

Organização de programas em Java

Orientação a Objetos – DCC025

Gleiph Ghiotto Lima de Menezes

gleiph.ghiotto@ufjf.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?

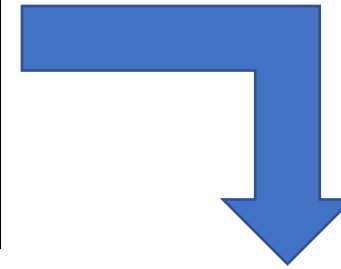
Instalação do JDK

- Download do JDK
 - <https://www.oracle.com/br/java/technologies/downloads/>
- Versão LTS mais recente para plataforma Java SE
 - JDK 21
- 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



Programa em
Linguagem de
alto nível

Compilador

Programa em
Linguagem de
Máquina



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

javac

Bytecode

```
00101110011110
10010110000110
10011100011010
```

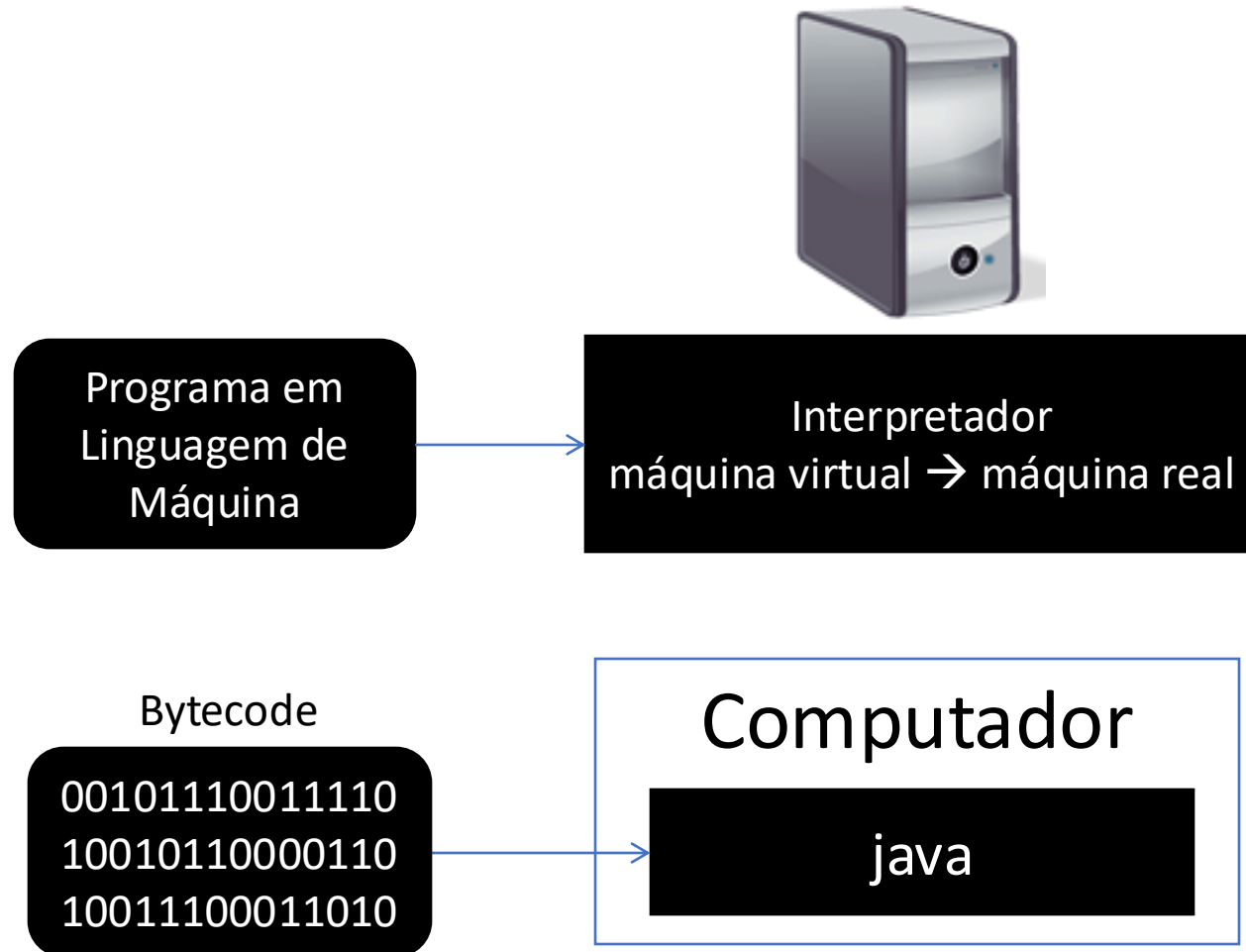
Compilação

```

DCC025 — -bash — 80x24
[Gleiphs-MacBook-Pro:DCC025 gleiph$ ls -la
total 8
drwxr-xr-x  3 gleiph  staff   96 Mar 12 02:50 .
drwx-----+ 92 gleiph  staff 2944 Mar 12 02:50 ..
-rw-r--r--@  1 gleiph  staff  111 Mar 12 02:49 AloMundo.java
[Gleiphs-MacBook-Pro:DCC025 gleiph$ javac AloMundo.java
[Gleiphs-MacBook-Pro:DCC025 gleiph$ ls -la
total 16
drwxr-xr-x  4 gleiph  staff   128 Mar 12 02:50 .
drwx-----+ 92 gleiph  staff 2944 Mar 12 02:50 ..
-rw-r--r--  1 gleiph  staff   420 Mar 12 02:50 AloMundo.class
-rw-r--r--@  1 gleiph  staff   111 Mar 12 02:49 AloMundo.java
Gleiphs-MacBook-Pro:DCC025 gleiph$

