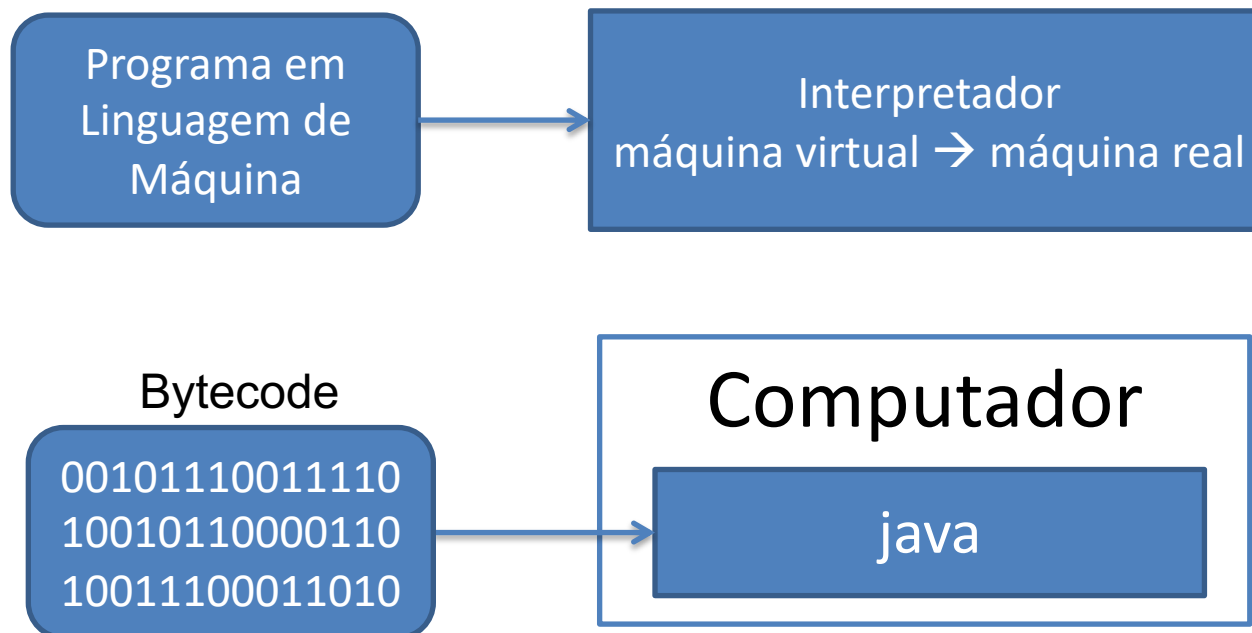
c:\Users\leomurta\prog1>dir
O volume na unidade C é OS
O Número de Série do Volume é 3A6F-C6C3

Pasta de c:\Users\leomurta\prog1
26/03/2012  10:55    <DIR>          .
26/03/2012  10:55    <DIR>          ..
26/03/2012  10:55                420 A!oMundo.class
26/03/2012  10:36                111 A!oMundo.java
                2 arquivo(s)                531 bytes
                2 pasta(s)  153.176.616.960 bytes disponíveis

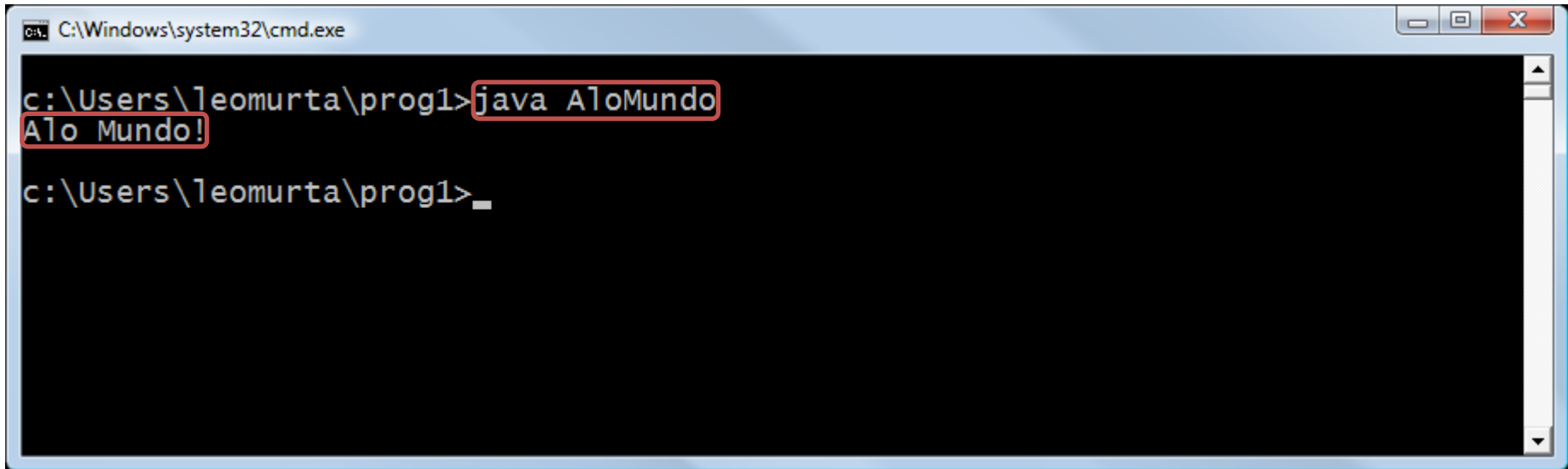
c:\Users\leomurta\prog1>

```

Execução



Execução



```
C:\Windows\system32\cmd.exe  
c:\Users\leomurta\prog1>java AtoMundo  
Ato Mundo!  
