

Sobre a Disciplina

Qual o Objetivo da Disciplina?



- Aprender os conceitos de **programação orientada a objetos**
- **Praticar** os conceitos usando uma linguagem de programação orientada a objetos: **Java**
 - Diferentemente de outras linguagens (C++, Python, PHP, etc), não tem como programar em Java sem utilizar orientação a objetos
- **Resolução de problemas** usando orientação a objetos
 - Laboratórios
 - Trabalhos Práticos
 - *Java*
 - *GUI*
 - *Banco de Dados*
 - *Android*

Assuntos



■ Introdução

- Classes, objetos, atributos, métodos, sobrecarga, tipos de dados, métodos construtores, métodos destrutores, etc

■ Classes utilitárias

- Wrapper classes, generic collections, entrada e saída

■ Conceitos

- Pacotes, encapsulamento, herança, polimorfismo, classes abstratas, interfaces, tratamento de exceções, arquivos, banco de dados

■ Prática

- Aplicativos em Janelas (GUI)
- Apps: Android

Avaliação



- 10 Laboratórios
 - Média dos Laboratórios (ML)
- 2 Trabalhos Práticos (TP1 e TP2)
 - TP1: Modelagem, Programação Java e Programação em Janelas
 - TP2: Programação na Plataforma Android
- 1 Prova Parcial (PP)
- 1 Prova Final (PF)
- Média dos Exercícios (MEE):
 - $MEE = (ML + TP1 + TP2 + PP) / 4$
- Média Final (MF):
 - $MF = (2 * MEE + PF) / 3$

Trabalhos Práticos e Laboratórios

- Trabalhos e laboratórios são **individuais**
 - Pode discutir e tirar dúvidas entre si, mas sem cópia de código
- Plágio
 - Será verificado com trabalhos atuais, antigos e disponíveis na Internet usando um **sistema automático de detecção**
 - Caso o plágio seja confirmado, a nota do trabalho será zero tanto para o aluno que copiou quanto para o aluno que disponibilizou o trabalho para ser copiado
- Avaliação das Implementações
 - Qualidade do código, execução, conformidade com o assunto especificado e pontos extras (positivos ou negativos)



**Ctrl+C e
Ctrl+V é
Plágio**

Laboratórios Presencial



- ▣ Laboratório presencial nas quarta-feiras
 - ▣ Obrigatório
 - ▣ Será tirado presença e alunos poderão ser reprovado por falta
 - ▣ O professor estará presente para tirar dúvidas e ajudar nas soluções
 - ▣ Alguns laboratórios só poderão ser entregues durante o horário da aula



- ColabWeb da UFAM (Moodle)
 - Será usado para **disponibilizar o material** da disciplina e trabalhos
 - <http://colabweb.ufam.edu.br/>

- Se você ainda não possui conta
 - Entre no site → Criar uma conta → Preencha os dados → E-Mail Continue com os passos seguintes...

- Se você já possui conta
 - A partir da página principal do ColabWeb:
 - Cursos → Computação → Cursos 202X/X → Nome da Disciplina
 - → Inscrever-se

Dúvidas



- ▣ A forma principal para tirar dúvidas será o **e-mail**:
 - ▣ Professor: horacio@icomp.ufam.edu.br
 - ▣ Monitoria:
 - **José Cordova**
 - E-Mail: jose.cordova@icomp.ufam.edu.br
 - Telegram: [@josemateusc](https://t.me/josemateusc)
 - Seg: 17h às 19h;
 - Sexta: 9h às 11h, 14h às 16h
 - *Pontos extras para quem procurar a monitoria*