```

Execução



Execução

```

DCC025 — -bash — 80x24
Gleiphs-MacBook-Pro:DCC025 gleiph$ java AloMundo
Alo mundo!
Gleiphs-MacBook-Pro:DCC025 gleiph$

```

VAMOS FAZER JUNTOS?

Java 21

```
gleiph@Gleiph-Legion5i: ~/Documents/UFJF/2024.1/DCC025/Projetos/ExemploMain
gleiph@Gleiph-Legion5i:~/Documents/UFJF/2024.1/DCC025/Projetos/ExemploMain$ ls -la
total 12
drwxrwxr-x 2 gleiph gleiph 4096 mar 18 00:03 .
drwxrwxr-x 4 gleiph gleiph 4096 mar 17 23:50 ..
-rw-rw-r-- 1 gleiph gleiph  53 mar 18 00:01 Hello.java
gleiph@Gleiph-Legion5i:~/Documents/UFJF/2024.1/DCC025/Projetos/ExemploMain$ cat Hello.java
void main(){
    System.out.println("Olá mundo");
}

gleiph@Gleiph-Legion5i:~/Documents/UFJF/2024.1/DCC025/Projetos/ExemploMain$ javac --release 21 --enable-preview -Xlint:preview
Hello.java
Hello.java:1: warning: [preview] unnamed classes are a preview feature and may be removed in a future release.
void main(){
^
1 warning
gleiph@Gleiph-Legion5i:~/Documents/UFJF/2024.1/DCC025/Projetos/ExemploMain$ java --enable-preview Hello
Olá mundo
gleiph@Gleiph-Legion5i:~/Documents/UFJF/2024.1/DCC025/Projetos/ExemploMain$
```

