

Curso Java COMPLETO

educandoweb.com (cupons de desconto)

Dr. Nelio Alves

Percorso de formação

Lógica de Programação
Algoritmos usando Java
(iniciante)

- Conceitos de programação
- Introdução à linguagem Java
- Estrutura sequencial
- Estrutura condicional
- Estruturas repetitivas

Java e Programação
Orientada a Objetos
(PARTE 1)

- Introdução à Programação Orientada a Objetos
- Construtores, palavra this, sobrecarga, encapsulamento
- Comportamento de memória, arrays, listas
- Tópicos especiais em Java
- BÔNUS - Nivelamento sobre Git e Github**
- Enumerações, composição
- Herança e polimorfismo
- Tratamento de exceções

PROJETO
Sistema jogo de xadrez

- Composição de objetos, herança, coleções, etc.
- Desenvolvimento em camadas
- Padrões de projetos

Java e Programação
Orientada a Objetos
(PARTE 2)
+
Programação Funcional
(lambda)

- Trabalhando com arquivos
- Interfaces
- Generics, Set, Map
- Programação funcional e expressões lambda
- BÔNUS - Nivelamento Álgebra Relacional, SQL, MySQL**
- Acesso a banco de dados com JDBC
- Interface Gráfica com JavaFX

PROJETO

Aplicação desktop com JavaFX e banco de dados MySQL com JDBC

Padrão MVC - Model View Controller

Padrão Camadas

FXML, SceneBuilder

Tratamento de eventos de UI

CRUD completo

Padrões de projetos

PROJETO

Web services com Spring Boot e JPA / Hibernate

BÔNUS - Nivelamento ORM com JPA / Hibernate

Padrão camadas

Web e HTTP

REST / web services

Modelo de domínio complexo

Maven

Banco de dados H2

Spring Data JPA

CRUD completo

Tratamento de exceções

PROJETO

Web services com Spring Boot e NoSQL (MongoDB)

BÔNUS - Nivelamento NoSQL e MongoDB

Modelo de domínio: agregados e desnормalização

Padrão DTO

Spring Data MongoDB

CRUD completo

Tratamento de exceções

Curso Java COMPLETO

Capítulo: Introdução sobre programação

<http://educandoweb.com>

Prof. Dr. Nelio Alves

Algoritmo, Automação, Programa de Computador

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Algoritmo

Sequência finita de instruções para se resolver um problema.

* aplica-se a diversas áreas de conhecimento

Exemplo:

Problema: lavar roupa suja

Algoritmo:

- 1) Colocar a roupa em um recipiente
- 2) Colocar um pouco de sabão e amaciante
- 3) Encher de água
- 4) Mexer tudo até dissolver todo o sabão
- 5) Deixar de molho por vinte minutos
- 6) Esfregar a roupa
- 7) Enxaguar
- 8) Torcer

Automação

Consiste em utilizar máquina(s) para executar o procedimento desejado de forma automática ou semiautomática.

Algoritmo:

- 1) Colocar a roupa em um recipiente
- 2) Colocar um pouco de sabão e amaciante
- 3) Encher de água
- 4) Mexer tudo até dissolver todo o sabão
- 5) Deixar de molho por vinte minutos
- 6) Esfregar a roupa
- 7) Enxaguar
- 8) Torcer



Mas o que algoritmo e
automação tem a ver com
programação de computadores?

Computador

- Hardware - parte física (a máquina em si)
- Software - parte lógica (programas)
 - Sistema operacional (Windows, Linux, iOS)
 - Aplicativos (apps de escritório, app de câmera, navegador web)
 - Jogos
 - Utilitários (Antivírus, compactador de arquivos)
 - Outros



Programa ~ Algoritmo

Programas de computador **são algoritmos** executados pelo computador (em linhas gerais).

Conclusão: o computador é uma máquina que **automatiza** a execução de **algoritmos**.

Qualquer algoritmo? Não. Apenas algoritmos computacionais:

- Processamento de dados
- Cálculos

Resumo da aula

- Algoritmo: sequência finita de instruções para se resolver um problema
- Automação: quando uma máquina realiza o algoritmo
- Computador:
 - hardware / software
 - máquina que automatiza algoritmos (de cálculo)
- Programa de computador: algoritmo executado pelo computador

O que é preciso para se fazer um programa de computador?

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Vamos precisar de:

- Uma **linguagem de programação**: regras **léxicas** e **sintáticas** para se escrever o programa
- Uma **IDE**: software para editar e testar o programa
- Um **compilador**: software para transformar o **código fonte** em **código objeto**
- Um **gerador de código** ou **máquina virtual**: software que permite que o programa seja executado

Linguagem de programação, léxica, sintática

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Vamos precisar:

- Uma **linguagem de programação**: regras **léxicas e sintáticas** para se escrever o programa
- Uma **IDE**: software para editar e testar o programa
- Um **compilador**: software para transformar o **código fonte** em **código objeto**
- Um **gerador de código** ou **máquina virtual**: software que permite que o programa seja executado

Linguagem de programação

É um conjunto de regras **léxicas** (ortografia) e **sintáticas** (gramática) para se escrever programas.

Léxica

Diz respeito à correção das **palavras "isoladas"** (ortografia).

Exemplo (Português):

cachorro

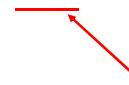
caxorro



Linguagem de programação:

main

maim



Sintática

Diz respeito à correção das **sentenças** (gramática).

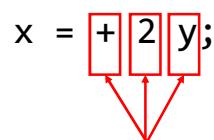
Exemplo (Português):

O cachorro está com fome.



Linguagem de programação:

$x = 2 + y;$



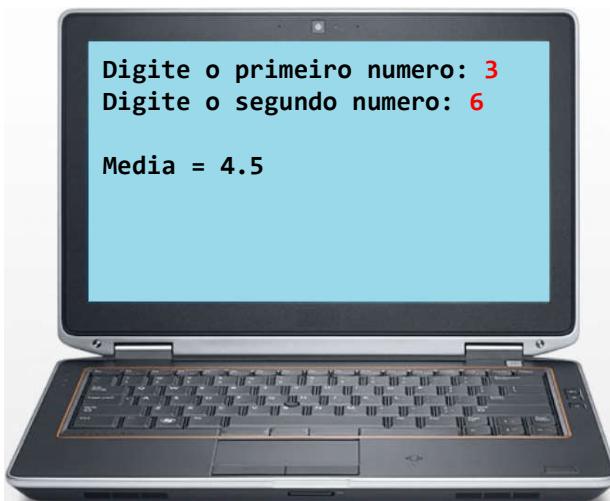
Linguagem de programação

Exemplos de linguagens de programação:

C, Pascal, C++, Java, C#, Python, Ruby, PHP, JavaScript, etc.

Exemplo de um programa:

Suponha um programa que solicita do usuário dois números e depois mostra a média aritmética deles:



Solução em linguagem C

```
#include <stdio.h>

int main() {
    double x, y, media;

    printf("Digite o primeiro numero: ");
    scanf("%lf", &x);
    printf("Digite o segundo numero: ");
    scanf("%lf", &y);
    media = (x + y) / 2.0;
    printf("Media = %.1f\n", media);
    return 0;
}
```

Solução em linguagem C++

```
#include <iostream>

using namespace std;

int main() {
    double x, y, media;

    cout << "Digite o primeiro numero: ";
    cin >> x;
    cout << "Digite o segundo numero: ";
    cin >> y;
    media = (x + y) / 2.0;
    cout << "Media = " << media << endl;
    return 0;
}
```

Solução em linguagem C#

```
using System;

namespace programa {
    class Program {
        static void Main(string[] args) {
            double x, y, media;

            Console.Write("Digite o primeiro numero: ");
            x = double.Parse(Console.ReadLine());
            Console.Write("Digite o segundo numero: ");
            y = double.Parse(Console.ReadLine());
            media = (x + y) / 2.0;
            Console.WriteLine("Media = " + media);
        }
    }
}
```

Solução em linguagem Java

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double x, y, media;

        System.out.print("Digite o primeiro numero: ");
        x = sc.nextDouble();
        System.out.print("Digite o segundo numero: ");
        y = sc.nextDouble();
        media = (x + y) / 2.0;
        System.out.println("Media = " + media);
        sc.close();
    }
}
```

Resumo da aula

- Linguagem: conjunto de regras léxicas e sintáticas para se escrever um programa
 - Léxica = ortografia. Palavras isoladas.
 - Sintática = gramática. Sentença como um todo.
- Exemplos de linguagens: C, Pascal, C++, Java, C#, Python, Ruby, PHP, JavaScript, etc.
- Exemplo de códigos feitos em linguagem C, C++, C# e Java

IDE: Ambiente de Desenvolvimento Integrado

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

IDE – Ambiente Integrado de Desenvolvimento

É um conjunto de softwares utilizado para a construção de programas.

Exemplos:

C/C++ : **Code Blocks**

Java : **Eclipse, NetBeans**

C# : **Microsoft Visual Studio**

Funcionalidades de uma IDE

- Edição de código fonte (endentação, autocompletar, destaque de palavras, etc.)
- Depuração e testes
- Construção do produto final (build)
- Sugestão de modelos (templates)
- Auxiliar em várias tarefas do seu projeto
- Etc.

Resumo da aula

- IDE: é um conjunto de softwares utilizado para a construção de programas
 - C/C++ : **Code Blocks**
 - Java : **Eclipse, NetBeans**
 - C# : **Microsoft Visual Studio**
- Uma IDE oferece várias funcionalidades para facilitar a construção dos programas

Compilação e interpretação

Código fonte e objeto

Máquina virtual

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Código fonte: é aquele escrito pelo programador em linguagem de programação

```
#include <stdio.h>

int main() {
    double x, y, media;
    printf("Digite o primeiro numero: ");
    scanf("%lf", &x);
    printf("Digite o segundo numero: ");
    scanf("%lf", &y);
    media = (x + y) / 2.0;
    printf("Media = %.1f\n", media);
    return 0;
}
```

```
#include <iostream>

using namespace std;

int main() {
    double x, y, media;
    cout << "Digite o primeiro numero: ";
    cin >> x;
    cout << "Digite o segundo numero: ";
    cin >> y;
    media = (x + y) / 2.0;
    cout << "Media = " << media << endl;
    return 0;
}
```

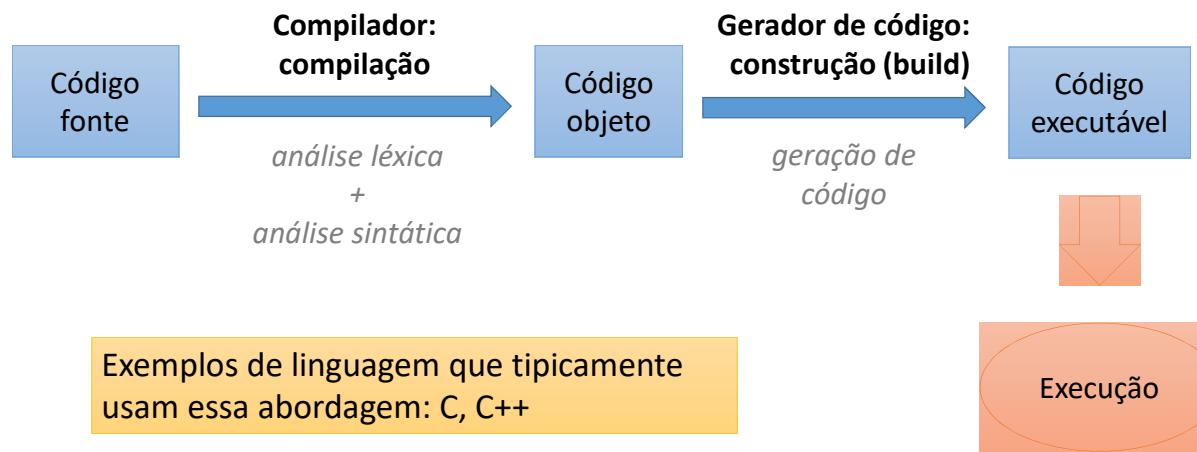
```
using System;
namespace programa {
    class Program {
        static void Main(string[] args) {
            double x, y, media;
            Console.WriteLine("Digite o primeiro numero: ");
            x = double.Parse(Console.ReadLine());
            Console.WriteLine("Digite o segundo numero: ");
            y = double.Parse(Console.ReadLine());
            media = (x + y) / 2.0;
            Console.WriteLine("Media = " + media);
        }
    }
}
```

```
import java.util.Scanner;

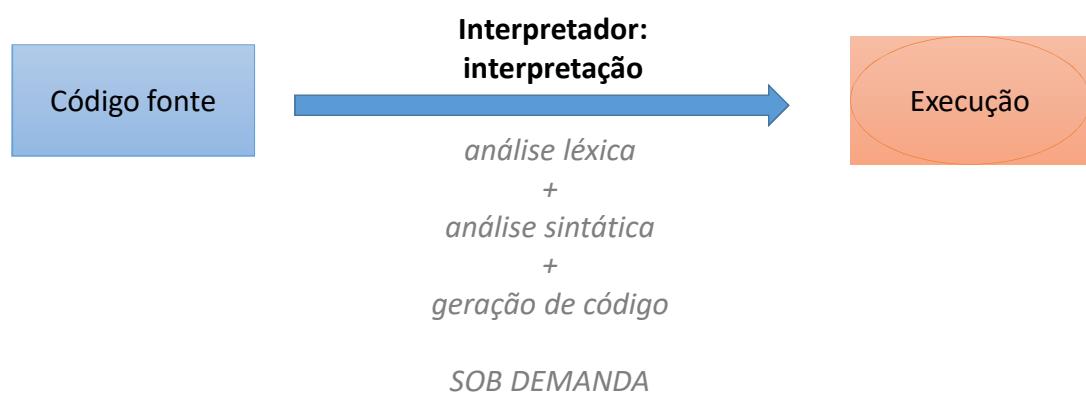
public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double x, y, media;
        System.out.print("Digite o primeiro numero: ");
        x = sc.nextDouble();
        System.out.print("Digite o segundo numero: ");
        y = sc.nextDouble();
        media = (x + y) / 2.0;
        System.out.println("Media = " + media);
        sc.close();
    }
}
```

Compilação

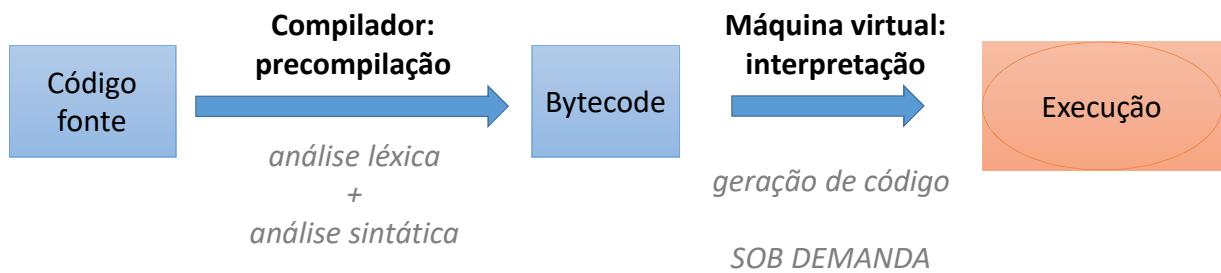


Interpretação



Exemplos de linguagem que tipicamente usam essa abordagem: PHP, JavaScript, Python, Ruby

Abordagem híbrida



Exemplos de linguagem que tipicamente usam essa abordagem: Java (JVM), C# (Microsoft .NET Framework)

Vantagens

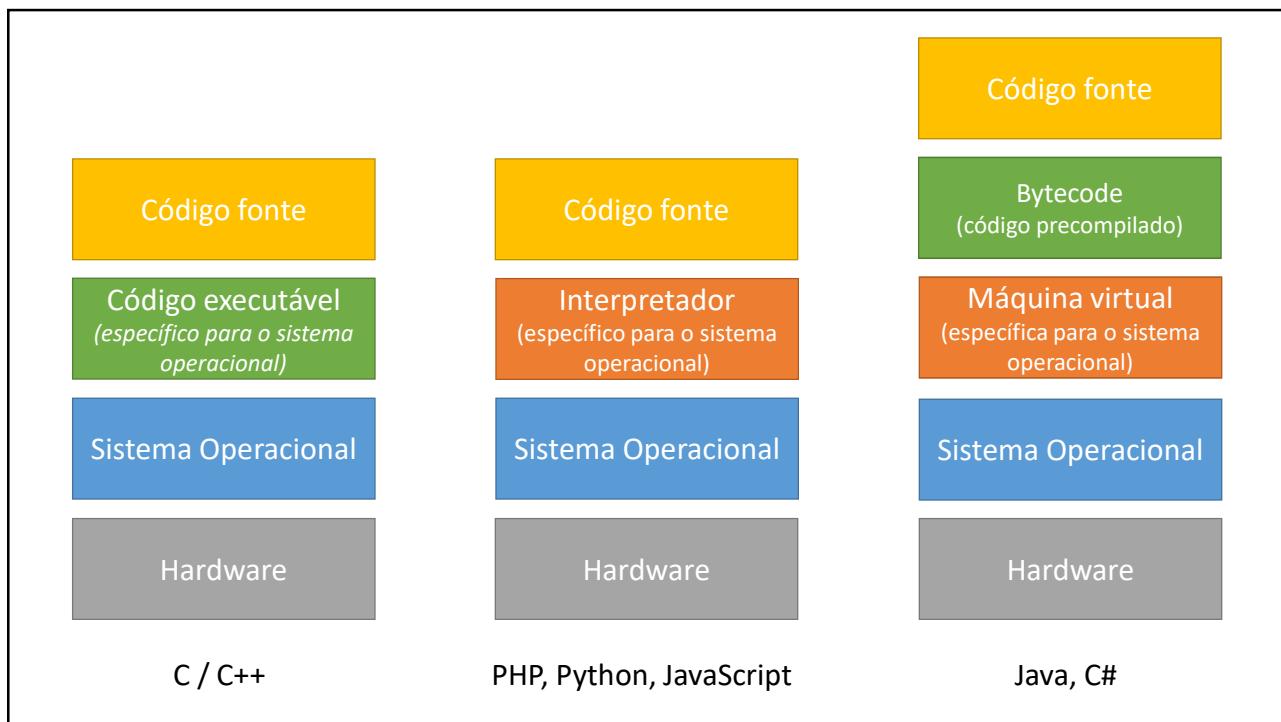
Compilação:

- velocidade do programa
- auxílio do compilador antes da execução

Abordagem híbrida

Interpretação:

- flexibilidade de manutenção do aplicativo em produção
- expressividade da linguagem
- código fonte não precisa ser recompilado para rodar em plataformas diferentes



Resumo da aula

- **Tipos de código**
 - Código fonte
 - Código objeto / bytecode
- **Modelos de execução:**
 - Compilação
 - Gerador de código
 - Interpretação
 - Abordagem híbrida
 - Máquina virtual

Curso Java COMPLETO

Capítulo: Introdução à linguagem Java

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Entendendo as versões do Java

<http://www.oracle.com/technetwork/java/javase>

LTS - Long Term Support

Java - contextualização

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

O que é Java?

- Linguagem de programação (regras sintáticas)
- Plataforma de desenvolvimento e execução
 - Bibliotecas (API)
 - Ambientes de execução

Histórico

- Problemas resolvidos e motivo de seu sucesso:
 - Ponteiros / gerenciamento de memória
 - Portabilidade falha: reescrever parte do código ao mudar de SO
 - Utilização em dispositivos diversos
 - Custo
- Criada pela Sun Microsystems no meio da década de 1990
- Adquirida pela Oracle Corporation em 2010



Aspectos notáveis

- Código compilado para bytecode e executado em máquina virtual (JVM)
- Portável, segura, robusta
- Roda em vários tipos de dispositivos
- Domina o mercado corporativo desde o fim do século 20
- Padrão Android por muitos anos



Edições

- Java ME - Java Micro Edition - dispositivos embarcados e móveis - IoT
 - <http://www.oracle.com/technetwork/java/javame>
- Java SE - Java Standard Edition - core - desktop e servidores
 - <http://www.oracle.com/technetwork/java/javase>
- Java EE - Java Enterprise Edition - aplicações corporativas
 - <http://www.oracle.com/technetwork/java/javaee>

Plataforma Java SE

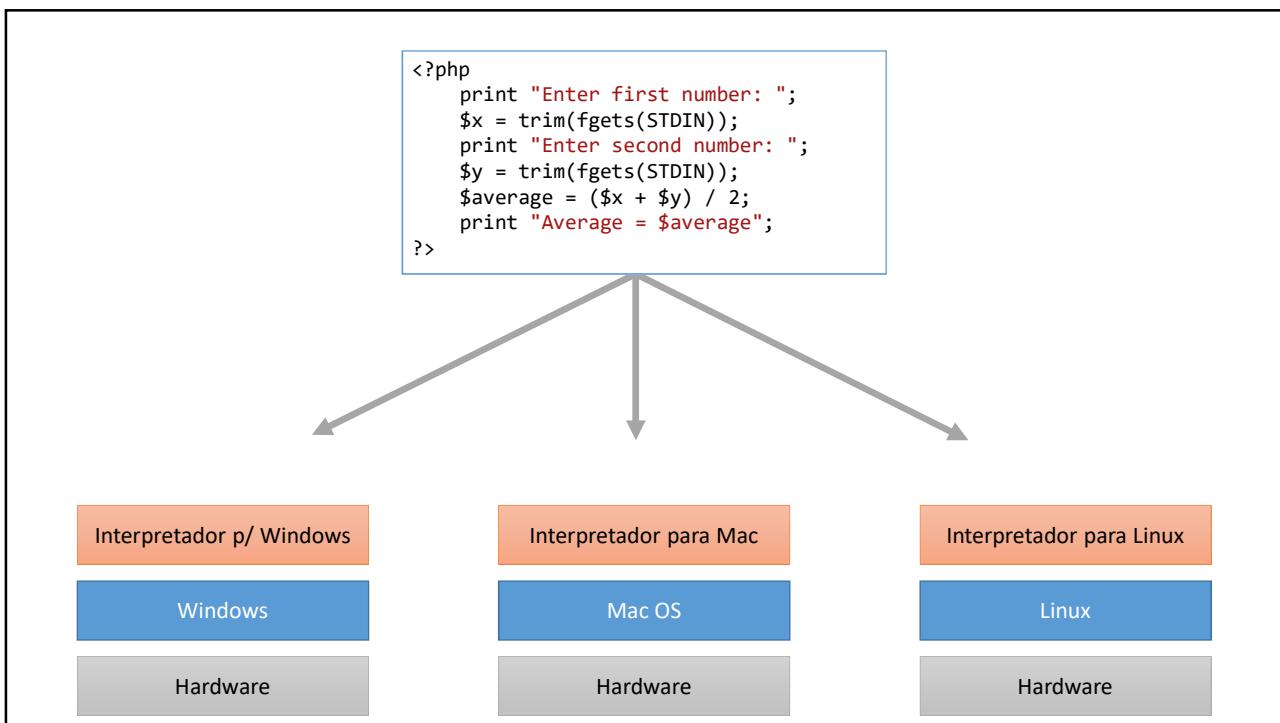
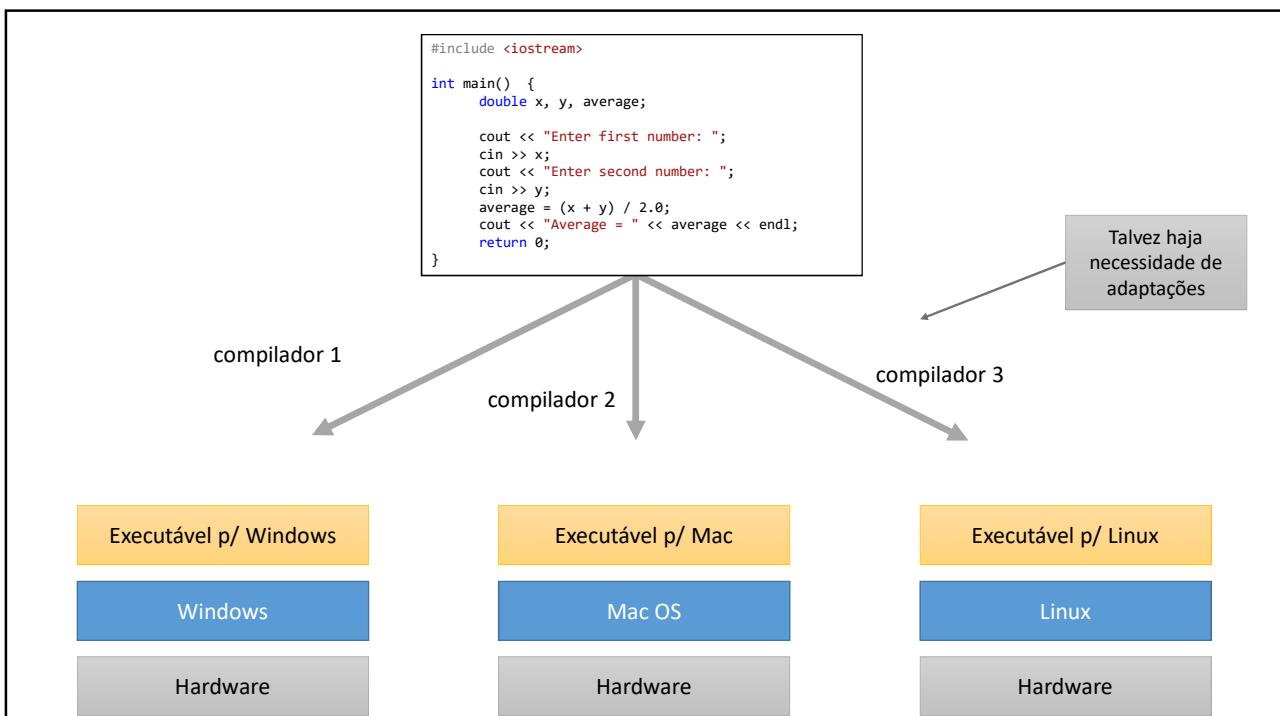
- Documentação
 - <https://docs.oracle.com/en/java/javase/11/>

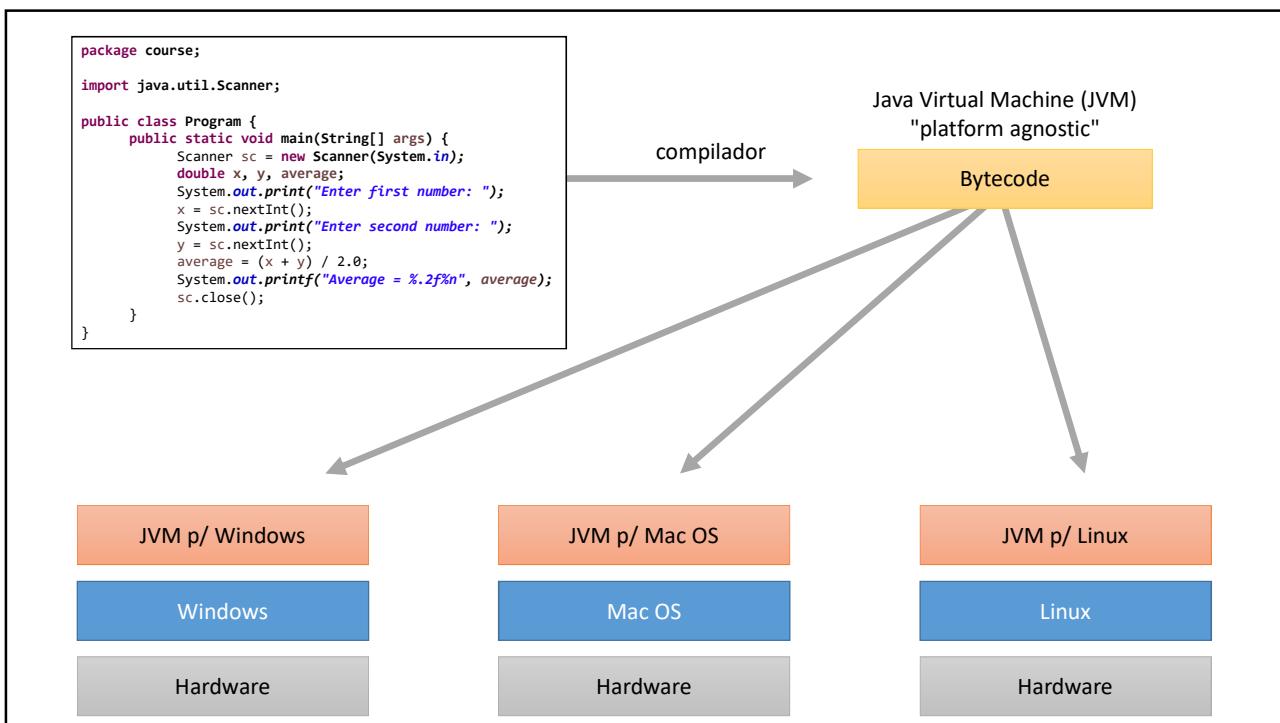
Plataforma Java SE

- JVM - Java Virtual Machine
 - Máquina virtual do Java - necessário para executar sistemas Java

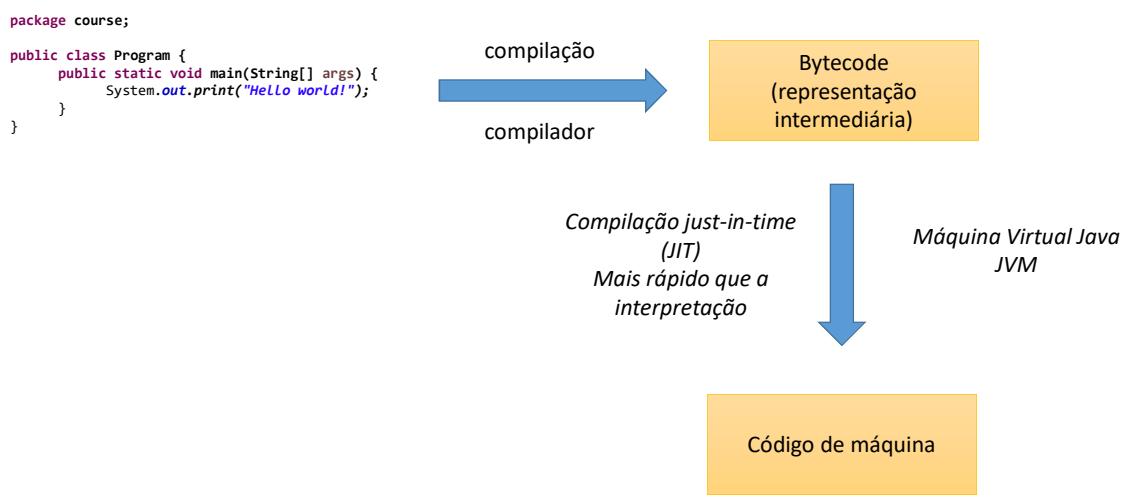
Compilação e interpretação

- Linguagens **compiladas**: C, C++
- Linguagens **interpretadas**: PHP, JavaScript
- Linguagens **pré-compiladas + máquina virtual**: Java, C#





Modelo de execução

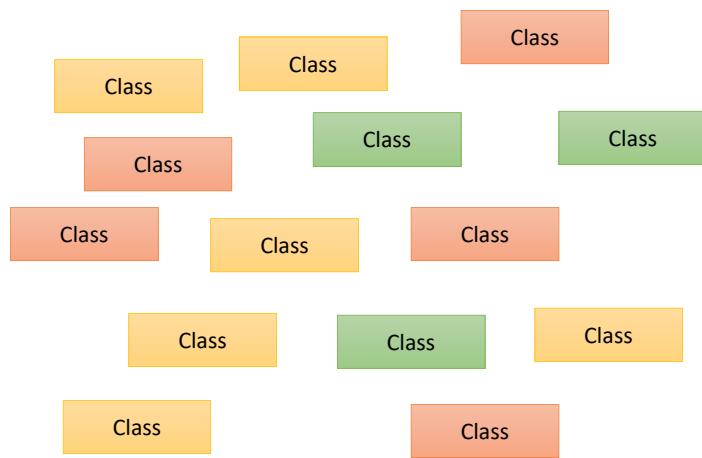


Estrutura de uma aplicação Java

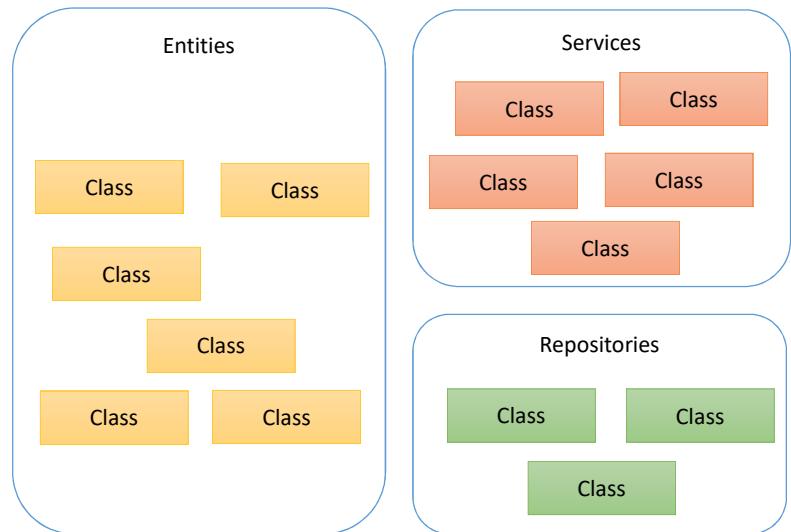
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

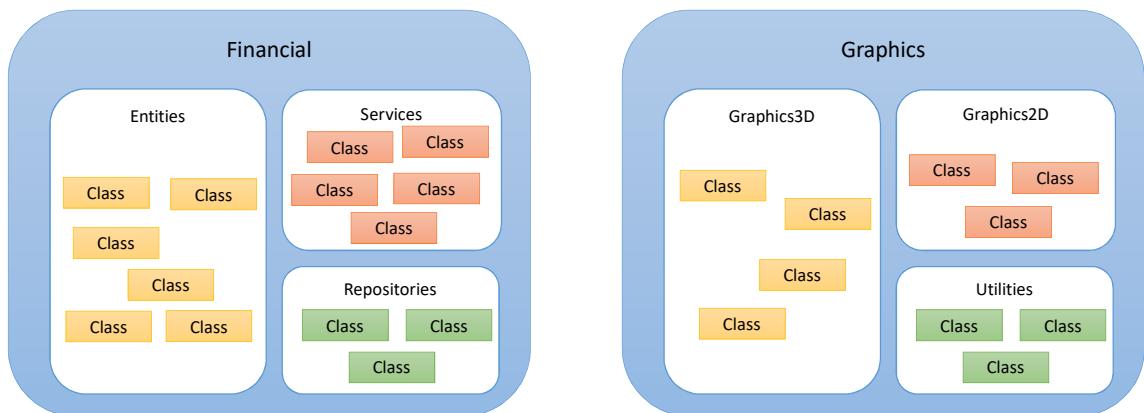
Uma aplicação é composta por classes



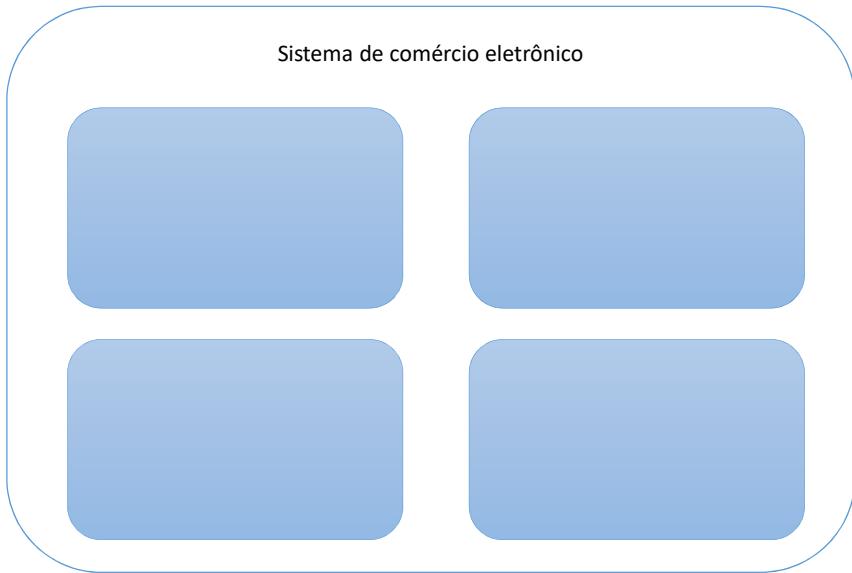
package = agrupamento LÓGICO de classes relacionadas



Módulo (Java 9+) = Agrupamento lógico de pacotes relacionados
Runtime = Agrupamento físico



Aplicação = Agrupamento de módulo relacionados



Instalação do Java e Eclipse no Windows

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Checklist

- Baixar e instalar o Java JDK
<https://www.oracle.com/java/technologies/javase-jdk11-downloads.html>
- Configurar variáveis de ambiente do sistema
 - Painel de Controle -> Variáveis de Ambiente
 - JAVA_HOME:
C:\Program Files\Java\jdk-11.0.4
 - Path: incluir
C:\Program Files\Java\jdk-11.0.4\bin
 - Testar no terminal de comando: `java -version`
- Baixar e descompactar o Eclipse
 - <https://www.eclipse.org/downloads/packages/>
 - Testar: rodar o Eclipse e escolher um "workspace" (pasta onde você vai salvar seus projetos)

Primeiro programa em Java e utilização básica do Eclipse

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

- Workspace (selecione a pasta aonde os projetos serão salvos)
- Mudar o layout: Window -> Perspective -> Open Perspective -> Java
- Zerar o layout: Window -> Perspective -> Reset Perspective
- Mostrar a aba Console: Window -> Show View -> Console
- Criar projeto: File -> New -> Java Project

- Criar classe:
 - Botão direito na pasta "src" -> New -> Class
 - Package: deixe em branco
 - Nome da classe: Main (com M maiúsculo)
 - Marque a opção: public static void main(String[] args)
- Mudar o tamanho da fonte:
 - CTRL +
 - CTRL -

Curso Java COMPLETO

Capítulo: Estrutura sequencial

<http://educandoweb.com.br>

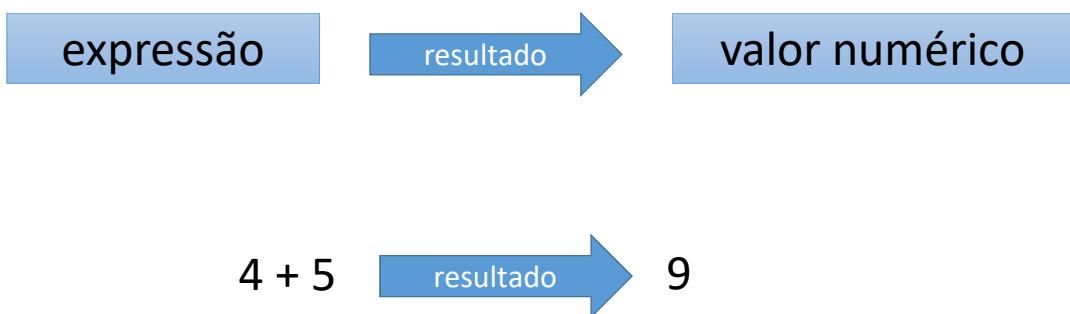
Prof. Dr. Nelio Alves

Expressões aritméticas

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Expressões aritméticas



Operadores aritméticos

C, C++,
Java, C# →

Operador	Significado
+	adição
-	subtração
*	multiplicação
/	divisão
%	resto da divisão ("mod")

Precedência: 1º lugar: * / %
 2º lugar: + -

Exemplos de expressões aritméticas

$$2 * 6 / 3 \quad \text{Resultado} = 4$$

$$3 + 2 * 4 \quad \text{Resultado} = 11$$

$$(3 + 2) * 4 \quad \text{Resultado} = 20$$

$$60 / (3 + 2) * 4 \quad \text{Resultado} = 48$$

$$60 / ((3 + 2) * 4) \quad \text{Resultado} = 3$$

Exemplos com o operador "mod"

$$14 \% 3 \quad \text{Resultado} = 2$$

$$19 \% 5 \quad \text{Resultado} = 4$$

Pois:

$$\begin{array}{r} 14 \\ \hline 2 \end{array} \quad \begin{array}{r} 3 \\ \hline 4 \end{array}$$

$$\begin{array}{r} 19 \\ \hline 4 \end{array} \quad \begin{array}{r} 5 \\ \hline 3 \end{array}$$

Variáveis e tipos primitivos em Java

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

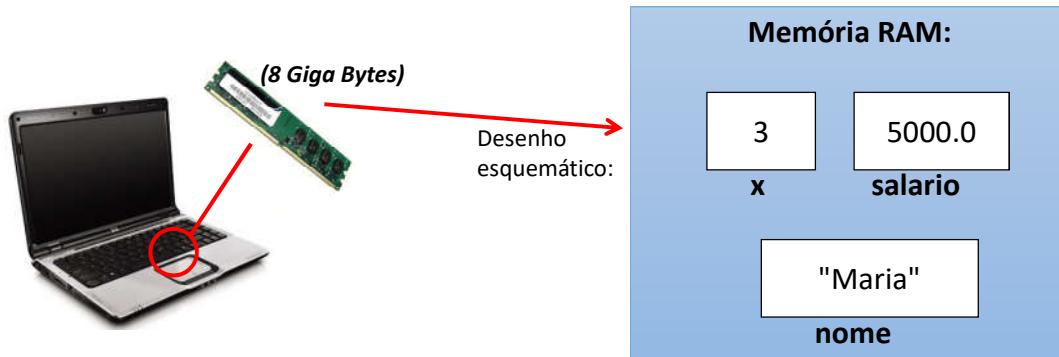
Visão geral

- Um programa de computador em execução lida com dados
- Como esses dados são armazenados?
- Em **VARIÁVEIS!**

Variáveis

Definição informal:

Em programação, uma variável é uma porção de memória (RAM) utilizada para armazenar dados durante a execução dos programas.



Declaração de variáveis

Sintaxe:

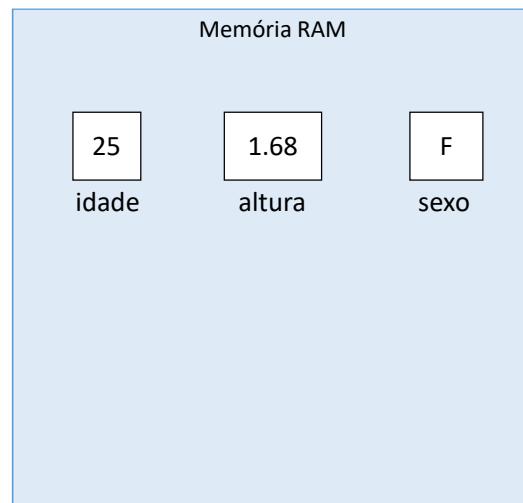
```
<tipo> <nome> = <valor inicial>;  
                                ^_____  
                                |       (opcional)
```

Exemplos:

```
int idade = 25;  
double altura = 1.68;  
char sexo = 'F';
```

Uma variável possui:

- Nome (ou identificador)
- Tipo
- Valor
- Endereço



Tipos primitivos em Java

Descrição	Tipo	Tamanho	Valores	Valor padrão
tipos numéricos inteiros	byte	8 bits	-128 a 127	0
	short	16 bits	-32768 a 32767	0
	int	32 bits	-2147483648 a 2147483647	0
	long	64 bits	-9223372036854770000 a 9223372036854770000	0L
tipos numéricos com ponto flutuante	float	32 bits	-1,4024E-37 a 3,4028E+38	0.0f
	double	64 bits	-4,94E-307 a 1,79E+308	0.0
um caractere Unicode	char	16 bits	'\u0000' a '\uFFFF'	'\u0000'
valor verdade	boolean	1 bit	{false, true}	false

String - cadeia de caracteres (palavras ou textos)

Veja: unicode-table.com

Exemplo: 'a' = '\u0061'

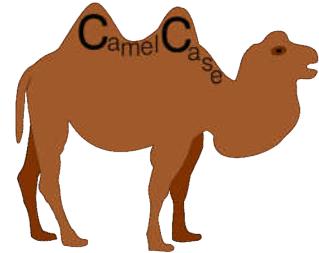
Um bit pode armazenar 2 valores possíveis (0 ou 1)

Cada bit = 2 possibilidades

8 bits:

Nomes de variáveis

- Não pode começar com dígito: use uma letra ou _
- Não pode ter espaço em branco
- Não usar acentos ou til
- Sugestão: use o padrão "camel case"



Errado:

```
int 5minutos;  
int salário;  
int salário do funcionario;
```

Correto:

```
int _5minutos;  
int salario;  
int salarioDoFuncionario;
```

Resumo da aula

- Conceito informal
- Declaração de variáveis: <tipo> <nome> = valor;
- Tipos primitivos:
 - **Números inteiros:** byte, short, int, long
 - **Números com ponto flutuante:** float, double
 - **Valor verdade:** boolean
 - **Um caractere Unicode:** char
- Tipo String: cadeia de caracteres (palavras, textos)
- Nomes de variáveis / padrão camel case

As três operações básicas de programação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Um programa de computador é capaz de realizar essencialmente três operações:

Entrada
de dados



Processamento
de dados



Saída
de dados



Entrada de dados

Usuário → Programa
(dentro de variáveis)



Dispositivo de ENTRADA



Também chamada de
LEITURA:

"O programa está lendo dados."

Processamento de dados

É quando o programa realiza os cálculos



O processamento de
dados se dá por um
comando chamado
ATRIBUIÇÃO

`media = (x + y) / 2.0;`

Saída de dados

Programa → Usuário



Dispositivo de SAÍDA



Também chamada de
ESCRITA:

"O programa está escrevendo dados."

Saída de dados em Java

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Saída de dados

Programa → Usuário



Dispositivo de SAÍDA



Também chamada de
ESCRITA:

"O programa está escrevendo dados."

Para escrever na tela um texto qualquer

Sem quebra de linha ao final:

```
System.out.print("Bom dia!");
```

Com quebra de linha ao final:

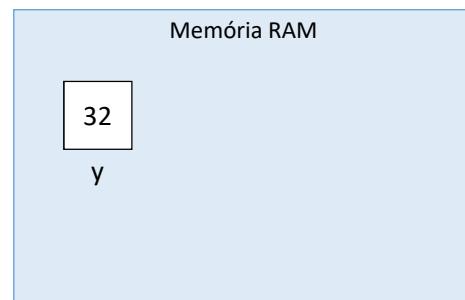
```
System.out.println("Bom dia!");
```

Para escrever o conteúdo de uma variável de algum tipo básico

Suponha uma variável tipo **int** declarada e iniciada:

```
int y = 32;
```

```
System.out.println(y);
```



Para escrever o conteúdo de uma variável com ponto flutuante

Suponha uma variável tipo **double** declarada e iniciada:

```
double x = 10.35784;
```

%n = quebra de linha
(independente de plataforma)

```
System.out.println(x);
```

```
System.out.printf("%.2f%n", x);
```

```
System.out.printf("%.4f%n", x);
```

Localidade do sistema

ATENÇÃO:

Para considerar o separador de decimais como ponto, **ANTES** da declaração do Scanner, faça:

```
Locale.setDefault(Locale.US);
```

Para concatenar vários elementos em um mesmo comando de escrita

Regra geral para **print** e **println**:

elemento1 + elemento2 + elemento3 + ... + elementoN

```
System.out.println("RESULTADO = " + x + " METROS");
```

Para concatenar vários elementos em um mesmo comando de escrita

Regra geral para **printf**:

"TEXT01 %f TEXT02 %f TEXT03", variavel1, variavel2

%f = ponto flutuante

%n = quebra de linha

```
System.out.printf("RESULTADO = %.2f metros%n", x);
```

MAIS INFORMAÇÕES: <https://docs.oracle.com/javase/tutorial/java/data/numberformat.html>

Para concatenar vários elementos em um mesmo comando de escrita

Regra geral para **printf**:

```
"TEXT01 %f TEXT02 %f TEXT03", variavel1, variavel2
```

%f = ponto flutuante

%d = inteiro

%s = texto

%n = quebra de linha

```
String nome = "Maria";
int idade = 31;
double renda = 4000.0;
System.out.printf("%s tem %d anos e ganha R$ %.2f reais%n", nome, idade, renda);
```

MAIS INFORMAÇÕES: <https://docs.oracle.com/javase/tutorial/java/data/numberformat.html>

Resumo da aula

- System.out.print
- System.out.println
- System.out.printf
 - %d
 - %f
 - %s
 - %n
- Locale
- Como concatenar vários elementos em um mesmo comando de escrita
- Exemplos

Exercício de fixação

Em um novo programa, inicie as seguintes variáveis:

```
String product1 = "Computer";
String product2 = "Office desk";

int age = 30;
int code = 5290;
char gender = 'F';

double price1 = 2100.0;
double price2 = 650.50;
double measure = 53.234567;
```

Em seguida, usando os valores das variáveis, produza a seguinte saída na tela do console:

```
Products:
Computer, which price is $ 2100,00
Office desk, which price is $ 650,50

Record: 30 years old, code 5290 and gender: F

Measue with eight decimal places: 53,23456700
Rouded (three decimal places): 53,235
US decimal point: 53.235
```

(correção na próxima página)

```
import java.util.Locale;

public class Main {

    public static void main(String[] args) {

        String product1 = "Computer";
        String product2 = "Office desk";

        byte age = 30;
        int code = 5290;
        char gender = 'F';

        double price1 = 2100.0;
        double price2 = 650.50;
        double measure = 53.234567;

        System.out.println("Products:");
        System.out.printf("%s, which price is $ %.2f%n", product1, price1);
        System.out.printf("%s, which price is $ %.2f%n", product2, price2);
        System.out.println();
        System.out.printf("Record: %d years old, code %d and gender: %c%n", age, code, gender);
        System.out.println();
        System.out.printf("Measue with eight decimal places: %.8f%n", measure);
        System.out.printf("Rouded (three decimal places): %.3f%n", measure);
        Locale.setDefault(Locale.US);
        System.out.printf("US decimal point: %.3f%n", measure);
    }
}
```

- Comentários de linha:
 - Começam com //
- Atalhos:
 - Importar classes: CTRL + SHIFT + O
 - Autoendentaçāo: CTRL + SHIFT + F
 - sysout CTRL + espaço

Processamento de dados em Java, Casting

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Processamento de dados

Comando de atribuição.

Sintaxe:

`<variável> = <expressão>;`

Lê-se “recebe”


REGRA:

- 1) A expressão é calculada
- 2) O resultado da expressão é armazenado na variável

Exemplo 1

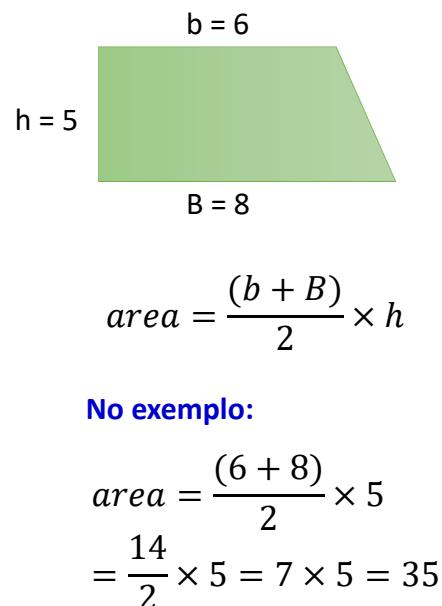
```
int x, y;  
  
x = 5;  
  
y = 2 * x;  
  
System.out.println(x);  
System.out.println(y);
```

Exemplo 2

```
int x;  
double y;  
  
x = 5;  
  
y = 2 * x;  
  
System.out.println(x);  
System.out.println(y);
```

Exemplo 3

```
double b, B, h, area;  
  
b = 6.0;  
B = 8.0;  
h = 5.0;  
  
area = (b + B) / 2.0 * h;  
  
System.out.println(area);
```



```
double b, B, h, area;  
  
b = 6.0;  
B = 8.0;  
h = 5.0;  
  
area = (b + B) / 2.0 * h;  
  
System.out.println(area);
```

Boa prática:

Sempre indique o tipo do número, se a expressão for de ponto flutuante (não inteira).

Para **double** use:
.0

Para **float** use:
f

```
float b, B, h, area;  
  
b = 6f;  
B = 8f;  
h = 5f;  
  
area = (b + B) / 2f * h;  
  
System.out.println(area);
```

Boa prática:

Sempre indique o tipo do número, se a expressão for de ponto flutuante (não inteira).

Para **double** use:
.0

Para **float** use:
f

Exemplo 4

```
int a, b;  
double resultado;  
  
a = 5;  
b = 2;  
  
resultado = a / b;  
  
System.out.println(resultado);
```

Casting

É a conversão explícita de um tipo para outro.

É necessário quando o compilador não é capaz de “adivinar” que o resultado de uma expressão deve ser de outro tipo.

Exemplo 4

```
int a, b;
double resultado;

a = 5;
b = 2;

resultado = a / b;

System.out.println(resultado);
```

Exemplo 4

```
int a, b;
double resultado;

a = 5;
b = 2;

resultado = (double) a / b;

System.out.println(resultado);
```

Exemplo 5

```
double a;  
int b;  
  
a = 5.0;  
b = a;  
  
System.out.println(b);
```

Entrada de dados em Java

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Entrada de dados

Usuário → Programa
(dentro de variáveis)



Dispositivo de ENTRADA



Também chamada de
LEITURA:

"O programa está lendo dados."

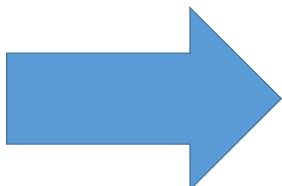


`int x;`



32

ENTER



Memória RAM

32

x

Scanner

Para fazer entrada de dados, nós vamos criar um objeto do tipo "Scanner" da seguinte forma:

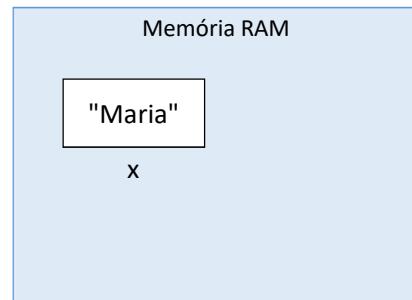
```
Scanner sc = new Scanner(System.in);  
import java.util.Scanner;  
faça sc.close() quando não precisar mais do objeto sc
```

Para ler uma palavra (texto sem espaços)

Suponha uma variável tipo **String** declarada:

```
String x;
```

```
x = sc.next();
```

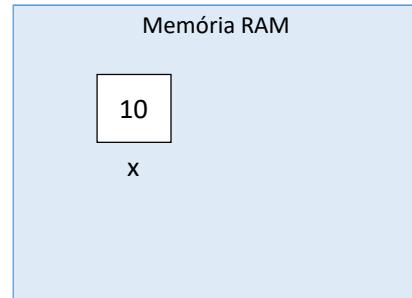


Para ler um número inteiro

Suponha uma variável tipo **int** declarada:

```
int x;
```

```
x = sc.nextInt();
```



Para ler um número com ponto flutuante

Suponha uma variável tipo **double** declarada:

```
double x;
```

```
x = sc.nextDouble();
```

Localidade do sistema

ATENÇÃO:

Para considerar o separador de decimais como ponto, **ANTES** da declaração do Scanner, faça:

```
Locale.setDefault(Locale.US);
```

Para ler um caractere

Suponha uma variável tipo **char** declarada:

```
char x;
```

```
x = sc.next().charAt(0);
```

Para ler vários dados na mesma linha

```
string x;
int y;
double z;
```

```
x = sc.next();
y = sc.nextInt();
z = sc.nextDouble();
```

Para ler um texto ATÉ A QUEBRA DE LINHA

```
import java.util.Scanner;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner sc = new Scanner(System.in);  
  
        String s1, s2, s3;  
  
        s1 = sc.nextLine();  
        s2 = sc.nextLine();  
        s3 = sc.nextLine();  
  
        System.out.println("DADOS DIGITADOS:");  
        System.out.println(s1);  
        System.out.println(s2);  
        System.out.println(s3);  
  
        sc.close();  
    }  
}
```

ATENÇÃO: quebra de linha pendente

Quando você usa um comando de leitura diferente do nextLine() e dá alguma quebra de linha, essa quebra de linha fica "pendente" na entrada padrão.

Se você então fizer um nextLine(), aquela quebra de linha pendente será absorvida pelo nextLine().

Solução:

Faça um nextLine() extra antes de fazer o nextLine() de seu interesse.

```
int x;  
String s1, s2, s3;  
  
x = sc.nextInt();  
s1 = sc.nextLine();  
s2 = sc.nextLine();  
s3 = sc.nextLine();  
  
System.out.println("DADOS DIGITADOS:");  
System.out.println(x);  
System.out.println(s1);  
System.out.println(s2);  
System.out.println(s3);
```

Resumo da aula

- Scanner
 - next()
 - nextInt()
 - nextDouble()
 - next().charAt(0)
- Locale
- Como ler até a quebra de linha
 - nextLine()
 - como limpar o buffer de leitura

Funções matemáticas em Java

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Algumas funções matemáticas em Java

Exemplo	Significado
A = Math.sqrt(x);	Variável A recebe a raiz quadrada de x
A = Math.pow(x, y);	Variável A recebe o resultado de x elevado a y
A = Math.abs(x);	Variável A recebe o valor absoluto de x

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

        double x = 3.0;
        double y = 4.0;
        double z = -5.0;
        double A, B, C;

        A = Math.sqrt(x);
        B = Math.sqrt(y);
        C = Math.sqrt(25.0);
        System.out.println("Raiz quadrada de " + x + " = " + A);
        System.out.println("Raiz quadrada de " + y + " = " + B);
        System.out.println("Raiz quadrada de 25 = " + C);

        A = Math.pow(x, y);
        B = Math.pow(x, 2.0);
        C = Math.pow(5.0, 2.0);
        System.out.println(x + " elevado a " + y + " = " + A);
        System.out.println(x + " elevado ao quadrado = " + B);
        System.out.println("5 elevado ao quadrado = " + C);

        A = Math.abs(y);
        B = Math.abs(z);
        System.out.println("Valor absoluto de " + y + " = " + A);
        System.out.println("Valor absoluto de " + z + " = " + B);
    }
}
```

Incluindo funções em expressões maiores

$$x = \frac{-b \pm \sqrt{\Delta}}{2.a}$$

$$\Delta = b^2 - 4ac$$

```
delta = Math.pow(b, 2.0) - 4*a*c;
```

```
x1 = (-b + Math.sqrt(delta)) / (2.0 * a);  
x2 = (-b - Math.sqrt(delta)) / (2.0 * a);
```

Funções matemáticas

- sqrt – raiz quadrada
- pow – potenciação
- abs – valor absoluto
- Exemplos

Maiores informações: `java.lang.Math`

Curso Java Completo

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Capítulo: Apresentação da linguagem Java e tópicos básicos

Exercícios sobre Estrutura Sequencial (entrada, processamento, saída)

* Exercícios obtidos do URI Online Judge: www.urionlinejudge.com.br

Atenção! Esses exercícios são:

- **OPCIONAIS** para quem já sabe Lógica de Programação em qualquer linguagem
- **NECESSÁRIOS** para alunos **iniciantes**

Exercício resolvido:

- <https://www.youtube.com/watch?v=Ah1Y6d6deq0>

Exercícios propostos:

Exercício 01

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1003.java>

Faça um programa para ler dois valores inteiros, e depois mostrar na tela a soma desses números com uma mensagem explicativa, conforme exemplos.

Exemplos:

Entrada:	Saída:
10 30	SOMA = 40

Entrada:	Saída:
-30 10	SOMA = -20

Entrada:	Saída:
0 0	SOMA = 0

Exercício 02

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1002.java>

Faça um programa para ler o valor do raio de um círculo, e depois mostrar o valor da área deste círculo com quatro casas decimais conforme exemplos.

Fórmula da área: area = $\pi \cdot \text{raio}^2$

Considere o valor de $\pi = 3.14159$

Exemplos:

Entrada:	Saída:
2.00	A=12.5664
100.64	A=31819.3103
150.00	A=70685.7750

Exercício 03

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1007.java>

Fazer um programa para ler quatro valores inteiros A, B, C e D. A seguir, calcule e mostre a diferença do produto de A e B pelo produto de C e D segundo a fórmula: DIFERENCA = (A * B - C * D).

Exemplos:

Entrada:	Saída:
5 6 7 8	DIFERENCA = -26
5 6 -7 8	DIFERENCA = 86

Exercício 04

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1008.java>

Fazer um programa que leia o número de um funcionário, seu número de horas trabalhadas, o valor que recebe por hora e calcula o salário desse funcionário. A seguir, mostre o número e o salário do funcionário, com duas casas decimais.

Exemplos:

Entrada:	Saída:
25 100 5.50	NUMBER = 25 SALARY = U\$ 550.00

Entrada:	Saída:
1 200 20.50	NUMBER = 1 SALARY = U\$ 4100.00

Entrada:	Saída:
6 145 15.55	NUMBER = 6 SALARY = U\$ 2254.75

Exercício 05

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1010.java>

Fazer um programa para ler o código de uma peça 1, o número de peças 1, o valor unitário de cada peça 1, o código de uma peça 2, o número de peças 2 e o valor unitário de cada peça 2. Calcule e mostre o valor a ser pago.

Exemplos:

Entrada:	Saída:
12 1 5.30 16 2 5.10	VALOR A PAGAR: R\$ 15.50

Entrada:	Saída:
13 2 15.30 161 4 5.20	VALOR A PAGAR: R\$ 51.40

Entrada:	Saída:
1 1 15.10 2 1 15.10	VALOR A PAGAR: R\$ 30.20

Exercício 06

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1012.java>

Fazer um programa que leia três valores com ponto flutuante de dupla precisão: A, B e C. Em seguida, calcule e mostre:

- a) a área do triângulo retângulo que tem A por base e C por altura.
- b) a área do círculo de raio C. ($\pi = 3.14159$)
- c) a área do trapézio que tem A e B por bases e C por altura.
- d) a área do quadrado que tem lado B.
- e) a área do retângulo que tem lados A e B.

Exemplos:

Entrada:	Saída:
3.0 4.0 5.2	TRIANGULO: 7.800 CIRCULO: 84.949 TRAPEZIO: 18.200 QUADRADO: 16.000 RETANGULO: 12.000

Entrada:	Saída:
12.7 10.4 15.2	TRIANGULO: 96.520 CIRCULO: 725.833 TRAPEZIO: 175.560 QUADRADO: 108.160 RETANGULO: 132.080

Curso Java COMPLETO

Capítulo: Estrutura condicional

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Expressões comparativas

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Expressões comparativas

expressão

resultado

valor verdade

$5 > 10$

resultado

Falso

Operadores comparativos

C, C++,
Java, C# →

Operador	Significado
>	maior
<	menor
\geq	maior ou igual
\leq	menor ou igual
$=$	igual
\neq	diferente

Exemplos de expressões comparativas

(suponha x igual a 5)

X > 0 Resultado: V

X == 3 Resultado: F

$10 \leq 30$ Resultado: V

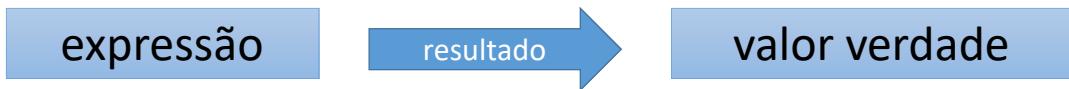
X != 2 Resultado: V

Expressões lógicas

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Expressões lógicas



Operadores lógicos

C, C++,
Java, C# →

Operador	Significado
&&	E
	OU
!	NÃO

Ideia por trás do operador "E"

Você pode obter uma habilitação de motorista se:

- For aprovado no exame psicotécnico,
E
- For aprovado no exame de legislação,
E
- For aprovado no exame de direção

**Todas condições
devem ser
verdadeiras!**

Exemplos de expressões lógicas

(suponha x igual a 5)

$$X \leq 20 \quad \&\& \quad X == 10 \quad \text{Resultado: F}$$



$$X > 0 \quad \&\& \quad X != 3 \quad \text{Resultado: V}$$



$$X \leq 20 \quad \&\& \quad X == 10 \quad \&\& \quad X != 3 \quad \text{Resultado: F}$$



Tabela verdade do operador "E"

A	B	A & B
F	F	F
F	V	F
V	F	F
V	V	V

Ideia por trás do operador "OU"

Você pode estacionar na vaga especial se:

- For idoso(a),
OU
- For uma pessoa com deficiência,
OU
- For uma gestante

**Pelo menos uma
condição deve
ser verdadeira!**

Exemplos de expressões lógicas

(suponha x igual a 5)

$$X == 10 \quad || \quad X \leq 20 \quad \text{Resultado: V}$$



$$X > 0 \quad || \quad X \neq 3 \quad \text{Resultado: V}$$



$$X \leq 0 \quad || \quad X \neq 3 \quad || \quad X \neq 5 \quad \text{Resultado: V}$$

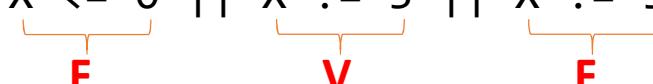


Tabela verdade do operador "OU"

A	B	A B
F	F	F
F	V	V
V	F	V
V	V	V

Ideia por trás do operador "NÃO"

Você tem direito a receber uma bolsa de estudos se você:

NÃO

- Possuir renda maior que \$ 3000,00



**O operador
"NÃO" inverte a
condição**

Exemplos de expressões lógicas

(suponha x igual a 5)

$$!(X == 10)$$

F

Resultado: V

$$!(X >= 2)$$

V

Resultado: F

Exemplos de expressões lógicas
(suponha x igual a 5)

$$!(X \leq 20 \ \&\& \ X == 10)$$

The expression is evaluated as follows:
1. Evaluate the inner condition $X \leq 20$. Since $x = 5$, this is true (V).
2. Evaluate the second condition $X == 10$. Since $x = 5$, this is false (F).
3. Evaluate the conjunction ($\&\&$) of the two conditions. Since one is false, the result is false (F).
4. Finally, evaluate the negation (!) of the conjunction result. Since it is false, the final result is true (V).

Resultado: V

Tabela verdade do operador "NÃO"

A	! A
F	V
V	F

Estrutura condicional

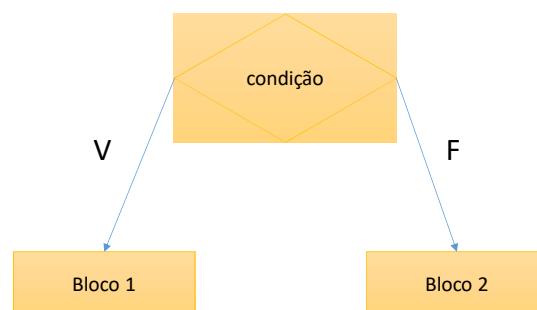
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Conceito

Estrutura condicional:

É uma **estrutura de controle** que permite definir que um certo **bloco de comandos** somente será executado dependendo de uma **condição**



Sintaxe da estrutura condicional

Simples:

```
if ( <condição> ) {  
    <comando 1>  
    <comando 2>  
}
```

REGRA:

V: executa o bloco de comandos
F: pula o bloco de comandos

*Importante:
Repare na endentação!*

Sintaxe da estrutura condicional

Composta:

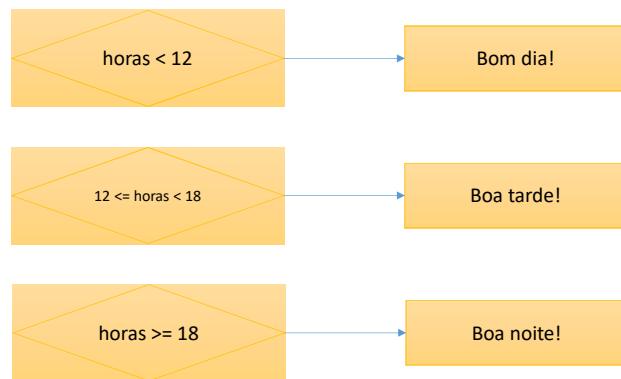
```
if ( <condição> ) {  
    <comando 1>  
    <comando 2>  
}  
else {  
    <comando 3>  
    <comando 4>  
}
```

REGRA:

V: executa somente o bloco do **if**
F: executa somente o bloco do **else**

*Importante:
Repare na endentação!*

E se eu tiver mais de duas possibilidades?



Encadeamento de estruturas condicionais

```
if ( condição 1 ) {  
    comando 1  
    comando 2  
}  
else {  
    if ( condição 2 ) {  
        comando 3  
        comando 4  
    }  
    else {  
        comando 5  
        comando 6  
    }  
}
```

*Importante:
Repare na endentação!*

Encadeamento de estruturas condicionais

```
if ( condição 1 ) {  
    comando 1  
    comando 2  
}  
else if ( condição 2 ) {  
    comando 3  
    comando 4  
}  
else if ( condição 3 ) {  
    comando 5  
    comando 6  
}  
else {  
    comando 7  
    comando 8  
}
```

*Importante:
Repare na endentação!*

Sintaxe opcional: operadores de atribuição cumulativa

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Problema exemplo

Uma operadora de telefonia cobra R\$ 50.00 por um plano básico que dá direito a 100 minutos de telefone. Cada minuto que exceder a franquia de 100 minutos custa R\$ 2.00. Fazer um programa para ler a quantidade de minutos que uma pessoa consumiu, daí mostrar o valor a ser pago.

Entrada	Saída
22	Valor a pagar: R\$ 50.00

Entrada	Saída
103	Valor a pagar: R\$ 56.00

Operadores de atribuição cumulativa

$a += b;$	$a = a + b;$
$a -= b;$	$a = a - b;$
$a *= b;$	$a = a * b;$
$a /= b;$	$a = a / b;$
$a \%= b;$	$a = a \% b;$

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

public class Main {

    public static void main(String[] args) {

        Locale.setDefault(Locale.US);
        Scanner sc = new Scanner(System.in);

        int minutos = sc.nextInt();

        double conta = 50.0;
        if (minutos > 100) {
            conta += (minutos - 100) * 2.0;
        }

        System.out.printf("Valor da conta = R$ %.2f%n", conta);

        sc.close();
    }
}
```

Sintaxe opcional: estrutura switch-case

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Estrutura switch-case

Quando se tem várias opções de fluxo a serem tratadas com base no valor de uma variável, ao invés de várias estruturas if-else encadeadas, alguns preferem utilizar a estrutura switch-case.

Problema exemplo

Fazer um programa para ler um valor inteiro de 1 a 7 representando um dia da semana (sendo 1=domingo, 2=segunda, e assim por diante). Escrever na tela o dia da semana correspondente, conforme exemplos.

Entrada	Saída
1	Dia da semana: domingo
4	Dia da semana: quarta
9	Dia da semana: valor inválido

```

import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int x = sc.nextInt();
        String dia;

        if (x == 1) {
            dia = "domingo";
        } else if (x == 2) {
            dia = "segunda";
        } else if (x == 3) {
            dia = "terca";
        } else if (x == 4) {
            dia = "quarta";
        } else if (x == 5) {
            dia = "quinta";
        } else if (x == 6) {
            dia = "sexta";
        } else if (x == 7) {
            dia = "sabado";
        } else {
            dia = "valor invalido";
        }
        System.out.println("Dia da semana: " + dia);
        sc.close();
    }
}

import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int x = sc.nextInt();
        String dia;

        switch (x) {
            case 1:
                dia = "domingo";
                break;
            case 2:
                dia = "segunda";
                break;
            case 3:
                dia = "terca";
                break;
            case 4:
                dia = "quarta";
                break;
            case 5:
                dia = "quinta";
                break;
            case 6:
                dia = "sexta";
                break;
            case 7:
                dia = "sabado";
                break;
            default:
                dia = "valor invalido";
                break;
        }
        System.out.println("Dia da semana: " + dia);
        sc.close();
    }
}

```

Sintaxe do switch-case

```

switch ( expressão ) {
    case valor1:
        comando1
        comando2
        break;
    case valor2:
        comando3
        comando4
        break;

    default:
        comando5
        comando6
        break;
}

```

Expressão condicional ternária

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Expressão condicional ternária

Estrutura opcional ao if-else quando se deseja decidir um **VALOR** com base em uma condição.

Sintaxe:

```
( condição ) ? valor_se_verdadeiro : valor_se_falso
```

Exemplos:

(2 > 4) ? 50 : 80  80

(10 != 3) ? "Maria" : "Alex"  "Maria"

Demo

```
double preco = 34.5;
double desconto;
if (preco < 20.0) {
    desconto = preco * 0.1;
}
else {
    desconto = preco * 0.05;
}
```

```
double preco = 34.5;
double desconto = (preco < 20.0) ? preco * 0.1 : preco * 0.05;
```

Escopo e inicialização

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Checklist

- Escopo de uma variável: é a região do programa onde a variável é válida, ou seja, onde ela pode ser referenciada.
- Uma variável não pode ser usada se não for iniciada.
- Falaremos de escopo de métodos no Capítulo 5

Demo

```
double price = sc.nextDouble();

if (price > 100.0) {
    double discount = price * 0.1;
}

System.out.println(discount);
```

Curso Java Completo

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Capítulo: Apresentação da linguagem Java e tópicos básicos

Exercícios sobre Estrutura Condicional (if-else)

* Exercícios obtidos do URI Online Judge: www.urionlinejudge.com.br

Atenção! Esses exercícios são:

- **OPCIONAIS** para quem já sabe Lógica de Programação em qualquer linguagem
- **NECESSÁRIOS** para alunos **iniciantes**

Exercícios resolvidos:

<https://www.youtube.com/watch?v=SRyQZBaA-s>

<https://www.youtube.com/watch?v=3lhkB5l8P6E>

<https://www.youtube.com/watch?v=UjCVlcKccdQ>

Exercícios propostos:

Exercício 01

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/cond01.java>

Fazer um programa para ler um número inteiro, e depois dizer se este número é negativo ou não.

Exemplos:

Entrada:	Saída:
-10	NEGATIVO
8	NAO NEGATIVO
0	NAO NEGATIVO

Exercício 02

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/cond02.java>

Fazer um programa para ler um número inteiro e dizer se este número é par ou ímpar.

Exemplos:

Entrada:	Saída:
12	PAR

Entrada:	Saída:
-27	IMPAR

Entrada:	Saída:
0	PAR

Exercício 03

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1044.java>

Leia 2 valores inteiros (A e B). Após, o programa deve mostrar uma mensagem "Sao Multiplos" ou "Nao sao Multiplos", indicando se os valores lidos são múltiplos entre si. Atenção: os números devem poder ser digitados em ordem crescente ou decrescente.

Exemplos:

Entrada:	Saída:
6 24	Sao Multiplos

Entrada:	Saída:
6 25	Nao sao Multiplos

Entrada:	Saída:
24 6	Sao Multiplos

Exercício 04

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1046.java>

Leia a hora inicial e a hora final de um jogo. A seguir calcule a duração do jogo, sabendo que o mesmo pode começar em um dia e terminar em outro, tendo uma duração mínima de 1 hora e máxima de 24 horas.

Exemplos:

Entrada:	Saída:
16 2	O JOGO DUROU 10 HORA(S)
0 0	O JOGO DUROU 24 HORA(S)
2 16	O JOGO DUROU 14 HORA(S)

Exercício 05

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1038.java>

Com base na tabela abaixo, escreva um programa que leia o código de um item e a quantidade deste item. A seguir, calcule e mostre o valor da conta a pagar.

CÓDIGO	ESPECIFICAÇÃO	PREÇO
1	Cachorro Quente	R\$ 4.00
2	X-Salada	R\$ 4.50
3	X-Bacon	R\$ 5.00
4	Torrada simples	R\$ 2.00
5	Refrigerante	R\$ 1.50

Exemplos:

Entrada:	Saída:
3 2	Total: R\$ 10.00
2 3	Total: R\$ 13.50

Exercício 06

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1037.java>

Você deve fazer um programa que leia um valor qualquer e apresente uma mensagem dizendo em qual dos seguintes intervalos ($[0,25]$, $(25,50]$, $(50,75]$, $(75,100]$) este valor se encontra. Obviamente se o valor não estiver em nenhum destes intervalos, deverá ser impressa a mensagem “Fora de intervalo”.

Exemplos:

Entrada:	Saída:
25.01	Intervalo $(25,50]$
25.00	Intervalo $[0,25]$
100.00	Intervalo $(75,100]$
-25.02	Fora de intervalo

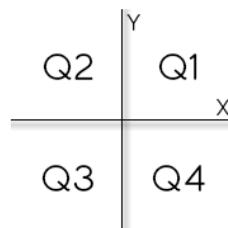
Exercício 07

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1041.java>

Leia 2 valores com uma casa decimal (x e y), que devem representar as coordenadas de um ponto em um plano. A seguir, determine qual o quadrante ao qual pertence o ponto, ou se está sobre um dos eixos cartesianos ou na origem ($x = y = 0$).

Se o ponto estiver na origem, escreva a mensagem “Origem”.

Se o ponto estiver sobre um dos eixos escreva “Eixo X” ou “Eixo Y”, conforme for a situação.



Exemplos:

Entrada:	Saída:
4.5 -2.2	Q4
0.1 0.1	Q1
0.0 0.0	Origem

Exercício 08

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1051.java>

Em um país imaginário denominado Lisarb, todos os habitantes ficam felizes em pagar seus impostos, pois sabem que nele não existem políticos corruptos e os recursos arrecadados são utilizados em benefício da população, sem qualquer desvio. A moeda deste país é o Rombus, cujo símbolo é o R\$.

Leia um valor com duas casas decimais, equivalente ao salário de uma pessoa de Lisarb. Em seguida, calcule e mostre o valor que esta pessoa deve pagar de Imposto de Renda, segundo a tabela abaixo.

Renda	Imposto de Renda
de 0.00 a R\$ 2000.00	Isento
de R\$ 2000.01 até R\$ 3000.00	8 %
de R\$ 3000.01 até R\$ 4500.00	18 %
acima de R\$ 4500.00	28 %

Lembre que, se o salário for R\$ 3002.00, a taxa que incide é de 8% apenas sobre R\$ 1000.00, pois a faixa de salário que fica de R\$ 0.00 até R\$ 2000.00 é isenta de Imposto de Renda. No exemplo fornecido (abaixo), a taxa é de 8% sobre R\$ 1000.00 + 18% sobre R\$ 2.00, o que resulta em R\$ 80.36 no total. O valor deve ser impresso com duas casas decimais.

Exemplos:

Entrada:	Saída:
3002.00	R\$ 80.36

Entrada:	Saída:
1701.12	Isento

Entrada:	Saída:
4520.00	R\$ 355.60

Curso Java COMPLETO

Capítulo: Estruturas repetitivas

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Como utilizar o Debug no Eclipse (execução passo a passo)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Como executar o debug do Eclipse

- Para marcar uma linha de breakpoint:
 - Run -> Toggle Breakpoint
- Para iniciar o debug:
 - Botão direito na classe -> Debug as -> Java Application
- Para executar uma linha:
 - F6
- Para interromper o debug:



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

public class Main {

    public static void main(String[] args) {
        Locale.setDefault(Locale.US);
        Scanner sc = new Scanner(System.in);

        double largura = sc.nextDouble();
        double comprimento = sc.nextDouble();
        double metroQuadrado = sc.nextDouble();

        double area = largura * comprimento;
        double preco = area * metroQuadrado;

        System.out.printf("AREA = %.2f%n", area);
        System.out.printf("PRECO = %.2f%n", preco);

        sc.close();
    }
}
```

Estrutura repetitiva "enquanto" (while)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Estrutura repetitiva "enquanto"

É uma **estrutura de controle** que **repete** um bloco de comandos **enquanto** uma **condição** for verdadeira.