Referências



- **Slides da disciplina**
- DEITEL, Harvey M. e DEITEL, Paul J. Java – Como Programar, 8ª edição. Editora Pearson, 2010.
- SCHILDT, Herbert. Java Para Iniciantes - Crie, Compile e Execute Programas Java Rapidamente, 5ª edição. Bookman, 2013.
- DEITEL, Paul J. e DEITEL, Harvey M. Java: Java How To Program (Early Objects) (10th Edition). Prentice Hall, 2014.
- GOSLING, J.; JOY, B.; STEELE, G.; BRACHA, G.; BUCKLEY, A. The Java Language Specification, Java SE 8 Edition. Oracle, 2015. Online: <https://docs.oracle.com/javase/specs/>
- ORACLE. Código Fonte do Java 8. Disponível online: <http://hg.openjdk.java.net/jdk8/jdk8/jdk/file/687fd7c7986d>

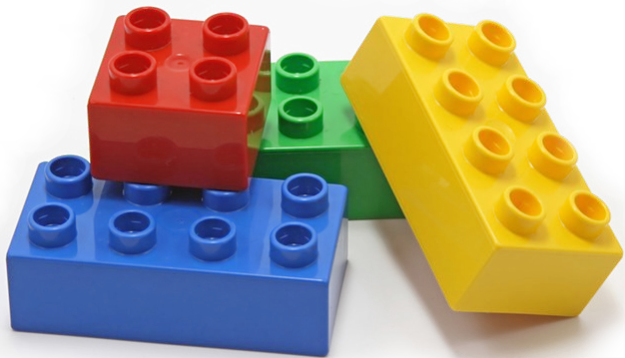
Projeto DevTITANS



- Capacitação e Desenvolvimento em Tecnologias Android para Sistemas Embarcados
 - Aprender não só a criar um app
 - Mas aprender a criar/personalizar um Android novo

- Aluno de PP/TAP pode?
 - Sim
 - Quando chegarmos na parte que precisa de Android, vocês já estarão prontos como conteúdo da disciplina até o momento (Java)

- Inscrições
 - Serão abertas mês que vem!
 - Curso noturno



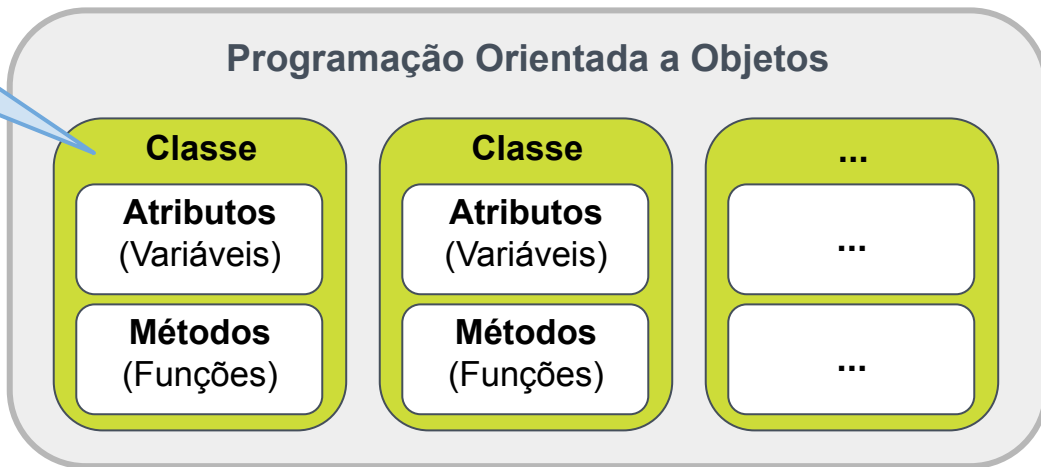
Programação Orientada a Objetos



Mudança de Enfoque



Um **objeto** é uma **variável** cujo tipo é uma classe



Ponto de Vista da OO

- O mundo real é composto de **objetos**. Para onde você olha, você vê objetos!
 - Uma cadeira, um carro, computador, celular, etc
- **Objetos pertencem a uma classe**
 - A maçã que você comeu no café, é um objeto da classe das maçãs (que é uma subclasse das frutas)
 - O carro que você usou hoje é um objeto da classe dos carros
 - Todas as cadeiras em uma sala de aula são objetos da classe das cadeiras
- Todos os **objetos** têm coisas em **comuns**:
 - Possuem **atributos**/propriedades: tamanho, forma, cor, peso, etc
 - Possuem **comportamentos** (métodos): corre, freia, dorme, etc