Notepad x IDE

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



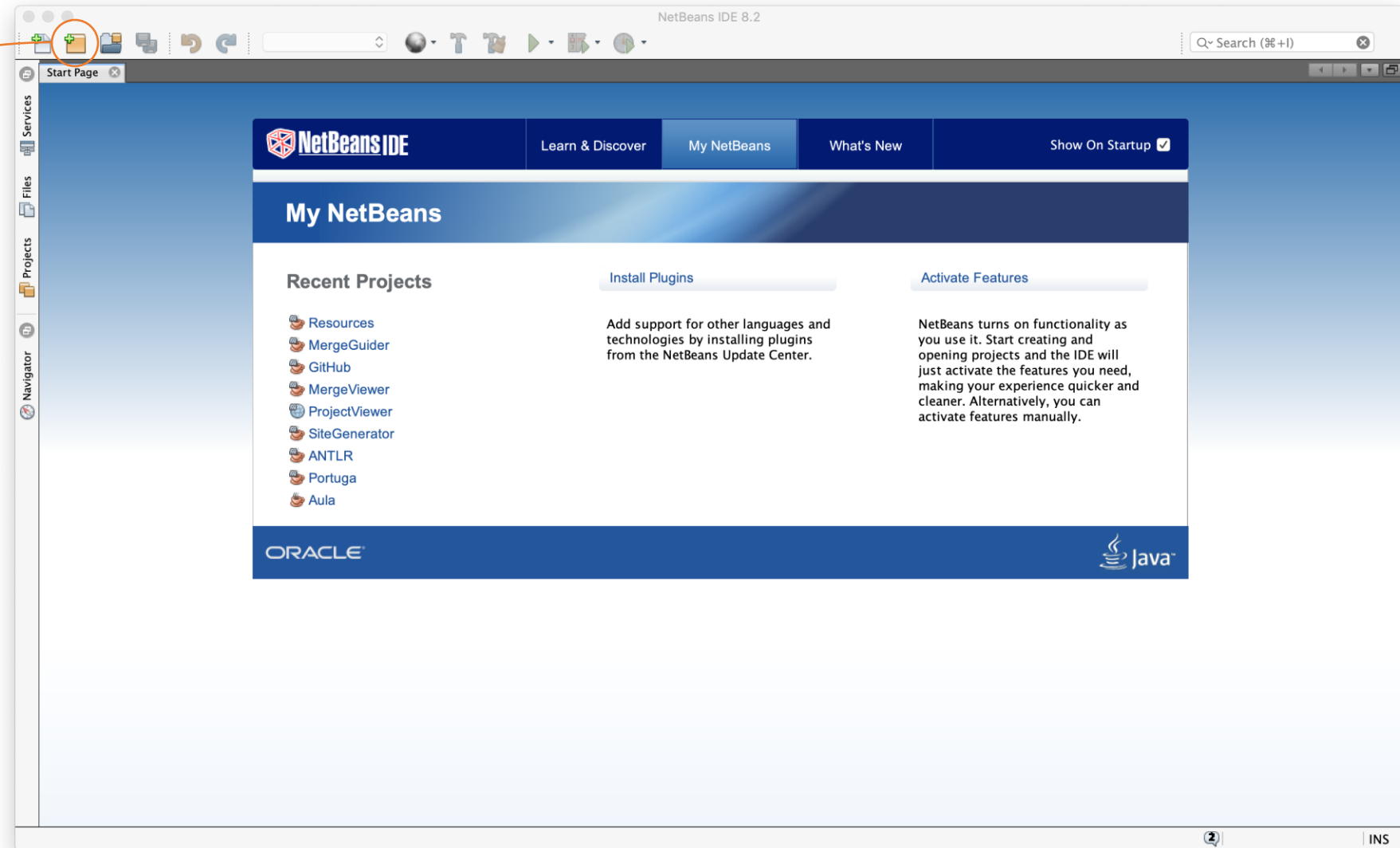
Instalação do NetBeans

- Usaremos o NetBeans neste curso
- Download do NetBeans
 - <http://netbeans.org/downloads>
 - Importante: baixar a distribuição Java SE na versão mais recente



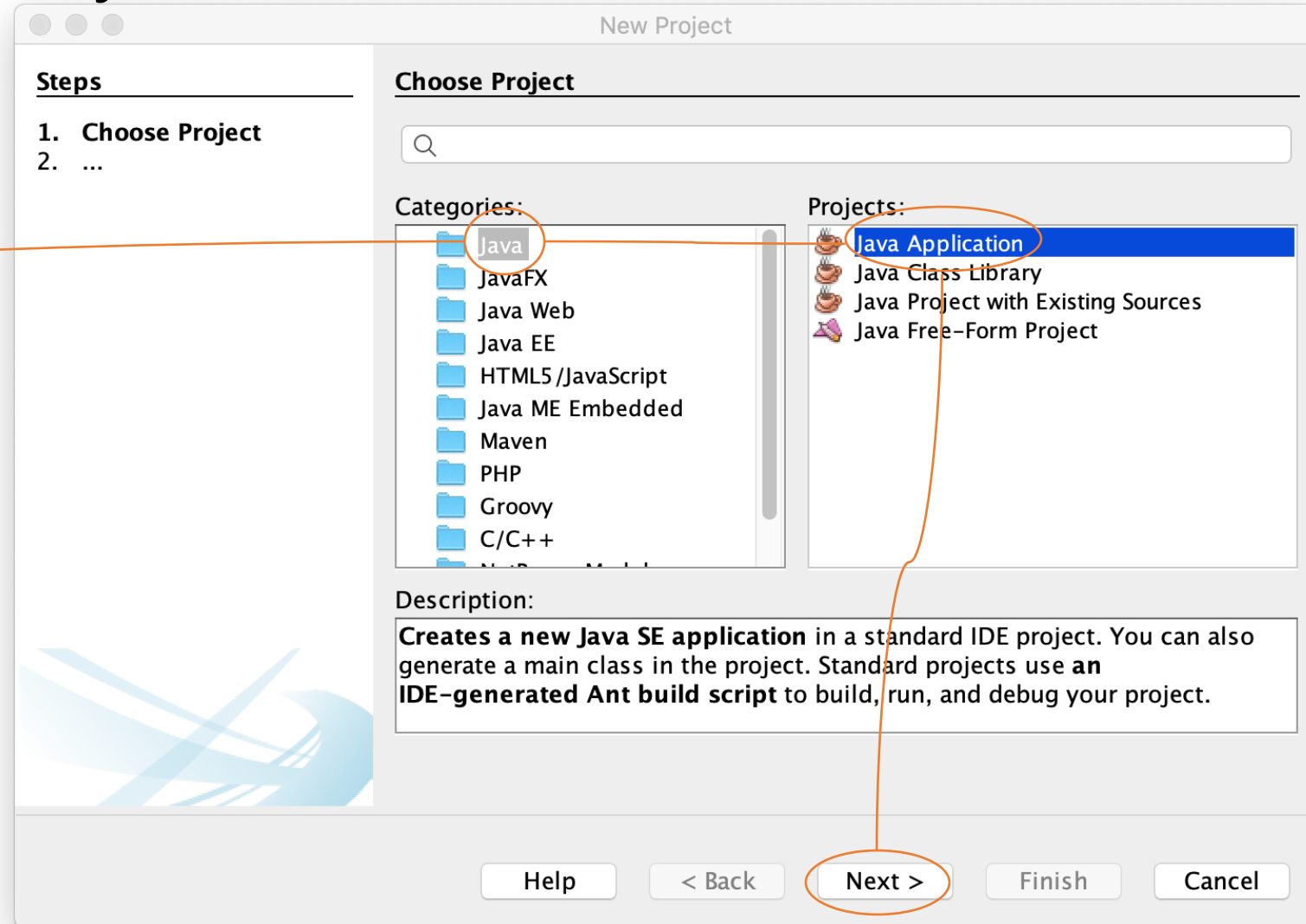
Criando o projeto no NetBeans...