Quando usar: quando **não** se sabe previamente a quantidade de repetições que será realizada.

Problema exemplo:

Fazer um programa que lê números inteiros até que um zero seja lido. Ao final mostra a soma dos números lidos.

Entrada	Saída
5	
2	
4	
0	

Sintaxe / regra

```
while ( condição ) {  
    comando 1  
    comando 2  
}
```

Regra:

V: executa e volta
F: pula fora

Resumo da aula

- Estrutura repetitiva "enquanto"
- Recomendada quando não se sabe previamente a quantidade de repetições
- Regra:
 - V: executa e volta
 - F: pula fora

Exercício de testes de mesa com while

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

```
x = 5;  
y = 0;  
while (x > 2) {  
    System.out.print(x);  
    y = y + x;  
    x = x - 1;  
}
```

x **y** **i**

Tela:

```
x = 2;  
y = 0;  
while (x < 60) {  
    System.out.println(x);  
    x = x * 2;  
    y = y + 10;  
}
```

x **y** **i**

Tela:

```
x = 100;  
y = 100;  
while (x != y) {  
    System.out.print("olha");  
    x = Math.sqrt(y);  
}
```

x **y** **i**

Tela:

```

x = 0;
while (x < 5) {
    y = x * 3;
    System.out.print(y);
    x = x + 1;
}
System.out.println("Fim");

```

x **y** **i**

Tela:



```

x = 2;
y = 10;
System.out.println("Olá");
while (x < y) {
    System.out.println(x + "-" + y);
    x = x * 2;
    y = y + 1;
}

```

x **y** **i**

Tela:



```

x = 4;
y = 0;
i = 0;
while (i < x) {
    i = i + 1;
    y = y + i;
    System.out.print(i);
    System.out.println(y);
}

```

x **y** **i**

Tela:



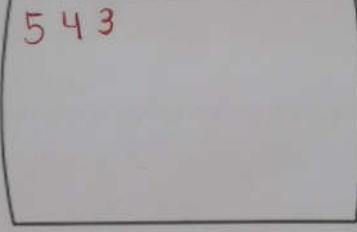
```

x = 5;
y = 0;
while (x > 2) {
    System.out.print(x);
    y = y + x;
    x = x - 1;
}

```

x **y** **i**

Tela:



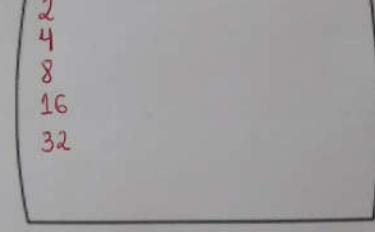
```

x = 2;
y = 0;
while (x < 60) {
    System.out.println(x);
    x = x * 2;
    y = y + 10;
}

```

x **y** **i**

Tela:



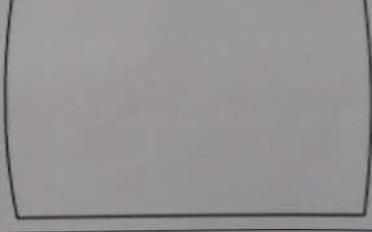
```

x = 100;
y = 100;
while (x != y) {
    System.out.print("olha");
    x = Math.sqrt(y);
}

```

x **y** **i**

Tela:



```

x = 0;
while (x < 5) {
    y = x * 3;
    System.out.print(y);
    x = x + 1;
}
System.out.println("Fim");

```

0x2	0x3 6	
8x5	8x12	

x y i

Tela:

036912 Fim

```

x = 2;
y = 10;
System.out.println("Olá");
while (x < y) {
    System.out.println(x + "-" + y);
    x = x * 2;
    y = y + 1;
}

```

2x8	10-11-12	
16	13	

x y i

Tela:

Olá
2-10
4-11
8-12

```

x = 4;
y = 0;
i = 0;
while (i < x) {
    i = i + 1;
    y = y + i;
    System.out.print(i);
    System.out.println(y);
}

```

4	0x8	0x2
	8x10	8x4

x y i

Tela:

11
23
36
410

Estrutura repetitiva "para" (for)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Estrutura repetitiva "para"

É uma **estrutura de controle** que **repete** um bloco de comandos **para** um certo **intervalo de valores**.

Quando usar: quando se sabe previamente a quantidade de repetições, ou o intervalo de valores.

Por exemplo:

Fazer um programa que lê um valor inteiro N e depois N números inteiros. Ao final, mostra a soma dos N números lidos

Entrada	Saída
3	
5	
2	
4	11

Sintaxe / regra

Executa somente na primeira vez

V: executa e volta
F: pula fora

Executa toda vez depois de voltar

```
for ( início ; condição ; incremento ) {  
    comando 1  
    comando 2  
}
```

Importante

Perceba que a estrutura "para" é ótima para se fazer uma repetição baseada em uma CONTAGEM:

Resultado na tela:

```
for (int i=0; i<5; i++) {  
    System.out.println("Valor de i: " + i);  
}
```

```
Valor de i: 0  
Valor de i: 1  
Valor de i: 2  
Valor de i: 3  
Valor de i: 4
```

Contagem regressiva

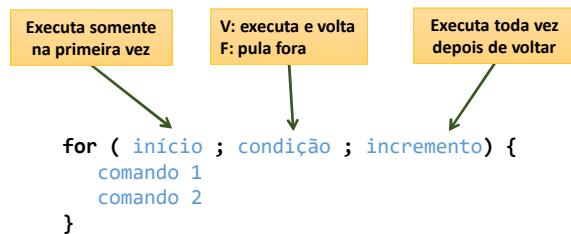
Resultado na tela:

```
for (int i=4; i>=0; i--) {  
    System.out.println("Valor de i: " + i);  
}
```

```
Valor de i: 4  
Valor de i: 3  
Valor de i: 2  
Valor de i: 1  
Valor de i: 0
```

Resumo da aula

- Estrutura repetitiva "para"
- Usar quando se sabe previamente a quantidade de repetições
- Ótimo para fazer contagens (progressiva ou regressiva)
- Regra:



Exercício de testes de mesa com for

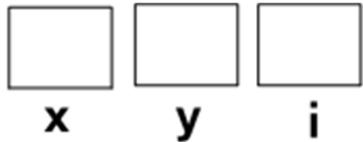
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

```

x = 4;
y = x + 2;
for (i=0; i<x; i++) {
    System.out.print(x+" "+y);
    y = y + i;
}

```

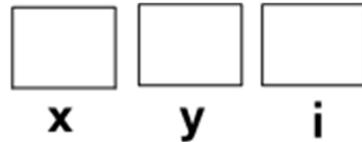


Tela:

```

for (i=1; i<5; i++){
    y = i - 1;
    x = i * 10;
    System.out.print(i);
}

```

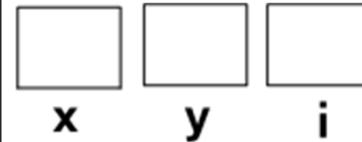


Tela:

```

y = 10;
for (i=0; i<4; i++){
    System.out.print(i);
    y = y + i;
    System.out.println(y);
}

```

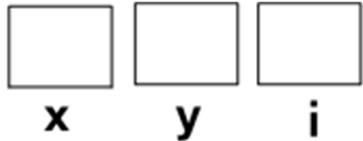


Tela:

```

x = 4;
y = 0;
for (i=0; i<x; i++) {
    System.out.print(i);
    System.out.println(x);
    y = y + 10;
}

```

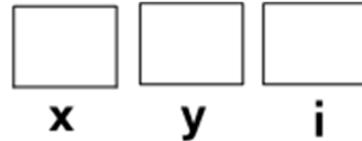


Tela:

```

x = 4;
y = 0;
for (i=0; i<x; i++) {
    y = y + i;
}
System.out.println(y);

```

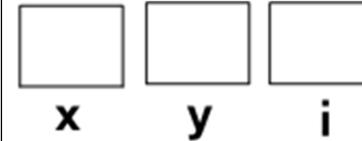


Tela:

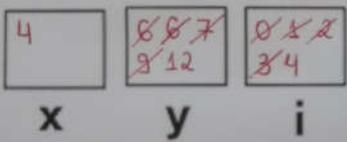
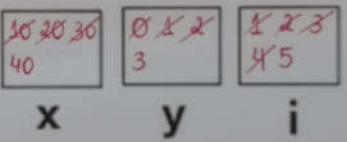
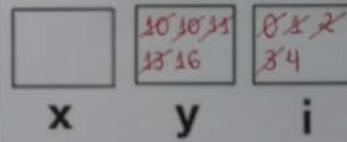
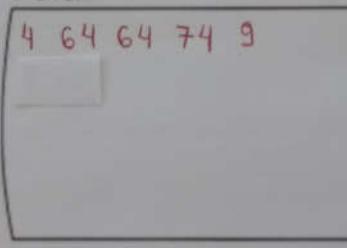
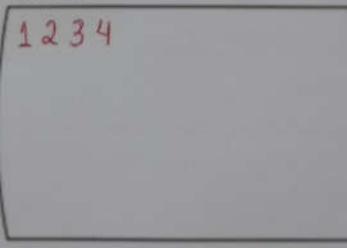
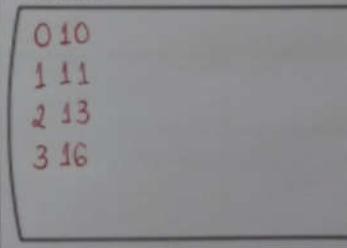
```

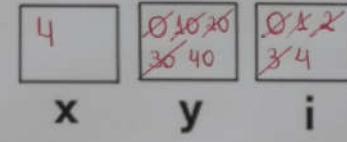
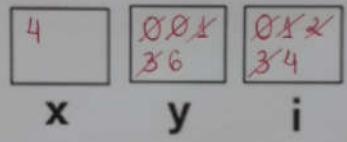
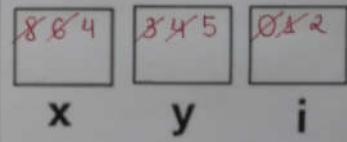
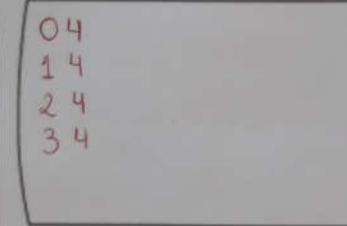
x = 8;
y = 3;
for (i=0; y<x; i++){
    x = x - 2;
    y = y + 1;
    System.out.println(i);
}

```



Tela:

<pre>x = 4; y = x + 2; for (i=0; i<x; i++) { System.out.print(x+" "+y); y = y + i; }</pre>	<pre>for (i=1; i<5; i++){ y = i - 1; x = i * 10; System.out.print(i); }</pre>	<pre>y = 10; for (i=0; i<4; i++){ System.out.print(i); y = y + i; System.out.println(y); }</pre>
 <p>x y i</p>	 <p>x y i</p>	 <p>x y i</p>
<p>Tela:</p> 	<p>Tela:</p> 	<p>Tela:</p> 

<pre>x = 4; y = 0; for (i=0; i<x; i++) { System.out.print(i); System.out.println(x); y = y + 10; }</pre>	<pre>x = 4; y = 0; for (i=0; i<x; i++) { y = y + i; } System.out.println(y);</pre>	<pre>x = 8; y = 3; for (i=0; y<x; i++){ x = x - 2; y = y + 1; System.out.println(i); }</pre>
 <p>x y i</p>	 <p>x y i</p>	 <p>x y i</p>
<p>Tela:</p> 	<p>Tela:</p> 	<p>Tela:</p> 

Estrutura repetitiva "faça-enquanto"

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Estrutura repetitiva "faça-enquanto"

Menos utilizada, mas em alguns casos se encaixa melhor ao problema.

O bloco de comandos executa pelo menos uma vez, pois a condição é verificada no final.

Sintaxe / regra

```
do {  
    comando 1  
    comando 2  
} while ( condição );
```

Regra:

V: volta

F: pula fora

Problema exemplo:

Fazer um programa para ler uma temperatura em Celsius e mostrar o equivalente em Fahrenheit. Perguntar se o usuário deseja repetir (s/n). Caso o usuário digite "s", repetir o programa.

$$\text{Fórmula: } F = \frac{9C}{5} + 32$$

Exemplo:

```
Digite a temperatura em Celsius: 30.0  
Equivalente em Fahrenheit: 86.0  
Deseja repetir (s/n)? s  
Digite a temperatura em Celsius: 21.0  
Equivalente em Fahrenheit: 69.8  
Deseja repetir (s/n)? s  
Digite a temperatura em Celsius: -10.5  
Equivalente em Fahrenheit: 13.1  
Deseja repetir (s/n)? n
```

Resumo da aula

- Estrutura repetitiva "faça-enquanto"
- O bloco de comandos executa pelo menos uma vez, pois a condição é verificada no final.
- Regra:
 - V: volta
 - F: pula fora

```
do {  
    comando 1  
    comando 2  
} while ( condição );
```

```
import java.util.Locale;  
import java.util.Scanner;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Locale.setDefault(Locale.US);  
        Scanner sc = new Scanner(System.in);  
  
        char resp;  
        do {  
            System.out.print("Digite a temperatura em Celsius: ");  
            double C = sc.nextDouble();  
            double F = 9.0 * C / 5.0 + 32.0;  
            System.out.printf("Equivalente em Fahrenheit: %.1f%n", F);  
            System.out.print("Deseja repetir (s/n)? ");  
            resp = sc.next().charAt(0);  
        } while (resp != 'n');  
  
        sc.close();  
    }  
}
```

Curso Java Completo

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Capítulo: Apresentação da linguagem Java e tópicos básicos

Exercícios sobre estrutura repetitiva while

* Exercícios obtidos do URI Online Judge: www.urionlinejudge.com.br

Atenção! Esses exercícios são:

- **OPCIONAIS** para quem já sabe Lógica de Programação em qualquer linguagem
- **NECESSÁRIOS** para alunos **iniciantes**

Exercícios resolvidos:

<https://www.youtube.com/watch?v=r3qCFqaNHds>

<https://www.youtube.com/watch?v=vT0QEDHK2yU>

Exercícios propostos:

Exercício 01

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1114.java>

Escreva um programa que repita a leitura de uma senha até que ela seja válida. Para cada leitura de senha incorreta informada, escrever a mensagem "Senha Invalida". Quando a senha for informada corretamente deve ser impressa a mensagem "Acesso Permitido" e o algoritmo encerrado. Considere que a senha correta é o valor 2002.

Exemplo:

Entrada:	Saída:
2200	Senha Invalida
1020	Senha Invalida
2022	Senha Invalida
2002	Acesso Permitido

Exercício 02

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1115.java>

Escreva um programa para ler as coordenadas (X,Y) de uma quantidade indeterminada de pontos no sistema cartesiano. Para cada ponto escrever o quadrante a que ele pertence. O algoritmo será encerrado quando pelo menos uma de duas coordenadas for NULA (nesta situação sem escrever mensagem alguma).

Exemplo:

Entrada:	Saída:
2 2 3 -2 -8 -1 -7 1 0 2	primeiro quarto terceiro segundo

Exercício 03

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1134.java>

Um Posto de combustíveis deseja determinar qual de seus produtos tem a preferência de seus clientes. Escreva um algoritmo para ler o tipo de combustível abastecido (codificado da seguinte forma: 1.Álcool 2.Gasolina 3.Diesel 4.Fim). Caso o usuário informe um código inválido (fora da faixa de 1 a 4) deve ser solicitado um novo código (até que seja válido). O programa será encerrado quando o código informado for o número 4. Deve ser escrito a mensagem: "MUITO OBRIGADO" e a quantidade de clientes que abasteceram cada tipo de combustível, conforme exemplo.

Exemplo:

Entrada:	Saída:
8 1 7 2 2 4	MUITO OBRIGADO Alcool: 1 Gasolina: 2 Diesel: 0

Curso Java Completo

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Capítulo: Apresentação da linguagem Java e tópicos básicos

Exercícios sobre estrutura repetitiva for

* Exercícios obtidos do URI Online Judge: www.urionlinejudge.com.br

Atenção! Esses exercícios são:

- **OPCIONAIS** para quem já sabe Lógica de Programação em qualquer linguagem
- **NECESSÁRIOS** para alunos **iniciantes**

Exercícios resolvidos:

<https://www.youtube.com/watch?v=JTa8WEhV38E>
<https://www.youtube.com/watch?v=RVJnkOyc7Kk>

Exercícios propostos:

Exercício 01

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1067.java>

Leia um valor inteiro X ($1 \leq X \leq 1000$). Em seguida mostre os ímpares de 1 até X, um valor por linha, inclusive o X, se for o caso.

Exemplo:

Entrada:	Saída:
8	1 3 5 7

Exercício 02

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1072.java>

Leia um valor inteiro N. Este valor será a quantidade de valores inteiros X que serão lidos em seguida. Mostre quantos destes valores X estão dentro do intervalo [10,20] e quantos estão fora do intervalo, mostrando essas informações conforme exemplo (use a palavra "in" para dentro do intervalo, e "out" para fora do intervalo).

Exemplo:

Entrada:	Saída:
5 14 123 10 -25 32	2 in 3 out

Exercício 03

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1079.java>

Leia 1 valor inteiro N, que representa o número de casos de teste que vem a seguir. Cada caso de teste consiste de 3 valores reais, cada um deles com uma casa decimal. Apresente a média ponderada para cada um destes conjuntos de 3 valores, sendo que o primeiro valor tem peso 2, o segundo valor tem peso 3 e o terceiro valor tem peso 5.

Exemplo:

Entrada:	Saída:
3 6.5 4.3 6.2 5.1 4.2 8.1 8.0 9.0 10.0	5.7 6.3 9.3

Exercício 04

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1116.java>

Fazer um programa para ler um número N. Depois leia N pares de números e mostre a divisão do primeiro pelo segundo. Se o denominador for igual a zero, mostrar a mensagem "divisao impossivel".

Exemplo:

Entrada:	Saída:
3 3 -2 -8 0 0 8	-1.5 divisao impossivel 0.0

Exercício 05

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1153.java>

Ler um valor N. Calcular e escrever seu respectivo fatorial. Fatorial de N = N * (N-1) * (N-2) * (N-3) * ... * 1. Lembrando que, por definição, fatorial de 0 é 1.

Exemplos:

Entrada:	Saída:
4	24
1	1
5	120
0	1

Exercício 06

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1157.java>

Ler um número inteiro N e calcular todos os seus divisores.

Exemplo:

Entrada:	Saída:
6	1 2 3 6

Exercício 07

Correção: <https://github.com/acenelio/nivelamento-java/blob/master/src/uri1143.java>

Fazer um programa para ler um número inteiro positivo N. O programa deve então mostrar na tela N linhas, começando de 1 até N. Para cada linha, mostrar o número da linha, depois o quadrado e o cubo do valor, conforme exemplo.

Exemplo:

Entrada:	Saída:
5	1 1 1 2 4 8 3 9 27 4 16 64 5 25 125

Curso Java COMPLETO

Capítulo: Outros tópicos básicos sobre Java

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Restrições e convenções para nomes

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Restrições para nomes de variáveis

- Não pode começar com dígito: use uma letra ou _
- Não usar acentos ou til
- Não pode ter espaço em branco
- Sugestão: use nomes que tenham um significado

Errado:

```
int 5minutes;  
int salário;  
int salario do funcionario;
```

Correto:

```
int _5minutes;  
int salario;  
int salarioDoFuncionario;
```

Convenções

- Camel Case: lastName
 - pacotes
 - atributos
 - métodos
 - variáveis e parâmetros
- Pascal Case: ProductService
 - classes

```
package entities;

public class Account {

    private String holder;
    private Double balance;

    public Account(String holder, Double balance) {
        this.holder = holder;
        this.balance = balance;
    }

    public String getHolder() {
        return holder;
    }

    public void deposit(double amount) {
        balance += amount;
    }

    public void withdraw(double amount) {
        balance -= amount;
    }
}
```

Operadores bitwise

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Operadores bitwise

Operador	Significado
&	Operação "E" bit a bit
	Operação "OU" bit a bit
^	Operação "OU-exclusivo" bit a bit

C1	C2	C1 E C2
F	F	F
F	V	F
V	F	F
V	V	V

C1	C2	C1 OU C2
F	F	F
F	V	V
V	F	V
V	V	V

C1	C2	C1 XOR C2
F	F	F
F	V	V
V	F	V
V	V	F

Demo

(89) 0101 1001 }
 (60) 0011 1100 }
 }
 &: 0001 1000 (24)
 |: 0111 1101 (125)
 ^: 0110 0101 (101)

```
int n1 = 89;  
int n2 = 60;  
System.out.println(n1 & n2);  
System.out.println(n1 | n2);  
System.out.println(n1 ^ n2);
```

Aplicação comum: verificar bit

(89) 01~~01~~ 1001 } &: 0000 0000 (0)
(32) 0010 0000

(113) 01~~11~~ 0001 } &: 0010 0000 (32)
(32) 0010 0000

```
package course;  
  
import java.util.Scanner;  
  
public class Program {  
  
    public static void main(String[] args) {  
  
        Scanner sc = new Scanner(System.in);  
  
        int mask = 0b100000;  
        int n = sc.nextInt();  
  
        if ((n & mask) != 0) {  
            System.out.println("6th bit is true!");  
        }  
        else {  
            System.out.println("6th bit is false");  
        }  
  
        sc.close();  
    }  
}
```

Funções interessantes para String

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Checklist

- Formatar: toLowerCase(), toUpperCase(), trim()
- Recortar: substring(inicio), substring(inicio, fim)
- Substituir: Replace(char, char), Replace(string, string)
- Buscar: IndexOf, LastIndexOf
- str.Split(" ")

```

String original = "abcde FGHIJ ABC abc DEFG  ";

String s01 = original.toLowerCase();
String s02 = original.toUpperCase();
String s03 = original.trim();
String s04 = original.substring(2);
String s05 = original.substring(2, 9);
String s06 = original.replace('a', 'x');
String s07 = original.replace("abc", "xy");
int i = original.indexOf("bc");
int j = original.lastIndexOf("bc");

System.out.println("Original: -" + original + "-");
System.out.println("toLowerCase: -" + s01 + "-");
System.out.println("toUpperCase: -" + s02 + "-");
System.out.println("trim: -" + s03 + "-");
System.out.println("substring(2): -" + s04 + "-");
System.out.println("substring(2, 9): -" + s05 + "-");
System.out.println("replace('a', 'x'): -" + s06 + "-");
System.out.println("replace('abc', 'xy'): -" + s07 + "-");
System.out.println("Index of 'bc': " + i);
System.out.println("Last index of 'bc': " + j);

```

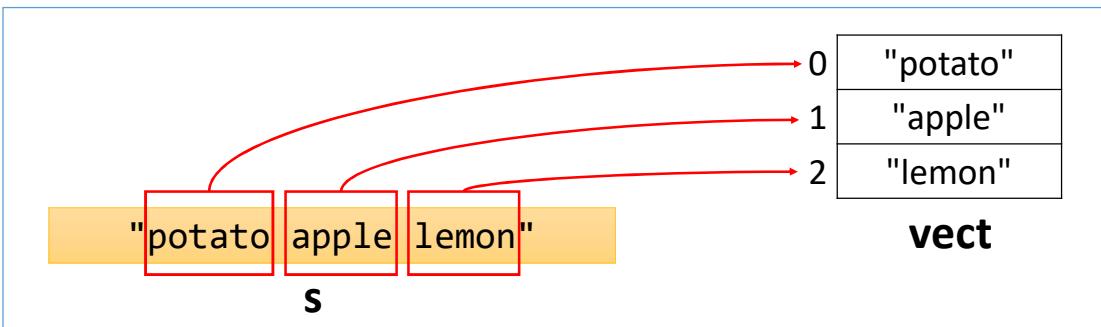
Operação split

```

String s = "potato apple lemon";

String[] vect = s.split(" ");
String word1 = vect[0];
String word2 = vect[1];
String word3 = vect[2];

```



Comentários em Java (básico)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

```
package course;

import java.util.Locale;
import java.util.Scanner;

/*
Este programa calcula as raízes de uma equação do segundo grau
Os valores dos coeficientes devem ser digitados um por linha
*/
public class Program {
    public static void main(String[] args) {

        Locale.setDefault(Locale.US);
        Scanner sc = new Scanner(System.in);

        double a, b, c, delta;

        System.out.println("Digite os valores dos coeficientes:");
        a = sc.nextDouble();
        b = sc.nextDouble();
        c = sc.nextDouble();

        delta = b * b - 4 * a * c; // cálculo do valor de delta
```

Funções (sintaxe)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Funções

- Representam um processamento que possui um significado
 - Math.sqrt(double)
 - System.out.println(string)
- Principais vantagens: modularização, delegação e reaproveitamento
- Dados de entrada e saída
 - Funções podem receber dados de entrada (parâmetros ou argumentos)
 - Funções podem ou não retornar uma saída
- Em orientação a objetos, funções em classes recebem o nome de "métodos"

Problema exemplo

Fazer um programa para ler três números inteiros e mostrar na tela o maior deles.

Exemplo:

```
Enter three numbers:  
5  
8  
3  
Higher = 8
```

```
package course;  
  
import java.util.Scanner;  
  
public class Program {  
    public static void main(String[] args) {  
  
        Scanner sc = new Scanner(System.in);  
  
        System.out.println("Enter three numbers:");  
        int a = sc.nextInt();  
        int b = sc.nextInt();  
        int c = sc.nextInt();  
  
        if (a > b && a > c) {  
            System.out.println("Higher = " + a);  
        } else if (b > c) {  
            System.out.println("Higher = " + b);  
        } else {  
            System.out.println("Higher = " + c);  
        }  
  
        sc.close();  
    }  
}
```

```
package course;

import java.util.Scanner;

public class Program {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter three numbers:");
        int a = sc.nextInt();
        int b = sc.nextInt();
        int c = sc.nextInt();

        int higher = max(a, b, c);

        showResult(higher);
        sc.close();
    }

    public static int max(int x, int y, int z) {
        int aux;
        if (x > y && x > z) {
            aux = x;
        } else if (y > z) {
            aux = y;
        } else {
            aux = z;
        }
        return aux;
    }

    public static void showResult(int value) {
        System.out.println("Higher = " + value);
    }
}
```

Curso Programação Orientada a Objetos com Java

Capítulo: Classes, atributos, métodos, membros estáticos

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Resolvendo um problema sem orientação a objetos

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Problema exemplo

Fazer um programa para ler as medidas dos lados de dois triângulos X e Y (suponha medidas válidas). Em seguida, mostrar o valor das áreas dos dois triângulos e dizer qual dos dois triângulos possui a maior área.

A fórmula para calcular a área de um triângulo a partir das medidas de seus lados a, b e c é a seguinte (fórmula de Heron):

$$\text{area} = \sqrt{p(p-a)(p-b)(p-c)} \quad \text{onde} \quad p = \frac{a+b+c}{2}$$

Exemplo:

```
Enter the measures of triangle X:
```

```
3.00
4.00
5.00
```

```
Enter the measures of triangle Y:
```

```
7.50
4.50
4.02
```

```
Triangle X area: 6.0000
```

```
Triangle Y area: 7.5638
```

```
Larger area: Y
```

```
package application;

import java.util.Locale;
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        Locale.setDefault(Locale.US);
        Scanner sc = new Scanner(System.in);

        double xA, xB, xC, yA, yB, yC;

        System.out.println("Enter the measures of triangle X: ");
        xA = sc.nextDouble();
        xB = sc.nextDouble();
        xC = sc.nextDouble();
        System.out.println("Enter the measures of triangle Y: ");
        yA = sc.nextDouble();
        yB = sc.nextDouble();
        yC = sc.nextDouble();

        double p = (xA + xB + xC) / 2.0;
        double areaX = Math.sqrt(p * (p - xA) * (p - xB) * (p - xC));

        p = (yA + yB + yC) / 2.0;
        double areaY = Math.sqrt(p * (p - yA) * (p - yB) * (p - yC));

        System.out.printf("Triangle X area: %.4f%n", areaX);
        System.out.printf("Triangle Y area: %.4f%n", areaY);

        if (areaX > areaY) {
            System.out.println("Larger area: X");
        } else {
            System.out.println("Larger area: Y");
        }
        sc.close();
    }
}
```

Criando uma classe com três atributos para representar melhor o triângulo

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Discussão

Triângulo é uma entidade com três atributos: a, b, c.

Estamos usando três variáveis distintas para representar cada triângulo:

double aX, bX, cX, aY, bY, cY;

Para melhorar isso, vamos usar uma CLASSE para representar um triângulo.

Memória:

3.00	4.00	5.00
aX	bX	cX
7.50	4.50	4.02
aY	bY	cY

Classe

- É um tipo estruturado que pode conter (membros):
 - Atributos (dados / campos)
 - Métodos (funções / operações)
- A classe também pode prover muitos outros recursos, tais como:
 - Construtores
 - Sobrecarga
 - Encapsulamento
 - Herança
 - Polimorfismo
- Exemplos:
 - Entidades: Produto, Cliente, Triangulo
 - Serviços: ProdutoService, ClienteService, EmailService, StorageService
 - Controladores: ProdutoController, ClienteController
 - Utilitários: Calculadora, Compactador
 - Outros (views, repositórios, gerenciadores, etc.)

```
package entities;
```

```
public class Triangle {  
  
    public double a;  
    public double b;  
    public double c;  
}
```

```
double aX, bX, cX, aY, bY, cY;
```

3.0	4.0	5.0
aX	bX	cX
7.5	4.5	4.02
aY	bY	cY



```
Triangle x, y;  
x = new Triangle();  
y = new Triangle();
```

x →	3.0	4.0	5.0
	a	b	c
y →	7.5	4.5	4.02
	a	b	c

```

package application;

import java.util.Locale;
import java.util.Scanner;

import entities.Triangle;

public class Program {

    public static void main(String[] args) {

        Locale.setDefault(Locale.US);
        Scanner sc = new Scanner(System.in);

        Triangle x, y;
        x = new Triangle();
        y = new Triangle();

        System.out.println("Enter the measures of triangle X: ");
        x.a = sc.nextDouble();
        x.b = sc.nextDouble();
        x.c = sc.nextDouble();
        System.out.println("Enter the measures of triangle Y: ");
        y.a = sc.nextDouble();
        y.b = sc.nextDouble();
        y.c = sc.nextDouble();

        double p = (x.a + x.b + x.c) / 2.0;
        double areaX = Math.sqrt(p * (p - x.a) * (p - x.b) * (p - x.c));

        p = (y.a + y.b + y.c) / 2.0;
        double areaY = Math.sqrt(p * (p - y.a) * (p - y.b) * (p - y.c));
        ...
    }
}

```

Instanciação

(alocação dinâmica de memória)

```

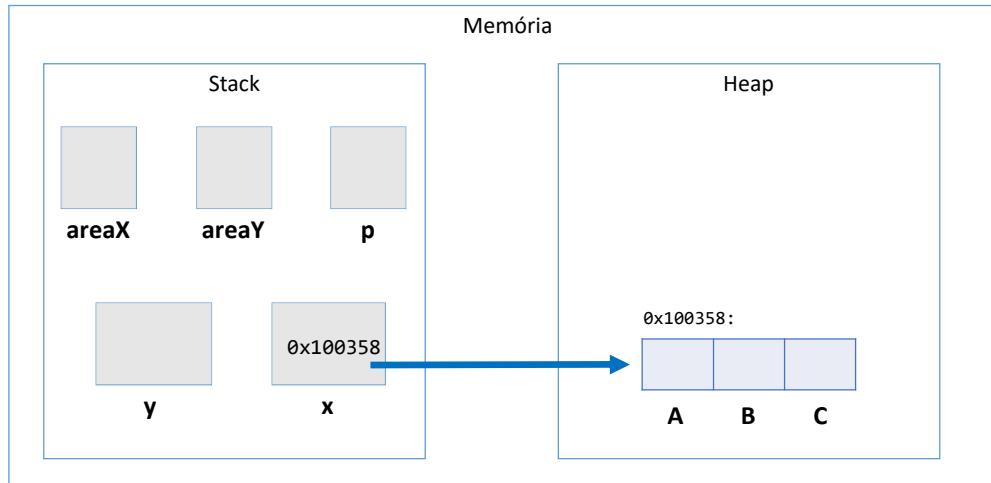
double areaX, areaY, p;
Triangle x, y;

```

```

x = new Triangle();

```



Classes, objetos, atributos

- Classe: é a definição do tipo

```
package course;  
  
public class Triangle {  
  
    public double a;  
    public double b;  
    public double c;  
}
```

- Objetos: são instâncias da classe



Criando um método para obtermos os benefícios de reutilização e delegação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

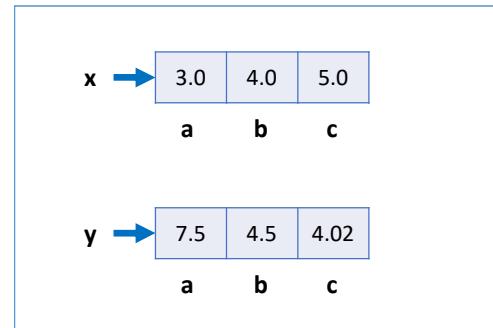
Discussão

Com o uso de CLASSE, agora nós temos uma variável composta do tipo "Triangle" para representar cada triângulo:

```
Triangle x, y;
x = new Triangle();
y = new Triangle();
```

Agora vamos melhorar nossa CLASSE, acrescentando nela um MÉTODO para calcular a área.

