

Java - Aula 03

Elementos básicos da linguagem Java

Cristiano Amaral Maffort

`cristiano@cefetmg.br`

`maffort@gmail.com`

Técnico em Informática

Departamento de Computação

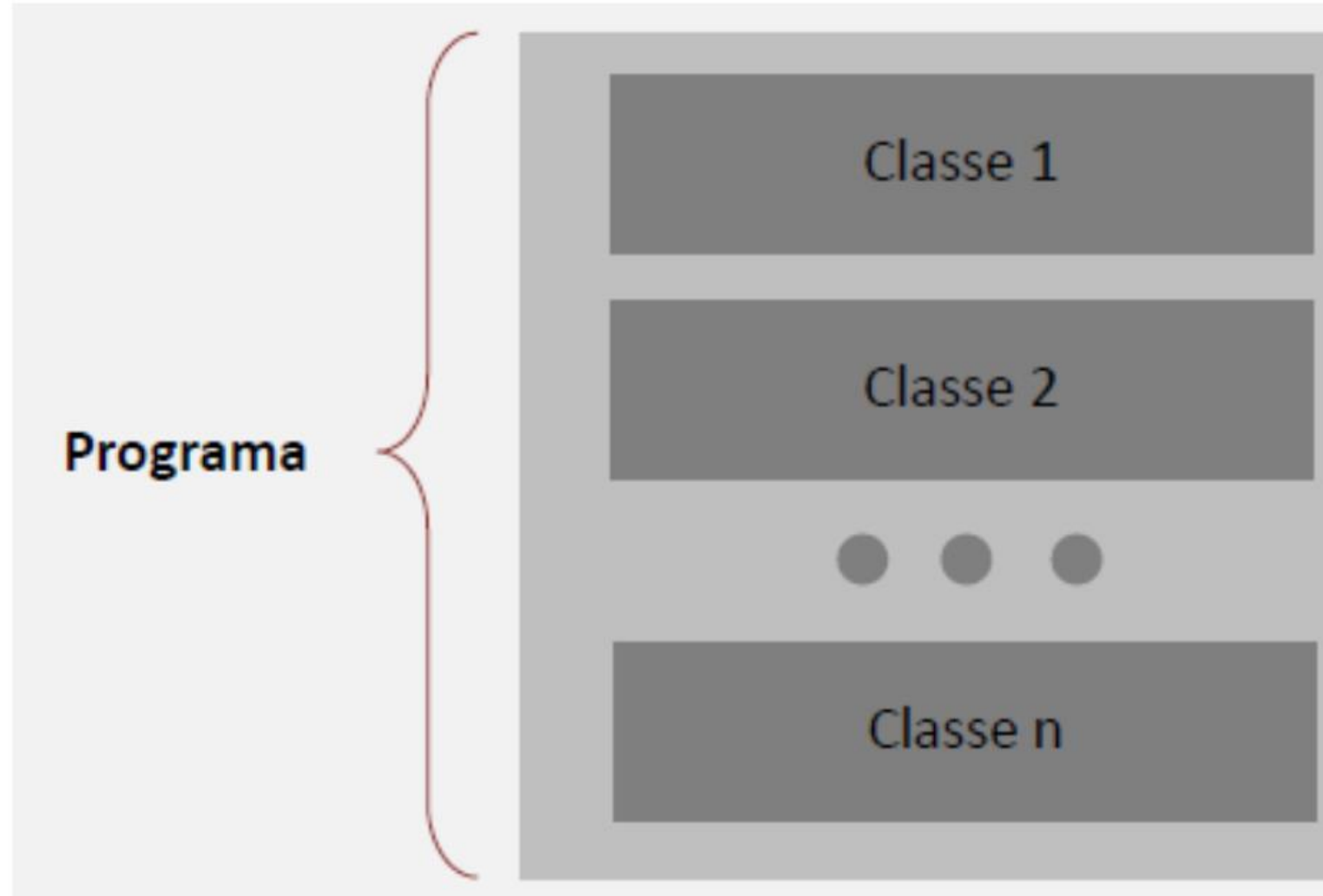
CEFET-MG – Belo Horizonte



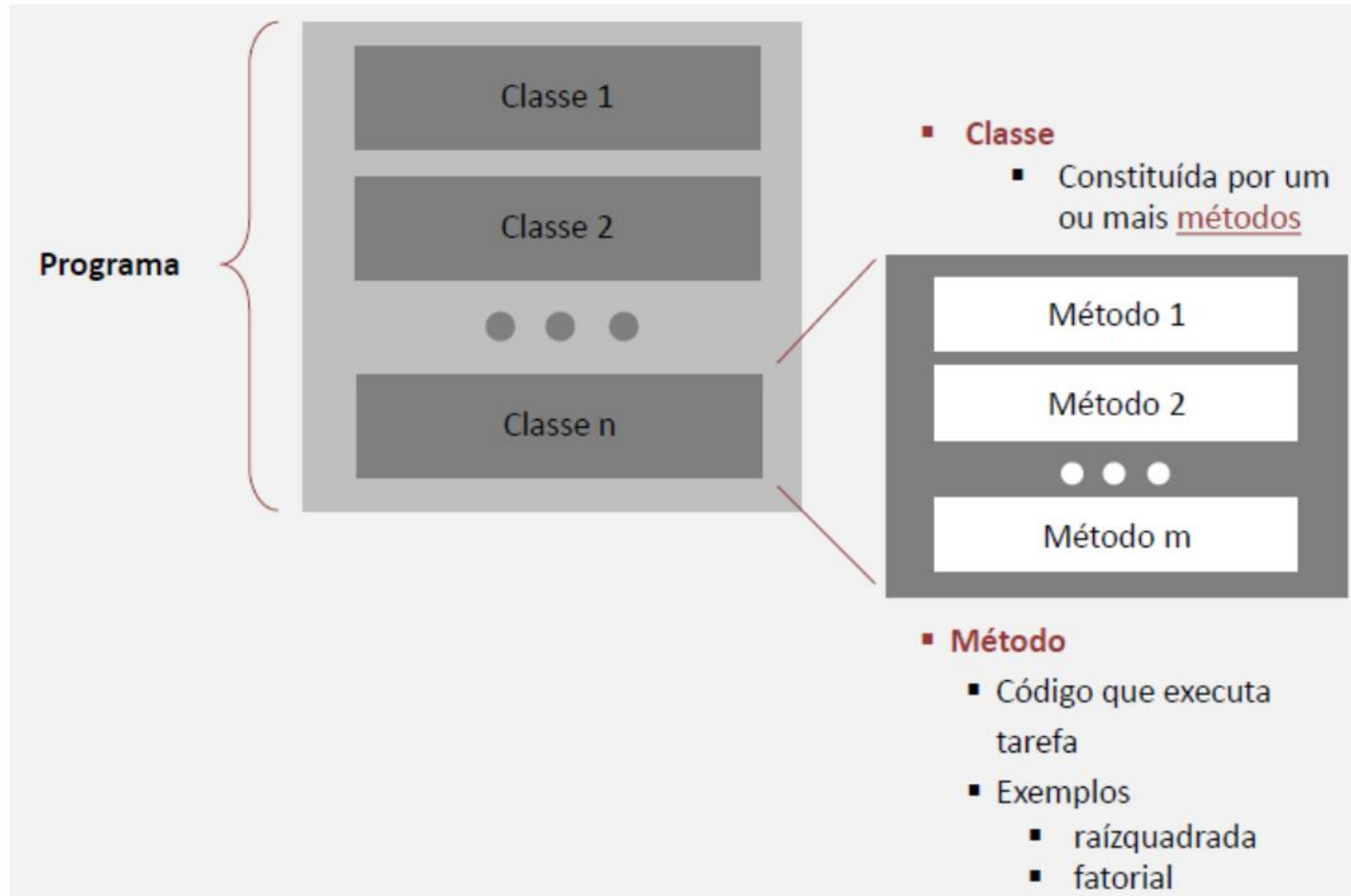
Agenda

- Elementos básicos de um programa
- Tipos de dados
- Variáveis
- Operadores
- Entrada e saída