Clicar neste
ícone para
**criar um
novo
projeto**



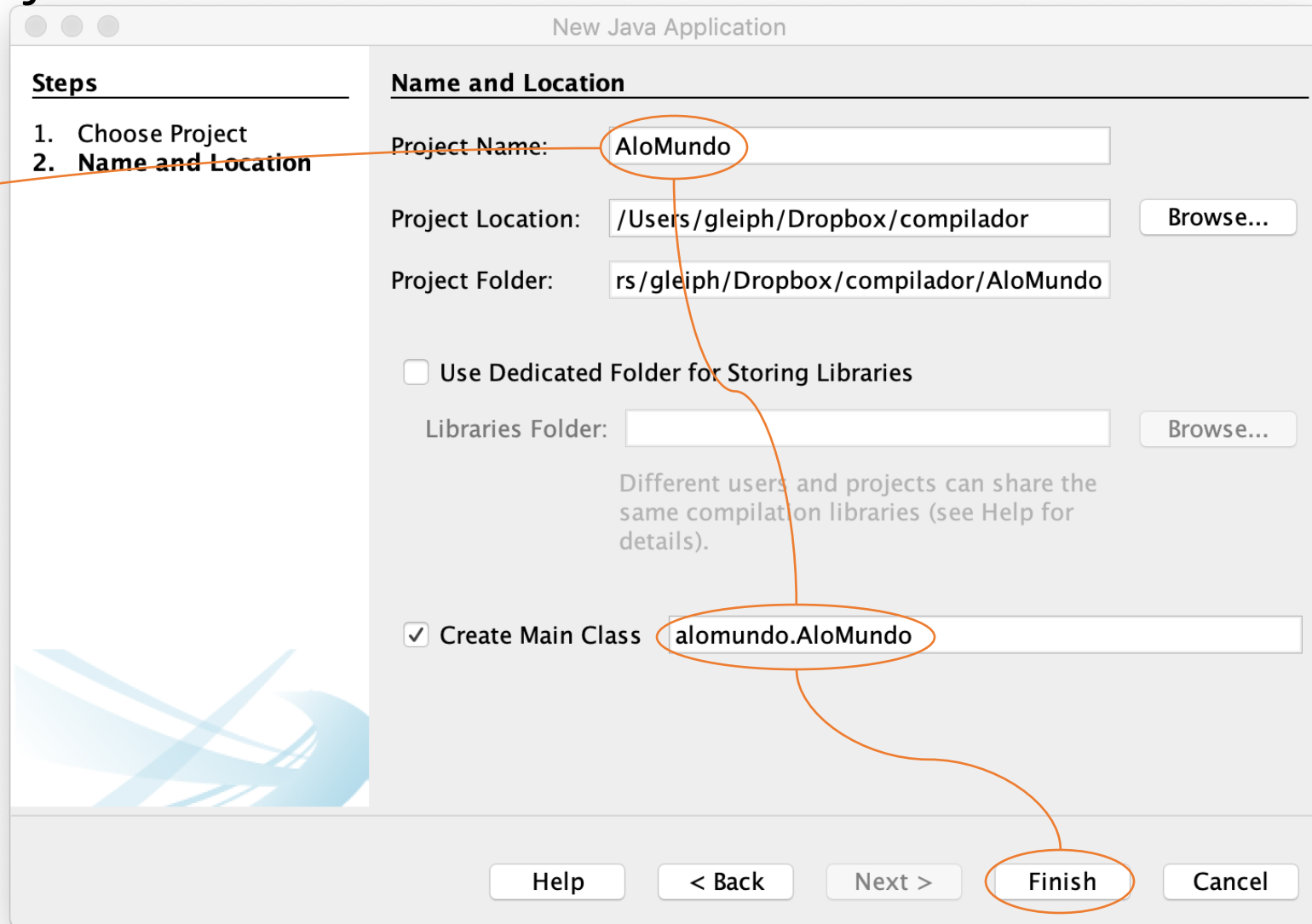
Criando o projeto no NetBeans...