Memória:



```
package application;

import java.util.Locale;
import java.util.Scanner;

import entities.Triangle;

public class Program {

    public static void main(String[] args) {
        Locale.setDefault(Locale.US);
        Scanner sc = new Scanner(System.in);

        Triangle x, y;
        x = new Triangle();
        y = new Triangle();

        System.out.println("Enter the measures of triangle X: ");
        x.a = sc.nextDouble();
        x.b = sc.nextDouble();
        x.c = sc.nextDouble();
        System.out.println("Enter the measures of triangle Y: ");
        y.a = sc.nextDouble();
        y.b = sc.nextDouble();
        y.c = sc.nextDouble();

        double areaX = x.area();
        double areaY = y.area();
        (...)
```

```

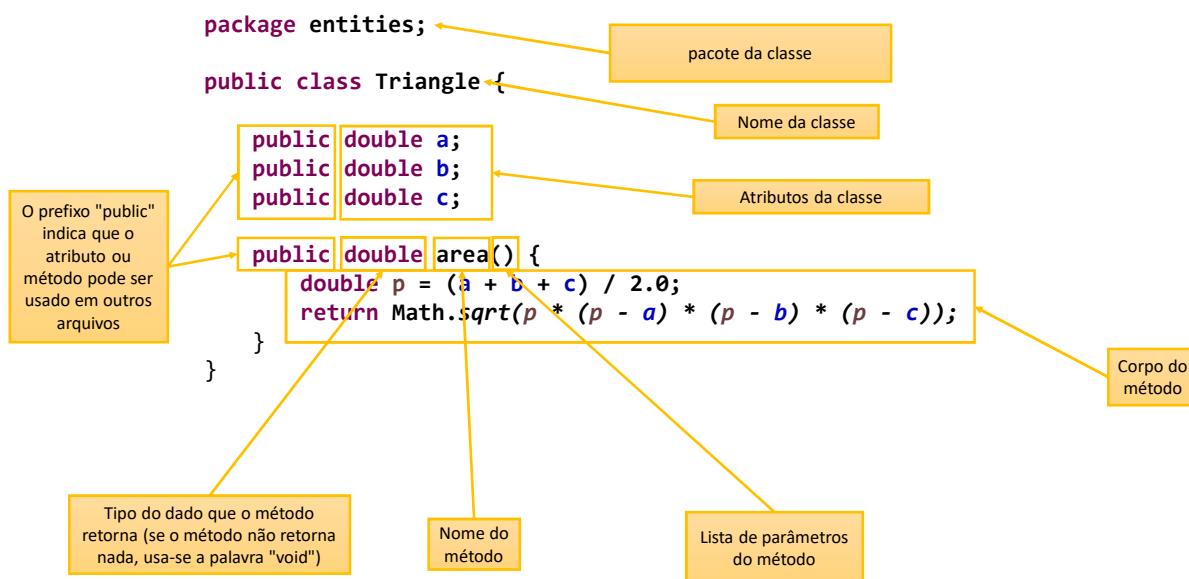
package entities;

public class Triangle {

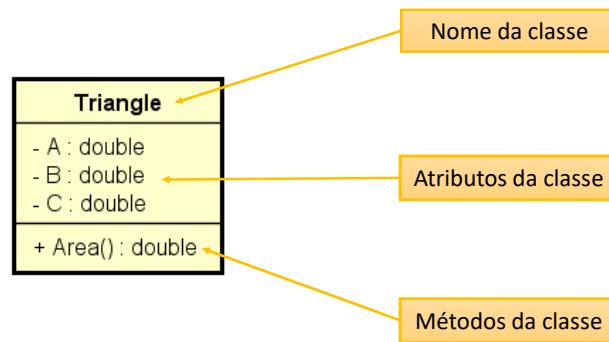
    public double a;
    public double b;
    public double c;

    public double area() {
        double p = (a + b + c) / 2.0;
        return Math.sqrt(p * (p - a) * (p - b) * (p - c));
    }
}

```



Projeto da classe (UML)



DISCUSSÃO

Quais são os benefícios de se calcular a área de um triângulo por meio de um MÉTODO dentro da CLASSE Triangle?

- 1) Reaproveitamento de código:** nós eliminamos o código repetido (cálculo das áreas dos triângulos x e y) no programa principal.
- 2) Delegação de responsabilidades:** quem deve ser responsável por saber como calcular a área de um triângulo é o próprio triângulo. A lógica do cálculo da área não deve estar em outro lugar.

Começando a resolver um segundo problema exemplo

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Outro exemplo

Fazer um programa para ler os dados de um produto em estoque (nome, preço e quantidade no estoque). Em seguida:

- Mostrar os dados do produto (nome, preço, quantidade no estoque, valor total no estoque)
- Realizar uma entrada no estoque e mostrar novamente os dados do produto
- Realizar uma saída no estoque e mostrar novamente os dados do produto

Para resolver este problema, você deve criar uma CLASSE conforme projeto ao lado:

(veja exemplo na próxima página)

Product
- Name : string
- Price : double
- Quantity : int
+ TotalValueInStock() : double
+ AddProducts(quantity : int) : void
+ RemoveProducts(quantity : int) : void

Example:

```

Enter product data:
Name: TV
Price: 900.00
Quantity in stock: 10

Product data: TV, $ 900.00, 10 units, Total: $ 9000.00

Enter the number of products to be added in stock: 5

Updated data: TV, $ 900.00, 15 units, Total: $ 13500.00

Enter the number of products to be removed from stock: 3

Updated data: TV, $ 900.00, 12 units, Total: $ 10800.00

```

Example:

```

Enter product data:
Name: TV
Price: 900.00
Quantity in stock: 10

Product data: TV, $ 900.00, 10 units, Total: $ 9000.00

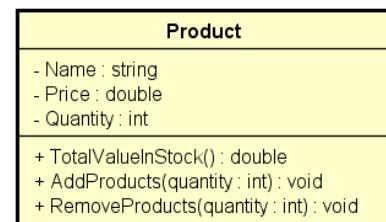
Enter the number of products to be added in stock: 5

Updated data: TV, $ 900.00, 15 units, Total: $ 13500.00

Enter the number of products to be removed from stock: 3

Updated data: TV, $ 900.00, 12 units, Total: $ 10800.00

```



Object e `toString`

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Discussão

- Toda classe em Java é uma subclasse da classe Object
- Object possui os seguintes métodos:
 - `getClass`- retorna o tipo do objeto
 - `equals` - compara se o objeto é igual a outro
 - `hashCode` - retorna um código hash do objeto
 - `toString` - converte o objeto para string

```
package entities;

public class Product {

    public String name;
    public double price;
    public int quantity;

    public double totalValueInStock() {
        return price * quantity;
    }

    public void addProducts(int quantity) {
        this.quantity += quantity;
    }

    public void removeProducts(int quantity) {
        this.quantity -= quantity;
    }

    public String toString() {
        return name
            + ", $ "
            + String.format("%.2f", price)
            + " "
            + quantity
            + " units, Total: $ "
            + String.format("%.2f", totalValueInStock());
    }
}
```

Finalizando o programa

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

```
package application;

import java.util.Locale;
import java.util.Scanner;

import entities.Product;

public class Program {
    public static void main(String[] args) {
        Locale.setDefault(Locale.US);
        Scanner sc = new Scanner(System.in);

        Product product = new Product();
        System.out.println("Enter product data: ");
        System.out.print("Name: ");
        product.name = sc.nextLine();
        System.out.print("Price: ");
        product.price = sc.nextDouble();
        System.out.print("Quantity in stock: ");
        product.quantity = sc.nextInt();

        System.out.println();
        System.out.println("Product data: " + product);

        System.out.println();
        System.out.print("Enter the number of products to be added in stock: ");
        int quantity = sc.nextInt();
        product.addProducts(quantity);

        System.out.println();
        System.out.print("Enter the number of products to be removed from stock: ");
        quantity = sc.nextInt();
        product.removeProducts(quantity);

        System.out.println();
        System.out.println("Updated data: " + product);

        sc.close();
    }
}
```

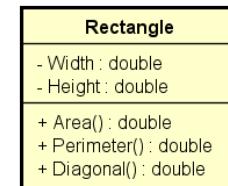
Exercícios de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Exercício 1

Fazer um programa para ler os valores da largura e altura de um retângulo. Em seguida, mostrar na tela o valor de sua área, perímetro e diagonal. Usar uma classe como mostrado no projeto ao lado.



Exemplo:

```
Enter rectangle width and height:  

3.00  

4.00  

AREA = 12.00  

PERIMETER = 14.00  

DIAGONAL = 5.00
```

Exercício 2

Fazer um programa para ler os dados de um funcionário (nome, salário bruto e imposto). Em seguida, mostrar os dados do funcionário (nome e salário líquido). Em seguida, aumentar o salário do funcionário com base em uma porcentagem dada (somente o salário bruto é afetado pela porcentagem) e mostrar novamente os dados do funcionário. Use a classe projetada abaixo.

Exemplo:

```
Name: Joao Silva  

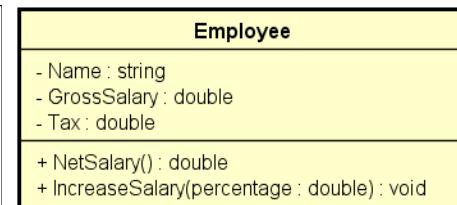
Gross salary: 6000.00  

Tax: 1000.00  

Employee: Joao Silva, $ 5000.00  

Which percentage to increase salary? 10.0  

Updated data: Joao Silva, $ 5600.00
```



Exercício 3

Fazer um programa para ler o nome de um aluno e as três notas que ele obteve nos três trimestres do ano (primeiro trimestre vale 30 e o segundo e terceiro valem 35 cada). Ao final, mostrar qual a nota final do aluno no ano. Dizer também se o aluno está aprovado (PASS) ou não (FAILED) e, em caso negativo, quantos pontos faltam para o aluno obter o mínimo para ser aprovado (que é 60% da nota). Você deve criar uma classe Student para resolver este problema.

Exemplos:

Entrada:	Saída:
Alex Green 27.00 31.00 32.00	FINAL GRADE = 90.00 PASS
Alex Green 17.00 20.00 15.00	FINAL GRADE = 52.00 FAILED MISSING 8.00 POINTS

Membros estáticos - PARTE 1

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

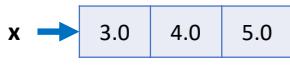
Product
- Name : string
- Price : double
- Quantity : int
+ TotalValueInStock() : double
+ AddProducts(quantity : int) : void
+ RemoveProducts(quantity : int) : void

membros
=
atributos e métodos

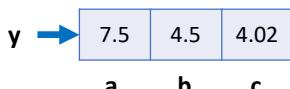
Membros estáticos

- Também chamados membros de classe
 - Em oposição a membros e instância
- São membros que fazem sentido independentemente de objetos. Não precisam de objeto para serem chamados. São chamados a partir do próprio nome da classe.
- Aplicações comuns:
 - Classes utilitárias  **Math.sqrt(double)**
 - Declaração de constantes
- Uma classe que possui somente membros estáticos, pode ser uma classe estática também. Esta classe não poderá ser instanciada.

```
Triangle x, y;
x = new Triangle();
y = new Triangle();
```



x.area() → 6.0



y.area() → 7.5638

Problema exemplo

Fazer um programa para ler um valor numérico qualquer, e daí mostrar quanto seria o valor de uma circunferência e do volume de uma esfera para um raio daquele valor. Informar também o valor de PI com duas casas decimais.

Exemplo:

```
Enter radius: 3.0
Circumference: 18.85
Volume: 113.10
PI value: 3.14
```

Checklist

- Versão 1: métodos na própria classe do programa
 - Nota: dentro de um método estático você não pode chamar membros de instância da mesma classe.
- Versão 2: classe Calculator com membros de instância
- Versão 3: classe Calculator com método estático

```

package application;

import java.util.Locale;
import java.util.Scanner;

public class Program {

    public static final double PI = 3.14159;

    public static void main(String[] args) {
        Locale.setDefault(Locale.US);
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter radius: ");
        double radius = sc.nextDouble();

        double c = circumference(radius);
        double v = volume(radius);

        System.out.printf("Circumference: %.2f%n", c);
        System.out.printf("Volume: %.2f%n", v);
        System.out.printf("PI value: %.2f%n", PI);

        sc.close();
    }

    public static double circumference(double radius) {
        return 2.0 * PI * radius;
    }

    public static double volume(double radius) {
        return 4.0 * PI * radius * radius * radius / 3.0;
    }
}

```

VERSÃO 1

```
package util;

public class Calculator {

    public final double PI = 3.14159;

    public double circumference(double radius) {
        return 2.0 * PI * radius;
    }

    public double volume(double radius) {
        return 4.0 * PI * radius * radius * radius / 3.0;
    }
}
```

VERSAO 2

```
Calculator calc = new Calculator();

System.out.print("Enter radius: ");
double radius = sc.nextDouble();

double c = calc.circumference(radius);

double v = calc.volume(radius);

System.out.printf("Circumference: %.2f%n", c);
System.out.printf("Volume: %.2f%n", v);
System.out.printf("PI value: %.2f%n", calc.PI);
```

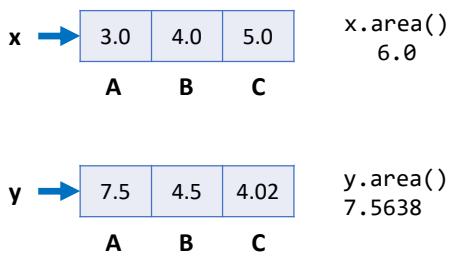
Membros estáticos - PARTE 2

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Discussão

- No problema dos triângulos, cada triângulo possui sua área.
- Area() é uma operação concernente ao objeto: cada triângulo possui sua área.
- Já no caso da calculadora, os valores dos cálculos não mudam para calculadoras diferentes, ou seja, são cálculos estáticos. O valor de Pi também é estático.



```
Calculator calc1 = new Calculator();
Calculator calc2 = new Calculator();

calc1.PI
3.14
calc1.circumference(3.0)
18.85

calc2.PI
3.14
calc2.circumference(3.0)
18.85
```

```
package util;

public class Calculator {

    public static final double PI = 3.14159;

    public static double circumference(double radius) {
        return 2.0 * PI * radius;
    }

    public static double volume(double radius) {
        return 4.0 * PI * radius * radius * radius / 3.0;
    }
}
```

VERSAO 3

```
System.out.print("Enter radius: ");
double radius = sc.nextDouble();

double c = Calculator.circumference(radius);
double v = Calculator.volume(radius);

System.out.printf("Circumference: %.2f%n", c);
System.out.printf("Volume: %.2f%n", v);
System.out.printf("PI value: %.2f%n", Calculator.PI);
```

Exercício de fixação

Faça um programa para ler a cotação do dólar, e depois um valor em dólares a ser comprado por uma pessoa em reais. Informar quantos reais a pessoa vai pagar pelos dólares, considerando ainda que a pessoa terá que pagar 6% de IOF sobre o valor em dólar. Criar uma classe **CurrencyConverter** para ser responsável pelos cálculos.

Exemplo:

What is the dollar price? **3.10**
 How many dollars will be bought? **200.00**
 Amount to be paid in reais = 657.20

Correção do exercício de fixação

```
package util;
public class CurrencyConverter {
    public static double IOF = 0.06;
    public static double dollarToReal(double amount, double dollarPrice) {
        return amount * dollarPrice * (1.0 + IOF);
    }
}
```

```
package application;
import java.util.Locale;
import java.util.Scanner;
import util.CurrencyConverter;
public class Program {
    public static void main(String[] args) {
        Locale.setDefault(Locale.US);
        Scanner sc = new Scanner(System.in);
        System.out.print("What is the dollar price? ");
        double dollarPrice = sc.nextDouble();
        System.out.print("How many dollars will be bought? ");
        double amount = sc.nextDouble();
        double result = CurrencyConverter.dollarToReal(amount, dollarPrice);
        System.out.printf("Amount to be paid in reais = %.2f\n", result);
        sc.close();
    }
}
```

Correção dos exercícios de fixação (assunto: classes, atributos e métodos)

Exercício 1:

Classe Rectangle.java (pacote entities):

```
package entities;

public class Rectangle {

    public double width;
    public double height;

    public double area() {
        return width * height;
    }

    public double perimeter() {
        return 2 * (width + height);
    }

    public double diagonal() {
        return Math.sqrt(width * width + height * height);
    }
}
```

Classe Program.java (pacote application):

```
package application;

import java.util.Locale;
import java.util.Scanner;
import entities.Rectangle;

public class Program {
    public static void main(String[] args) {

        Locale.setDefault(Locale.US);
        Scanner sc = new Scanner(System.in);

        Rectangle rect = new Rectangle();

        System.out.println("Enter rectangle width and height:");
        rect.width = sc.nextDouble();
        rect.height = sc.nextDouble();

        System.out.printf("AREA = %.2f%n", rect.area());
        System.out.printf("PERIMETER = %.2f%n", rect.perimeter());
        System.out.printf("DIAGONAL = %.2f%n", rect.diagonal());
        sc.close();
    }
}
```

Exercício 2:

Classe Employee.java (pacote entities):

```
package entities;

public class Employee {

    public String name;
    public double grossSalary;
    public double tax;

    public double netSalary() {
        return grossSalary - tax;
    }

    public void increaseSalary(double percentage) {
        grossSalary += grossSalary * percentage / 100.0;
    }

    public String toString() {
        return name + ", $" + String.format("%.2f", netSalary());
    }
}
```

Classe Program.java (pacote application):

```
package application;

import java.util.Locale;
import java.util.Scanner;
import entities.Employee;

public class Program {

    public static void main(String[] args) {

        Locale.setDefault(Locale.US);
        Scanner sc = new Scanner(System.in);

        Employee emp = new Employee();

        System.out.print("Name: ");
        emp.name = sc.nextLine();
        System.out.print("Gross salary: ");
        emp.grossSalary = sc.nextDouble();
        System.out.print("Tax: ");
        emp.tax = sc.nextDouble();

        System.out.println();
        System.out.println("Employee: " + emp);
        System.out.println();
        System.out.print("Which percentage to increase salary? ");
        double percentage = sc.nextDouble();
        emp.increaseSalary(percentage);

        System.out.println();
        System.out.println("Updated data: " + emp);
        sc.close();
    }
}
```

Exercício 3:

Classe Student.java (pacote entities):

```
package entities;

public class Student {

    public String name;
    public double grade1;
    public double grade2;
    public double grade3;

    public double finalGrade() {
        return grade1 + grade2 + grade3;
    }

    public double missingPoints() {
        if (finalGrade() < 60.0) {
            return 60.0 - finalGrade();
        }
        else {
            return 0.0;
        }
    }
}
```

Classe Program.java (pacote application):

```
package application;

import java.util.Locale;
import java.util.Scanner;

import entities.Student;

public class Program {

    public static void main(String[] args) {

        Locale.setDefault(Locale.US);
        Scanner sc = new Scanner(System.in);

        Student student = new Student();

        student.name = sc.nextLine();
        student.grade1 = sc.nextDouble();
        student.grade2 = sc.nextDouble();
        student.grade3 = sc.nextDouble();

        System.out.printf("FINAL GRADE: %.2f%n", student.finalGrade());

        if (student.finalGrade() < 60.0) {
            System.out.println("FAILED");
            System.out.printf("MISSING %.2f POINTS%n", student.missingPoints());
        }
        else {
            System.out.println("PASS");
        }
        sc.close();
    }
}
```

Curso Programação Orientada a Objetos com Java

Capítulo: Construtores, palavra this, sobrecarga, encapsulamento

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Construtores

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Construtor

- É uma operação especial da classe, que executa no momento da instanciação do objeto
- Usos comuns:
 - Iniciar valores dos atributos
 - Permitir ou obrigar que o objeto receba dados / dependências no momento de sua instanciação (injeção de dependência)
- Se um construtor customizado não for especificado, a classe disponibiliza o construtor padrão:
`Product p = new Product();`
- É possível especificar mais de um construtor na mesma classe (sobrecarga)

Problema exemplo

```

Enter product data:
Name: TV
Price: 900.00
Quantity in stock: 10

Product data: TV, $ 900.00, 10 units, Total: $ 9000.00

Enter the number of products to be added in stock: 5

Updated data: TV, $ 900.00, 15 units, Total: $ 13500.00

Enter the number of products to be removed from stock: 3

Updated data: TV, $ 900.00, 12 units, Total: $ 10800.00
  
```

Product
- Name : string
- Price : double
- Quantity : int
+ TotalValueInStock() : double
+ AddProducts(quantity : int) : void
+ RemoveProducts(quantity : int) : void

```

package application;

import java.util.Locale;
import java.util.Scanner;

import entities.Product;

public class Program {
    public static void main(String[] args) {
        Locale.setDefault(Locale.US);
        Scanner sc = new Scanner(System.in);

        Product product = new Product();
        System.out.println("Enter product data: ");
        System.out.print("Name: ");
        product.name = sc.nextLine();
        System.out.print("Price: ");
        product.price = sc.nextDouble();
        System.out.print("Quantity in stock: ");
        product.quantity = sc.nextInt();

        System.out.println();
        System.out.println("Product data: " + product);

        System.out.println();
        System.out.print("Enter the number of products to be added in stock: ");
        int quantity = sc.nextInt();
        product.addProducts(quantity);

        System.out.println();
        System.out.println("Updated data: " + product);

        System.out.println();
        System.out.print("Enter the number of products to be removed from stock: ");
        quantity = sc.nextInt();
        product.removeProducts(quantity);

        System.out.println();
        System.out.println("Updated data: " + product);

        sc.close();
    }
}

```

```

package entities;

public class Product {

    public String name;
    public double price;
    public int quantity;

    public double totalValueInStock() {
        return price * quantity;
    }

    public void addProducts(int quantity) {
        this.quantity += quantity;
    }

    public void removeProducts(int quantity) {
        this.quantity -= quantity;
    }

    public String toString() {
        return name
            + ", $ "
            + String.format("%.2f", price)
            + ", "
            + quantity
            + " units, Total: $ "
            + String.format("%.2f", totalValueInStock());
    }
}

```

Proposta de melhoria

Quando executamos o comando abaixo, instanciamos um produto "product" com seus atributos "vazios":

```
product = new Product();
```



Entretanto, faz sentido um produto que não tem nome? Faz sentido um produto que não tem preço?

Com o intuito de evitar a existência de produtos sem nome e sem preço, é possível fazer com que seja "obrigatória" a iniciação desses valores?

```
package entities;

public class Product {

    public String name;
    public double price;
    public int quantity;

    public Product(String name, double price, int quantity) {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }
    (...)
```

```
System.out.println("Enter product data: ");
System.out.print("Name: ");
String name = sc.nextLine();
System.out.print("Price: ");
double price = sc.nextDouble();
System.out.print("Quantity in stock: ");
int quantity = sc.nextInt();
Product product = new Product(name, price, quantity);
```

Palavra this

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

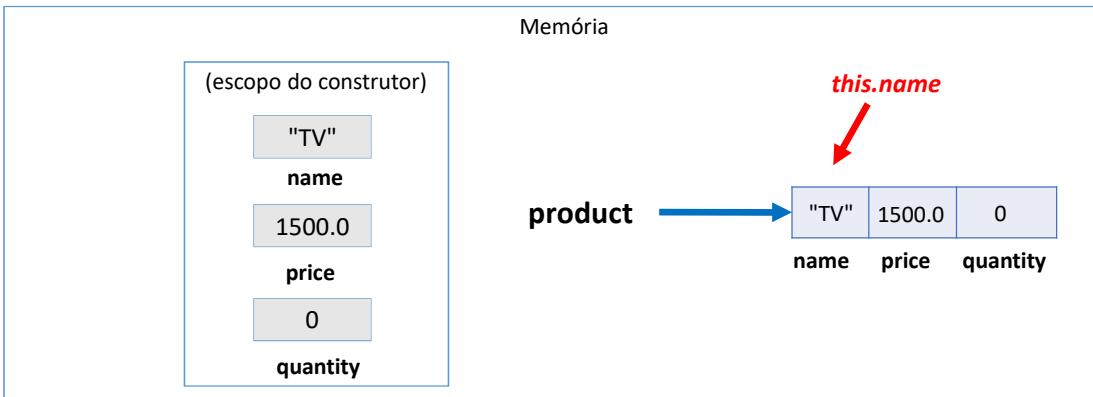
Palavra this

- É uma referência para o próprio objeto
- Usos comuns:
 - Diferenciar atributos de variáveis locais
 - Passar o próprio objeto como argumento na chamada de um método ou construtor

Diferenciar atributos de variáveis locais

```
Product product = new Product("TV", 1500.0, 0);
```

```
public Product(String name, double price, int quantity) {
    this.name = name;
    this.price = price;
    this.quantity = quantity;
}
```



Passar o próprio objeto como argumento na chamada de um método ou construtor

```
public class ChessMatch {
    ...
    placeNewPiece('e', 1, new King(board, Color.WHITE, this));
    ...
}
```

Sobrecarga

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Sobrecarga

- É um recurso que uma classe possui de oferecer mais de uma operação com o mesmo nome, porém com diferentes listas de parâmetros.

Proposta de melhoria

- Vamos criar um construtor opcional, o qual recebe apenas nome e preço do produto. A quantidade em estoque deste novo produto, por padrão, deverá então ser iniciada com o valor zero.
- Nota: é possível também incluir um construtor padrão

```
package entities;

public class Product {

    public String name;
    public double price;
    public int quantity;

    public Product() {}

    public Product(String name, double price, int quantity) {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }

    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }
    (...)
```

Encapsulamento

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Encapsulamento

- É um princípio que consiste em esconder detalhes de implementação de uma classe, expondo apenas operações seguras e que mantenham os objetos em um estado consistente.
- Regra de ouro: o objeto deve sempre estar em um estado consistente, e a própria classe deve garantir isso.

Analogia:



Regra geral básica

- Um objeto **NÃO** deve expor nenhum atributo (modificador de acesso **private**)
- Os atributos devem ser acessados por meio de métodos get e set
 - Padrão JavaBeans: <https://en.wikipedia.org/wiki/JavaBeans>

Padrão para implementação de getters e setters

```
private String name;
private double price;

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public double getPrice() {
    return price;
}

public void setPrice(double price) {
    this.price = price;
}
```

```
package entities;

public class Product {

    private String name;
    private double price;
    private int quantity;

    public Product() {
    }

    public Product(String name, double price, int quantity) {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }

    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public int getQuantity() {
        return quantity;
    }

    (...)
```

Gerando automaticamente
construtores, getters e setters
com Eclipse

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Comandos

- Botão direito -> Source -> Generate Constructor using Fields
- Botão direito -> Source -> Generate Getters and Setters

Modificadores de acesso

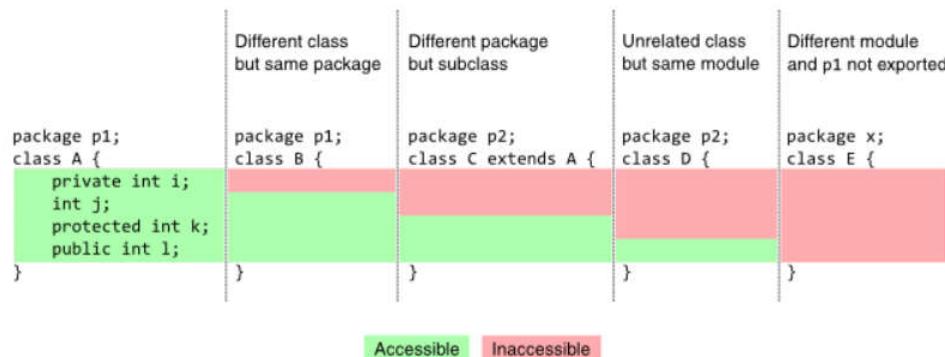
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Modificadores de acesso

- <https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html>
- **private**: o membro só pode ser acessado na **própria classe**
- (nada): o membro só pode ser acessado nas classes do **mesmo pacote**
- **protected**: o membro só pode ser acessado no **mesmo pacote**, bem como em **subclasses de pacotes diferentes**
- **public**: o membro é acessado por todas classes (ao menos que ele resida em um módulo diferente que não exporte o pacote onde ele está)

<https://stackoverflow.com/questions/215497/in-java-difference-between-package-private-public-protected-and-private>



Exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Em um banco, para se cadastrar uma conta bancária, é necessário informar o número da conta, o nome do titular da conta, e o valor de depósito inicial que o titular depositou ao abrir a conta. Este valor de depósito inicial, entretanto, é opcional, ou seja: se o titular não tiver dinheiro a depositar no momento de abrir sua conta, o depósito inicial não será feito e o saldo inicial da conta será, naturalmente, zero.

Importante: uma vez que uma conta bancária foi aberta, o número da conta nunca poderá ser alterado. Já o nome do titular pode ser alterado (pois uma pessoa pode mudar de nome por ocasião de casamento, por exemplo).

Por fim, o saldo da conta não pode ser alterado livremente. É preciso haver um mecanismo para proteger isso. O saldo só aumenta por meio de depósitos, e só diminui por meio de saques. Para cada saque realizado, o banco cobra uma taxa de \$ 5.00. Nota: a conta pode ficar com saldo negativo se o saldo não for suficiente para realizar o saque e/ou pagar a taxa.

Você deve fazer um programa que realize o cadastro de uma conta, dando opção para que seja ou não informado o valor de depósito inicial. Em seguida, realizar um depósito e depois um saque, sempre mostrando os dados da conta após cada operação.

(exemplos nas próximas páginas)

EXAMPLE 1

```
Enter account number: 8532
Enter account holder: Alex Green
Is there na initial deposit (y/n)? y
Enter initial deposit value: 500.00

Account data:
Account 8532, Holder: Alex Green, Balance: $ 500.00

Enter a deposit value: 200.00
Updated account data:
Account 8532, Holder: Alex Green, Balance: $ 700.00

Enter a withdraw value: 300.00
Updated account data:
Account 8532, Holder: Alex Green, Balance: $ 395.00
```

EXAMPLE 2

```
Enter account number: 7801
Enter account holder: Maria Brown
Is there na initial deposit (y/n)? n

Account data:
Account 7801, Holder: Maria Brown, Balance: $ 0.00

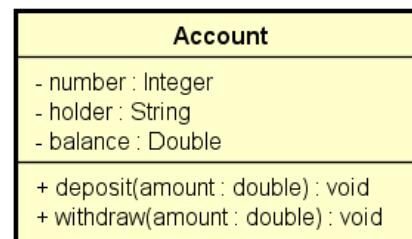
Enter a deposit value: 200.00
Updated account data:
Account 7801, Holder: Maria Brown, Balance: $ 200.00

Enter a withdraw value: 198.00
Updated account data:
Account 7801, Holder: Maria Brown, Balance: $ -3.00
```

Correção do exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves



<https://github.com/acenelio/encapsulation1-java>

Curso Programação Orientada a Objetos com Java

Capítulo: Comportamento de memória, arrays, listas

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Tipos referência vs. tipos valor

<http://educandoweb.com.br>

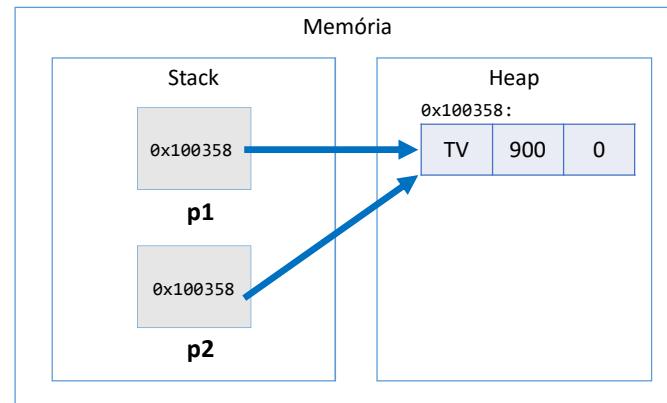
Prof. Dr. Nelio Alves

Classes são tipos referência

Variáveis cujo tipo são classes não devem ser entendidas como caixas, mas sim “tentáculos” (ponteiros) para caixas

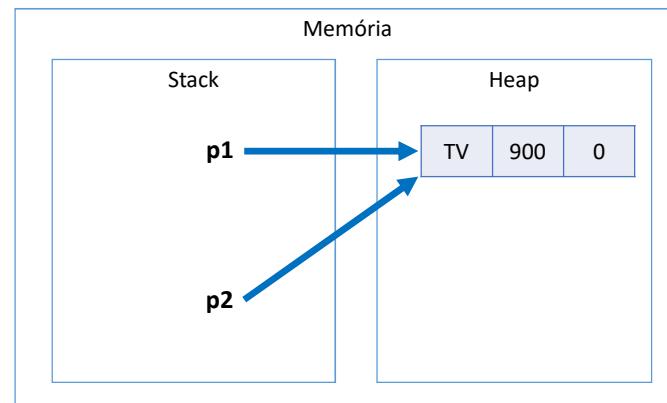
```
Product p1, p2;  
  
p1 = new Product("TV", 900.00, 0);  
  
p2 = p1;
```

**p2 = p1;
"p2 passa a apontar para onde
p1 aponta"**



Desenho simplificado

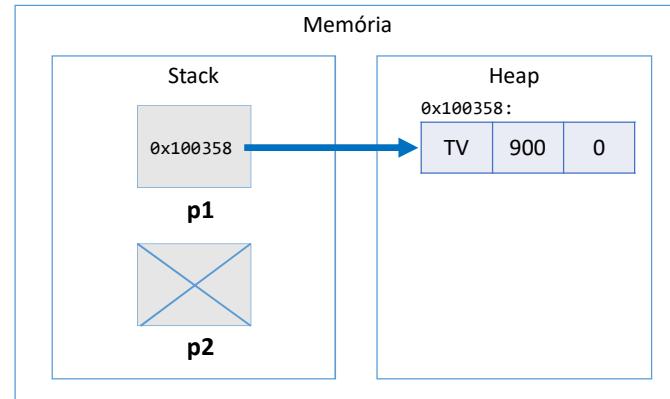
```
Product p1, p2;  
  
p1 = new Product("TV", 900.00, 0);  
  
p2 = p1;
```



Valor "null"

Tipos referência aceitam o valor "null", que indica que a variável aponta pra ninguém.

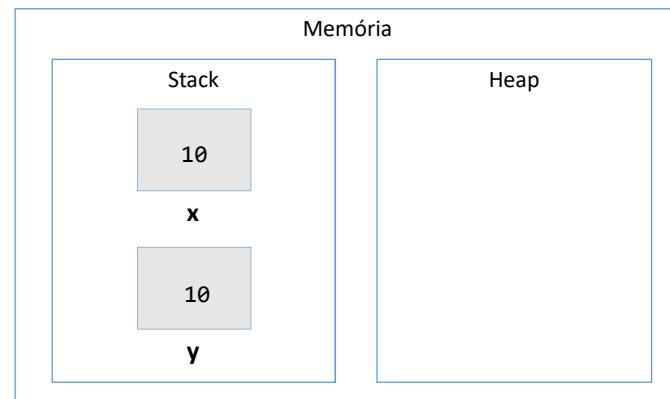
```
Product p1, p2;
p1 = new Product("TV", 900.00, 0);
p2 = null;
```



Tipos primitivos são tipos valor

Em Java, tipos primitivos são tipos valor. Tipos valor são CAIXAS e não ponteiros.

```
double x, y;
x = 10;
y = x;
y = x;
"y recebe uma CÓPIA de x"
```



Type	Contains	Default	Size	Range
boolean	true or false	false	1 bit	NA
char	Unicode character	\u0000	16 bits	\u0000 to \uFFFF
byte	Signed integer	0	8 bits	-128 to 127
short	Signed integer	0	16 bits	-32768 to 32767
int	Signed integer	0	32 bits	-2147483648 to 2147483647
long	Signed integer	0	64 bits	-9223372036854775808 to 9223372036854775807
float	IEEE 754 floating point	0.0	32 bits	±1.4E-45 to ±3.4028235E+38
double	IEEE 754 floating point	0.0	64 bits	±4.9E-324 to ±1.7976931348623157E+308

Tipos primitivos e inicialização

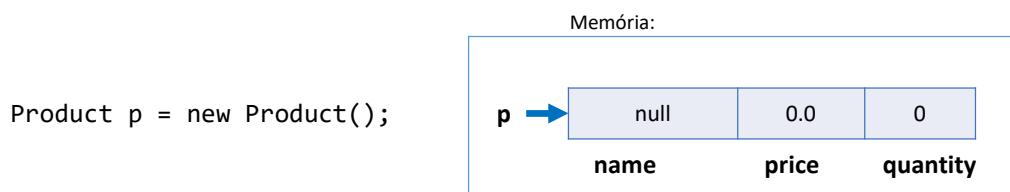
- Demo:

```
int p;
System.out.println(p); // erro: variável não iniciada

p = 10;
System.out.println(p);
```

Valores padrão

- Quando alocamos (new) qualquer tipo estruturado (classe ou array), são atribuídos valores padrão aos seus elementos
 - números: 0
 - boolean: false
 - char: caractere código 0
 - objeto: null



Tipos referência vs. tipos valor

CLASSE	TIPO PRIMITIVO
Vantagem: usufrui de todos recursos OO	Vantagem: é mais simples e mais performático
Variáveis são ponteiros	Variáveis são caixas
Objetos precisam ser instanciados usando new, ou apontar para um objeto já existente.	Não instancia. Uma vez declarados, estão prontos para uso.
Aceita valor null	Não aceita valor null
Y = X; "Y passa a apontar para onde X aponta"	Y = X; "Y recebe uma cópia de X"
Objetos instanciados no heap	"Objetos" instanciados no stack
Objetos não utilizados são desalocados em um momento próximo pelo garbage collector	"Objetos" são desalocados imediatamente quando seu escopo de execução é finalizado

Desalocação de memória - garbage collector e escopo local

<http://educandoweb.com.br>

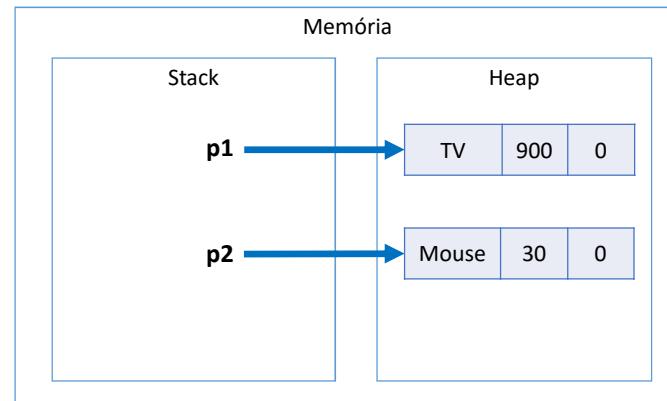
Prof. Dr. Nelio Alves

Garbage collector

- É um processo que automatiza o gerenciamento de memória de um programa em execução
- O garbage collector monitora os objetos alocados dinamicamente pelo programa (no heap), desalocando aqueles que não estão mais sendo utilizados.