Ponto de Vista da OO

Exemplo:

Classe	Carro
Atributos	ano modelo placa cor
Métodos	acelerar() frear() buzinar()

Objetos da classe Carro



seuCarro

1950
Fusca
NAN-1892
Vermelho



meuCarro

1985
DeLorean
OUTATIME
Cinza

Ponto de Vista da OO



- Na POO, **objeto** é uma entidade que **combina** uma estrutura de dados (**atributos**) e um comportamento funcional (**métodos**)
- Sistemas são estruturados a partir de objetos que existem no domínio do problema
- Um objeto é **criado** com base em uma **classe**
 - Assim como, em C, uma variável pode ser criada a partir de uma *struct* (tipo)

Ponto de Vista da OO

“O paradigma da orientação a objetos visualiza um sistema de **software** como uma **coleção** de agentes interconectados chamados **objetos**. Cada objeto é responsável por realizar tarefas específicas. É através da interação entre objetos que uma tarefa computacional é realizada.”

(Bezerra, 2015)



Objetivos da OO



- **Diminuir a distância** conceitual entre o **mundo real** (domínio do problema) e o **modelo abstrato** de solução (domínio da solução)
 - Diminuição do “gap semântico”
- Trabalhar com **noções intuitivas** (objetos e ações) durante todo o ciclo de vida do desenvolvimento
 - Atrasando ao máximo a introdução de conceitos de implementação

Benefícios



■ Modelagem

- Facilita a **compreensão** do problema
- Melhora a **interação** entre o cliente e o analista
- Melhora a interação entre o analista e o desenvolvedor
- Aumenta a **consistência** interna dos resultados da análise
- Usa uma **representação** básica consistente para a análise, projeto e implementação (UML – Unified Modeling Language)

■ Implementação

- Programa passa a ser um conjunto de “**blocos**”
- Facilita a **manutenção** e **expansão** do código
- Melhora a **legibilidade**
- Facilita a **reutilização** de códigos

Conceitos



- Orientação a objetos possui uma série de **conceitos**, muitos das quais você já pode estar familiarizado:
 - Classes, Objetos, Atributos, Métodos, Pacotes, Interfaces, Mensagens
 - Modularidade, Encapsulamento
 - Construtores e Destrutores
 - Polimorfismo, Sobrecarga, Sobreposição
 - Herança e Hierarquia de Classes
 - Tratamento de Exceções