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



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:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

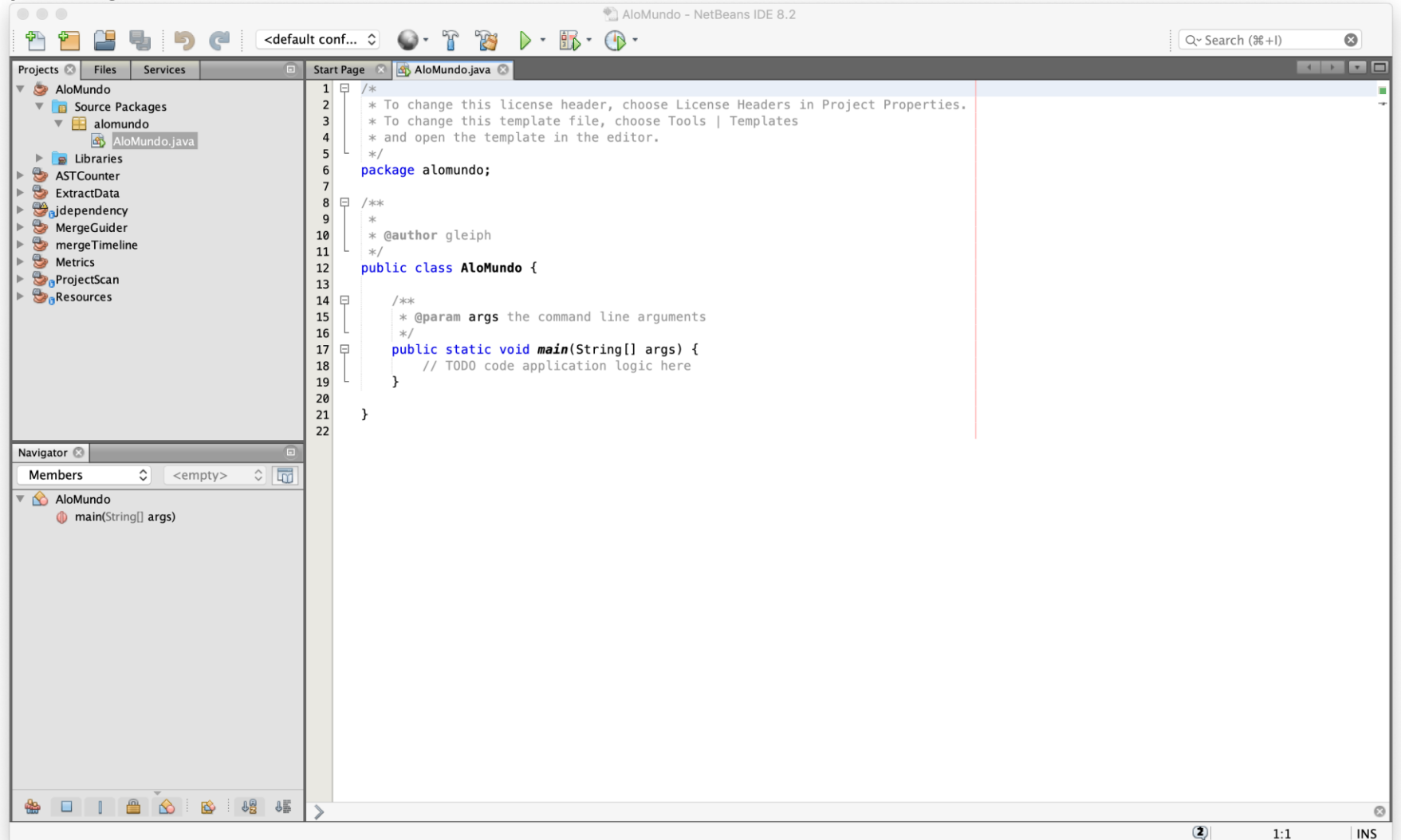
Libraries Folder:

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

☒ Create Main Class

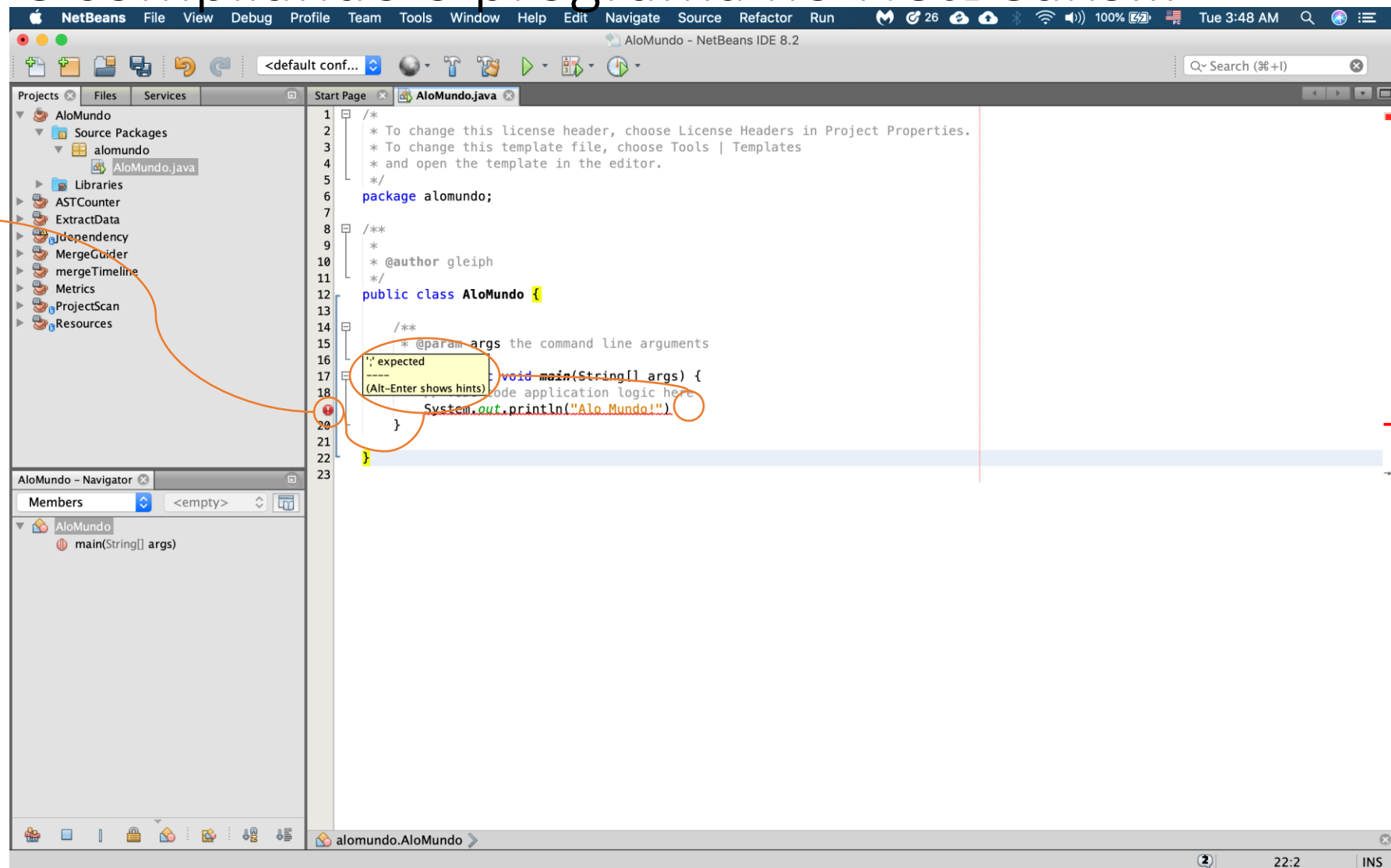
Criando o projeto no NetBeans...

Geração automática do esqueleto do programa



Escrevendo e compilando o programa no NetBeans...