c:\Users\leomurta\prog1>_
```

VAMOS FAZER JUNTOS?

Notepad x IDE

- Dificuldades do Notepad
 - Editor básico, sem ajuda para programar
 - Compilação externa
 - Execução externa
- *Integrated Development Environment (IDE)*



Apache NetBeans



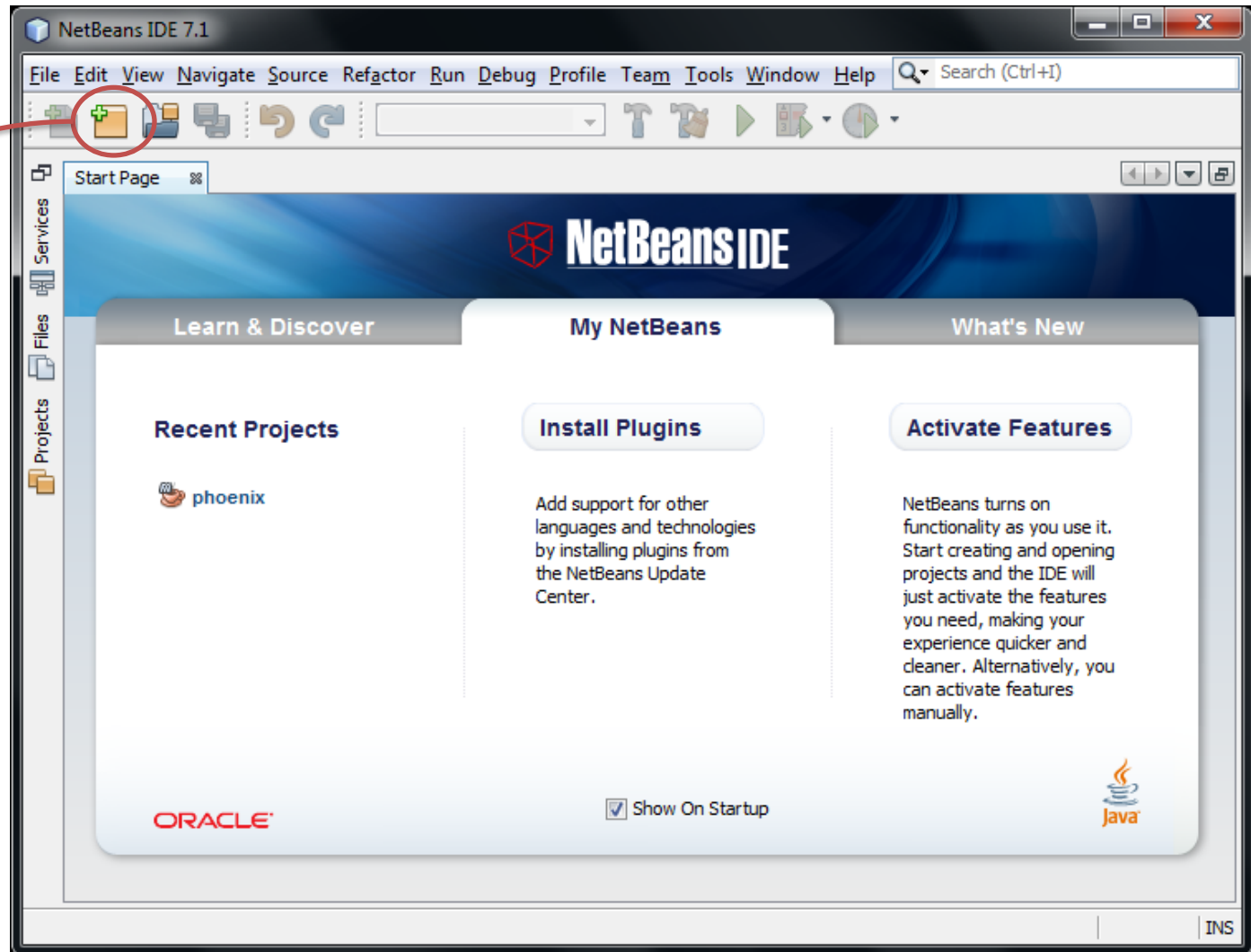
Instalação do NetBeans

- Usaremos o NetBeans neste curso
- Download do NetBeans
 - <https://netbeans.apache.org/download/index.html>
 - Importante: baixar a versão mais recente



Criando o projeto no NetBeans...