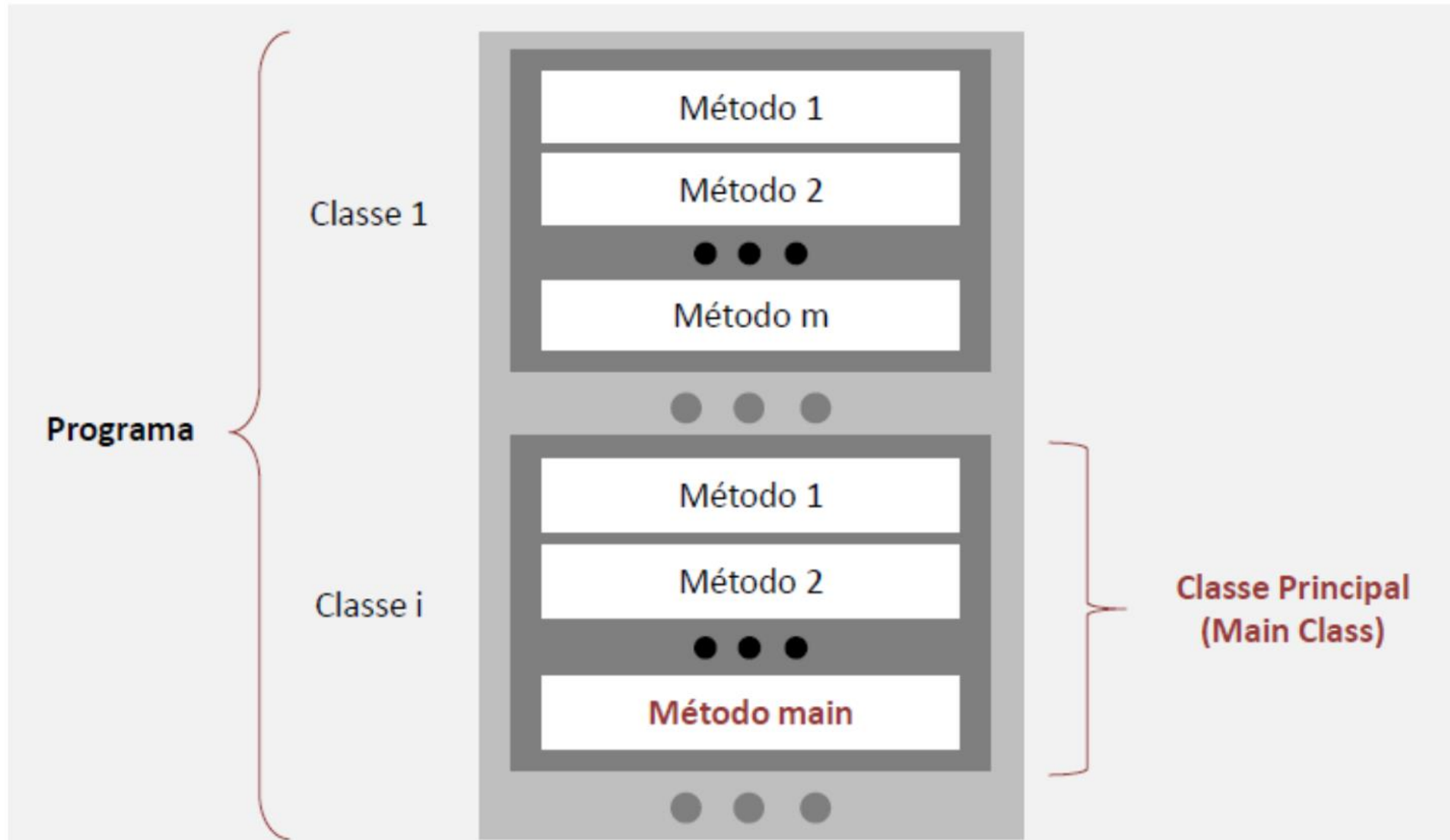
Elementos básicos de um programa Java



Elementos básicos de um programa Java



Elementos básicos de um programa Java



Elementos básicos de um programa Java

package especifica o nome do pacote que a classe pertencerá (assunto futuro)