Desalocação por garbage collector

```
Product p1, p2;
p1 = new Product("TV", 900.00, 0);
p2 = new Product("Mouse", 30.00, 0);
```

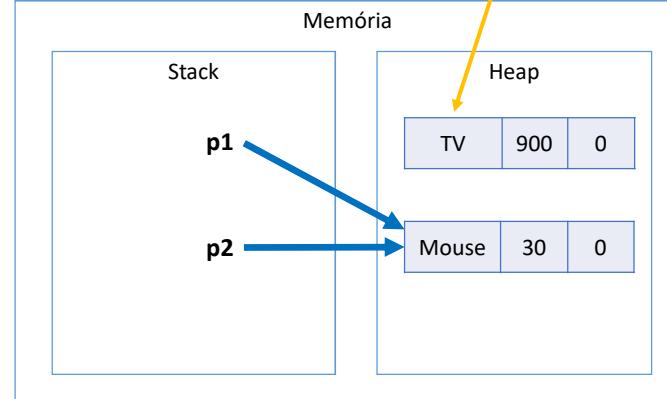


Desalocação por garbage collector

```
Product p1, p2;
p1 = new Product("TV", 900.00, 0);
p2 = new Product("Mouse", 30.00, 0);
```

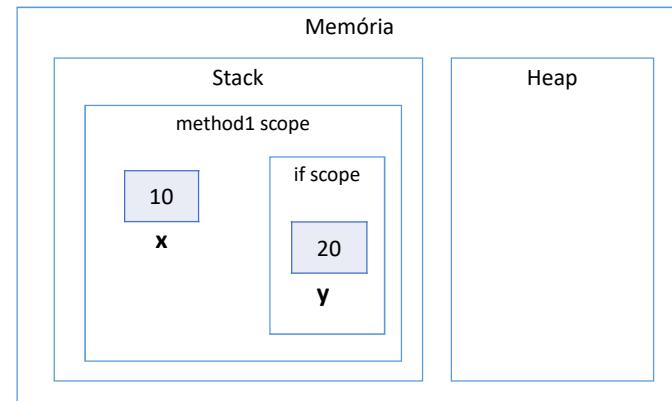
p1 = p2;

Será desalocado
pelo garbage
collector



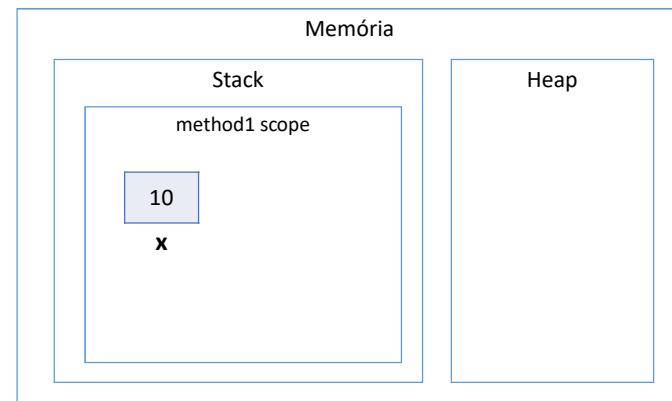
Desalocação por escopo

```
void method1() {  
    int x = 10;  
    if (x > 0) {  
        int y = 20;  
    }  
    System.out.println(x);  
}
```



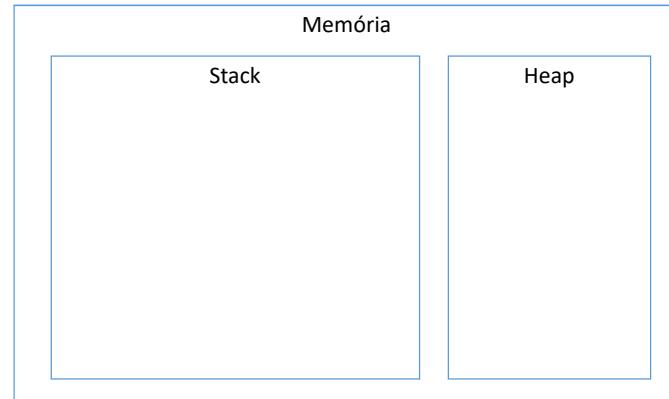
Desalocação por escopo

```
void method1() {  
    int x = 10;  
    if (x > 0) {  
        int y = 20;  
    }  
    System.out.println(x);  
}
```



Desalocação por escopo

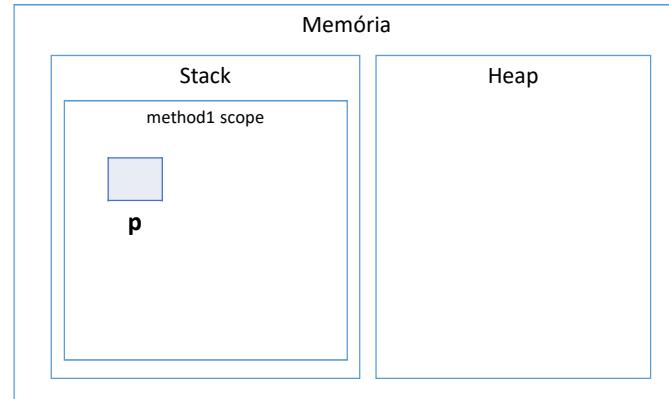
```
void method1() {
    int x = 10;
    if (x > 0) {
        int y = 20;
    }
    System.out.println(x);
}
```



Outro exemplo

```
void method1() {
    Product p = method2();
    System.out.println(p.Name);
}

Product method2() {
    Product prod = new Product("TV", 900.0, 0);
    return prod;
}
```



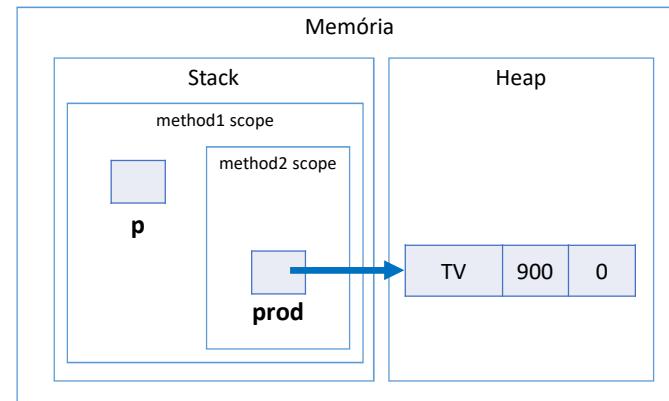
Outro exemplo

```

void method1() {
    Product p = method2();
    System.out.println(p.Name);
}

Product method2() {
    Product prod = new Product("TV", 900.0, 0);
    return prod;
}

```



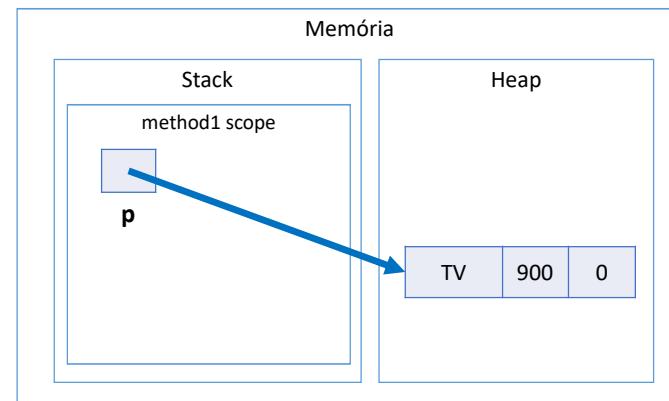
Outro exemplo

```

void method1() {
    Product p = method2();
    System.out.println(p.Name);
}

Product method2() {
    Product prod = new Product("TV", 900.0, 0);
    return prod;
}

```



Resumo

- Objetos alocados dinamicamente, quando não possuem mais referência para eles, serão desalocados pelo garbage collector
- Variáveis locais são desalocadas imediatamente assim que seu escopo local sai de execução

Vetores - Parte 1

<http://educandoweb.com.br>

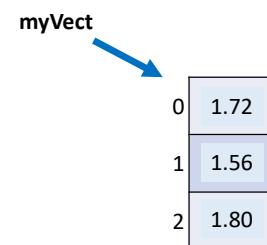
Prof. Dr. Nelio Alves

Checklist

- Revisão do conceito de vetor
- Declaração e instanciação
- Manipulação de vetor de elementos tipo valor (tipo primitivo)
- Manipulação de vetor de elementos tipo referência (classe)
- Acesso aos elementos
- Propriedade length

Vetores

- Em programação, "vetor" é o nome dado a arranjos unidimensionais
- Arranjo (array) é uma estrutura de dados:
 - Homogênea (dados do mesmo tipo)
 - Ordenada (elementos acessados por meio de posições)
 - Alocada de uma vez só, em um bloco contíguo de memória
- Vantagens:
 - Acesso imediato aos elementos pela sua posição
- Desvantagens:
 - Tamanho fixo
 - Dificuldade para se realizar inserções e deleções

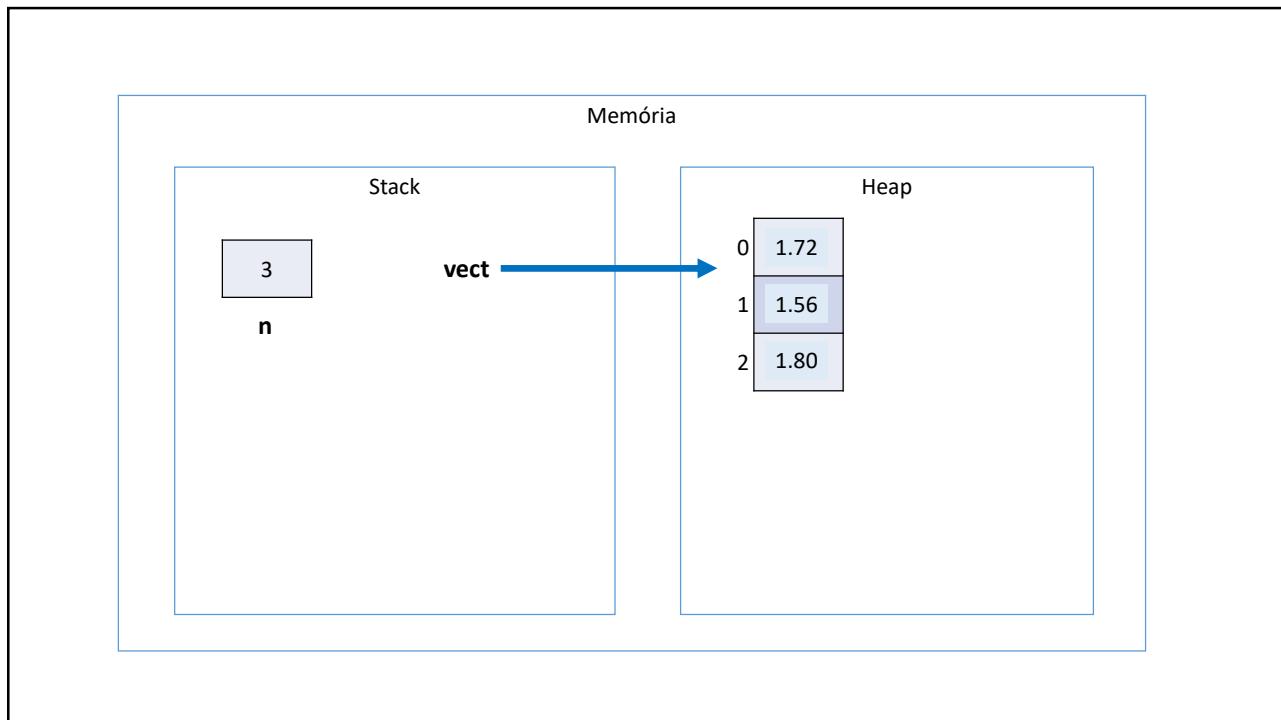


Problema exemplo 1

Fazer um programa para ler um número inteiro N e a altura de N pessoas. Armazene as N alturas em um vetor. Em seguida, mostrar a altura média dessas pessoas.

Example

Input:	Output:
3 1.72 1.56 1.80	AVERAGE HEIGHT = 1.69



```

package application;

import java.util.Locale;
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        Locale.setDefault(Locale.US);
        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();
        double[] vect = new double[n];

        for (int i=0; i<n; i++) {
            vect[i] = sc.nextDouble();
        }

        double sum = 0.0;
        for (int i=0; i<n; i++) {
            sum += vect[i];
        }
        double avg = sum / n;

        System.out.printf("AVERAGE HEIGHT: %.2f%n", avg);
        sc.close();
    }
}

```

Vetores - Parte 2

<http://educandoweb.com.br>

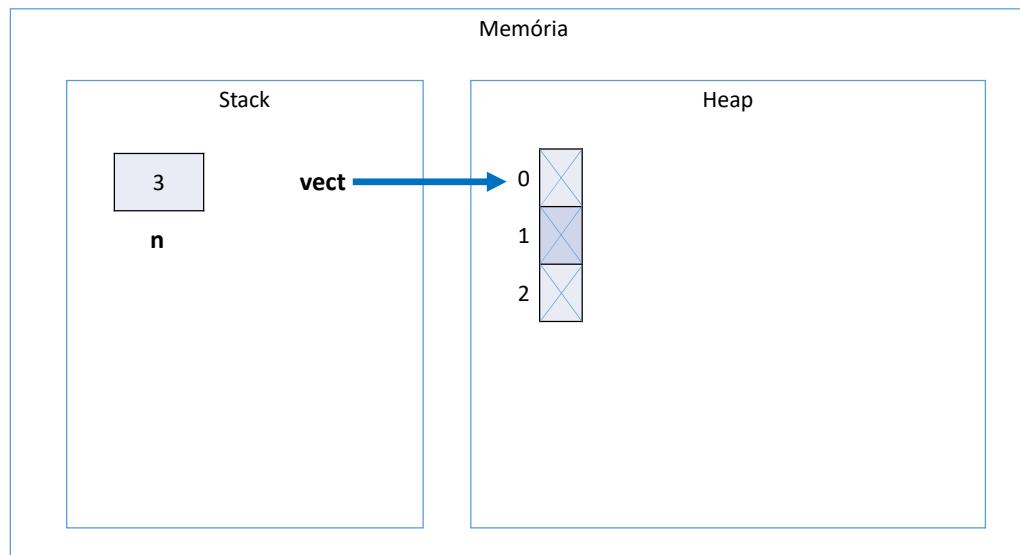
Prof. Dr. Nelio Alves

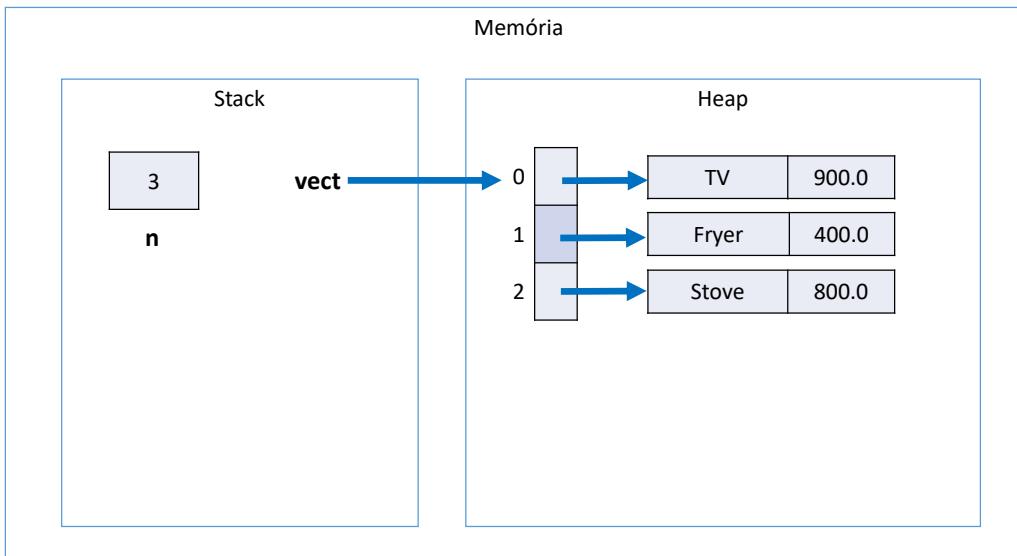
Problema exemplo 2

Fazer um programa para ler um número inteiro N e os dados (nome e preço) de N Produtos. Armazene os N produtos em um vetor. Em seguida, mostrar o preço médio dos produtos.

Example

Input:	Output:
3 TV 900.00 Fryer 400.00 Stove 800.00	AVERAGE PRICE = 700.00





```

package application;

import java.util.Locale;
import java.util.Scanner;

import entities.Product;

public class Program {

    public static void main(String[] args) {
        Locale.setDefault(Locale.US);
        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();
        Product[] vect = new Product[n];

        for (int i=0; i<vect.length; i++) {
            sc.nextLine();
            String name = sc.nextLine();
            double price = sc.nextDouble();
            vect[i] = new Product(name, price);
        }

        double sum = 0.0;
        for (int i=0; i<vect.length; i++) {
            sum += vect[i].getPrice();
        }
        double avg = sum / vect.length;

        System.out.printf("AVERAGE PRICE = %.2f%n", avg);
        sc.close();
    }
}
  
```

Exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

A dona de um pensionato possui dez quartos para alugar para estudantes, sendo esses quartos identificados pelos números 0 a 9.

Fazer um programa que inicie com todos os dez quartos vazios, e depois leia uma quantidade N representando o número de estudantes que vão alugar quartos (N pode ser de 1 a 10). Em seguida, registre o aluguel dos N estudantes. Para cada registro de aluguel, informar o nome e email do estudante, bem como qual dos quartos ele escolheu (de 0 a 9). Suponha que seja escolhido um quarto vago. Ao final, seu programa deve imprimir um relatório de todas ocupações do pensionato, por ordem de quarto, conforme exemplo.

How many rooms will be rented? **3**

Rent #1:

Name: **Maria Green**
 Email: **maria@gmail.com**
 Room: **5**

Rent #2:

Name: **Marco Antonio**
 Email: **marco@gmail.com**
 Room: **1**

Rent #3:

Name: **Alex Brown**
 Email: **alex@gmail.com**
 Room: **8**

Busy rooms:

1: Marco Antonio, marco@gmail.com
 5: Maria Green, maria@gmail.com
 8: Alex Brown, alex@gmail.com

Sugestão

```
if (vect[i] != null)
```



(correção na próxima página)

```

package entities;

public class Rent {

    private String name;
    private String email;

    public Rent(String name, String email) {
        this.name = name;
        this.email = email;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String toString() {
        return name + ", " + email;
    }
}

```

```

package application;

import java.text.ParseException;
import java.util.Scanner;

import entities.Rent;

public class Program {

    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        Rent[] vect = new Rent[10];
        System.out.print("How many rooms will be rented? ");
        int n = sc.nextInt();
        for (int i=1; i<n; i++) {
            System.out.println();
            System.out.print("Rent # " + i + ":");
            System.out.print("Name: ");
            sc.nextLine();
            String name = sc.nextLine();
            System.out.print("Email: ");
            String email = sc.nextLine();
            System.out.print("Room: ");
            int room = sc.nextInt();
            vect[room] = new Rent(name, email);
        }
        System.out.println();
        System.out.println("Busy rooms:");
        for (int i=0; i<10; i++) {
            if (vect[i] != null) {
                System.out.println(i + ": " + vect[i]);
            }
        }
        sc.close();
    }
}

```

Boxing, unboxing e wrapper classes

<http://educandoweb.com.br>

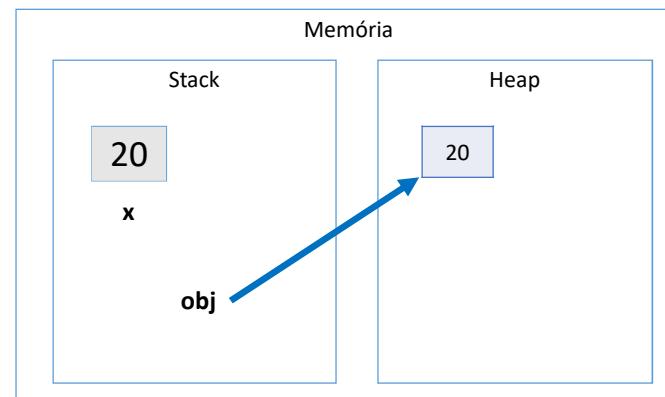
Prof. Dr. Nelio Alves

Boxing

- É o processo de conversão de um objeto tipo valor para um objeto tipo referência compatível

```
int x = 20;
```

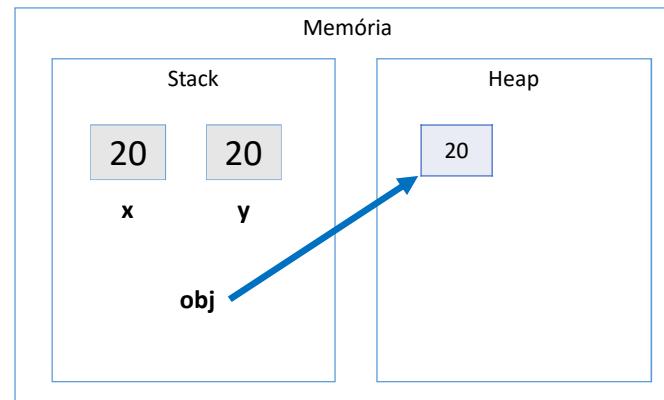
```
Object obj = x;
```



Unboxing

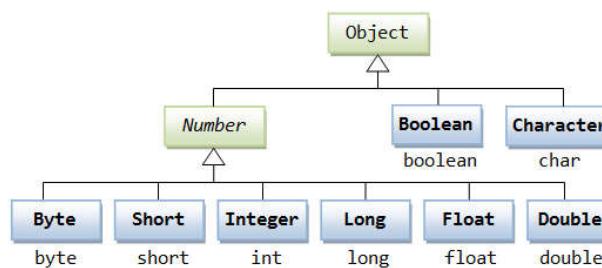
- É o processo de conversão de um objeto tipo referência para um objeto tipo valor compatível

```
int x = 20;
Object obj = x;
int y = (int) obj;
```



Wrapper classes

- São classes equivalentes aos tipos primitivos
- Boxing e unboxing é natural na linguagem
- Uso comum: campos de entidades em sistemas de informação (IMPORTANTE!)
 - Pois tipos referência (classes) aceitam valor null e usufruem dos recursos OO



Demo

```
Integer x = 10;  
int y = x * 2;
```

```
public class Product {  
  
    public String name;  
    public Double price;  
    public Integer quantity;  
  
    (...)
```

Laço "for each"

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Laço "for each"

Sintaxe opcional e simplificada para percorrer coleções

Sintaxe:

```
for (Tipo apelido : coleção) {  
    <comando 1>  
    <comando 2>  
}
```

Demo

Leitura: "para cada objeto 'obj' contido em vect, faça:"

```
String[] vect = new String[] {"Maria", "Bob", "Alex"};  
  
for (int i=0; i< vect.length; i++) {  
    System.out.println(vect[i]);  
}  
  
for (String obj : vect) {  
    System.out.println(obj);  
}
```

Listas - Parte 1

<http://educandoweb.com.br>

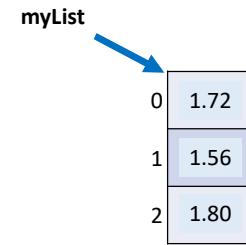
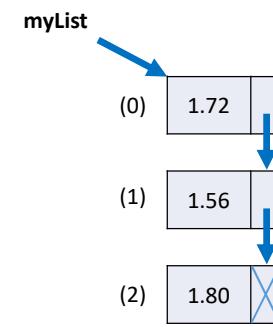
Prof. Dr. Nelio Alves

Checklist

- Conceito de lista
- Tipo **List** - Declaração, instanciação
- Demo
- Referência: <https://docs.oracle.com/javase/10/docs/api/java/util/List.html>
- Assuntos pendentes:
 - interfaces
 - generics
 - predicados (lambda)

Listas

- Lista é uma estrutura de dados:
 - Homogênea (dados do mesmo tipo)
 - Ordenada (elementos acessados por meio de posições)
 - Inicia vazia, e seus elementos são alocados sob demanda
 - Cada elemento ocupa um "nó" (ou nodo) da lista
- Tipo (interface): List
- Classes que implementam: ArrayList, LinkedList, etc.
- Vantagens:
 - Tamanho variável
 - Facilidade para se realizar inserções e deleções
- Desvantagens:
 - Acesso sequencial aos elementos *



(desenho simplificado)

Listas - Parte 2

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Demo

- Tamanho da lista: `size()`
- Obter o elemento de uma posição: `get(position)`
- Inserir elemento na lista: `add(obj)`, `add(int, obj)`
- Remover elementos da lista: `remove(obj)`, `remove(int)`, `removeIf(Predicate)`
- Encontrar posição de elemento: `indexOf(obj)`, `lastIndexOf(obj)`
- Filtrar lista com base em predicado:
`List<Integer> result = list.stream().filter(x -> x > 4).collect(Collectors.toList());`
- Encontrar primeira ocorrência com base em predicado:
`Integer result = list.stream().filter(x -> x > 4).findFirst().orElse(null);`
- Assuntos pendentes:
 - interfaces
 - generics
 - predicados (lambda)

```
package application;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

public class Program {

    public static void main(String[] args) {
        List<String> list = new ArrayList<>();

        list.add("Maria");
        list.add("Alex");
        list.add("Bob");
        list.add("Anna");
        list.add(2, "Marco");

        System.out.println(list.size());
        for (String x : list) {
            System.out.println(x);
        }
        System.out.println("-----");
        list.removeIf(x -> x.charAt(0) == 'M');
        for (String x : list) {
            System.out.println(x);
        }
        System.out.println("-----");
        System.out.println("Index of Bob: " + list.indexOf("Bob"));
        System.out.println("Index of Marco: " + list.indexOf("Marco"));
        System.out.println("-----");
        List<String> result = list.stream().filter(x -> x.charAt(0) == 'A').collect(Collectors.toList());
        for (String x : result) {
            System.out.println(x);
        }
        System.out.println("-----");
        String name = list.stream().filter(x -> x.charAt(0) == 'J').findFirst().orElse(null);
        System.out.println(name);
    }
}
```

Exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Fazer um programa para ler um número inteiro N e depois os dados (id, nome e salario) de N funcionários. Não deve haver repetição de id.

Em seguida, efetuar o aumento de X por cento no salário de um determinado funcionário. Para isso, o programa deve ler um id e o valor X. Se o id informado não existir, mostrar uma mensagem e abortar a operação. Ao final, mostrar a listagem atualizada dos funcionários, conforme exemplos.

Lembre-se de aplicar a técnica de encapsulamento para não permitir que o salário possa ser mudado livremente. Um salário só pode ser aumentado com base em uma operação de aumento por porcentagem dada.

(exemplo na próxima página)

How many employees will be registered? **3**

Employee #1:

Id: **333**

Name: **Maria Brown**

Salary: **4000.00**

Employee #2:

Id: **536**

Name: **Alex Grey**

Salary: **3000.00**

Employee #3:

Id: **772**

Name: **Bob Green**

Salary: **5000.00**

Enter the employee id that will have salary increase : **536**

Enter the percentage: **10.0**

List of employees:

333, Maria Brown, 4000.00

536, Alex Grey, 3000.00

772, Bob Green, 5000.00

How many employees will be registered? **2**

Employee #1:

Id: **333**

Name: **Maria Brown**

Salary: **4000.00**

Employee #2:

Id: **536**

Name: **Alex Grey**

Salary: **3000.00**

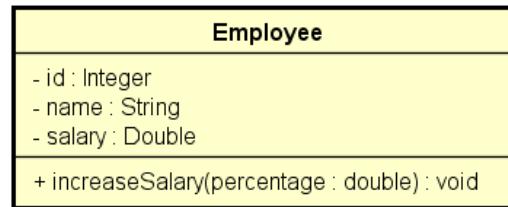
Enter the employee id that will have salary increase: **776**

This id does not exist!

List of employees:

333, Maria Brown, 4000.00

536, Alex Grey, 3000.00



<https://github.com/acenelio/list1-java>

Matrizes

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Checklist

- Revisão do conceito de matriz
- Declaração e instanciação
- Acesso aos elementos / como percorrer uma matriz
- Propriedade length

Matrizes

- Em programação, "matriz" é o nome dado a arranjos bidimensionais
 - Atenção: "vetor de vetores"
- Arranjo (array) é uma estrutura de dados:
 - Homogênea (dados do mesmo tipo)
 - Ordenada (elementos acessados por meio de posições)
 - Alocada de uma vez só, em um bloco contíguo de memória
- Vantagens:
 - Acesso imediato aos elementos pela sua posição
- Desvantagens:
 - Tamanho fixo
 - Dificuldade para se realizar inserções e deleções

`myMat`



	0	1	2	3
0	3.5	17.0	12.3	8.2
1	4.1	6.2	7.5	2.9
2	11.0	9.5	14.8	21.7

Exercício resolvido

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

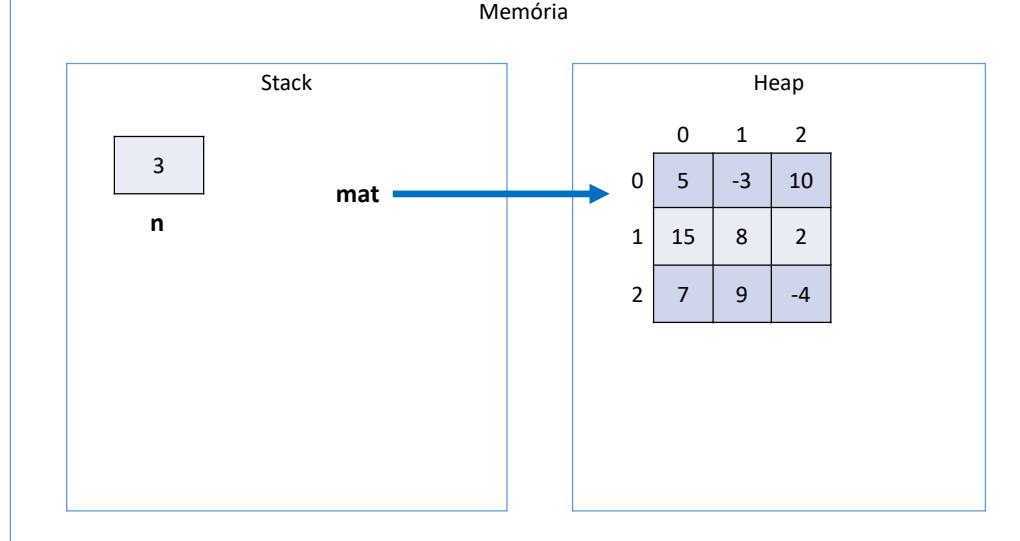
Exercício resolvido

Fazer um programa para ler um número inteiro N e uma matriz de ordem N contendo números inteiros. Em seguida, mostrar a diagonal principal e a quantidade de valores negativos da matriz.

Example

Input:	Output:
3 5 -3 10 15 8 2 7 9 -4	Main diagonal: 5 8 -4 Negative numbers = 2

<https://github.com/acenelio/matrix1-java>



Exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Fazer um programa para ler dois números inteiros M e N, e depois ler uma matriz de M linhas por N colunas contendo números inteiros, podendo haver repetições. Em seguida, ler um número inteiro X que pertence à matriz. Para cada ocorrência de X, mostrar os valores à esquerda, acima, à direita e abaixo de X, quando houver, conforme exemplo.

Example

```
3 4
10 8 15 12
21 11 23 8
14 5 13 19
8
Position 0,1:
Left: 10
Right: 15
Down: 11
Position 1,3:
Left: 23
Up: 12
Down: 19
```

<https://github.com/acenelio/matrix2-java>

Exercícios de fixação sobre vetores

ATENÇÃO: nos exemplos, os dados em **vermelho** representam os dados que o usuário vai digitar.

Problema "negativos"

Correção: <https://github.com/acenelio/curso-algoritmos/blob/master/java/negativos.java>

Faça um programa que leia um número inteiro positivo N (máximo = 10) e depois N números inteiros e armazene-os em um vetor. Em seguida, mostrar na tela todos os números negativos lidos.

Exemplo:

```
Quantos numeros voce vai digitar? 6
```

```
Digite um numero: 8
```

```
Digite um numero: -2
```

```
Digite um numero: 9
```

```
Digite um numero: 10
```

```
Digite um numero: -3
```

```
Digite um numero: -7
```

```
NUMEROS NEGATIVOS:
```

```
-2
```

```
-3
```

```
-7
```

Problema "soma_vetor"

Correção: https://github.com/acenelio/curso-algoritmos/blob/master/java/soma_vetor.java

Faça um programa que leia N números reais e armazene-os em um vetor. Em seguida:

- Imprimir todos os elementos do vetor
- Mostrar na tela a soma e a média dos elementos do vetor

Exemplo:

```
Quantos numeros voce vai digitar? 4
```

```
Digite um numero: 8.0
```

```
Digite um numero: 4.0
```

```
Digite um numero: 10.0
```

```
Digite um numero: 14.0
```

```
VALORES = 8.0 4.0 10.0 14.0
```

```
SOMA = 36.00
```

```
MEDIA = 9.00
```

Problema "alturas"

Correção: <https://github.com/acenelio/curso-algoritmos/blob/master/java/alturas.java>

Fazer um programa para ler nome, idade e altura de N pessoas, conforme exemplo. Depois, mostrar na tela a altura média das pessoas, e mostrar também a porcentagem de pessoas com menos de 16 anos, bem como os nomes dessas pessoas caso houver.

Exemplo:

```
Quantas pessoas serao digitadas? 5
```

```
Dados da 1a pessoa:
```

```
Nome: Joao
```

```
Idade: 15
```

```
Altura: 1.82
```

```
Dados da 2a pessoa:
```

```
Nome: Maria
```

```
Idade: 16
```

```
Altura: 1.60
```

```
Dados da 3a pessoa:
```

```
Nome: Teresa
```

```
Idade: 14
```

```
Altura: 1.58
```

```
Dados da 4a pessoa:
```

```
Nome: Carlos
```

```
Idade: 21
```

```
Altura: 1.65
```

```
Dados da 5a pessoa:
```

```
Nome: Paulo
```

```
Idade: 17
```

```
Altura: 1.78
```

```
Altura média: 1.69
```

```
Pessoas com menos de 16 anos: 40.0%
```

```
Joao
```

```
Teresa
```

Problema "numeros_pares"

Correção: https://github.com/acenelio/curso-algoritmos/blob/master/java/numeros_pares.java

Faça um programa que leia N números inteiros e armazene-os em um vetor. Em seguida, mostre na tela todos os números pares, e também a quantidade de números pares.

Exemplo:

```
Quantos numeros voce vai digitar? 6
```

```
Digite um numero: 8
```

```
Digite um numero: 2
```

```
Digite um numero: 11
```

```
Digite um numero: 14
```

```
Digite um numero: 13
```

```
Digite um numero: 20
```

```
NUMEROS PARES:
```

```
8 2 14 20
```

```
QUANTIDADE DE PARES = 4
```

Problema "maior_posicao"

Correção: https://github.com/acenelio/curso-algoritmos/blob/master/java/maior_posicao.java

Faça um programa que leia N números reais e armazene-os em um vetor. Em seguida, mostrar na tela o maior número do vetor (supor não haver empates). Mostrar também a posição do maior elemento, considerando a primeira posição como 0 (zero).

Exemplo:

```
Quantos numeros voce vai digitar? 6
```

```
Digite um numero: 8.0
```

```
Digite um numero: 4.0
```

```
Digite um numero: 10.0
```

```
Digite um numero: 14.0
```

```
Digite um numero: 13.0
```

```
Digite um numero: 7.0
```

```
MAIOR VALOR = 14.0
```

```
POSICAO DO MAIOR VALOR = 3
```

Problema "soma_vetores"

Correção: https://github.com/acenelio/curso-algoritmos/blob/master/java/soma_vetores.java

Faça um programa para ler dois vetores A e B, contendo N elementos cada. Em seguida, gere um terceiro vetor C onde cada elemento de C é a soma dos elementos correspondentes de A e B. Imprima o vetor C gerado.

Exemplo:

```
Quantos valores vai ter cada vetor? 6
```

```
Digite os valores do vetor A:
```

```
8
```

```
2
```

```
11
```

```
14
```

```
13
```

```
20
```

```
Digite os valores do vetor B:
```

```
5
```

```
10
```

```
3
```

```
1
```

```
10
```

```
7
```

```
VETOR RESULTANTE:
```

```
13
```

```
12
```

```
14
```

```
15
```

```
23
```

```
27
```

Problema "abaixo_da_media"

Correção: https://github.com/acenelio/curso-algoritmos/blob/master/java/abaixo_da_media.java

Fazer um programa para ler um número inteiro N e depois um vetor de N números reais. Em seguida, mostrar na tela a média aritmética de todos elementos com três casas decimais. Depois mostrar todos os elementos do vetor que estejam abaixo da média, com uma casa decimal cada.