Clicar neste
ícone para
criar um
novo
projeto



Criando o projeto no NetBeans...

Selecionar categoria **Java** e projeto do tipo **Java Application**, e clicar em **Next** ao final

Steps

1. Choose Project
2. ...

Choose Project

Categories:

- Java
- JavaFX
- Java Web
- Java EE
- Java Card
- Java ME
- Maven
- PHP
- Groovy
- C/C++
- NetBeans Modules
- Samples

Projects:

- Java Application
- Java Class Library
- Java Project with Existing Sources
- Java Free-Form Project

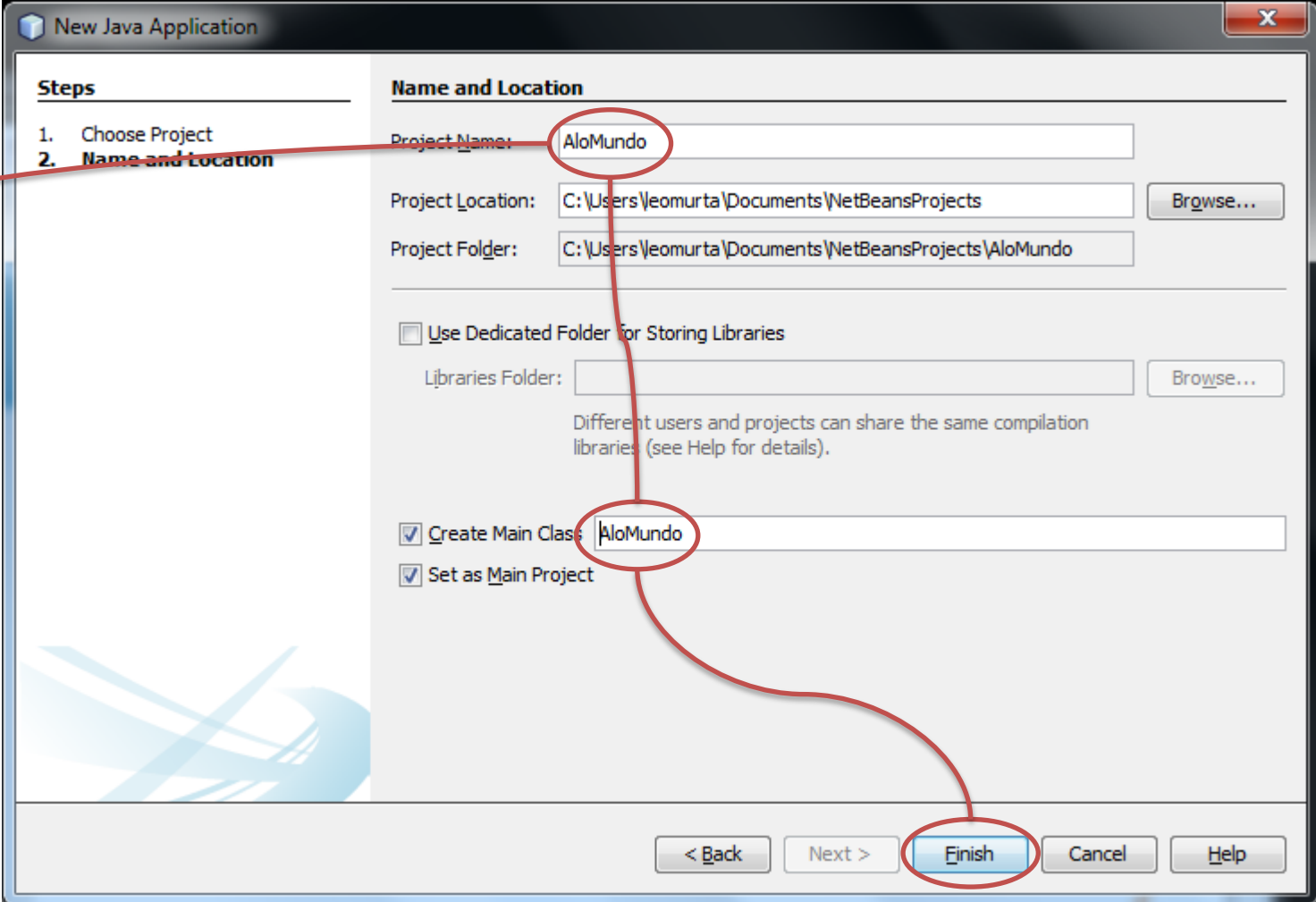
Description:

Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use an IDE-generated Ant build script to build, run, and debug your project.

< Back **Next >** Finish Cancel Help

Criando o projeto no NetBeans...

Definir o nome do projeto e da classe principal, e clicar em **Finish** ao final



Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: AloMundo

Project Location: C:\Users\leomurta\Documents\NetBeansProjects Browse...

Project Folder: C:\Users\leomurta\Documents\NetBeansProjects\AloMundo

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder: Browse...