- Semelhantes a pastas/diretórios de arquivos;
- Tem objetivo de organizar as classes;
- Caso seja omitido, classe ficará no pacote *default* (padrão).

```
1 package br.cefetmg.aula03;
2
3 public class PrimeiroPrograma {
4
5     public static void main(String[] args) {
6         System.out.println("Hello World!");
7     }
8
9 }
```

Elementos básicos de um programa Java

public é um modificador de acesso à classe e seus membros (assunto futuro)

class define uma nova classe, cujo nome vem logo depois do operador

O arquivo de código-fonte deve ser salvo exatamente com o mesmo nome da classe.

```
1 package br.cefetmg.aula03;
2
3 public class PrimeiroPrograma {
4
5     public static void main(String[] args) {
6         System.out.println("Hello World!");
7     }
8
9 }
```

Elementos básicos de um programa Java

A execução do programa é iniciada a partir do método **main**

Se for necessário passar algum argumento, via prompt de comando, para o programa, ele será passado como parâmetro do método **main**, em um array/vetor de String.

static (assunto futuro)

```
1 package br.cefetmg.aula03;
2
3 public class PrimeiroPrograma {
4
5     public static void main(String[] args) {
6         System.out.println("Hello World!");
7     }
8
9 }
```

Elementos básicos de um programa Java

```
1 package br.cefetmg.aula03;
2
3 public class PrimeiroPrograma {
4
5     public static void main(String[] args) {
6         System.out.println("Hello World!");
7     }
8
9 }
```

System: é uma classe Java que contém diversos recursos utilitários, como acesso a recursos de entrada e saída por console.

out: objeto que provê acesso a recurso de saída via console.

println: método que exibe mensagem no console e, ao final da mensagem, adiciona uma quebra de linha

Convenção de nomes em Java

- Case-sensitive
 - Java distingue letras maiúsculas e minúsculas
- Nome de Classes
 - Pascal Case
 - Cada palavra do nome da classe começa com letra maiúscula;
 - Não há espaço entre as palavras.
 - Exemplos:
 - `TextField`
 - `Integer`
 - `ActionEvent`
 - `Aluno`

Identificadores

- Um identificador é o nome dado a um método, a uma variável ou a qualquer outro item definido pelo usuário.
- Regras gerais para formação do nome dos identificadores:
 - Não pode ser uma palavra reservada (como int, for, return, public etc)
 - Podem começar com qualquer letra do alfabeto
 - Incluindo um sublinhado (*underline*, `_`) ou um cifrão (\$).
 - Em seguida, pode haver uma letra, um dígito, um cifrão ou um sublinhado
 - Exemplos:
 - `$up`
 - `_top`
 - `my_var`

Declaração de variáveis

- CamelCase
 - Primeira palavra do nome da variável é minúscula;
 - Se tiver mais de uma palavra, da segunda palavra em diante inicia com letra maiúscula
- Exemplos:
 - `int x, y;`
 - `String nomeAluno;`
 - `long dataNascimento;`
- **Boa prática:**
 - Atribuir nomes de identificadores que reflitam o significado ou o uso dos itens que estão sendo nomeados

Definição de métodos

- CamelCase
- Exemplos:
 - `System.exit();`
 - `System.currentTimeMillis();`
 - `System.gc();`