- Estes e outros conceitos **serão vistos na prática** usando a linguagem Java



Introdução à Linguagem Java



A Linguagem Java

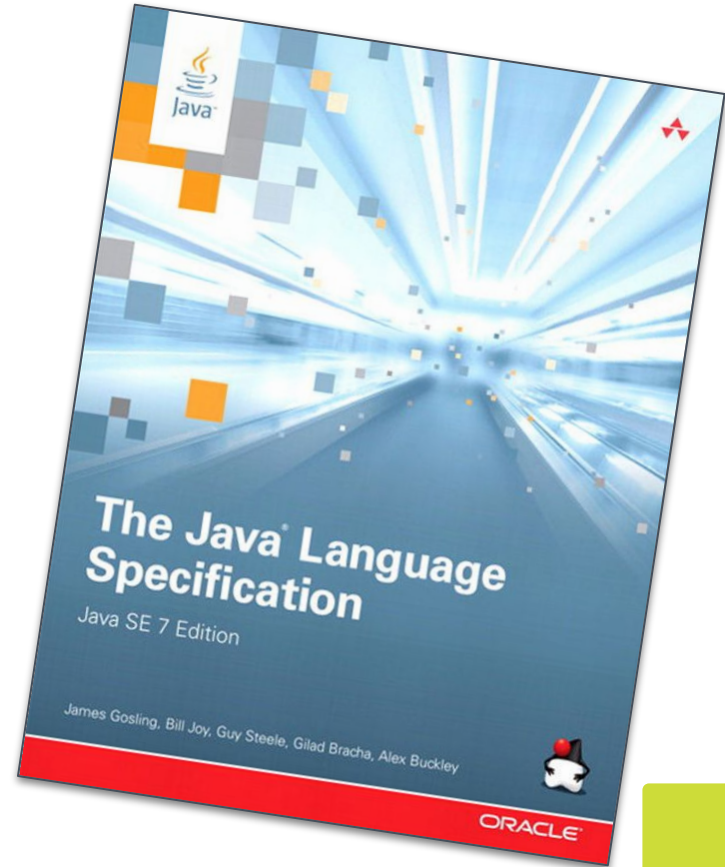
Hi! I'm Duke, the
Java Mascot



- The Java Language Specification (2020):
 - **Java** é uma linguagem de **uso geral**, **concorrente**, baseada em classes e **orientada a objetos**
 - Criada para ser **simples**
 - **Parecida com C/C++**, mas organizada de forma diferente
 - Possui **tipagem forte e estática**
 - Linguagem de **alto nível**, em que os detalhes de representação da máquina não estão disponíveis ao programador
 - Possui **gerenciamento automático de memória** (coletor de lixo)
 - É compilada para um **bytecode**, executado por uma **máquina virtual**

A Linguagem Java

- GOSLING, J.; JOY, B.; STEELE, G.; BRACHA, G.; BUCKLEY, A. SMITH, D.; BIERMAN, G. ***The Java Language Specification***, Java SE 14 Edition. Oracle, 2020.
- Disponível online:
 - <https://docs.oracle.com/javase/specs/>



Compilação Virtual



■ Linguagens compiladas

- C/C++, Rust, Pascal, Fortran
- Programas são compilados para um **código binário** (linguagem de máquina) executado diretamente pelo processador do computador

■ Linguagens interpretadas

- JavaScript, PHP, Bash
- Programas são **lidos por um interpretador** (um programa) que lê o código-fonte e diz para o computador o que ele deve fazer

Compilação Virtual

▣ Java: Compilação Virtual

- ▣ O código-fonte é **compilado para o código binário** de uma máquina virtual.
- ▣ Este código compilado é chamado de **bytecode**
- ▣ Uma máquina virtual (um programa) **lê o bytecode** e diz para o computador o que ele deve fazer



*Nota: em sua implementação atual, a linguagem **Python** também usa um esquema de compilação virtual*

Compilação Virtual

Código-Fonte
(Carro.java)

```
class Carro {  
    int ano;  
    String modelo;  
    String placa;  
    String cor;  
}
```

Compilação
Virtual
(javac)



Bytecode
(Carro.class)

0000	ca	fe	ba	be	00	00
0010	43	61	72	72	6f	07
0020	6c	61	6e	67	2f	4f
0030	64	65	6c	6f	01	00
0040	67	2f	53	74	72	69
0050	61	01	00	03	61	6e
0060	6e	69	74	3e	01	00

Máquina
Virtual
(JVM)



Windows



Linux



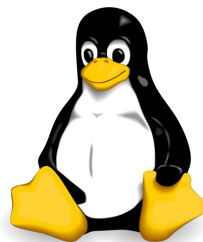
Android



Java: Onde Funciona

- Java funciona em diversos dispositivos:

- Sistemas Operacionais
- Smartphones
- TVs
- Sensores
- Navegadores
- Servidores Web
- Lego MindStorms
- Blu-Ray
- Muitos outros