Different users and projects can share the same compilation libraries (see Help for details).

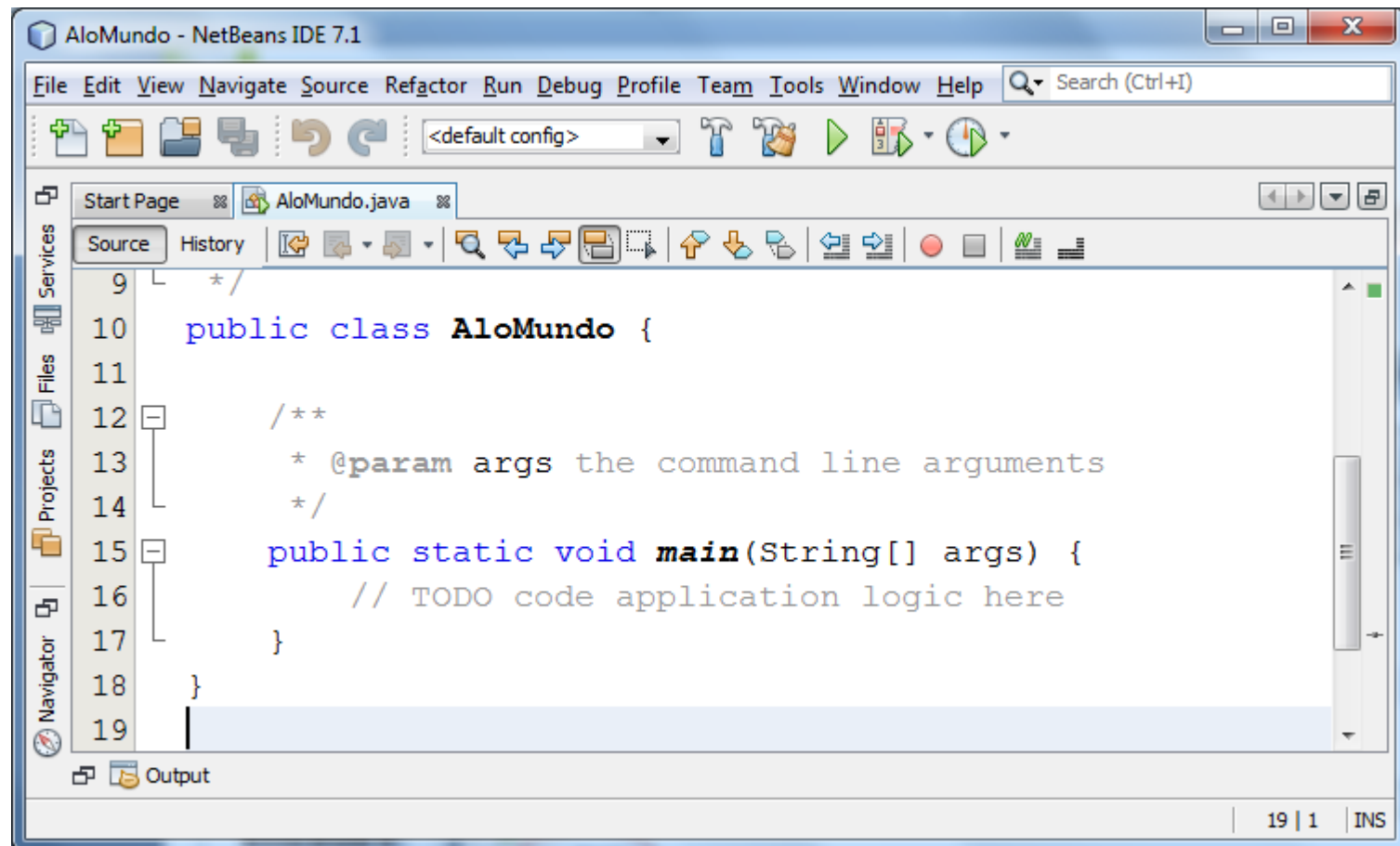
☒ Create Main Class AloMundo

☒ Set as Main Project