Exemplo:

Quantos elementos vai ter o vetor? **4**

Digite um numero: **10.0**

Digite um numero: **15.5**

Digite um numero: **13.2**

Digite um numero: **9.8**

MEDIA DO VETOR = **12.125**

ELEMENTOS ABAIXO DA MEDIA:

10.0

9.8

Problema "media_pares"

Correção: https://github.com/acenelio/curso-algoritmos/blob/master/java/media_pares.java

Fazer um programa para ler um vetor de N números inteiros. Em seguida, mostrar na tela a média aritmética somente dos números pares lidos, com uma casa decimal. Se nenhum número par for digitado, mostrar a mensagem "NENHUM NUMERO PAR"

Exemplo 1:

Quantos elementos vai ter o vetor? **6**

Digite um numero: **8**

Digite um numero: **2**

Digite um numero: **11**

Digite um numero: **14**

Digite um numero: **13**

Digite um numero: **20**

MEDIA DOS PARES = **11.0**

Exemplo 2:

Quantos elementos vai ter o vetor? **3**

Digite um numero: **7**

Digite um numero: **9**

Digite um numero: **11**

NENHUM NUMERO PAR

Problema "mais_velho"

Correção: https://github.com/acenelio/curso-algoritmos/blob/master/java/mais_velho.java

Fazer um programa para ler um conjunto de nomes de pessoas e suas respectivas idades. Os nomes devem ser armazenados em um vetor, e as idades em um outro vetor. Depois, mostrar na tela o nome da pessoa mais velha.

Exemplo:

```
Quantas pessoas voce vai digitar? 5
```

```
Dados da 1a pessoa:
```

```
Nome: Joao
```

```
Idade: 16
```

```
Dados da 2a pessoa:
```

```
Nome: Maria
```

```
Idade: 21
```

```
Dados da 3a pessoa:
```

```
Nome: Teresa
```

```
Idade: 15
```

```
Dados da 4a pessoa:
```

```
Nome: Carlos
```

```
Idade: 23
```

```
Dados da 5a pessoa:
```

```
Nome: Paulo
```

```
Idade: 17
```

```
PESSOA MAIS VELHA: Carlos
```

Problema "aprovados"

Correção: <https://github.com/acenelio/curso-algoritmos/blob/master/java/aprovados.java>

Fazer um programa para ler um conjunto de N nomes de alunos, bem como as notas que eles tiraram no 1º e 2º semestres. Cada uma dessas informações deve ser armazenada em um vetor. Depois, imprimir os nomes dos alunos aprovados, considerando aprovados aqueles cuja média das notas seja maior ou igual a 6.0 (seis).

Exemplo:

```
Quantos alunos serao digitados? 4
```

```
Digite nome, primeira e segunda nota do 1o aluno:
```

```
Joao Silva
```

```
7.0
```

```
8.5
```

```
Digite nome, primeira e segunda nota do 2o aluno:
```

```
Maria Teixeira
```

```
9.2
```

```
6.5
```

```
Digite nome, primeira e segunda nota do 3o aluno:
```

```
Carlos Carvalho
```

```
5.0
```

```
6.0
```

```
Digite nome, primeira e segunda nota do 4o aluno:
```

```
Teresa Pires
```

```
5.5
```

```
6.5
```

```
Alunos aprovados:
```

```
Joao Silva
```

```
Maria Teixeira
```

```
Teresa Pires
```

Problema "dados_pessoas"

Correção: https://github.com/acenelio/curso-algoritmos/blob/master/java/dados_pessoas.java

Tem-se um conjunto de dados contendo a altura e o gênero (M, F) de N pessoas. Fazer um programa que calcule e escreva a maior e a menor altura do grupo, a média de altura das mulheres, e o número de homens.

Exemplo:

```
Quantas pessoas serão digitadas? 5
Altura da 1a pessoa: 1.70
Gênero da 1a pessoa: F
Altura da 2a pessoa: 1.83
Gênero da 2a pessoa: M
Altura da 3a pessoa: 1.54
Gênero da 3a pessoa: M
Altura da 4a pessoa: 1.61
Gênero da 4a pessoa: F
Altura da 5a pessoa: 1.75
Gênero da 5a pessoa: F
Menor altura = 1.54
Maior altura = 1.83
Media das alturas das mulheres = 1.69
Número de homens = 2
```



Trabalhando com data-hora

Nelio Alves

<https://devsuperior.com.br>

1

Conceitos importantes

- **Data-[hora] local:**

ano-mês-dia-[hora] sem fuso horário
[hora] opcional

- **Data-hora global:**

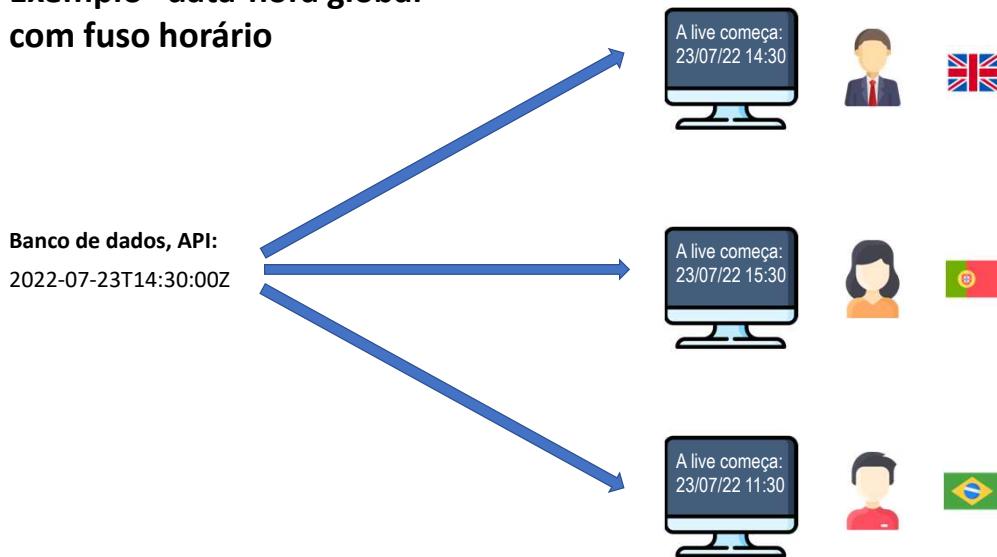
ano-mês-dia-hora com fuso horário

- **Duração:**

tempo decorrido entre duas data-horas

2

Exemplo "data-hora global" com fuso horário



3

Quando usar?

- **Data-[hora] local:**

Quando o momento exato não interessa a pessoas de outro fuso horário.

Uso comum: sistemas de região única, Excel.

Data de nascimento: "15/06/2001"

Data-hora da venda: "13/08/2022 às 15:32" (presumindo não interessar fuso horário)

- **Data-hora global:**

Quando o momento exato interessa a pessoas de outro fuso horário.

Uso comum: sistemas multi-região, web.

Quando será o sorteio? "21/08/2022 às 20h (horário de São Paulo)"

Quando o comentário foi postado? "há 17 minutos"

Quando foi realizada a venda? "13/08/2022 às 15:32 (horário de São Paulo)"

Início e fim do evento? "21/08/2022 às 14h até 16h (horário de São Paulo)"

4

Timezone (fuso horário)

- GMT - Greenwich Mean Time
 - Horário de Londres
 - Horário do padrão UTC - Coordinated Universal Time
 - Também chamado de "Z" time, ou Zulu time
- Outros fuso horários são relativos ao GMT/UTC:
 - São Paulo: GMT-3
 - Manaus: GMT-4
 - Portugal: GMT+1
- Muitas linguagens/tecnologias usam nomes para as timezones:
 - "US/Pacific"
 - "America/Sao_Paulo"
 - etc.

5

Padrão ISO 8601

Data-[hora] local:

2022-07-21
2022-07-21T14:52
2022-07-22T14:52:09
2022-07-22T14:52:09.4073

Data-hora global:

2022-07-23T14:52:09Z
2022-07-23T14:52:09.254935Z
2022-07-23T14:52:09-03:00

6

Operações importantes com data-hora

- Instanciação
 - (agora) → Data-hora
 - Texto ISO 8601 → Data-hora
 - Texto formato customizado → Data-hora
 - dia, mês, ano, [horário] → Data-hora local
- Formatação
 - Data-hora → Texto ISO 8601
 - Data-hora → Texto formato customizado

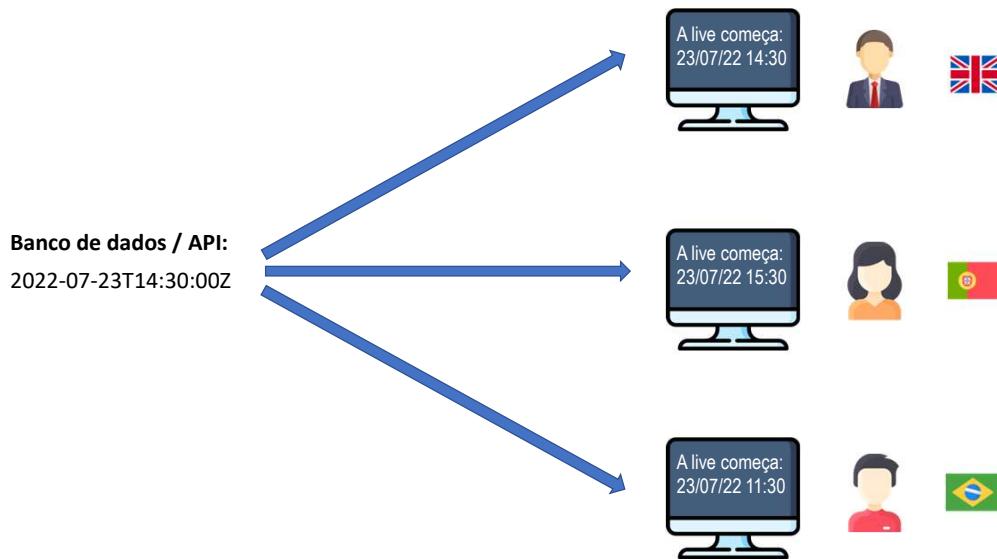
7

Operações importantes com data-hora

- Converter data-hora global para local
 - Data-hora global, timezone (sistema local) → Data-hora local
- Obter dados de uma data-hora local
 - Data-hora local → dia, mês, ano, horário
- Cálculos com data-hora
 - Data-hora +/- tempo → Data-hora
 - Data-hora 1, Data-hora 2 → Duração

8

Boa prática: armazene UTC, mostre local



9

Principais tipos Java (versão 8+)

- Data-hora local
 - LocalDate
 - LocalDateTime
- Data-hora global
 - Instant
- Duração
 - Duration
- Outros
 - ZoneId
 - ChronoUnit

10

Demo

<https://github.com/devsuperior/date-time-java>

Curso Java Completo

Capítulo: Tópicos especiais em Java

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Date

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Date

Representa um INSTANTE

Pacote java.util

<https://docs.oracle.com/javase/10/docs/api/java/util/Date.html>

Um objeto Date internamente armazena:

- O número de milissegundos desde a meia noite do dia 1 de janeiro de 1970 GMT (UTC)
 - *GMT: Greenwich Mean Time (time zone)*
 - *UTC: Coordinated Universal Time (time standard)*

SimpleDateFormat

- <https://docs.oracle.com/javase/10/docs/api/java/text/SimpleDateFormat.html>
- Define formatos para conversão entre Date e String
 - dd/MM/yyyy -> 23/07/2018
 - dd/MM/yyyy HH:mm:ss -> 23/07/2018 15:42:07

Padrão ISO 8601 e classe Instant

- Formato: yyyy-MM-ddTHH:mm:ssZ
- Exemplo: "2018-06-25T15:42:07Z"
- Date y3 = Date.from(Instant.parse("2018-06-25T15:42:07Z"));

```

SimpleDateFormat sdf1 = new SimpleDateFormat("dd/MM/yyyy");
SimpleDateFormat sdf2 = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
SimpleDateFormat sdf3 = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
sdf3.setTimeZone(TimeZone.getTimeZone("GMT"));

Date x1 = new Date();
Date x2 = new Date(System.currentTimeMillis());
Date x3 = new Date(0L);
Date x4 = new Date(1000L * 60L * 60L * 5L);

Date y1 = sdf1.parse("25/06/2018");
Date y2 = sdf2.parse("25/06/2018 15:42:07");
Date y3 = Date.from(Instant.parse("2018-06-25T15:42:07Z"));

System.out.println("x1: " + x1);
System.out.println("x2: " + x2);
System.out.println("x3: " + x3);
System.out.println("x4: " + x4);
System.out.println("y1: " + y1);
System.out.println("y2: " + y2);
System.out.println("y3: " + y3);
System.out.println("-----");
System.out.println("x1: " + sdf2.format(x1));
System.out.println("x2: " + sdf2.format(x2));
System.out.println("x3: " + sdf2.format(x3));
System.out.println("x4: " + sdf2.format(x4));
System.out.println("y1: " + sdf2.format(y1));
System.out.println("y2: " + sdf2.format(y2));
System.out.println("y3: " + sdf2.format(y3));
System.out.println("-----");
System.out.println("x1: " + sdf3.format(x1));
System.out.println("x2: " + sdf3.format(x2));
System.out.println("x3: " + sdf3.format(x3));
System.out.println("x4: " + sdf3.format(x4));
System.out.println("y1: " + sdf3.format(y1));
System.out.println("y2: " + sdf3.format(y2));
System.out.println("y3: " + sdf3.format(y3));

```

*Demo:
Criação e
impressão
de datas.*

Manipulando uma data com Calendar

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Somando uma unidade de tempo

```
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");

Date d = Date.from(Instant.parse("2018-06-25T15:42:07Z"));

System.out.println(sdf.format(d));

Calendar cal = Calendar.getInstance();
cal.setTime(d);
cal.add(Calendar.HOUR_OF_DAY, 4);
d = cal.getTime();

System.out.println(sdf.format(d));
```

Obtendo uma unidade de tempo

```
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");

Date d = Date.from(Instant.parse("2018-06-25T15:42:07Z"));

System.out.println(sdf.format(d));

Calendar cal = Calendar.getInstance();
cal.setTime(d);
int minutes = cal.get(Calendar.MINUTE);
int month = 1 + cal.get(Calendar.MONTH);

System.out.println("Minutes: " + minutes);
System.out.println("Month: " + month);
```



Formação Desenvolvedor Moderno

Módulo: Git e Github

Capítulo: Introdução ao Git e Github

<https://devsuperior.com.br>



1

Favor instalar o Visual Studio Code

<https://code.visualstudio.com/download>

Vídeo instalação Windows:

<https://youtu.be/ZloHacwWjLI>

Vídeo instalação Ubuntu:

<https://youtu.be/THdaC99oNxw>

2

Visão geral de Git e Github

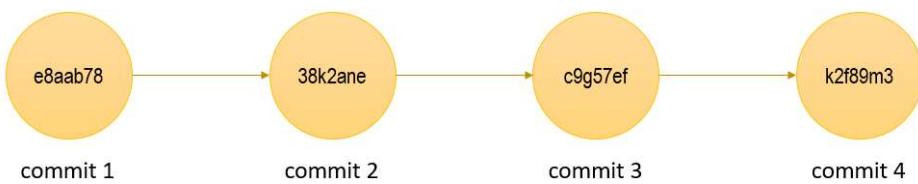
<https://devsuperior.com.br>

Prof. Dr. Nelio Alves

3

Git

GIT - é um sistema de versionamento: você controla as modificações de um projeto por meio de versões chamadas "commits".



4

Github

É um serviço online de hospedagem de repositórios Git remotos.

- Possui uma interface gráfica web: github.com
- É uma plataforma social (usuários, página de perfil, seguidores, colaboração, etc.). Dica: currículo!
- Maior serviço do mundo de hospedagem de projetos de código aberto
- Modelo de cobrança: gratuito para projetos de código aberto, pago para projetos privados
- Alternativas: BitBucket, GitLab, etc.

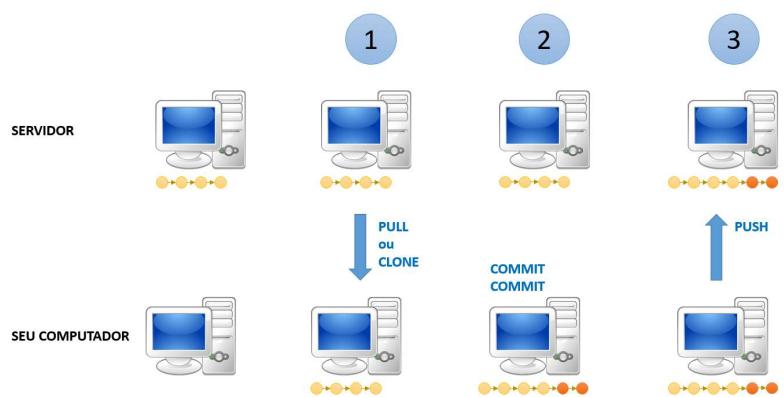
5

Repositório remoto e local

Um projeto controlado pelo Git é chamado de **repositório** de versionamento.

Tipicamente uma cópia "oficial" do repositório fica salvo em um **servidor (repositório remoto)**.

Cada pessoa que trabalha no projeto pode fazer uma cópia do repositório para seu computador (**repositório local**). A pessoa então faz suas alterações no projeto (novos commits) e depois salva as alterações no servidor.

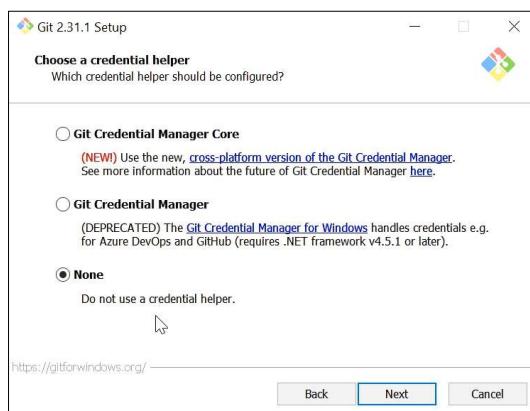


6

Instalação do Git no computador

<https://git-scm.com/downloads>

Importante: não escolha
Gerenciador de credenciais



7

Configurando sua identificação no Git

```
git config --global user.name "Seu nome"  
git config --global user.email "Seu email de cadastro do Github"  
git config --list
```

8

Configuração para ver arquivos ocultos (Windows)

Iniciar -> Opções do explorador de arquivos

DESMARCAR: "Ocultar as extensões dos tipos de arquivos conhecidos"

MARCAR: "Mostrar arquivos, pastas e unidades ocultas"

9

Configurar chave SSH para o Github

SSH é um protocolo para comunicação de dados com segurança.

O Github aboliu a autenticação somente com usuário e senha.

A ideia básica é cadastrar previamente quais computadores podem acessar o Github em seu nome. Outros computadores não conseguem acessar.

Para isto você deve:

- (1) Gerar uma chave SSH no seu computador
- (2) Cadastrar essa chave no seu Github

10

Passo a passo: salvar primeira versão de um projeto no Github

Considerando que agora seu ambiente já está todo configurado (usuário e email, visualização de arquivos ocultos, chave SSH), sempre que você criar um novo projeto, os passos básicos serão estes (troque os parâmetros em azul pelos seus dados):

```
git init  
git add .  
git commit -m "Mensagem explicativa"  
git branch -M main  
git remote add origin git@github.com:seuusuario/seurepositorio.git  
git push -u origin main
```

11

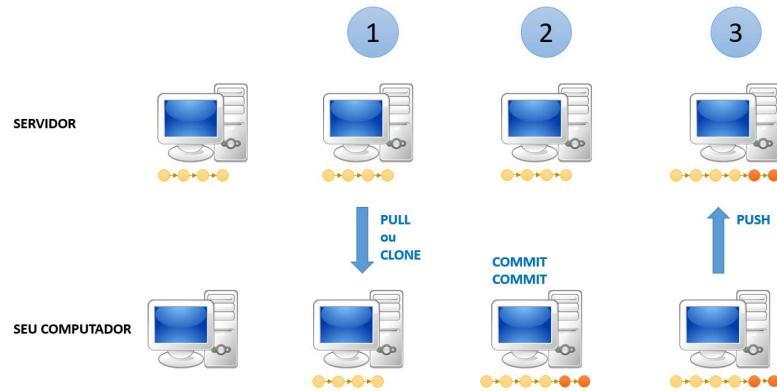
Passo a passo: salvar uma nova versão

```
git status  
git add .  
git commit -m "Mensagem explicativa"  
git push
```

12

Demo: clonar e modificar um projeto de um repositório remoto que você tem permissão para alterar

```
git clone git@github.com:seuusuario/seurepositorio.git  
git add .  
git commit -m "Mensagem explicativa"  
git push
```



13

Verificando o histórico de versões

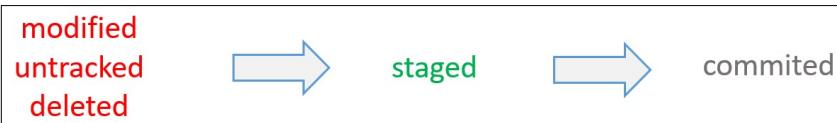
```
git log
```

Listagem resumida:

```
git log --oneline
```

14

Git status, git add e stage



15

Git diff

- Comando que mostra a diferença entre arquivos modificados
- Dica: utilizar o VS Code, que mostra graficamente as diferenças

16

Git checkout

- Permite modificar temporariamente os arquivos do projeto ao estado de um dado commit ou branch
- Código do commit, HEAD
 - Cada commit possui um código, que pode ser utilizado para referenciar o commit
 - O último commit do histórico do branch corrente também pode ser referenciado pela palavra HEAD
 - É possível referenciar um commit N versões antes de HEAD usando ~N, por exemplo:
 - HEAD~1 (penúltimo commit)
 - HEAD~2 (antepenúltimo commit)
- IMPORTANTE: antes de fazer o checkout para voltar para HEAD, certifique-se de que não haja mudanças nos arquivos. Se você acidentalmente mudou alguma coisa, desfaça as modificações usando:

```
git reset  
git clean -df  
git checkout -- .
```

17

Arquivo .gitignore

- É um arquivo que indica o que NÃO deve ser salvo pelo Git.
- Geralmente o arquivo .gitignore fica salvo na pasta principal do repositório. Mas também é possível salvar outros arquivos .gitignore em subpastas do repositório, para indicar o que deve ser ignorado por cada subpasta.

18

Casos comuns de arquivos que não devem ser salvos pelo Git:

- Arquivos compilados

Linguagens compiladas (C, C++, Java, C#, etc.) geram arquivos de código compilado para executar o programa localmente.

- Arquivos de bibliotecas externas usadas no projeto

Projetos reais utilizam bibliotecas externas (programas prontos disponíveis na Internet). Por exemplo, projetos JavaScript com NPM tipicamente salvam uma subpasta "node_modules" na pasta do seu projeto.

- Arquivos de configuração da sua IDE

IDE's podem salvar uma subpasta com arquivos de configuração na pasta do projeto (exemplo: .vscode).

- Arquivos de configuração do seu sistema

Por exemplo, sistemas Mac podem gravar uma subpasta .ds_store na pasta do projeto.

Exemplos de projetos com .gitignore

<https://github.com/acenelio/composition1-java>

<https://github.com/acenelio/dsmovie>



Formação Desenvolvedor Moderno

Módulo: Git e Github

Capítulo: Introdução ao Git e Github

<https://devsuperior.com.br>



1

Favor instalar o Visual Studio Code

<https://code.visualstudio.com/download>

Vídeo instalação Windows:

<https://youtu.be/ZloHacwWjLI>

Vídeo instalação Ubuntu:

<https://youtu.be/THdaC99oNxw>

2

Visão geral de Git e Github

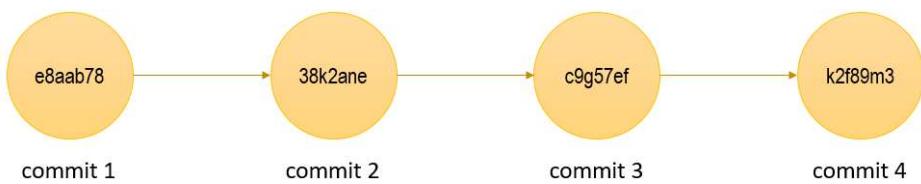
<https://devsuperior.com.br>

Prof. Dr. Nelio Alves

3

Git

GIT - é um sistema de versionamento: você controla as modificações de um projeto por meio de versões chamadas "commits".



4

Github

É um serviço online de hospedagem de repositórios Git remotos.

- Possui uma interface gráfica web: github.com
- É uma plataforma social (usuários, página de perfil, seguidores, colaboração, etc.). Dica: currículo!
- Maior serviço do mundo de hospedagem de projetos de código aberto
- Modelo de cobrança: gratuito para projetos de código aberto, pago para projetos privados
- Alternativas: BitBucket, GitLab, etc.

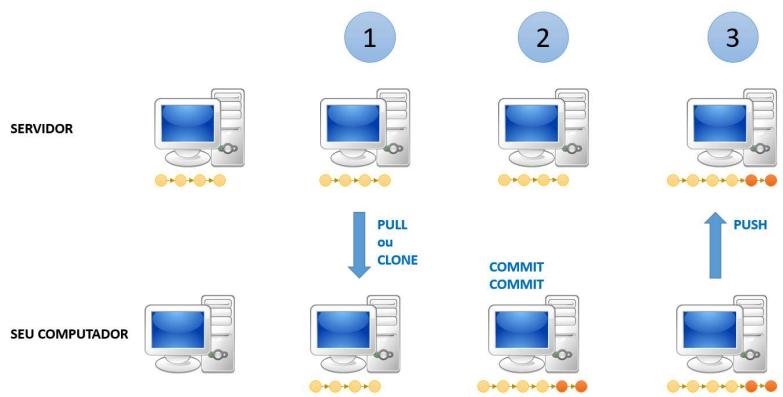
5

Repositório remoto e local

Um projeto controlado pelo Git é chamado de **repositório** de versionamento.

Tipicamente uma cópia "oficial" do repositório fica salvo em um **servidor (repositório remoto)**.

Cada pessoa que trabalha no projeto pode fazer uma cópia do repositório para seu computador (**repositório local**). A pessoa então faz suas alterações no projeto (novos commits) e depois salva as alterações no servidor.

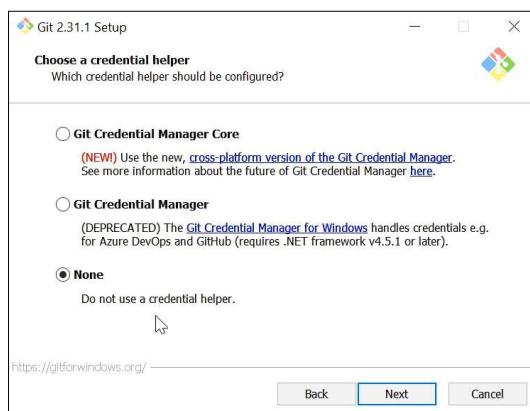


6

Instalação do Git no computador

<https://git-scm.com/downloads>

Importante: não escolha
Gerenciador de credenciais



7

Configurando sua identificação no Git

```
git config --global user.name "Seu nome"  
git config --global user.email "Seu email de cadastro do Github"  
git config --list
```

8

Configuração para ver arquivos ocultos (Windows)

Iniciar -> Opções do explorador de arquivos

DESMARCAR: "Ocultar as extensões dos tipos de arquivos conhecidos"

MARCAR: "Mostrar arquivos, pastas e unidades ocultas"

9

Configurar chave SSH para o Github

SSH é um protocolo para comunicação de dados com segurança.

O Github aboliu a autenticação somente com usuário e senha.

A ideia básica é cadastrar previamente quais computadores podem acessar o Github em seu nome. Outros computadores não conseguem acessar.

Para isto você deve:

- (1) Gerar uma chave SSH no seu computador
- (2) Cadastrar essa chave no seu Github

10

Passo a passo: salvar primeira versão de um projeto no Github

Considerando que agora seu ambiente já está todo configurado (usuário e email, visualização de arquivos ocultos, chave SSH), sempre que você criar um novo projeto, os passos básicos serão estes (troque os parâmetros em azul pelos seus dados):

```
git init  
git add .  
git commit -m "Mensagem explicativa"  
git branch -M main  
git remote add origin git@github.com:seuusuario/seurepositorio.git  
git push -u origin main
```

11

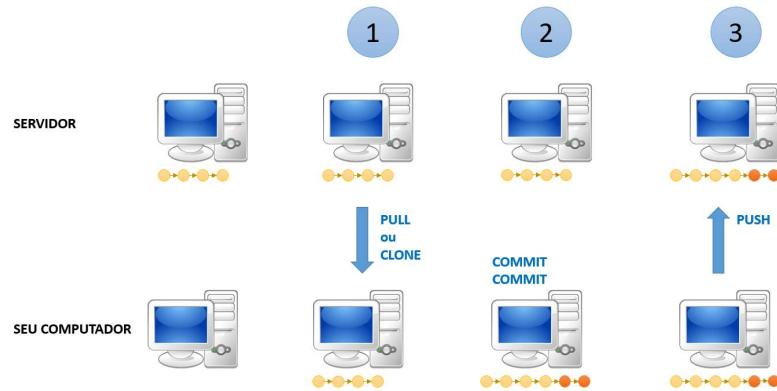
Passo a passo: salvar uma nova versão

```
git status  
git add .  
git commit -m "Mensagem explicativa"  
git push
```

12

Demo: clonar e modificar um projeto de um repositório remoto que você tem permissão para alterar

```
git clone git@github.com:seuusuario/seurepositorio.git  
git add .  
git commit -m "Mensagem explicativa"  
git push
```



13

Verificando o histórico de versões

```
git log
```

Listagem resumida:

```
git log --oneline
```

14

Git status, git add e stage



15

Git diff

- Comando que mostra a diferença entre arquivos modificados
- Dica: utilizar o VS Code, que mostra graficamente as diferenças

16

Git checkout

- Permite modificar temporariamente os arquivos do projeto ao estado de um dado commit ou branch
- Código do commit, HEAD
 - Cada commit possui um código, que pode ser utilizado para referenciar o commit
 - O último commit do histórico do branch corrente também pode ser referenciado pela palavra HEAD
 - É possível referenciar um commit N versões antes de HEAD usando ~N, por exemplo:
 - HEAD~1 (penúltimo commit)
 - HEAD~2 (antepenúltimo commit)
- IMPORTANTE: antes de fazer o checkout para voltar para HEAD, certifique-se de que não haja mudanças nos arquivos. Se você acidentalmente mudou alguma coisa, desfaça as modificações usando:

```
git reset  
git clean -df  
git checkout -- .
```

17

Arquivo .gitignore

- É um arquivo que indica o que NÃO deve ser salvo pelo Git.
- Geralmente o arquivo .gitignore fica salvo na pasta principal do repositório. Mas também é possível salvar outros arquivos .gitignore em subpastas do repositório, para indicar o que deve ser ignorado por cada subpasta.

18

Casos comuns de arquivos que não devem ser salvos pelo Git:

- **Arquivos compilados**

Linguagens compiladas (C, C++, Java, C#, etc.) geram arquivos de código compilado para executar o programa localmente.

- **Arquivos de bibliotecas externas usadas no projeto**

Projetos reais utilizam bibliotecas externas (programas prontos disponíveis na Internet). Por exemplo, projetos JavaScript com NPM tipicamente salvam uma subpasta "node_modules" na pasta do seu projeto.

- **Arquivos de configuração da sua IDE**

IDE's podem salvar uma subpasta com arquivos de configuração na pasta do projeto (exemplo: .vscode).

- **Arquivos de configuração do seu sistema**

Por exemplo, sistemas Mac podem gravar uma subpasta .ds_store na pasta do projeto.

Exemplos de projetos com .gitignore

<https://github.com/acenelio/composition1-java>

<https://github.com/acenelio/dsmovie>



DEVSUPERIOR

Formação Desenvolvedor Moderno Módulo: Git e Github

Capítulo: Resolvendo problemas comuns

<https://devsuperior.com.br>



1

Como remover arquivos da área de stage

```
git status
```

```
git reset
```

2

Como desfazer modificações não salvas

```
git status  
git reset  
git clean -df  
git checkout -- .
```

3

O que fazer quando abre o editor VIM

Estas ações podem abrir o editor VIM no terminal:

- Fazer um commit sem mensagem
- Fazer um merge de três vias

Habilitar o modo de edição:

```
i
```

Sair do VIM, salvando as alterações:

```
<ESC>  
:wq  
<ENTER>
```

Sair do VIM, descartando as alterações:

```
<ESC>  
:q!  
<ENTER>
```

4

Como desfazer o último commit

Desfazer último commit sem desfazer as modificações nos arquivos:

```
git status  
git reset --soft HEAD~1
```

5

Como deletar commits e também modificações nos arquivos

Voltar o projeto ao estado de um dado commit (deletar commits e alterações posteriores a esse commit)

```
git status  
git reset --hard < código do commit >
```

Voltar o projeto ao estado do penúltimo commit:

```
git status  
git reset --hard HEAD~1
```

ATENÇÃO: ação destrutiva!

6

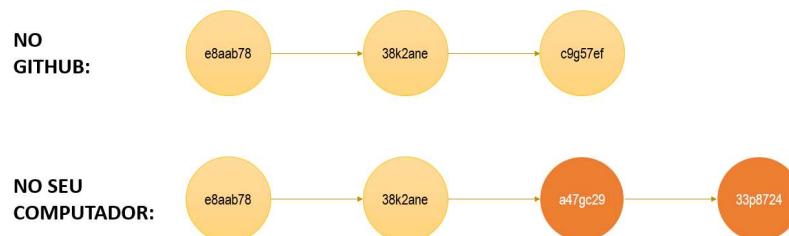
Como atualizar o repositório local em relação ao remoto

```
git status  
git pull <nome do remote> <nome do branch>
```

7

Como resolver push rejeitado

Não é permitido enviar um push se seu repositório local está atrasado em relação ao histórico do repositório remoto! Por exemplo:



Você tem que atualizar o repositório local:

```
git pull <nome do remote> <nome do branch>
```

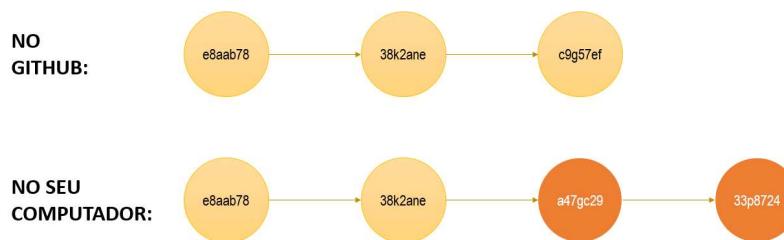
8

Resolvendo conflito

- Analise o código fonte
- Faça as edições necessárias
- Faça um novo commit

9

Como sobrescrever um histórico no Github



```
git push -f <nome do remote> <nome do branch>
```

ATENÇÃO: ação destrutiva!

10

Como apontar o projeto para outro repositório remoto

```
git remote set-url origin git@github.com:seuusuario/seurepositorio.git
```



DEVSUPERIOR

Formação Desenvolvedor Moderno Módulo: Git e Github

Capítulo: Resolvendo problemas comuns

<https://devsuperior.com.br>



1

Como remover arquivos da área de stage

```
git status
```

```
git reset
```

2

Como desfazer modificações não salvas

```
git status  
git reset  
git clean -df  
git checkout -- .
```

3

O que fazer quando abre o editor VIM

Estas ações podem abrir o editor VIM no terminal:

- Fazer um commit sem mensagem
- Fazer um merge de três vias

Habilitar o modo de edição:

```
i
```

Sair do VIM, salvando as alterações:

```
<ESC>  
:wq  
<ENTER>
```

Sair do VIM, descartando as alterações:

```
<ESC>  
:q!  
<ENTER>
```

4

Como desfazer o último commit

Desfazer último commit sem desfazer as modificações nos arquivos:

```
git status  
git reset --soft HEAD~1
```

5

Como deletar commits e também modificações nos arquivos

Voltar o projeto ao estado de um dado commit (deletar commits e alterações posteriores a esse commit)

```
git status  
git reset --hard < código do commit >
```

Voltar o projeto ao estado do penúltimo commit:

```
git status  
git reset --hard HEAD~1
```

ATENÇÃO: ação destrutiva!

6

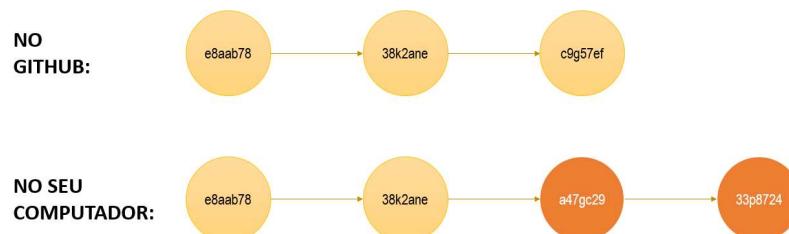
Como atualizar o repositório local em relação ao remoto

```
git status  
git pull <nome do remote> <nome do branch>
```

7

Como resolver push rejeitado

Não é permitido enviar um push se seu repositório local está atrasado em relação ao histórico do repositório remoto! Por exemplo:



Você tem que atualizar o repositório local:

```
git pull <nome do remote> <nome do branch>
```

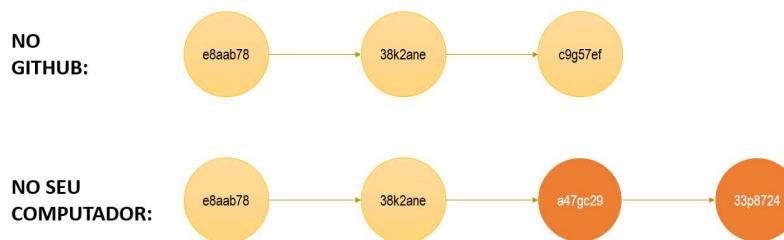
8

Resolvendo conflito

- Analise o código fonte
- Faça as edições necessárias
- Faça um novo commit

9

Como sobrescrever um histórico no Github



```
git push -f <nome do remote> <nome do branch>
```

ATENÇÃO: ação destrutiva!