Linux



MacOS



Windows



Android



TVs Digitais



Sensores



Servidores Web



Lego Mindstorms



Blu-Ray



Cartões



Bytecode

- Arquivos Java Compilados
 - Extensão **.class**

00000000	ca fe ba be 00 00 00 33	00 15 07 00 02 01 00 053.....
00000010	43 61 72 72 6f 07 00 04	01 00 10 6a 61 76 61 2f	Carro.....java/
00000020	6c 61 6e 67 2f 4f 62 6a	65 63 74 01 00 06 6d 6f	lang/Object...mo
00000030	64 65 6c 6f 01 00 12 4c	6a 61 76 61 2f 6c 61 6e	delo...Ljava/lan
00000040	67 2f 53 74 72 69 6e 67	3b 01 00 05 6d 61 72 63	g/String;...marc
00000050	61 01 00 03 61 6e 6f 01	00 01 49 01 00 06 3c 69	a...ano...I...<i
00000060	6e 69 74 3e 01 00 03 28	29 56 01 00 04 43 6f 64	nit>...()V...Cod
00000070	65 0a 00 03 00 0e 0c 00	0a 00 0b 01 00 0f 4c 69	e.....Li
00000080	6e 65 4e 75 6d 62 65 72	54 61 62 6c 65 01 00 12	neNumberTable...
00000090	4c 6f 63 61 6c 56 61 72	69 61 62 6c 65 54 61 62	LocalVariableTab
000000a0	6c 65 01 00 04 74 68 69	73 01 00 07 4c 43 61 72	le...this...LCar
000000b0	72 6f 3b 01 00 0a 53 6f	75 72 63 65 46 69 6c 65	ro;...SourceFile
000000c0	01 00 0a 43 61 72 72 6f	2e 6a 61 76 61 00 20 00	...Carro.java. .
000000d0	01 00 03 00 00 00 03 00	00 00 05 00 06 00 00 00

- Representação compacta de uma espécie de linguagem assembly
- Contém instruções**, uma tabela de símbolos e outras informações
 - Você pode ver os detalhes de um bytecode (incluindo as instruções) usando o comando: `javap -v <Arquivo.class>`
- Possuem **código binário** que será executado pela Máquina Virtual Java



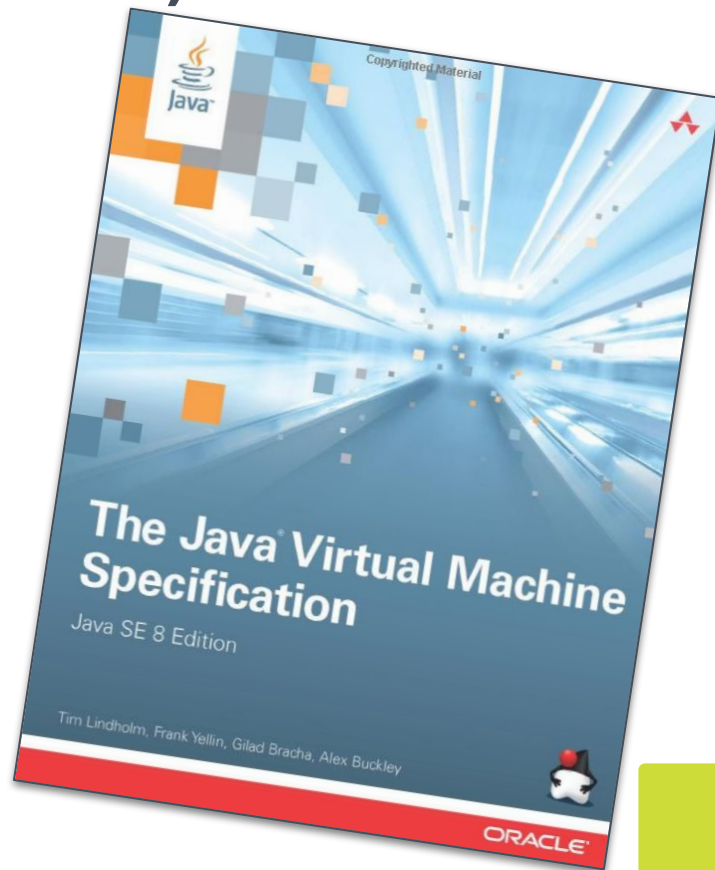
Máquina Virtual Java (JVM)