< Back Next > **Finish** Cancel Help

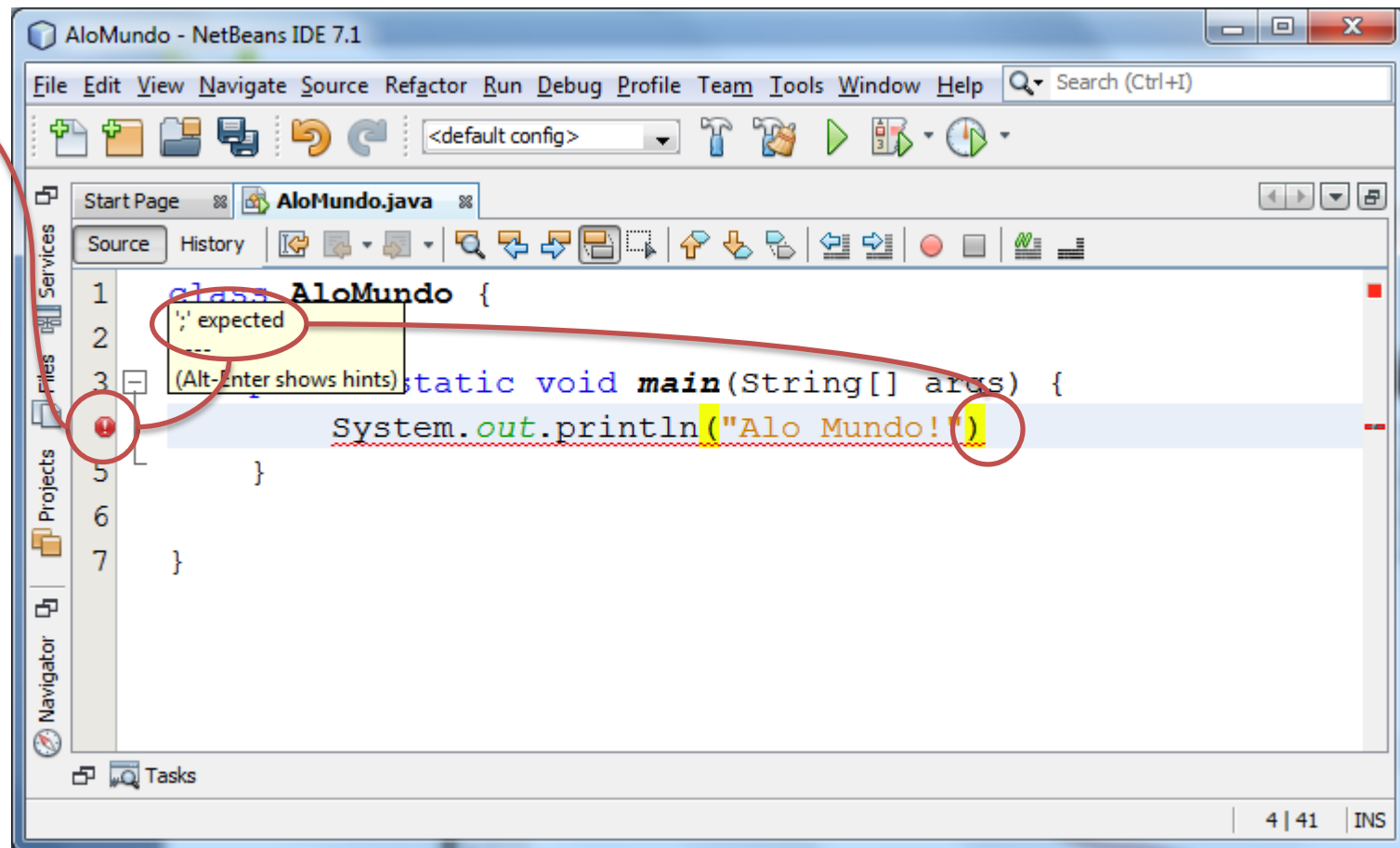
Criando o projeto no NetBeans...

Geração
automática
do
esqueleto
do
programa



Escrevendo e compilando o programa no NetBeans...