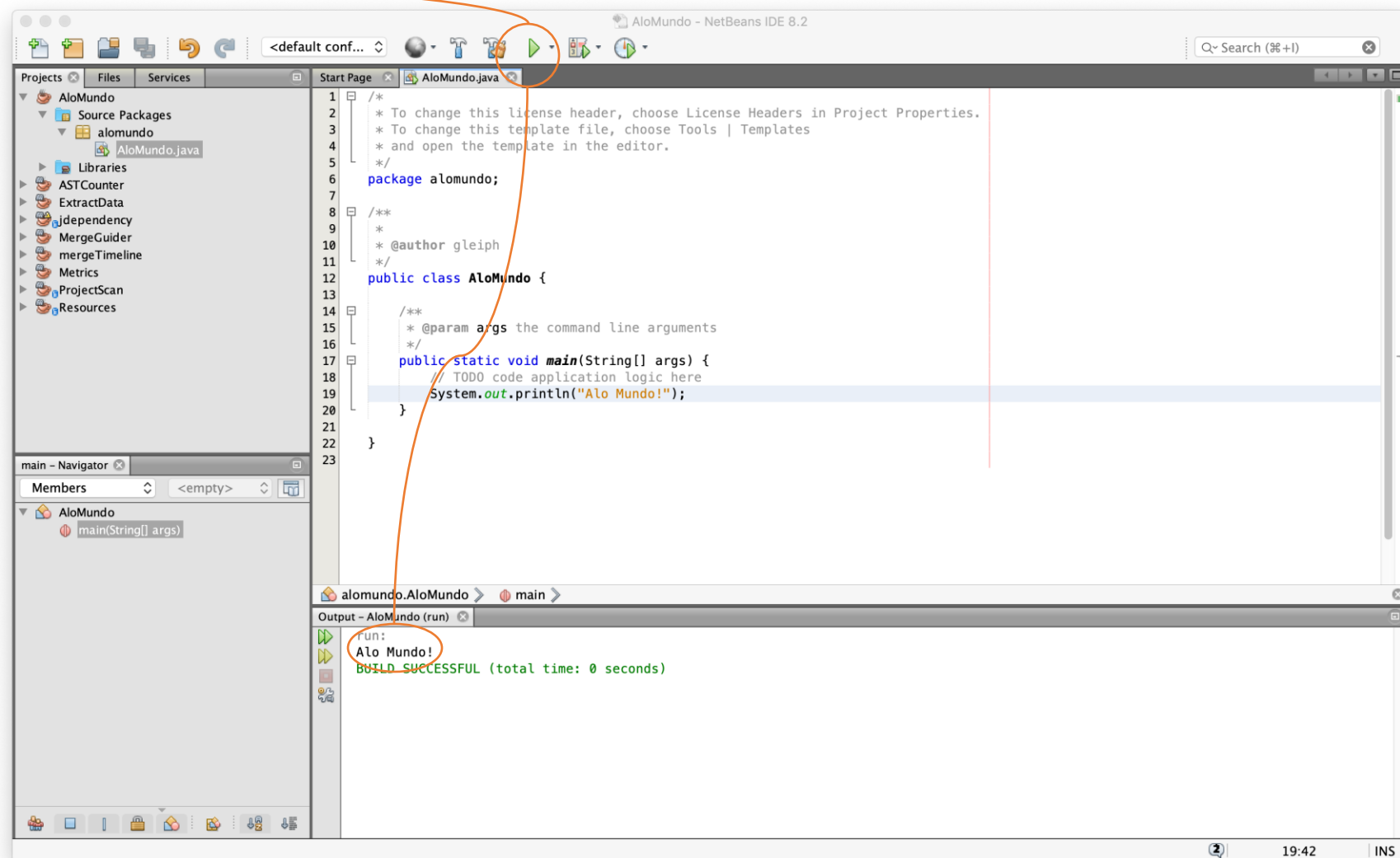
Compilação automática durante a edição do código e avisos sobre erros