Escopo e tempo de vida das variáveis

- Em Java, variáveis são declaradas dentro de qualquer bloco
 - Um bloco começa com uma chave de abertura ('{')
 - e termina com uma chave de fechamento ('}')
- Em Java, o bloco define um *escopo*
 - Sempre que iniciar um novo *bloco*, estará criando um novo *escopo*
- Um *escopo* determina que objetos estarão visíveis para outras partes do programa
 - Também determina o *tempo de vida* desses objetos
- As variáveis declaradas dentro de um escopo
 - não podem ser vistas por um código definido fora desse escopo.
 - Isso protege a variável contra acesso/modificação não autorizados

Escopo e tempo de vida das variáveis

- Java não suporta *escopo global*, como C - por exemplo
- Além do ***escopo de bloco***, Java define ***escopo de classe*** e ***escopo de método***
- O *escopo de classe* será abordado futuramente,
 - quando as classes forem apresentadas
- O escopo definido por um método começa com sua chave de abertura
 - E o tempo de vida dura até a chave de fechamento
 - No escopo de um método,
 - além das variáveis declaradas entre as chaves de abertura e fechamento desse método, se esse método tiver parâmetros, eles também estarão incluídos dentro do escopo do método.

Escopo e tempo de vida das variáveis

- Os escopos podem ser aninhados
 - Admitindo criação de blocos dentro de outros.
- Isso faz surgir escopos internos e externos a um bloco.
 - Via de regra, uma variável declarada no escopo externo poderá ser vista por um código que estiver dentro do escopo mais interno.
 - Entretanto, a recíproca não é verdadeira.
 - Variáveis declaradas no escopo interno não podem ser vistas fora dele.

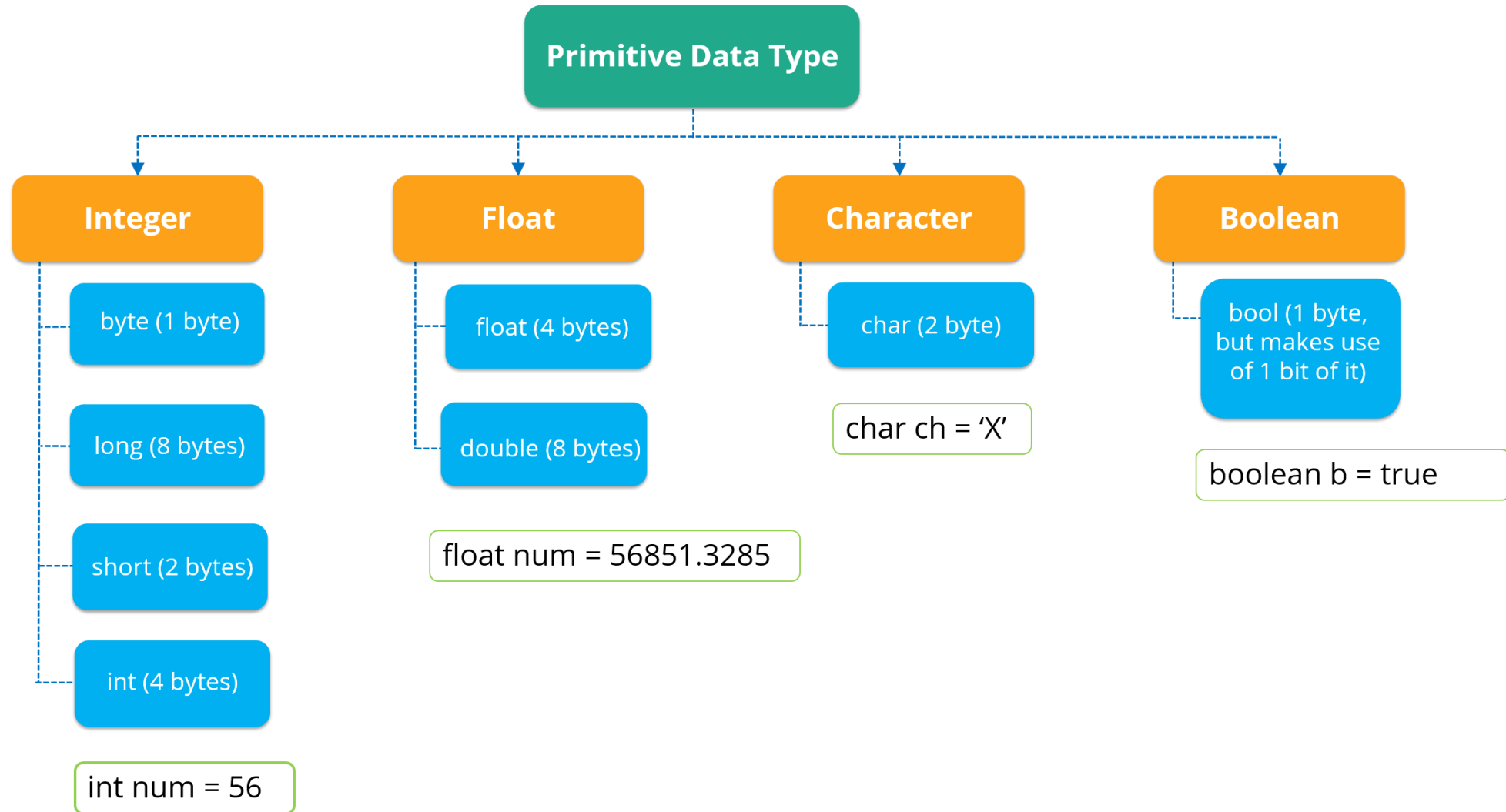
Escopo e tempo de vida das variáveis

```
1 package br.cefetmg.inf;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int x; // conhecida pelo código dentro de main
7
8         x = 10;
9
10        if (x == 10) { // inicia novo escopo
11            int y = 20; // conhecida apenas neste bloco
12
13            // tanto x quanto y são conhecidas aqui
14
15            System.out.println("x and y: " + x + " " + y);
16
17            x = y * 2;
18        }
19
20        // y = 100; // Erro! y não é conhecida aqui
21
22        // x ainda é conhecida aqui.
23
24        System.out.println("x is " + x);
25    }
26
27 }
```