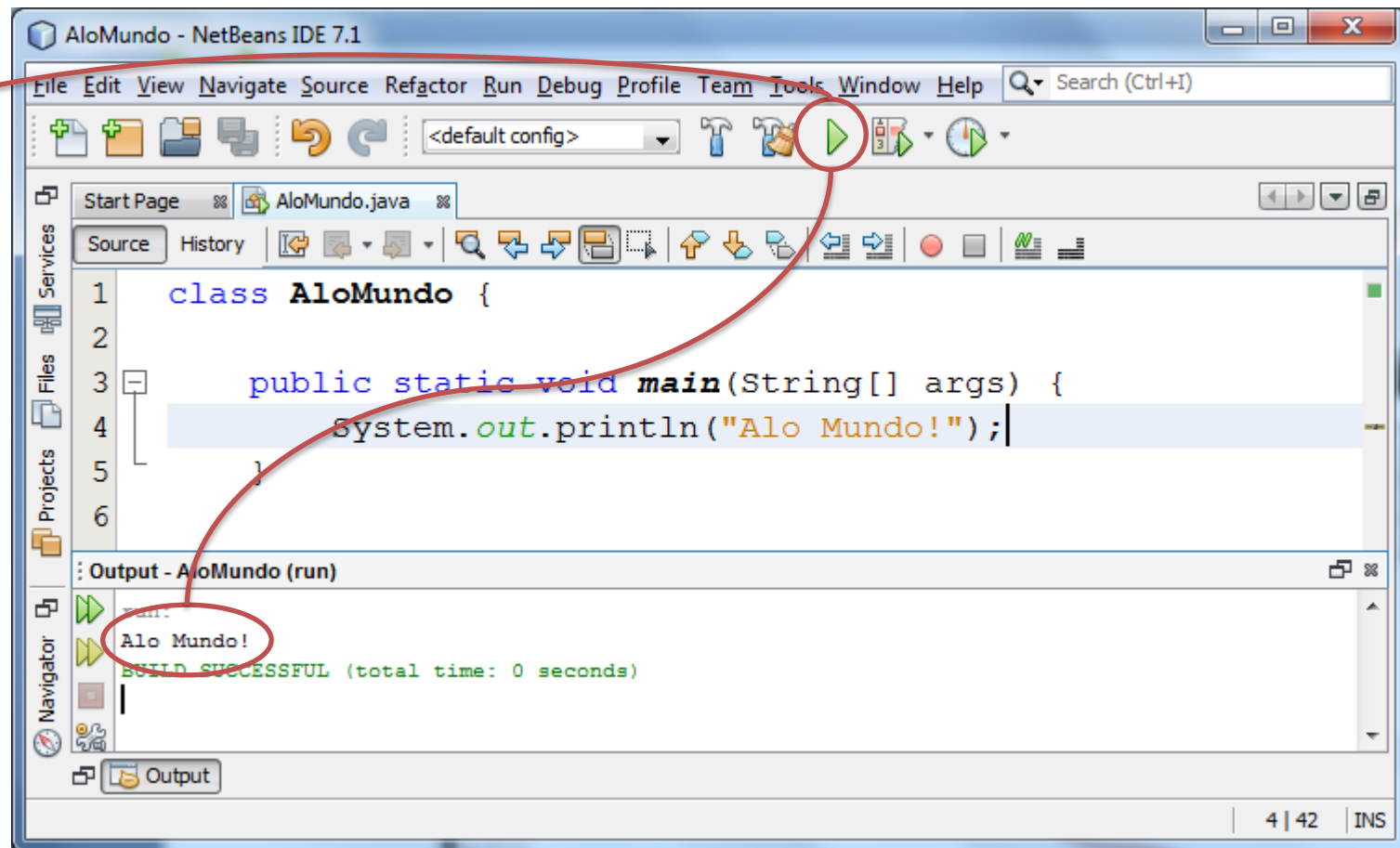
Compilação automática
durante a
edição do
código e
avisos
sobre erros



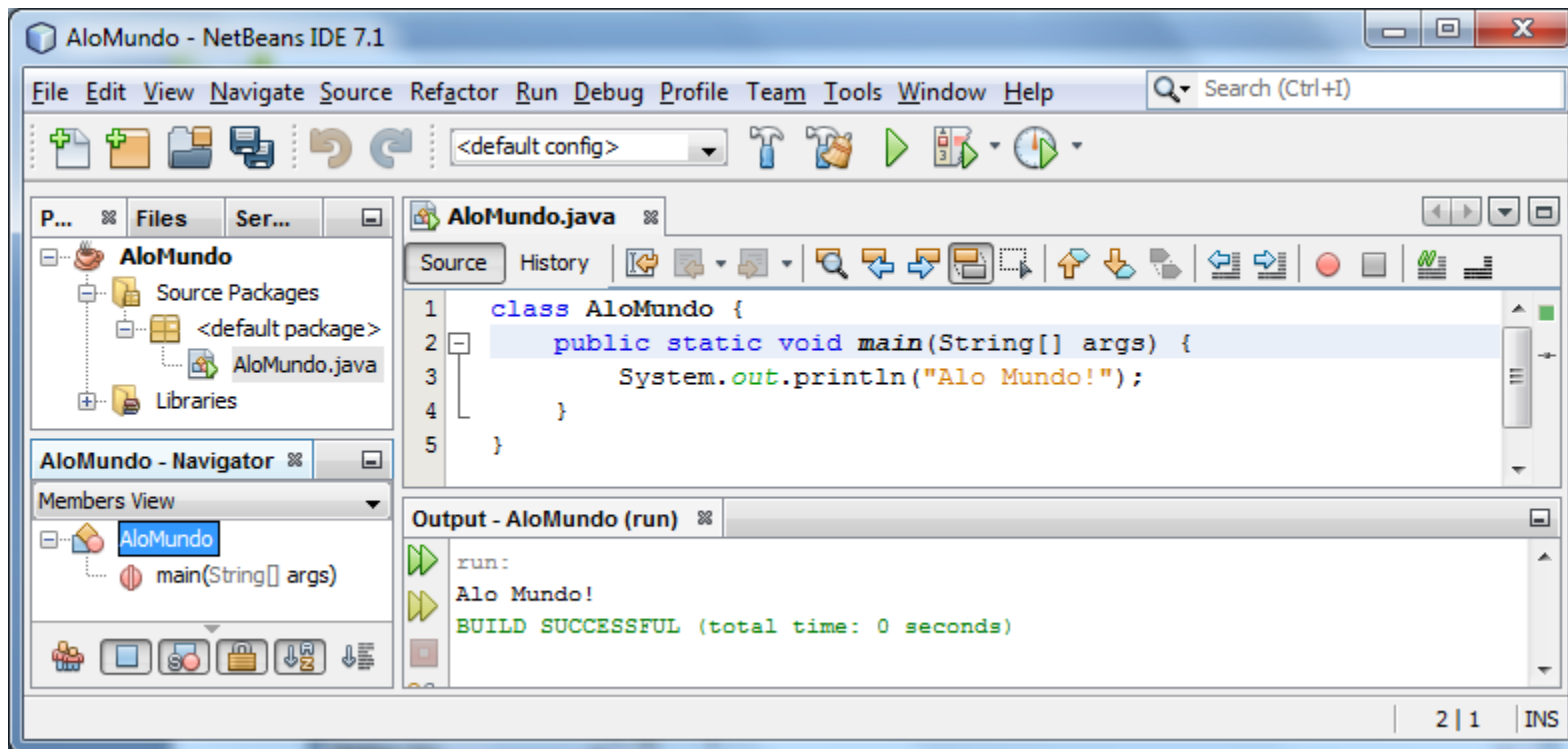
Executando o programa no NetBeans...