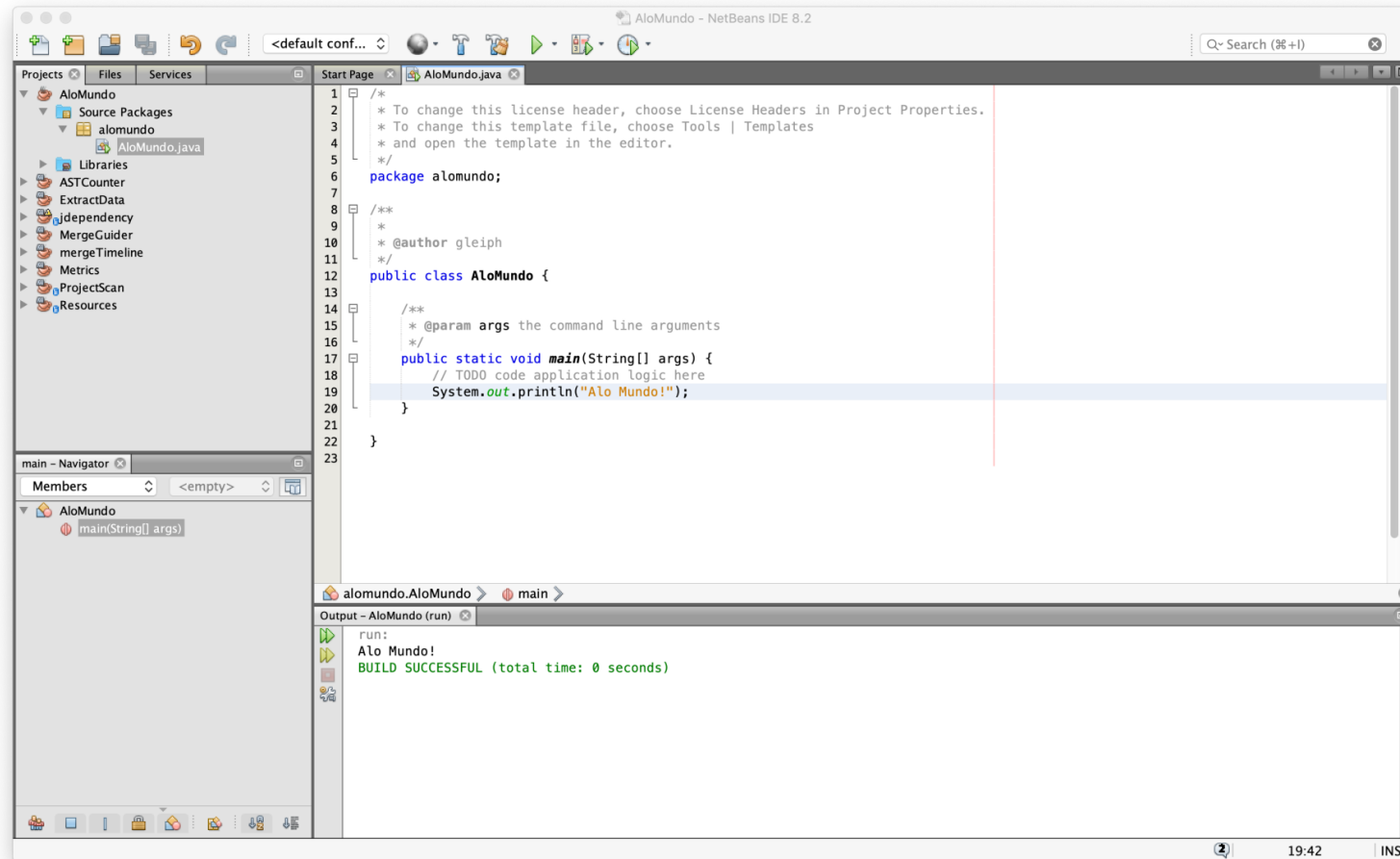
Executando o programa no NetBeans...

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...



The screenshot shows the NetBeans IDE interface. On the left, the 'Projects' pane displays a project named 'AloMundo' with a source package 'alomundo' containing the file 'AloMundo.java'. Below it, the 'main - Navigator' pane shows the 'main(String[] args)' method. The central editor displays the code for 'AloMundo.java':

```

1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package alomundo;
7
8  /**
9   *
10  * @author gleiph
11  */
12  public class AloMundo {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          // TODO code application logic here
19          System.out.println("Alo Mundo!");
20      }
21  }
22
23

```

At the bottom, the 'Output - AloMundo (run)' pane shows the execution results:

```

run:
Alo Mundo!
BUILD SUCCESSFUL (total time: 0 seconds)

```

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 triângulo 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)

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, qualquer 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)

Outros tipos de dados

- char
 - Caractere 16-bit (Unicode)
 - 'a', '1', '@'
- String
 - Texto de tamanho variável
 - "Aluno", "1234"
- boolean
 - Tipo lógico, com valores *true* ou *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 de variáveis seguem a notação ***Camel Case*** iniciando com minúsculas, com conectores (de, e, ou, etc.) omitidos
 - dataNascimento, valorProduto, etc.

Declaração de constantes

- 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* (`_`)

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

Orientação a Objetos – DCC025

Gleiph Ghiotto Lima de Menezes

gleiph.ghiotto@ufjf.br