10

Como apontar o projeto para outro repositório remoto

```
git remote set-url origin git@github.com:seuusuario/seurepositorio.git
```

Curso Programação Orientada a Objetos com Java

Capítulo: Enumerações, composição

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Enumerações

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Checklist

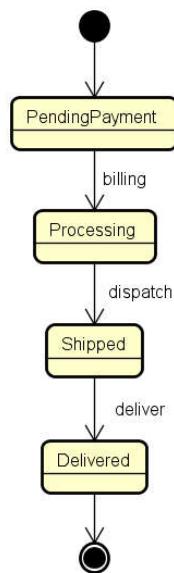
- Definição / discussão
- Exemplo: estados de um pedido
- Conversão de string para enum
- Representação UML

Enumerações

- É um tipo especial que serve para especificar de forma literal um conjunto de constantes relacionadas
- Palavra chave em Java: enum
- Vantagem: melhor semântica, código mais legível e auxiliado pelo compilador
- Referência: <https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html>

Exemplo

Ciclo de vida de um pedido.



```

package entities.enums;

public enum OrderStatus {
    PENDING_PAYMENT,
    PROCESSING,
    SHIPPED,
    DELIVERED;
}
  
```

```

package entities;

import java.util.Date;

import entities.enums.OrderStatus;

public class Order {
    private Integer id;
    private Date moment;
    private OrderStatus status;

    (...)

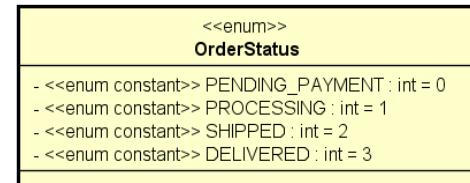
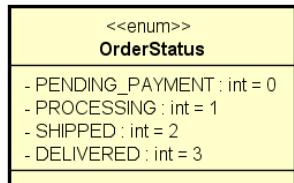
  
```

<https://github.com/acenelio/enum1-java>

Conversão de String para enum

```
OrderStatus os1 = OrderStatus.DELIVERED;  
OrderStatus os2 = OrderStatus.valueOf("DELIVERED");
```

Notação UML



Vamos falar um pouco de design

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Categorias de classes

- Em um sistema orientado a objetos, de modo geral "tudo" é objeto.
- Por questões de design tais como organização, flexibilidade, reuso, delegação, etc., há várias categorias de classes:

Views

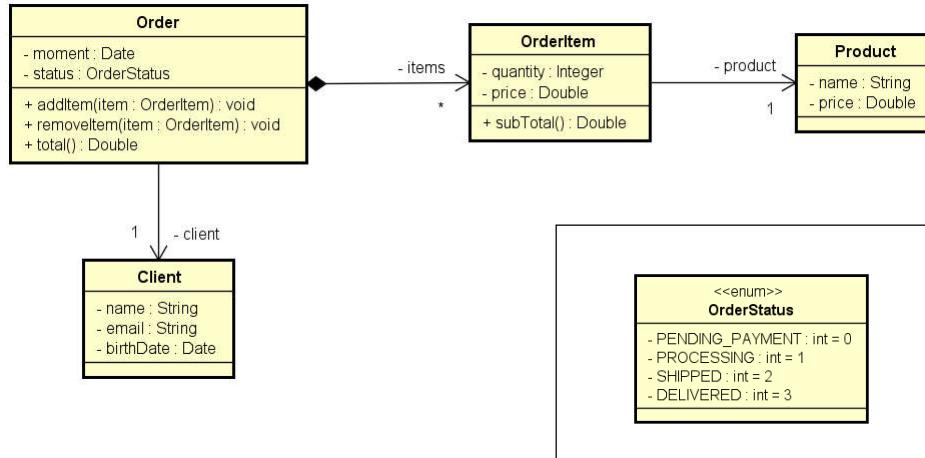
Controllers

Entities

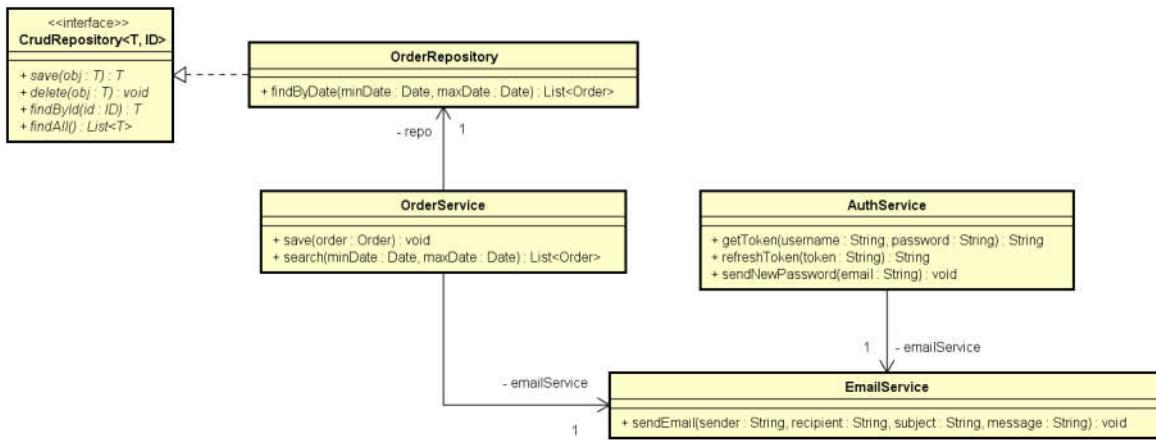
Services

Repositories

Entities



Services



Composição

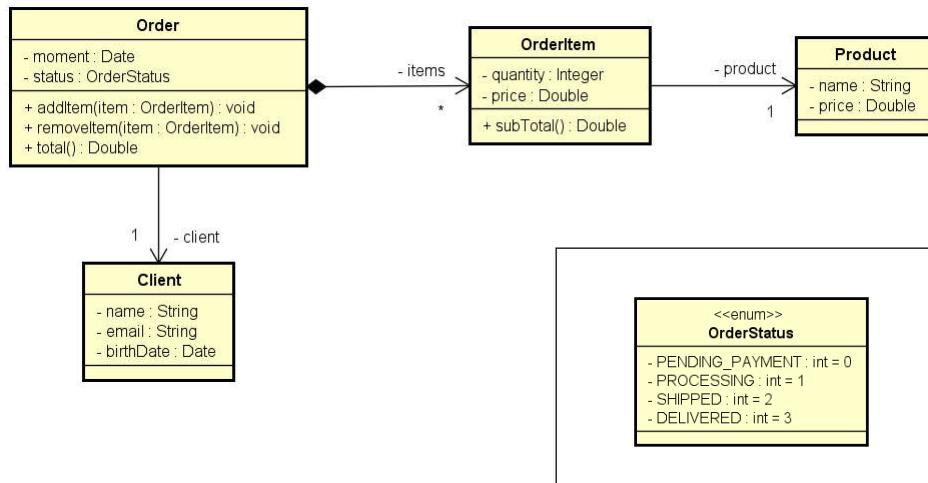
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

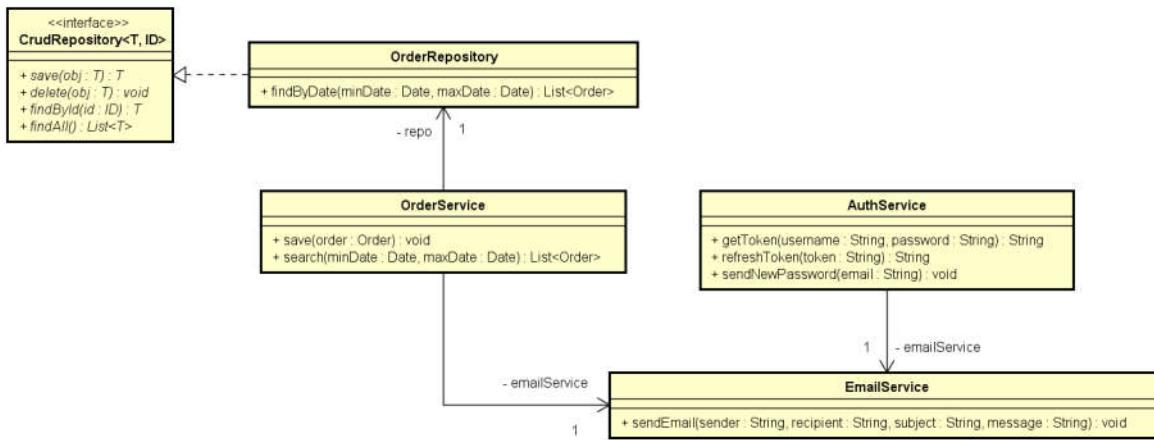
Composição

- É um tipo de associação que permite que um objeto contenha outro
- Relação "tem-um" ou "tem-vários"
- Vantagens
 - Organização: divisão de responsabilidades
 - Coesão
 - Flexibilidade
 - Reuso
- Nota: embora o símbolo UML para composição (todo-parte) seja o diamante preto, neste contexto estamos chamando de composição qualquer associação tipo "tem-um" e "tem-vários".

Entities



Services

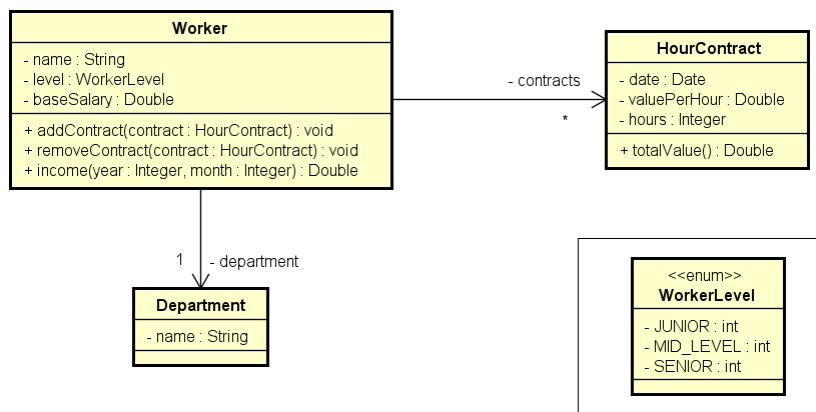


Exercício resolvido 1

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Ler os dados de um trabalhador com N contratos (N fornecido pelo usuário). Depois, solicitar do usuário um mês e mostrar qual foi o salário do funcionário nesse mês, conforme exemplo (próxima página).



```

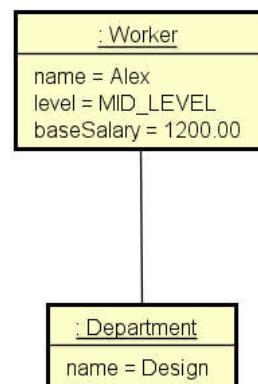
Enter department's name: Design
Enter worker data:
Name: Alex
Level: MID_LEVEL
Base salary: 1200.00
How many contracts to this worker? 3
Enter contract #1 data:
Date (DD/MM/YYYY): 20/08/2018
Value per hour: 50.00
Duration (hours): 20
Enter contract #2 data:
Date (DD/MM/YYYY): 13/06/2018
Value per hour: 30.00
Duration (hours): 18
Enter contract #3 data:
Date (DD/MM/YYYY): 25/08/2018
Value per hour: 80.00
Duration (hours): 10

Enter month and year to calculate income (MM/YYYY): 08/2018
Name: Alex
Department: Design
Income for 08/2018: 3000.00

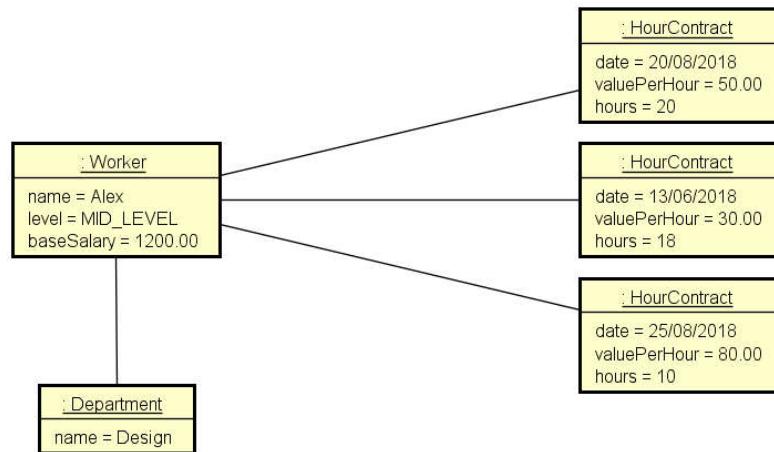
```

<https://github.com/acenelio/composition1-java>

Objects in memory:



Objects in memory:

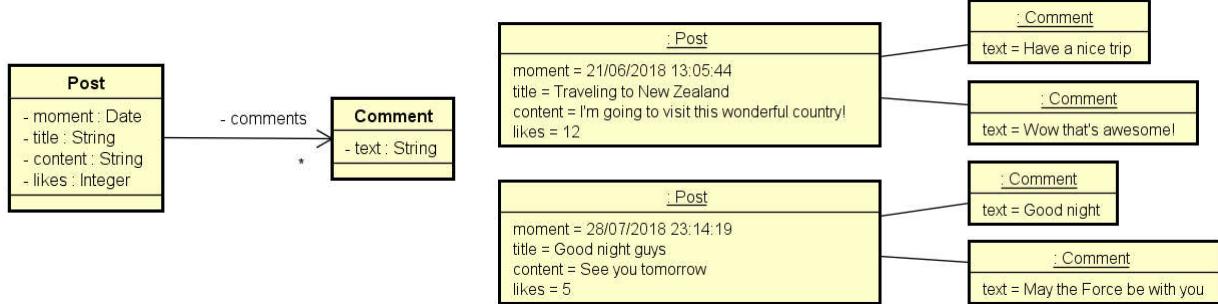


Exercício resolvido 2 (demo
StringBuilder)

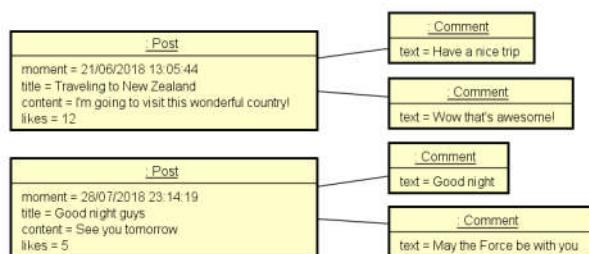
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Instancie manualmente (hard code) os objetos mostrados abaixo e mostre-os na tela do terminal, conforme exemplo.



Console output:



```

Traveling to New Zealand
12 Likes - 21/06/2018 13:05:44
I'm going to visit this wonderful country!
Comments:
Have a nice trip
Wow that's awesome!

Good night guys
5 Likes - 28/07/2018 23:14:19
See you tomorrow
Comments:
Good night
May the Force be with you

```

The console output displays the posts and their associated comments. The first post, titled 'Traveling to New Zealand', has 12 likes and two comments: 'Have a nice trip' and 'Wow that's awesome!'. The second post, titled 'Good night guys', has 5 likes and two comments: 'Good night' and 'May the Force be with you'.

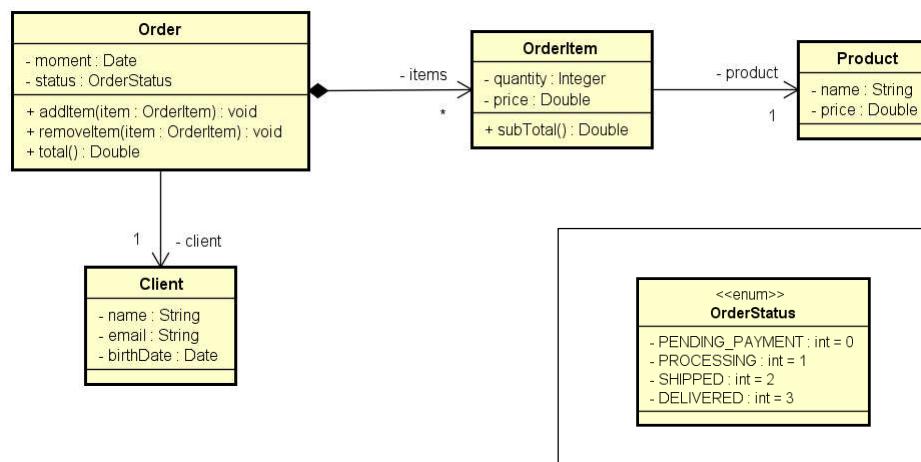
<https://github.com/acenelio/composition2-java>

Exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Ler os dados de um pedido com N itens (N fornecido pelo usuário). Depois, mostrar um sumário do pedido conforme exemplo (próxima página). Nota: o instante do pedido deve ser o instante do sistema: new Date()



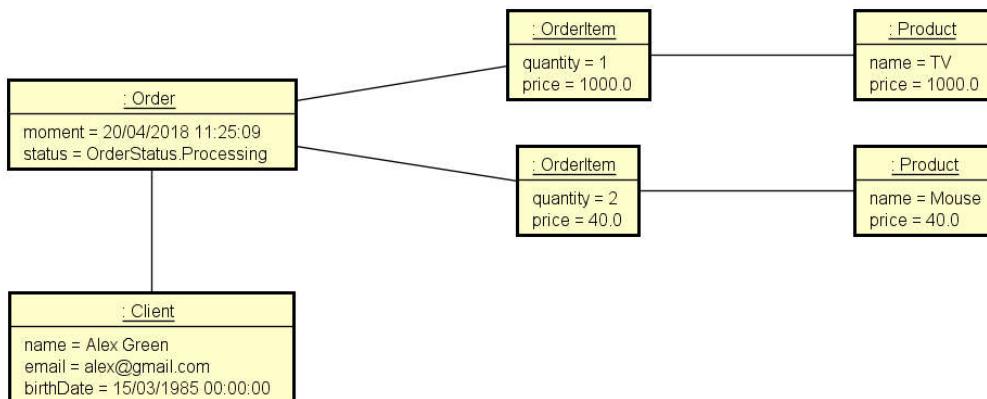
```

Enter cliente data:
Name: Alex Green
Email: alex@gmail.com
Birth date (DD/MM/YYYY): 15/03/1985
Enter order data:
Status: PROCESSING
How many items to this order? 2
Enter #1 item data:
Product name: TV
Product price: 1000.00
Quantity: 1
Enter #2 item data:
Product name: Mouse
Product price: 40.00
Quantity: 2

ORDER SUMMARY:
Order moment: 20/04/2018 11:25:09
Order status: PROCESSING
Client: Alex Green (15/03/1985) - alex@gmail.com
Order items:
TV, $1000.00, Quantity: 1, Subtotal: $1000.00
Mouse, $40.00, Quantity: 2, Subtotal: $80.00
Total price: $1080.00

```

Você deverá instanciar os objetos em memória da seguinte forma:



<https://github.com/acenelio/composition3-java>

Curso Programação Orientada a Objetos com Java

Capítulo: Herança e polimorfismo

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Herança

<http://educandoweb.com.br>

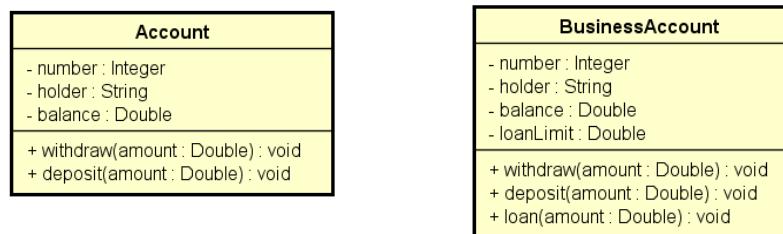
Prof. Dr. Nelio Alves

Herança

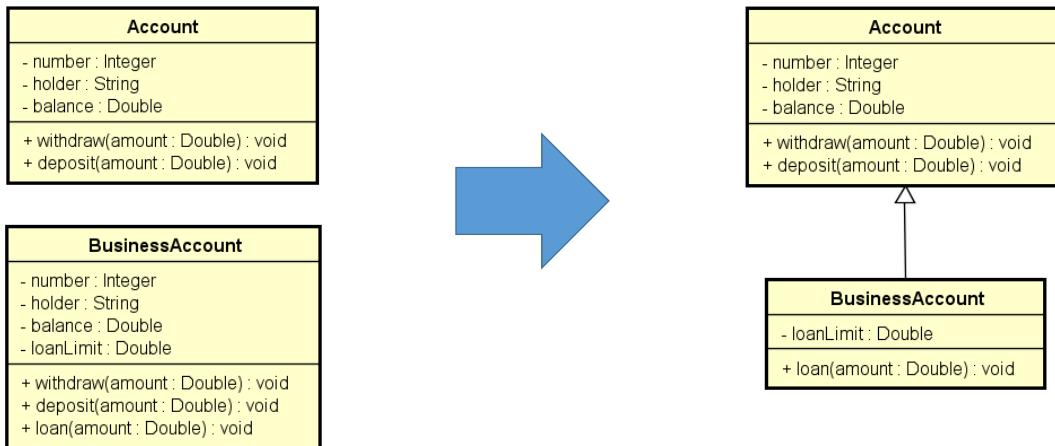
- É um tipo de associação que permite que uma classe herde *todos* dados e comportamentos de outra
- Definições importantes
- Vantagens
 - Reuso
 - Polimorfismo
- Sintaxe
 - class A extends B

Exemplo

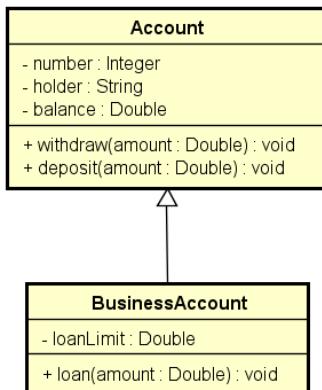
Suponha um negócio de banco que possui uma conta comum e uma conta para empresas, sendo que a conta para empresa possui todos membros da conta comum, mais um limite de empréstimo e uma operação de realizar empréstimo.



Herança permite o reuso de atributos e métodos (dados e comportamento)



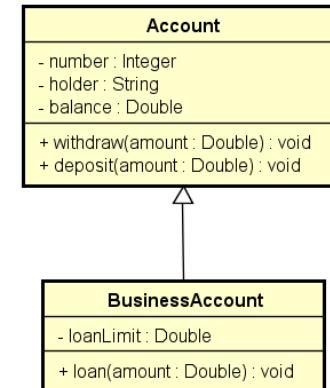
Definições importantes



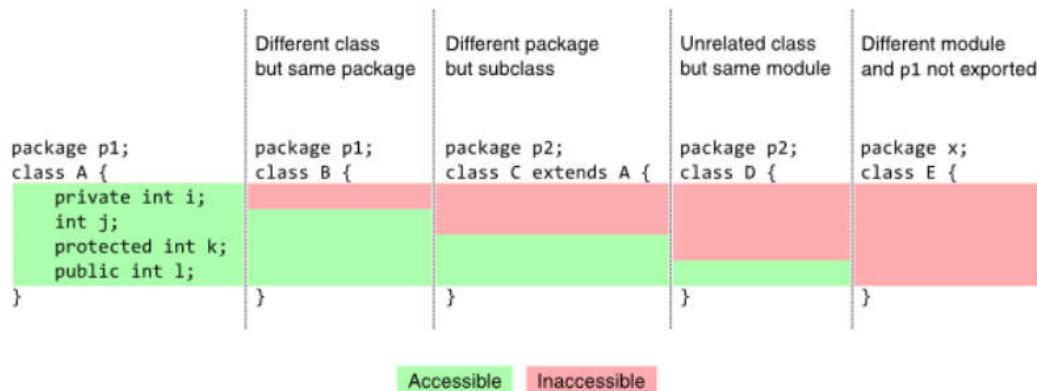
- Relação "é-um"
- Generalização/especialização
- Superclasse (classe base) / subclasse (classe derivada)
- Herança / extensão
- Herança é uma associação entre classes (e não entre objetos)

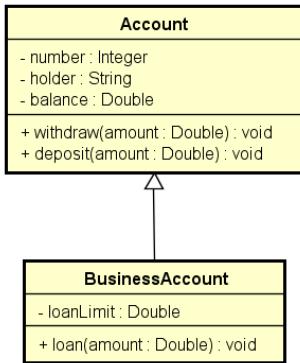
Demo

Vamos implementar as classes Account e BusinessAccount e fazer alguns testes.



Modificador de acesso protected





Suponha que, para realizar um empréstimo, é descontada uma taxa no valor de 10.0

Isso resulta em erro:

```
public void loan(double amount) {
    if (amount <= loanLimit) {
        balance += amount - 10.0;
    }
}
```

<https://github.com/acenelio/inheritance1-java>

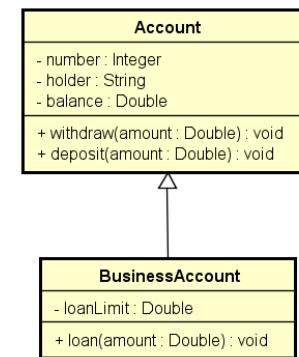
Upcasting e downcasting

<http://educandoweb.com.br>

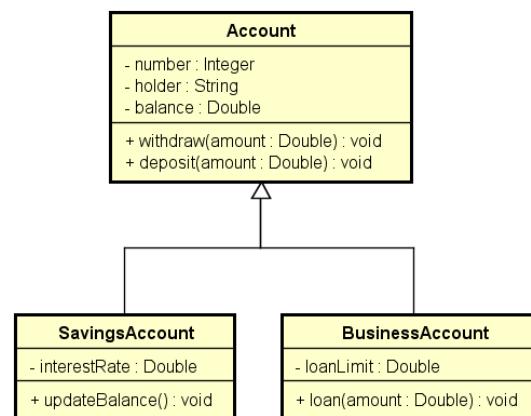
Prof. Dr. Nelio Alves

Checklist

- Upcasting
 - Casting da subclasse para superclasse
 - Uso comum: polimorfismo
- Downcasting
 - Casting da superclasse para subclasse
 - Palavra instanceof
 - Uso comum: métodos que recebem parâmetros genéricos (ex: Equals)



Example



<https://github.com/acenelio/inheritance2-java>

```
Account acc = new Account(1001, "Alex", 0.0);
BusinessAccount bacc = new BusinessAccount(1002, "Maria", 0.0, 500.0);

// UPCASTING

Account acc1 = bacc;
Account acc2 = new BusinessAccount(1003, "Bob", 0.0, 200.0);
Account acc3 = new SavingsAccount(1004, "Anna", 0.0, 0.01);

// DOWNCASTING

BusinessAccount acc4 = (BusinessAccount)acc2;
acc4.loan(100.0);

// BusinessAccount acc5 = (BusinessAccount)acc3;
if (acc3 instanceof BusinessAccount) {
    BusinessAccount acc5 = (BusinessAccount)acc3;
    acc5.loan(200.0);
    System.out.println("Loan!");
}

if (acc3 instanceof SavingsAccount) {
    SavingsAccount acc5 = (SavingsAccount)acc3;
    acc5.updateBalance();
    System.out.println("Update!");
}
```

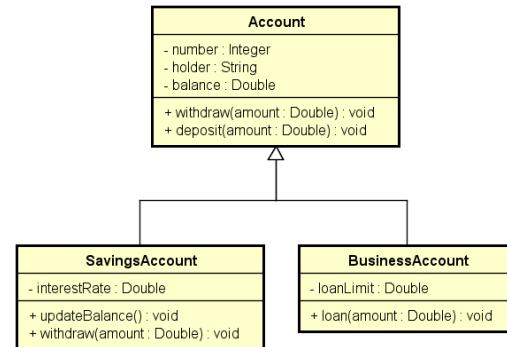
Sobreposição, palavra super,
anotação @Override

<http://educandoweb.com.br>

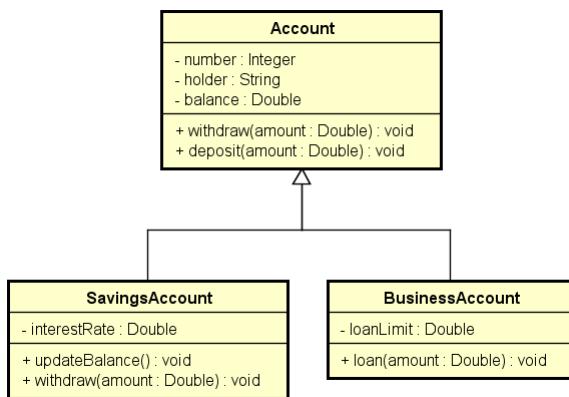
Prof. Dr. Nelio Alves

Sobreposição ou sobrescrita

- É a implementação de um método de uma superclasse na subclasse
- É fortemente recomendável usar a anotação @Override em um método sobreescrito
 - Facilita a leitura e compreensão do código
 - Avisamos ao compilador (boa prática)



Exemplo



Suponha que a operação de saque possui uma taxa no valor de 5.0. Entretanto, se a conta for do tipo poupança, esta taxa não deve ser cobrada.

Como resolver isso?

Resposta: sobrescrevendo o método withdraw na subclasse SavingsAccount

Account:

```
public void withdraw(double amount) {  
    balance -= amount + 5.0;  
}
```

SavingsAccount:

```
@Override  
public void withdraw(double amount) {  
    balance -= amount;  
}
```

Palavra super

É possível chamar a implementação da superclasse usando a palavra **super**.

Exemplo: suponha que, na classe BusinessAccount, a regra para saque seja realizar o saque normalmente da superclasse, e descontar mais 2.0.

```
@Override  
public void withdraw(double amount) {  
    super.withdraw(amount);  
    balance -= 2.0;  
}
```

Recordando: usando `super` em construtores

```
public class Account {  
    private Integer number;  
    private String holder;  
    private Double balance;  
  
    public Account(Integer number, String holder, Double balance) {  
        this.number = number;  
        this.holder = holder;  
        this.balance = balance;  
    }  
    (...)  
  
public class BusinessAccount extends Account {  
    private double loanLimit;  
  
    public BusinessAccount(Integer number, String holder, Double balance, double loanLimit) {  
        super(number, holder, balance);  
        this.loanLimit = loanLimit;  
    }  
    (...)
```

Código fonte desta aula

<https://github.com/acenelio/inheritance3-java>

Classes e métodos final

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Classes e métodos final

- Palavra chave: **final**
- **Classe**: evita que a classe seja herdada

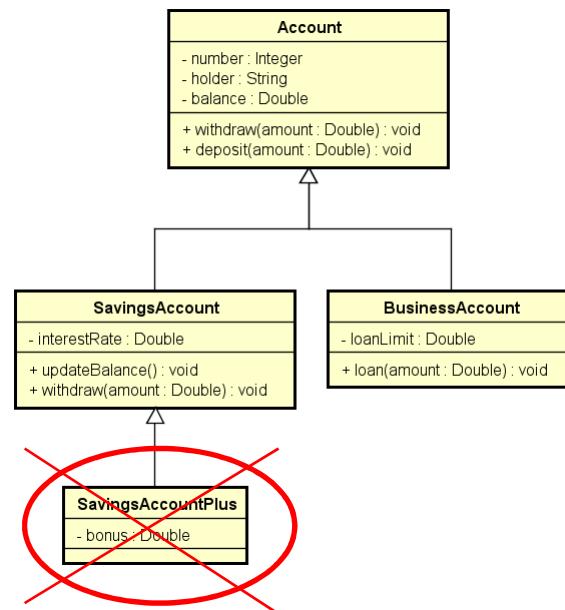
```
public final class SavingsAccount {
```

- **Método**: evita que o método sob seja sobreposto

Exemplo - Classe final

Suponha que você queira evitar que sejam criadas subclasses de SavingsAccount

```
public final class SavingsAccount {  
    (...)
```



Exemplo - método final

Suponha que você não queira que o método Withdraw de SavingsAccount seja sobreposto

```
@Override  
public final void withdraw(double amount) {  
    balance -= amount;  
}
```

Pra quê?

- Segurança: dependendo das regras do negócio, às vezes é desejável garantir que uma classe não seja herdada, ou que um método não seja sobreposto.
 - Geralmente convém acrescentar **final** em métodos sobrepostos, pois sobreposições múltiplas podem ser uma porta de entrada para inconsistências
- Performance: atributos de tipo de uma classe final são analisados de forma mais rápida em tempo de execução.
 - Exemplo clássico: String

Introdução ao polimorfismo

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Pilares da OOP

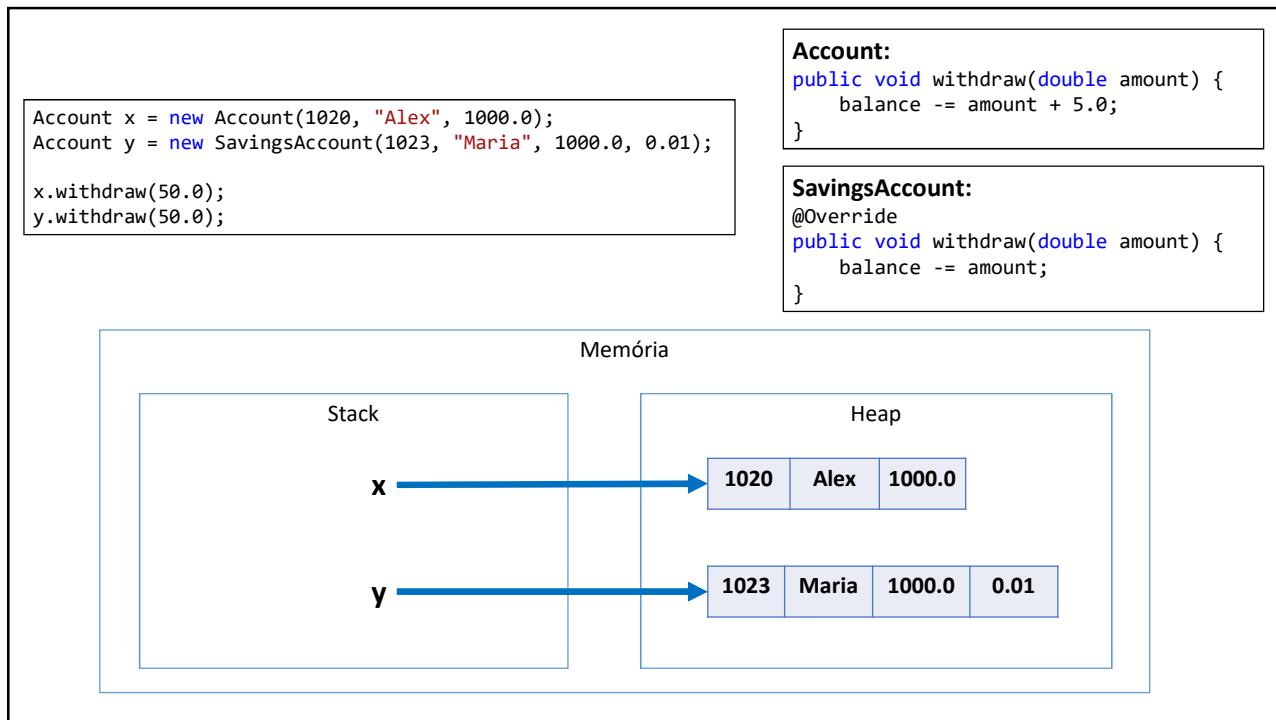
- Encapsulamento
- Herança
- Polimorfismo

Polimorfismo

Em Programação Orientada a Objetos, polimorfismo é recurso que permite que variáveis de um mesmo tipo mais genérico possam apontar para objetos de tipos específicos diferentes, tendo assim comportamentos diferentes conforme cada tipo específico.

```
Account x = new Account(1020, "Alex", 1000.0);
Account y = new SavingsAccount(1023, "Maria", 1000.0, 0.01);

x.withdraw(50.0);
y.withdraw(50.0);
```



Importante entender

- A associação do tipo específico com o tipo genérico é feita em tempo de execução (upcasting).
- O compilador não sabe para qual tipo específico a chamada do método Withdraw está sendo feita (ele só sabe que são duas variáveis tipo Account):

```
Account x = new Account(1020, "Alex", 1000.0);
Account y = new SavingsAccount(1023, "Maria", 1000.0, 0.01);

x.withdraw(50.0);
y.withdraw(50.0);
```

Exercício resolvido

<http://educandoweb.com.br>

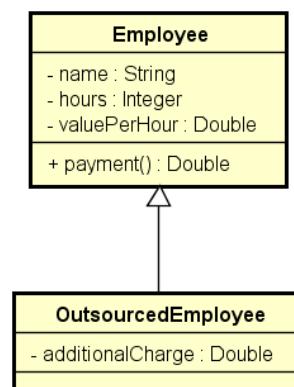
Prof. Dr. Nelio Alves

Uma empresa possui funcionários próprios e terceirizados. Para cada funcionário, deseja-se registrar nome, horas trabalhadas e valor por hora. Funcionários terceirizado possuem ainda uma despesa adicional.

O pagamento dos funcionários corresponde ao valor da hora multiplicado pelas horas trabalhadas, sendo que os funcionários terceirizados ainda recebem um bônus correspondente a 110% de sua despesa adicional.

Fazer um programa para ler os dados de N funcionários (N fornecido pelo usuário) e armazená-los em uma lista. Depois de ler todos os dados, mostrar nome e pagamento de cada funcionário na mesma ordem em que foram digitados.