Clicar neste ícone para executar o programa

No painel inferior ocorrerá a entrada e saída de dados



Escrevendo, compilando e executando o programa no NetBeans...



VAMOS FAZER JUNTOS?

Organização geral de um programa Java

- Nesse momento, abstrairemos um pouco a Orientação a Objetos
 - Depois veremos como isso funciona

```
import BIBLIOTECA EXTERNA;  
class NOME DO PROGRAMA {  
    public static void main(String[] args) {  
        CÓDIGO DO PROGRAMA  
    }  
}
```

Regras básicas

; no final dos comandos!

{ e } delimitam blocos!

Comentários

- Comentários são trechos do programa voltados para a leitura por humanos, e ignorados pela JVM
- Existem diferentes formas de escrever comentário
- **`/* COMENTÁRIO */`**
 - Conhecido como comentário de bloco
 - Tudo entre `/*` e `*/` é ignorado pelo interpretador
- **`// COMENTÁRIO`**
 - Conhecido como comentário de linha
 - Tudo na linha após `//` é ignorado pelo interpretador

Exemplo de programa em Java

```
import java.util.Scanner;

/* Este programa calcula a área
   de um triangulo retângulo */
class Triangulo {
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in); //Leitor do teclado
        int altura, base; //Dados de entrada
        float area; //Dados de saída

        System.out.print("Informe a altura: ");
        altura = teclado.nextInt();
        System.out.print("Informe a base: ");
        base = teclado.nextInt();
        area = 0.5f * altura * base;
        System.out.println("Área: " + area);
    }
}
```

Quais são os tipos de dados disponíveis?

- Em Java, toda variável tem que ter um tipo
- Com isso, o computador pode **reservar o espaço correto de memória**
- Os tipos básicos podem ser divididos em dois grupos
 - Tipos numéricos (inteiro e real)
 - Tipos não numéricos (caractere e booleano)
- Também existe texto como tipo complexo (classe)
 - String

Números inteiros

- byte
 - 8-bits (aceita valores de -128 a 127)
- short
 - 16-bits (aceita valores de -32.768 a 32.767)
- int
 - 32-bits (aceita valores de -2.147.483.648 a 2.147.483.647)
- long
 - 64-bit (aceita valores de -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807)
- Por padrão, qq número inteiro é do tipo int
 - Para forçar long, deve-se adicionar L ou l ao final (ex. 123L)

Exemplos de números inteiros

- byte
 - -5
 - 10
 - 120
- short
 - -1234
 - 10
 - 29090
- int
 - -12312312
 - 10
 - 345092834
- long
 - -12343212
 - 10
 - 45323565432L

Números reais

- float
 - Precisão simples 32-bits (IEEE 754 SPFP)
 - Precisão de 7 casas decimais com magnitude de 10^{38}
- double
 - Precisão dupla 64-bits (IEEE 754 DPFP)
 - Precisão de 15 casas decimais com magnitude de 10^{308}
- Por padrão, qq número real é do tipo double
 - Para forçar float, deve adicionar F ou f ao final (ex. 0.5f)
- Notação científica pode ser utilizada (ex. 0.5e3)

Exemplos de números reais

- float
 - -21.4f
 - 0.0000034f
 - 123456.0f
 - 0.6023e24f
 - 0.4e-3f
 - -0.5E2f
 - 15f
 - 15F
- double
 - 0.23e-94
 - 0.54336543454323e-7
 - 0.0000034
 - 0.4e-3
 - 0.4E-3d
 - 12345d
 - 15d
 - 15D

Outros tipos de dados

- char
 - Caractere 16-bit (Unicode)
- String
 - Texto de tamanho variável
- boolean
 - Tipo lógico, com valores *true* ou *false*

Exemplos de outros tipos de dados

- char
 - 'A'
 - 'b'
 - '4'
- String
 - ""
 - "Olá mundo!"