- Responsável pela independência de hardware e sistema operacional
- **É um computador abstrato**, implementado em um programa (java)
 - Possui um conjunto de instruções e registradores, como um processador
 - Possui e gerencia a memória
 - Assim como um computador não entende C (apenas o código compilado), a máquina virtual não entende Java, apenas o bytecode
- **A implementação JVM** mais usada hoje é a **HotSpot**
 - Implementada em C++
 - Possui código aberto e pode ser baixado no site do OpenJDK
 - <http://hg.openjdk.java.net/jdk/jdk14/file/6c954123ee8d>



Máquina Virtual Java (JVM)

- LINDHOLM, T.; YELLIN, F.; BRACHA, G.; BUCKLEY, A; SMITH, D. ***The Java Virtual Machine Specification***, Java SE 14 Edition. Oracle, 2020.
- Disponível online:
 - <https://docs.oracle.com/javase/specs/>



Instalação do Java



- Java pode ser baixado da Internet
 - <http://www.oracle.com/technetwork/java/javase/downloads/>
 - Java SE (Standard Edition): é a versão usada em computadores pessoais
- Existem **dois** pacotes principais: **JRE** e **JDK**
- **JRE**: Java Runtime Environment
 - Contém a **Máquina Virtual Java**
 - Permite executar os programas Java
 - Não possui o compilador (javac)

Instalação do Java



■ JDK: Java Development Kit

- Contém o JRE + **Compilador Java** (javac) + Outras ferramentas
- É o que usaremos para desenvolver aplicativos no curso

■ Instalação no Windows

- Acesse o site (slide anterior) → procure e faça download do .exe do instalador → abra o gerenciador de arquivos → vá na pasta de downloads → execute o instalador → next → next → next → accept → (desmarque os adwares) → next → finish → reinicie o windows.

■ Instalação no Linux

- **Já vem instalado**
- Se não, execute: `sudo apt install openjdk`

Desenvolvendo em Java - IDEs



■ Eclipse

- Software Livre. O mais popular. Um pouco complexo (e completo).
- Extremamente expansível através de pacotes
- <http://www.eclipse.org/>

■ IntelliJ

- Excelente IDE. Usado pelo Android. Pago, mas possui versão não-comercial.

■ NetBeans

- Software Livre. Relativamente mais fácil de aprender
- <http://netbeans.org/>

■ Outros editores

- Sublime, Visual Studio Code, Kate, Vi, Notepad



Java: Hello World!



Um Programa em Java



- Uma **aplicação é composta** por um **conjunto de classes**
 - O que será programado realmente são as classes
 - Cada classe fica em um arquivo separado
 - O **nome do arquivo** deve ser **igual ao nome da classe** e possuir **extensão .java**
 - Cada classe deve ser compilada, gerando o seu bytecode (**extensão .class**)
- Deve-se escolher uma classe que irá controlar o fluxo de execução do programa.
 - Esta classe terá o método “**main**”
 - A Máquina Virtual Java irá interpretar o bytecode da classe e executá-la
- As demais classes utilizadas na aplicação serão carregadas automaticamente no momento em que forem necessárias
 - Carga dinâmica de código

Método *main*

- A execução do programa começa por um **método** (função) **principal** chamado ***main***, similar ao C/C++
 - O método main controla o fluxo do programa, instanciando uma ou mais classes e invocando um ou mais métodos que forneçam a funcionalidade da aplicação

```
public static void main(String[] args)
```

*modificadores
de acesso*

retorno

*nome do
método*

parâmetros

- Ao invocar o interpretador Java, deve-se especificar o nome da classe a ser executada
 - A máquina virtual chama o método main definido nesta classe