Tipos de dados

- Java é uma ***linguagem fortemente tipada***
 - Todas as operações têm a compatibilidade de seus tipos verificada pelo compilador
 - Operações inválidas não serão compiladas
 - O tipo de um valor ou variável determina as operações que podem ser executadas nele
 - Uma operação aplicada a um tipo pode não ser permitida em outro
- A verificação dos tipos ajuda a impedir ocorrência de erros
 - além de melhorar a confiabilidade

Tipos primitivos de dados



Restrições aos tipos primitivos de dados

- Todos os tipos inteiros têm valores de sinal positivo e negativo.
- `char` é um tipo integral
 - tipos integrais contêm valores binários inteiros
 - o tipo `char` tem a finalidade de representar caracteres
 - os tipos inteiros tem a finalidade de representar quantidades numéricas inteiras
 - Java usa caracteres de 16 bits
 - para suporte a caracteres *Unicode*
 - *Unicode* possui todos os caracteres encontrados em todos os idiomas humanos
 - Os 127 valores do conjunto *ASCII* são simétricos ao conjunto *Unicode*

Restrições aos tipos primitivos de dados

- Tipos de ponto flutuante
 - Podem representar números fracionários
 - O tipo `float` tem tamanho de aproximadamente $3,4 \times 10^{38}$
 - O tipo `double` tem tamanho de aproximadamente $1,8 \times 10^{308}$
- Tipo booleano
 - O tipo `boolean` representa os valores *verdadeiro* e *falso*
 - Esses valores usam as palavras reservadas `true` e `false`

Tipos não primitivos de dados

- **String**: é uma sequência/conjunto de caracteres.
 - Os caracteres do conjunto devem estar inseridos entre aspas duplas.
- **Array**: uma estrutura homogênea de dados.
- **Classe**: basicamente, por hora, um tipo estruturado (assunto futuro).
- **Interface**: provê métodos que deverão ser implementador por classes (assunto futuro).