 - "4"
- boolean
 - true
 - false

Valores padrão

- Algumas linguagens não limpam o espaço de memória ao alocar uma nova variável
- Java toma esse cuidado para nós
 - Tipos numéricos são inicializados com 0
 - Tipo char é iniciado com ‘\u0000’
 - Tipo booleano é inicializado com *false*
- De qualquer forma, sempre inicialize as suas variáveis por precaução

Declaração de variáveis

- Para serem usadas, as variáveis precisam ser declaradas (criadas)
- Toda variável é declarada da seguinte forma:

```
TIPO NOME = VALOR INICIAL;
```

ou

```
TIPO NOME1, NOME2, ...;
```

Declaração de variáveis

- Os tipos são os que já vimos, assim como os valores iniciais possíveis
- Os nomes devem respeitar algumas regras
 - São sensíveis a caixa
 - Podem ter tamanho ilimitado (mas evite abusos)
 - Devem começar com letra, seguida de letras ou números
 - Não podem ter espaço nem acentos
 - Não podem ser uma palavra reservada da linguagem
- Usualmente nomes compostos de variáveis seguem a notação *Camel Case* iniciando com minúsculas, com conectores (de, e, ou, etc.) omitidos e demais palavras concatenadas iniciando com maiúsculas

Declaração de variáveis

- Um caso especial é referente a variáveis que nunca trocam de valor
 - Mais conhecidas como **constantes**
- Em Java, constantes são identificadas com o modificador *final* antes do tipo
- Usualmente, os nomes de constantes são em maiúsculas com as palavras separadas por *underscore* (`_`) quando for um nome composto

Atribuição de valores

- Em Java, o operador de igualdade (=) é usado para atribuir valores às variáveis
- Sempre na forma: **variável = valor ou expressão**
 - A expressão do lado direito é avaliada
 - O valor gerado é atribuído à variável

Como variáveis podem ser declaradas? (exemplos)

- `int idade = 15;`
- `int minutos = horas * 60;`
- `final float ACELERACAO_GRAVIDADE = 9.80665f;`
- `final double PI = 3.14159265358979;`
- `String melhorTimeFutebol = "Flamengo";`
- `boolean gostoJava = true;`
- `String nome, endereco, telefone;`
- `int ano, mes, dia;`

Entrada de dados

- Para entrada de dados, é necessário usar uma classe externa responsável por interpretar o que foi escrito
 - `java.util.Scanner`
- Para não ter que escrever o nome completo da classe a cada uso, é possível **importar a classe** para o seu programa
 - `import java.util.Scanner;`
 - A partir desse momento, a máquina virtual Java sabe onde encontrar a classe (no pacote `java.util`), e nós podemos chamá-la somente pelo nome `Scanner`

Entrada de dados

- Além de importar a classe `Scanner`, é necessário criar uma variável que permita acessá-la
 - `Scanner teclado = new Scanner(System.in);`
- A partir desse ponto, a variável *teclado* pode ser usada para ler o que foi digitado
 - O `Scanner` permite leitura individualizada para diferentes tipos de dados
 - A leitura só ocorre de fato após o usuário teclar *Enter*

Entrada de dados

Tipo de dado a ser lido	Método
byte	Scanner.nextByte()
short	Scanner.nextShort()
int	Scanner.nextInt()
long	Scanner.nextLong()
float	Scanner.nextFloat()
double	Scanner.nextDouble()
boolean	Scanner.nextBoolean()
String	Scanner.next() Scanner.nextLine()

Saída de dados

- A saída de dados é mais simples, acessando direto a classe que representa o sistema
 - `java.lang.System`
- O pacote `java.lang` não precisa ser importado, pois é visível automaticamente a todos os programas
- A partir da classe `System`, é possível escrever qualquer tipo de dados (`x`)
 - `System.out.print(x)`
 - `System.out.println(x)`

Exemplo de entrada e saída de dados

- `int nota = teclado.nextInt();`
- `nome = teclado.nextLine();`
- `altura = teclado.nextFloat();`
- `System.out.print("Java é muito legal!");`
- `System.out.println(123);`
- `System.out.println(teclado.nextLine());`

Exercícios

- Qual a saída do programa abaixo?

```
class Atribuicoes {  
    public static void main(String[] args) {  
        float x = 1.0f;  
        float y = 2.0f;  
        float z = 3.0f;  
  
        x = -x;  
        y = y - 1;  
        z = z + x;  
        z = z + x - y;  
        System.out.println("x = "+x+", y = "+y+", z = "+z);  
    }  
}
```


Exercícios

- Faça um programa para, a partir de um valor informado em centavos, indicar a menor quantidade de moedas que representa esse valor
 - Considere moedas de 1, 5, 10, 25 e 50 centavos, e 1 real
 - Exemplo: para o valor 290 centavos, a menor quantidade de moedas é 2 moedas de 1 real, 1 moeda de 50 centavos, 1 moeda de 25 centavos, 1 moeda de 10 centavos e 1 moeda de 5 centavos

Organização de programas em Java

Leonardo Gresta Paulino Murta
leomurta@ic.uff.br