Exemplo Completo

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

- A Este código deve ser salvo no arquivo texto:
 - `HelloApp.java`
- Use o compilador Java para compilar o arquivo:
 - `javac HelloApp.java`
- Será gerado um arquivo contendo o bytecode chamado
 - `HelloApp.class`
- Para executar, usa-se a máquina virtual:
 - `java HelloApp`
 - `Hello World !!`

Similar ao C

- Como a própria especificação diz, Java foi criado para ser semelhante a C/C++.
- Java aceita os tipos primitivos usados em C, além de um tipo próprio para Strings e um para valores Booleanos:

```
int a = 42;
float b = 1.21f;
double c = 1.21;
boolean d = false;
String nome = "Projeto de Programas";

// Impressão de Variáveis
System.out.println("O valor de 'a' é " + a);
```

Similar ao C

■ Comentários:

```
// Comentário de uma linha

/*
    Comentário de
    várias linhas ...
*/

/** JavaDoc - Usado para documentar o código */
```

Similar ao C

- Estruturas de Controle (if, for, while):

```
// Condicional
if (a > b) {
    System.out.println("A é maior do que B");
} else {
    System.out.println("A é menor ou igual a B");
}

// Loop For
for (int i=0; i<10; i++) {
    System.out.println("Loop " + i + ": " + nome);
}

// Loop While
while (a > 0) {
    System.out.print(a + " ");
    a--;
}
```

Similar ao C

- Estruturas de Controle (switch):

```
// Switch
float total;
char operacao = '+';

switch (operacao) {
    case '+':
        total = a + b;
        break;
    case '-':
        total = a - b;
        break;
    default:
        System.out.println("Operador desconhecido");
        total = 0;
}
System.out.println("Total da operação = " + total);
```


Similar ao C (quase)

■ Vetores e matrizes

O operador **new** é usado para alocar memória dinâmica, similar ao **malloc**.

```
// Para criar um vetor de inteiros (máx 50 elementos)
int vetor[] = new int[50];

// Para criar uma matriz de inteiros (máx 50*50 elementos)
int matriz[][] = new int[50][50];

// O restante é similar à Linguagem C. Exceto que em
// Java podemos usar o atributo 'length' de um vetor
// para pegar o tamanho máximo dele.
for (int i=0; i<vetor.length; i++) {
    vetor[i] = i * 2;
}
```

Similar ao C

- Impressão formatada (*printf*)

```
public class ImpressaoFormatada {  
    public static void main(String args[]) {  
  
        int velocidade = 88;  
        float energia = 1.21f;  
        String resultado = "viagem no tempo";  
        System.out.printf("A %d km/h com %.3f GW, temos %s.",  
                           velocidade, energia, resultado);  
    }  
}
```

```
$ javac ImpressaoFormatada.java
```

```
$ java ImpressaoFormatada
```

```
A 88 km/h com 1.210 gigawatts, temos viagem no tempo.
```

Leitura de dados do teclado

```
import java.util.Scanner;
```

Usa a classe Scanner

```
public class Dados {
```

```
    public static void main(String[] args) {
```

```
        Scanner scan = new Scanner(System.in);
```

Novo objeto Scanner, tendo o teclado como entrada (*in*)

```
        System.out.print("Digite seu nome: ");
```

```
        String nome = scan.next();
```

Lê uma string

```
        System.out.print("Digite sua idade: ");
```

```
        int idade = scan.nextInt();
```

Lê um inteiro

```
        System.out.print("Digite sua altura: ");
```

```
        float altura = scan.nextFloat();
```

Lê um float

```
        System.out.println("Olá " + nome + "! Você tem " + idade  
                           + " anos e tem " + altura + "m de altura.");
```

```
        scan.close();
```

```
    }
```

```
}
```

Laboratório

- Próxima Aula ...
 - Presença obrigatória
 - Será feito apenas no horário da disciplina
- Estará disponível no ColabWeb
 - bit.ly/pp-colabweb
- Instale em sua máquina/laptop
 - Java Development Kit (JDK)
 - Eclipse (ou seu IDE de preferência)