Operadores

- Possui os mesmos da linguagem C
- Operadores aritméticos: `+` `-` `/` `*` `%` `++` `--`
- Operadores relacionais: `>` `<` `>=` `<=` `==` `!=`
 - `>` `<` `>=` e `<=`: aplicados somente a tipos numéricos e `char`
 - `==` e `!=`: qualquer variável, incluindo `boolean`
- Operadores lógicos: `&` `|` `^` `!` `&&` `||`
 - `&` `|` `^` `!`: operações básicas AND, OR, XOR e NOT
 - `&&` e `||`: operadores de curto-circuito AND e OR
- Operador de atribuição: `=`
 - `var = expressão;` → valor da expressão (à direita) é calculado; somente depois é atribuído à variável (à esquerda)

Operadores Lógicos

p	q	p & q	p q	p ^ q	!p
Falso	Falso	Falso	Falso	Falso	Verdadeiro
Verdadeiro	Falso	Falso	Verdadeiro	Verdadeiro	Falso
Falso	Verdadeiro	Falso	Verdadeiro	Verdadeiro	Verdadeiro
Verdadeiro	Verdadeiro	Verdadeiro	Verdadeiro	Falso	Falso

Comentários em Java

- `//`: para comentar o que estiver após
 - até o final da linha
- `/* */`: para comentar o que estiver entre
 - incluindo quebras de linha
- `/** */`: para documentação (assunto futuro - JavaDoc)

```
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 br.cefetmg.inf;
7
8  /**
9   *
10   * @author Cristiano
11   */
12  public class Main {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          // TODO code application logic here
19      }
20
21  }
```

Entrada padrão de dados (console/teclado)

■ Scanner

- Utilizado para leitura de *streams* de texto, incluindo arquivos e strings;
- Mas também pode fazer leituras do teclado,
 - tratando-o como arquivo de texto.

■ Leitura

- Necessário invocar um método de um objeto Scanner que seja adequado ao tipo de dado a ler.
 - **Sintaxe:** `nomeObjeto.next<tipo de dado>()` ;
 - **Exemplo:**
 - `Scanner scan = new Scanner(System.in) ;`
 - `int idade = scan.nextInt() ;`

Entrada padrão de dados (console/teclado)

```
package br.cefetmg.aula03;

import java.util.Scanner;

public class EntradaScanner {
    public static void main(String[] args) {
        int numeroInteiro;
        float numeroReal;
        String primeiroNome;
        String nomeCompleto;

        Scanner input = new Scanner(System.in);

        numeroInteiro = input.nextInt();
        numeroReal = input.nextFloat();
        primeiroNome = input.next();
        nomeCompleto = input.nextLine();
    }
}
```

import: permite que uma classe utilize outra(s).

Semelhante ao include de C.

Saída padrão de dados (display/monitor)

- out é objeto da classe System
 - Representa dispositivo de saída *default*
- Métodos principais
 - print
 - println
 - printf

```
package br.cefetmg.aula03;

import java.util.Scanner;

public class EntradaScanner {
    public static void main(String[] args) {
        int numeroInteiro;
        float numeroReal;
        String primeiroNome;
        String nomeCompleto;

        Scanner input = new Scanner(System.in);

        System.out.print("Informe um número inteiro: ");
        numeroInteiro = input.nextInt();
        System.out.println("Número inteiro " + numeroInteiro + "\n");

        System.out.print("Informe um número real: ");
        numeroReal = input.nextFloat();
        System.out.printf("Número real %.2f\n\n", numeroReal);

        System.out.print("Informe o primeiro nome: ");
        primeiroNome = input.next();
        System.out.println("Nome " + primeiroNome);
    }
}
```

Bibliografia Obrigatória

- HORSTMANN, Cay S.; CORNELL, Gary. ***Core Java: Fundamentos***. 8. ed. São Paulo: Pearson Prentice Hall, 2020, p. 19 – 49.
- SCHILDT, Herbert; SKRIEN, Dale. ***Programação com Java: uma introdução abrangente***. Porto Alegre: AMGH, 2013, p. 42 – 115.