Construa o programa conforme projeto ao lado. Veja exemplo na próxima página.



```
Enter the number of employees: 3
```

```
Employee #1 data:
```

```
Outsourced (y/n)? n
```

```
Name: Alex
```

```
Hours: 50
```

```
Value per hour: 20.00
```

```
Employee #2 data:
```

```
Outsourced (y/n)? y
```

```
Name: Bob
```

```
Hours: 100
```

```
Value per hour: 15.00
```

```
Additional charge: 200.00
```

```
Employee #3 data:
```

```
Outsourced (y/n)? n
```

```
Name: Maria
```

```
Hours: 60
```

```
Value per hour: 20.00
```

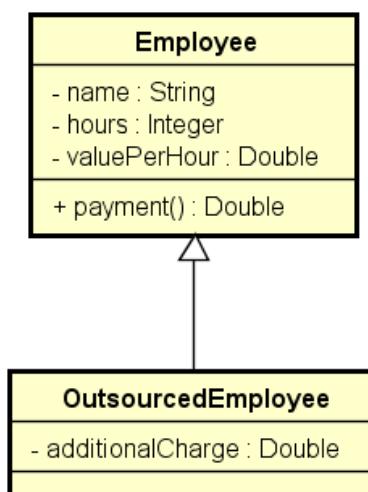
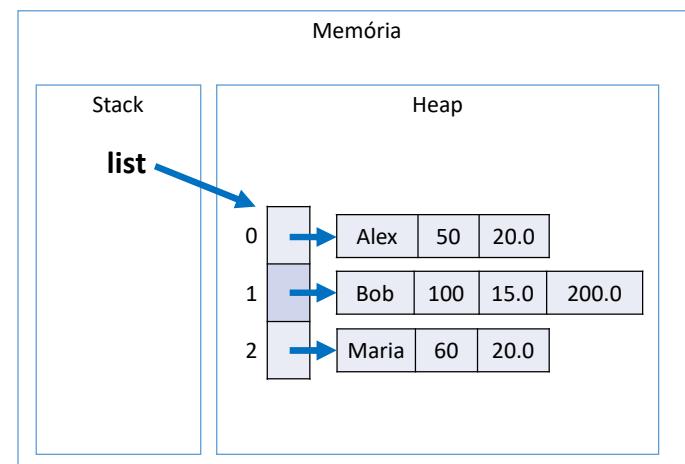
```
PAYMENTS:
```

```
Alex - $ 1000.00
```

```
Bob - $ 1720.00
```

```
Maria - $ 1200.00
```

<https://github.com/acenelio/inheritance4-java>



Exercício de fixação

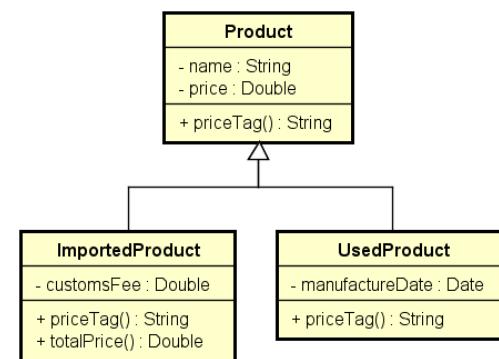
<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Fazer um programa para ler os dados de N produtos (N fornecido pelo usuário). Ao final, mostrar a etiqueta de preço de cada produto na mesma ordem em que foram digitados.

Todo produto possui nome e preço. Produtos importados possuem uma taxa de alfândega, e produtos usados possuem data de fabricação. Estes dados específicos devem ser acrescentados na etiqueta de preço conforme exemplo (próxima página). Para produtos importados, a taxa e alfândega deve ser acrescentada ao preço final do produto.

Favor implementar o programa conforme projeto ao lado.

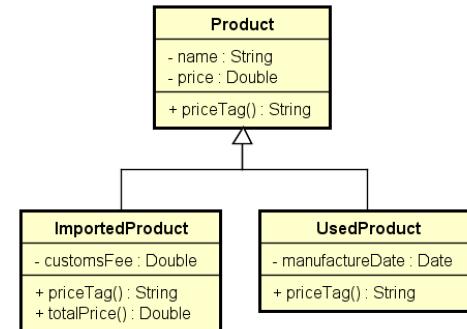


```

Enter the number of products: 3
Product #1 data:
Common, used or imported (c/u/i)? i
Name: Tablet
Price: 260.00
Customs fee: 20.00
Product #2 data:
Common, used or imported (c/u/i)? c
Name: Notebook
Price: 1100.00
Product #3 data:
Common, used or imported (c/u/i)? u
Name: Iphone
Price: 400.00
Manufacture date (DD/MM/YYYY): 15/03/2017

PRICE TAGS:
Tablet $ 280.00 (Customs fee: $ 20.00)
Notebook $ 1100.00
Iphone (used) $ 400.00 (Manufacture date: 15/03/2017)

```



<https://github.com/acenelio/inheritance5-java>

Classes abstratas

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Classes abstratas

- São classes que não podem ser instanciadas
- É uma forma de garantir herança total: somente subclasses não abstratas podem ser instanciadas, mas nunca a superclasse abstrata

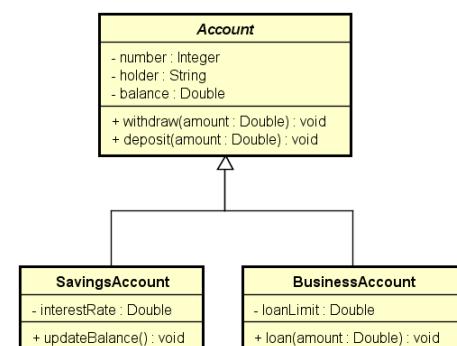
Exemplo

Suponha que em um negócio relacionado a banco, apenas contas poupança e contas para empresas são permitidas. Não existe conta comum.

Para garantir que contas comuns não possam ser instanciadas, basta acrescentarmos a palavra "abstract" na declaração da classe.

```
public abstract class Account {
    (...)
```

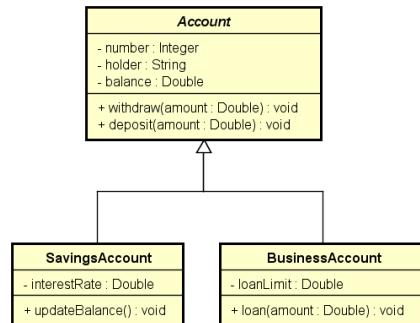
Notação UML: itálico



Vamos partir da implementação em: <https://github.com/acenelio/inheritance3-java>

Questionamento

- Se a classe Account não pode ser instanciada, por que simplesmente não criar somente SavingsAccount e BusinessAccount?
- Resposta:
 - **Reuso**
 - **Polimorfismo:** a superclasse classe genérica nos permite tratar de forma fácil e uniforme todos os tipos de conta, inclusive com polimorfismo se for o caso (como fizemos nos últimos exercícios). Por exemplo, você pode colocar todos tipos de contas em uma mesma coleção.
- Demo: suponha que você queira:
 - Totalizar o saldo de todas as contas.
 - Depositar 10.00 em todas as contas.



<https://github.com/acenelio/inheritance6-java>

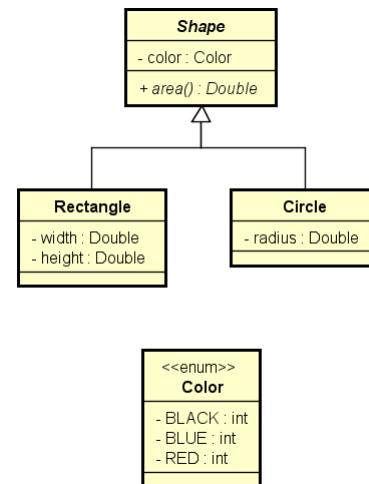
Métodos abstratos

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Métodos abstratos

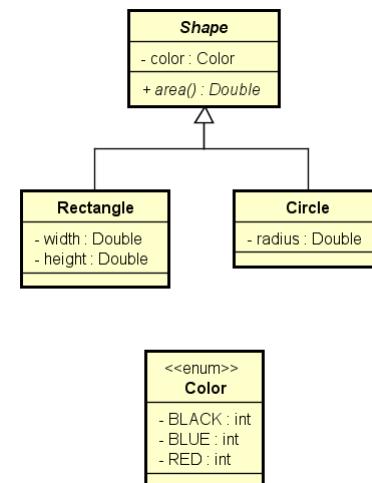
- São métodos que não possuem implementação.
- Métodos precisam ser abstratos quando a classe é genérica demais para conter sua implementação.
- Se uma classe possuir pelo menos um método abstrato, então esta classe também é abstrata.
- Notação UML: itálico
- Exercício resolvido



Fazer um programa para ler os dados de N figuras (N fornecido pelo usuário), e depois mostrar as áreas destas figuras na mesma ordem em que foram digitadas.

```

Enter the number of shapes: 2
Shape #1 data:
Rectangle or Circle (r/c)? r
Color (BLACK/BLUE/RED): BLACK
Width: 4.0
Height: 5.0
Shape #2 data:
Rectangle or Circle (r/c)? c
Color (BLACK/BLUE/RED): RED
Radius: 3.0
SHAPE AREAS:
20.00
28.27
  
```



<https://github.com/acenelio/inheritance7-java>

Exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Fazer um programa para ler os dados de N contribuintes (N fornecido pelo usuário), os quais podem ser pessoa física ou pessoa jurídica, e depois mostrar o valor do imposto pago por cada um, bem como o total de imposto arrecadado.

Os dados de pessoa física são: nome, renda anual e gastos com saúde. Os dados de pessoa jurídica são nome, renda anual e número de funcionários. As regras para cálculo de imposto são as seguintes:

Pessoa física: pessoas cuja renda foi abaixo de 20000.00 pagam 15% de imposto. Pessoas com renda de 20000.00 em diante pagam 25% de imposto. Se a pessoa teve gastos com saúde, 50% destes gastos são abatidos no imposto.

Exemplo: uma pessoa cuja renda foi 50000.00 e teve 2000.00 em gastos com saúde, o imposto fica: $(50000 * 25\%) - (2000 * 50\%) = \mathbf{11500.00}$

Pessoa jurídica: pessoas jurídicas pagam 16% de imposto. Porém, se a empresa possuir mais de 10 funcionários, ela paga 14% de imposto.

Exemplo: uma empresa cuja renda foi 400000.00 e possui 25 funcionários, o imposto fica: $400000 * 14\% = \mathbf{56000.00}$

```

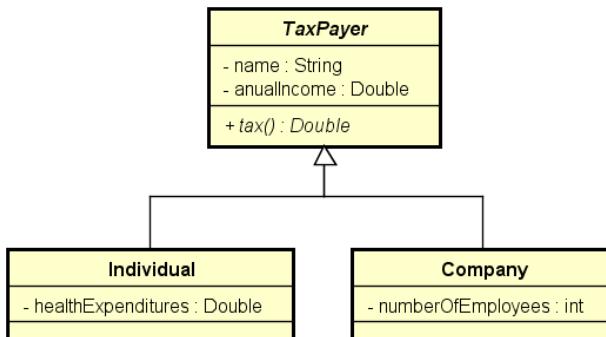
Enter the number of tax payers: 3
Tax payer #1 data:
Individual or company (i/c)? i
Name: Alex
Anual income: 50000.00
Health expenditures: 2000.00
Tax payer #2 data:
Individual or company (i/c)? c
Name: SoftTech
Anual income: 400000.00
Number of employees: 25
Tax payer #3 data:
Individual or company (i/c)? i
Name: Bob
Anual income: 120000.00
Health expenditures: 1000.00

TAXES PAID:
Alex: $ 11500.00
SoftTech: $ 56000.00
Bob: $ 29500.00

TOTAL TAXES: $ 97000.00

```

Correção



<https://github.com/acenelio/inheritance8-java>

Curso: Programação Orientada a Objetos com Java

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Capítulo: Projeto Sistema de Jogo de Xadrez

Objetivo geral:

- Aplicar os conhecimentos aprendidos até o momento no curso para a construção de um projeto

System design

<https://github.com/acenelio/chess-system-design>

Creating project and git repository

Checklist:

- Github: create a new project
 - **NOTE:** choose `.gitignore` type as Java
- Open a terminal in project folder, and perform the following commands:

```
git init
git remote add origin https://github.com/acenelio/chess-system-java.git
git pull origin master
git add .
git commit -m "Project created"
git push -u origin master
```

First class: Position

Checklist:

- Class Position [public]
- **OOP Topics:**
 - Encapsulation
 - Constructors
 - `ToString` (Object / overriding)

Starting to implement Board and Piece

Checklist:

- Classes Piece, Board [public]
- **OOP Topics:**
 - Associations
 - Encapsulation / Access Modifiers
- **Data Structures Topics:**
 - Matrix

Chess layer and printing the board

```
8 - - - - - - -  
7 - - - - - - -  
6 - - - - - - -  
5 - - - - - - -  
4 - - - - - - -  
3 - - - - - - -  
2 - - - - - - -  
1 - - - - - - -  
a b c d e f g h
```

Checklist:

- Methods: Board.Piece(row, column) and Board.Piece(position)
- Enum Chess.Color
- Class Chess.ChessPiece [public]
- Class Chess.ChessMatch [public]
- Class ChessConsole.UI
- **OOP Topics:**
 - Enumerations
 - Encapsulation / Access Modifiers
 - Inheritance
 - Downcasting
 - Static members
 - Layers pattern
- **Data Structures Topics:**
 - Matrix

Placing pieces on the board

Checklist:

- Method: Board.PlacePiece(piece, position)
- Classes: Rook, King [public]
- Method: ChessMatch.InitialSetup
- **OOP Topics:**
 - Inheritance
 - Overriding
 - Polymorphism (ToString)

BoardException and defensive programming

Checklist:

- Class BoardException [public]
- Methods: Board.PositionExists, Board.TherelsAPiece
- Implement defensive programming in Board methods
- **OOP Topics:**
 - Exceptions
 - Constructors (a string must be informed to the exception)

ChessException and ChessPosition

Checklist:

- Class ChessException [public]
- Class ChessPosition [public]
- Refactor ChessMatch.InitialSetup
- **OOP Topics:**
 - Exceptions
 - Encapsulation
 - Constructors (a string must be informed to the exception)
 - Overriding
 - Static members
 - Layers pattern

Little improvement in board printing

Color in terminal:

- Windows: Git Bash
- Mac: Google "osx terminal color"

Checklist:

- Place more pieces on the board
- Distinguish piece colors in UI.PrintPiece method

Moving pieces

Checklist:

- Method Board.RemovePiece
- Method UI.ReadChessPosition
- Method ChessMatch.PerformChessMove
 - Method ChessMatch.MakeMove
 - Method ChessMatch.ValidadeSourcePosition
- Write basic logic on Program.cs
- **OOP Topics:**
 - Exceptions
 - Encapsulation

Handling exceptions and clearing screen

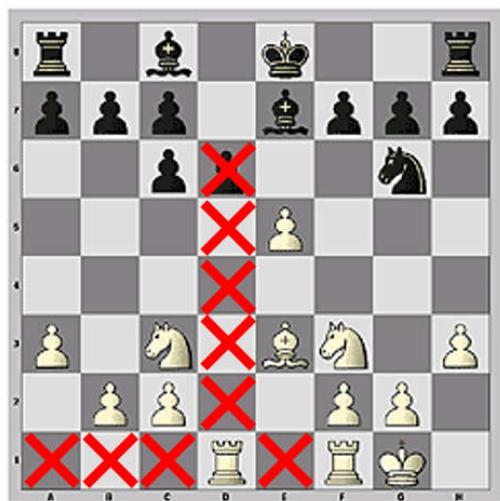
Clear screen using Java:

```
// https://stackoverflow.com/questions/2979383/java-clear-the-console
public static void clearScreen() {
    System.out.print("\033[H\033[2J");
    System.out.flush();
}
```

Checklist:

- ChessException
- InputMismatchException

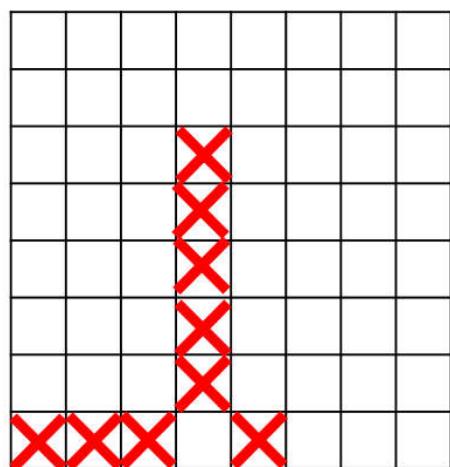
Possible moves of a piece



Input: a piece



Output: a boolean matrix of possible movements



Checklist:

- Methods in Piece:
 - PossibleMoves [abstract]
 - PossibleMove
 - IsThereAnyPossibleMove
 - Basic PossibleMove implementation for Rook and King
 - Update ChessMatch.ValidadeSourcePosition
 - **OOP Topics:**
 - Abstract method / class
 - Exceptions

Implementing possible moves of Rook

Checklist:

- Method ChessPiece.IsThereOpponentPiece(position) [protected]
- Implement Rook.PossibleMoves
- Method ChessMatch.ValidateTargetPosition
- **OOP Topics:**
 - Polymorphism
 - Encapsulation / access modifiers [protected]
 - Exceptions

Printing possible moves

Checklist:

- Method ChessMatch.PossibleMoves
- Method UI.PrintBoard [overload]
- Refactor main program logic
- **OOP Topics:**
 - Overloading

Implementing possible moves of King

Checklist:

- Method King.CanMove(position) [private]
- Implement King.PossibleMoves
- **OOP Topics:**
 - Encapsulation
 - Polymorphism

Switching player each turn

Checklist:

- Class ChessMatch:
 - Properties Turn, CurrentPlayer [private set]
 - Method NextTurn [private]
 - Update PerformChessMove
 - Update ValidadeSourcePosition
- Method UI.PrintMatch
- **OOP Topics:**
 - Encapsulation
 - Exceptions

Handling captured pieces

Checklist:

- Method UI.PrintCapturedPieces
- Update UI.PrintMatch
- Update Program logic
- Lists in ChessMatch: _piecesOnTheBoard, _capturedPieces
 - Update constructor
 - Update PlaceNewPiece
 - Update MakeMove
- **OOP Topics:**
 - Encapsulation
 - Constructors
- **Data Structures Topics:**
 - List

Check logic

Rules:

- Check means your king is under threat by at least one opponent piece
- You can't put yourself in check

Checklist:

- Property ChessPiece.ChessPosition [get]
- Class ChessMatch:
 - Method UndoMove
 - Property Check [private set]
 - Method Opponent [private]
 - Method King(color) [private]
 - Method TestCheck
 - Update PerformChessMove
- Update UI.PrintMatch

Checkmate logic

Checklist:

- Class ChessMatch:
 - Property Checkmate [private set]
 - Method TestCheckmate [private]
 - Update PerformChessMove
- Update UI.PrintMatch
- Update Program logic

Piece move count

Checklist:

- Class ChessPiece:
 - Property MoveCount [private set]
 - Method IncreaseMoveCount [internal]
 - Method DecreaseMoveCount [internal]
- Class ChessMatch:
 - Update MakeMove
 - Update UndoMove
- **OOP Topics:**
 - Encapsulation

Pawn

Checklist:

- Class Pawn
- Update ChessMatch.InitialSetup
- **OOP Topics:**
 - Encapsulation
 - Inheritance
 - Polymorphism

Bishop

Checklist:

- Class Bishop
- Update ChessMatch.InitialSetup
- **OOP Topics:**
 - Encapsulation
 - Inheritance
 - Polymorphism

Knight

Checklist:

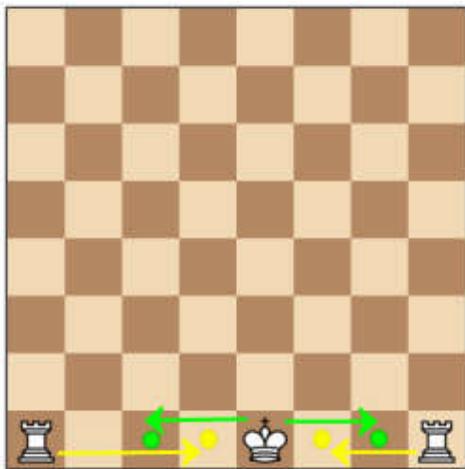
- Class Knight
- Update ChessMatch.InitialSetup
- **OOP Topics:**
 - Encapsulation
 - Inheritance
 - Polymorphism

Queen

Checklist:

- Class Queen
- Update ChessMatch.InitialSetup
- **OOP Topics:**
 - Encapsulation
 - Inheritance
 - Polymorphism

Special move - Castling



Checklist:

- Update King
- Update ChessMatch.MakeMove
- Update ChessMatch.UndoMove

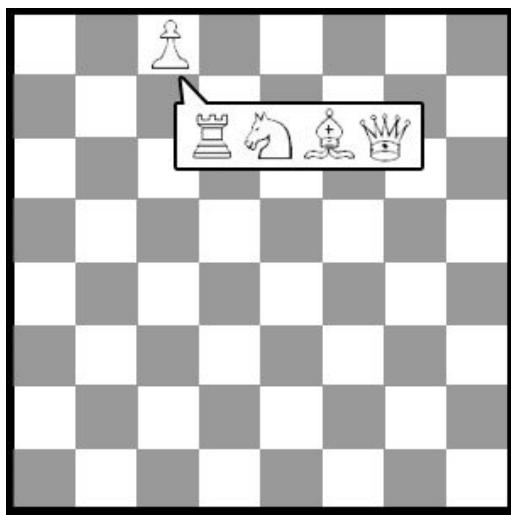
Special move - En Passant



Checklist:

- Register a pawn which can be captured by en passant on next turn
 - Property ChessMatch.EnPassantVulnerable
 - Update ChessMatch.PerformChessMove
- Update Pawn.PossibleMoves
- Update ChessMatch.MakeMove
- Update ChessMatch.UndoMove
- Update ChessMatch.InitialSetup

Special move - Promotion



Checklist:

- Property ChessMatch.Promoted
- Update ChessMatch.PerformChessMove
- Method ChessMatch.ReplacePromotedPiece
- Update Program logic

Curso Programação Orientada a Objetos com Java

Capítulo: Trabalhando com arquivos

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Lendo arquivo texto com classes File e Scanner

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Classes

- File - Representação abstrata de um arquivo e seu caminho
 - <https://docs.oracle.com/javase/10/docs/api/java/io/File.html>
- Scanner - Leitor de texto
 - <https://docs.oracle.com/javase/10/docs/api/java/util/Scanner.html>
- IOException (Exception)
 - <https://docs.oracle.com/javase/10/docs/api/java/io/IOException.html>

```
package application;

import java.io.File;
import java.io.IOException;
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {

        File file = new File("C:\\\\temp\\\\in.txt");
        Scanner sc = null;
        try {
            sc = new Scanner(file);
            while (sc.hasNextLine()) {
                System.out.println(sc.nextLine());
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
        finally {
            if (sc != null) {
                sc.close();
            }
        }
    }
}
```

FileReader e BufferedReader

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Classes

- FileReader (stream de leitura de caracteres a partir de arquivos)
 - <https://docs.oracle.com/javase/10/docs/api/java/io/FileReader.html>
- BufferedReader (mais rápido)
 - <https://docs.oracle.com/javase/10/docs/api/java/io/BufferedReader.html>
 - <https://stackoverflow.com/questions/9648811/specific-difference-between-bufferedreader-and-filereader>

```
package application;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class Program {

    public static void main(String[] args) {

        String path = "C:\\temp\\in.txt";
        BufferedReader br = null;
        FileReader fr = null;

        try {
            fr = new FileReader(path);
            br = new BufferedReader(fr);

            String line = br.readLine();

            while (line != null) {
                System.out.println(line);
                line = br.readLine();
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            try {
                if (br != null)
                    br.close();
                if (fr != null)
                    fr.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Bloco try-with-resources

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Bloco try-with-resources

- É um bloco try que declara um ou mais recursos, e garante que esses recursos serão fechados ao final do bloco
- Disponível no Java 7 em diante
- <https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html>

```
package application;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class Program {

    public static void main(String[] args) {
        String path = "C:\\\\temp\\\\in.txt";

        try (BufferedReader br = new BufferedReader(new FileReader(path))) {
            String line = br.readLine();
            while (line != null) {
                System.out.println(line);
                line = br.readLine();
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

FileWriter e BufferedWriter

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Classes

- **FileWriter** (stream de escrita de caracteres em de arquivos)
 - <https://docs.oracle.com/javase/10/docs/api/java/io/FileWriter.html>
 - Cria/recria o arquivo: **new FileWriter(path)**
 - Acrescenta ao arquivo existente: **new FileWriter(path, true)**
- **BufferedWriter** (mais rápido)
 - <https://docs.oracle.com/javase/10/docs/api/java/io/BufferedWriter.html>

```
package application;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

public class Program {

    public static void main(String[] args) {

        String[] lines = new String[] { "Good morning", "Good afternoon", "Good night" };

        String path = "C:\\temp\\out.txt";

        try (BufferedWriter bw = new BufferedWriter(new FileWriter(path))) {

            for (String line : lines) {
                bw.write(line);
                bw.newLine();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Manipulando pastas com File

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

```
package application;

import java.io.File;
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter a folder path: ");
        String strPath = sc.nextLine();

        File path = new File(strPath);

        File[] folders = path.listFiles(File::isDirectory);
        System.out.println("FOLDERS:");
        for (File folder : folders) {
            System.out.println(folder);
        }

        File[] files = path.listFiles(File::isFile);
        System.out.println("FILES:");
        for (File file : files) {
            System.out.println(file);
        }

        boolean success = new File(strPath + "\\subdir").mkdir();
        System.out.println("Directory created successfully: " + success);

        sc.close();
    }
}
```

Informações do caminho do arquivo

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

```
package application;

import java.io.File;
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter a folder path: ");
        String strPath = sc.nextLine();

        File path = new File(strPath);

        System.out.println("getPath: " + path.getPath());
        System.out.println("getParent: " + path.getParent());
        System.out.println("getName: " + path.getName());

        sc.close();
    }
}
```

Exercício proposto

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Fazer um programa para ler o caminho de um arquivo .csv contendo os dados de itens vendidos. Cada item possui um nome, preço unitário e quantidade, separados por vírgula. Você deve gerar um novo arquivo chamado "summary.csv", localizado em uma subpasta chamada "out" a partir da pasta original do arquivo de origem, contendo apenas o nome e o valor total para aquele item (preço unitário multiplicado pela quantidade), conforme exemplo.

Example:

Source file:

```
TV LED,1290.99,1  
Video Game Chair,350.50,3  
Iphone X,900.00,2  
Samsung Galaxy 9,850.00,2
```

Output file (out/summary.csv):

```
TV LED,1290.99  
Video Game Chair,1051.50  
Iphone X,1800.00  
Samsung Galaxy 9,1700.00
```

<https://github.com/acenelio/files1-java>



Java e Orientação a Objetos

Capítulo: Interfaces

<https://devsuperior.com.br>

Prof. Dr. Nelio Alves

1

Aviso

- A partir do Java 8, interfaces podem ter "default methods" ou "defender methods"
- Isso possui implicações conceituais e práticas, que serão discutidas mais à frente neste capítulo
- Primeiro vamos trabalhar com a definição "clássica" de interfaces. Depois vamos acrescentar o conceito de default methods.

2

Interface

Interface é um tipo que define um conjunto de operações que uma classe deve implementar.

A interface estabelece um **contrato** que a classe deve cumprir.

```
interface Shape {  
    double area();  
    double perimeter();  
}
```

Pra quê interfaces?

- Para criar sistemas com **baixo acoplamento e flexíveis**.

3

Problema exemplo

Uma locadora brasileira de carros cobra um valor por hora para locações de até 12 horas. Porém, se a duração da locação ultrapassar 12 horas, a locação será cobrada com base em um valor diário. Além do valor da locação, é acrescido no preço o valor do imposto conforme regras do país que, no caso do Brasil, é 20% para valores até 100.00, ou 15% para valores acima de 100.00. Fazer um programa que lê os dados da locação (modelo do carro, instante inicial e final da locação), bem como o valor por hora e o valor diário de locação. O programa deve então gerar a nota de pagamento (contendo valor da locação, valor do imposto e valor total do pagamento) e informar os dados na tela. Veja os exemplos.

4

Example 1:

Entre com os dados do aluguel
Modelo do carro: **Civic**
Retirada (dd/MM/yyyy hh:mm): **25/06/2018 10:30**
Retorno (dd/MM/yyyy hh:mm): **25/06/2018 14:40**
Entre com o preço por hora: **10.00**
Entre com o preço por dia: **130.00**
FATURA:
Pagamento basico: 50.00
Imposto: 10.00
Pagamento total: 60.00

Cálculos:

$$\text{Duração} = (25/06/2018 14:40) - (25/06/2018 10:30) = 4:10 = 5 \text{ horas}$$
$$\text{Pagamento básico} = 5 * 10 = 50$$

$$\text{Imposto} = 50 * 20\% = 50 * 0.2 = 10$$

5

Example 2:

Entre com os dados do aluguel
Modelo do carro: **Civic**
Retirada (dd/MM/yyyy hh:mm): **25/06/2018 10:30**
Retorno (dd/MM/yyyy hh:mm): **27/06/2018 11:40**
Entre com o preço por hora: **10.00**
Entre com o preço por dia: **130.00**
FATURA:
Pagamento basico: 390.00
Imposto: 58.50
Pagamento total: 448.50

Cálculos:

$$\text{Duração} = (27/06/2018 11:40) - (25/06/2018 10:30) = 2 \text{ days} + 1:10 = 3 \text{ days}$$
$$\text{Pagamento básico} = 3 * 130 = 390$$

$$\text{Imposto} = 390 * 15\% = 390 * 0.15 = 58.50$$

6

Solução do problema

educandoweb.com.br

Prof. Dr. Nelio Alves

7

(recordando - cap. Composição)

Views

Controllers

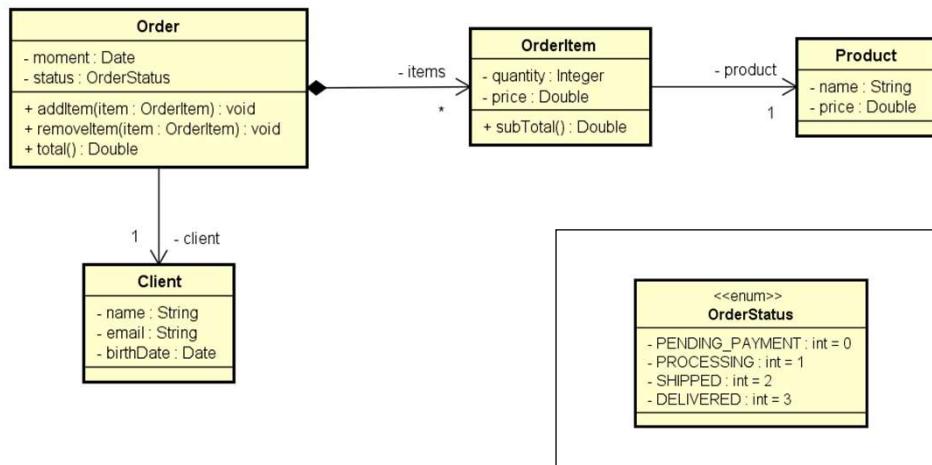
Entities

Services

Repositories

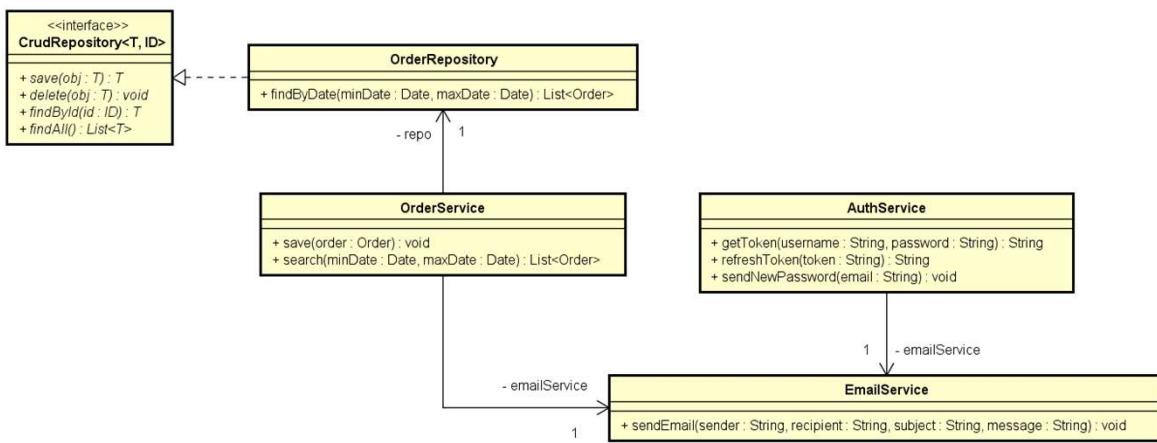
8

Entities



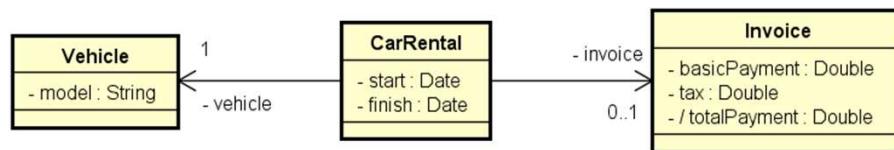
9

Services



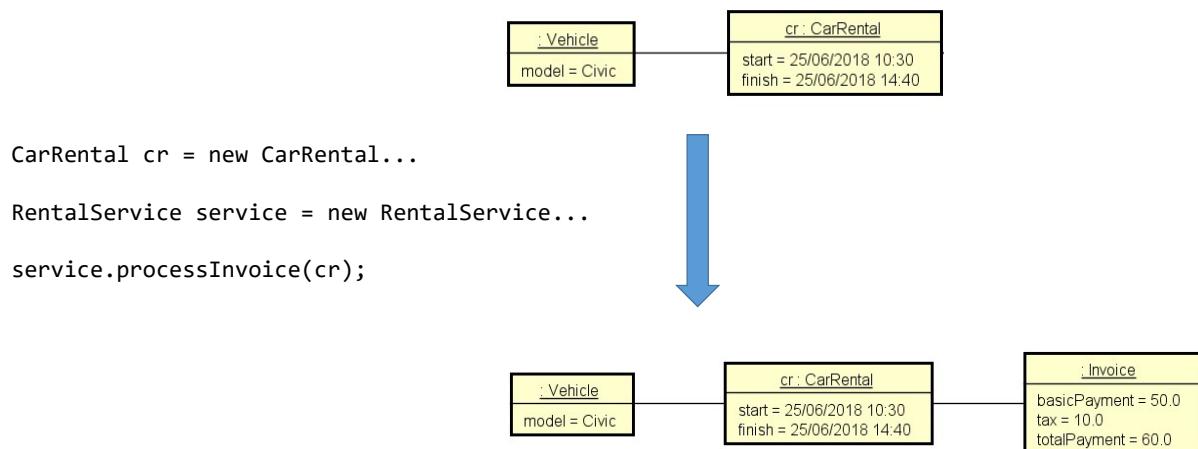
10

Domain layer design



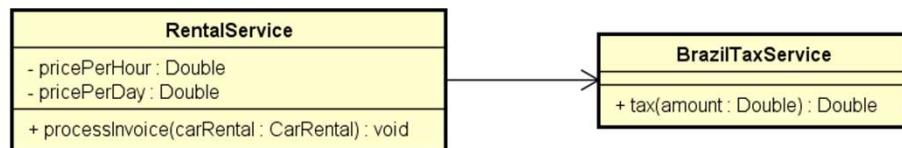
11

Agora precisamos criar um serviço para gerar a fatura



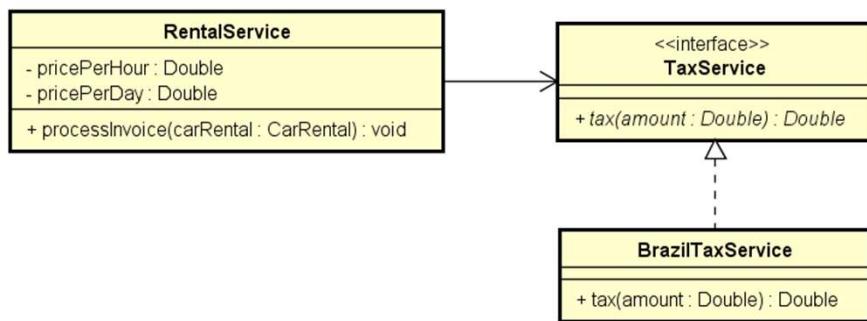
12

Service layer design (no interface)



13

Service layer design



14

Projeto no Github

<https://github.com/acenelio/interfaces1-java>

15

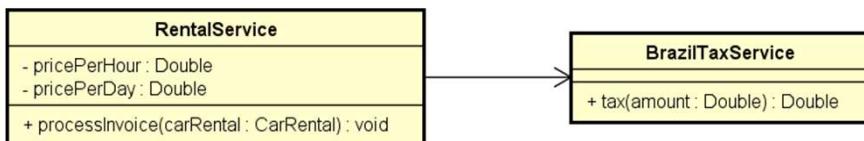
Inversão de controle, Injeção de dependência

educandoweb.com.br

Prof. Dr. Nelio Alves

16

- Acoplamento forte
- A classe RentalService conhece a dependência concreta
- Se a classe concreta mudar, é preciso mudar a classe RentalService

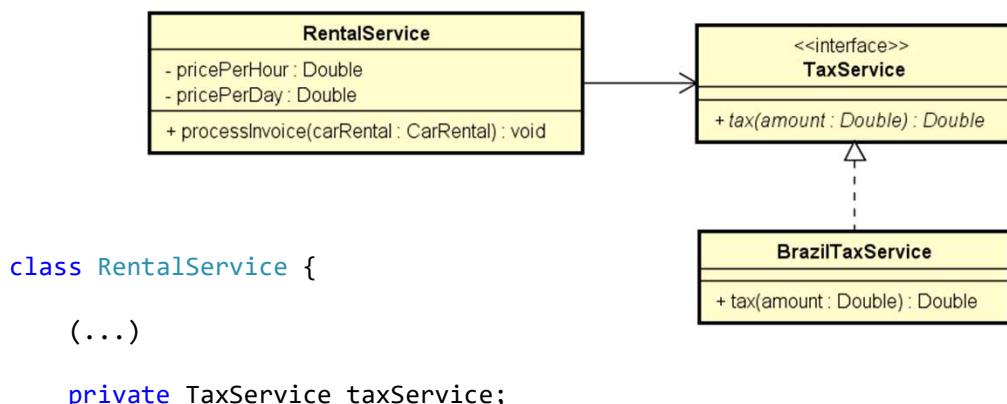


```

class RentalService {
    ...
    private BrazilTaxService taxService;
}
  
```

17

- Acoplamento fraco
- A classe RentalService não conhece a dependência concreta
- Se a classe concreta mudar, a classe RentalService não muda nada



18

Injeção de dependência por meio de construtor

```
class Program {
    static void Main(string[] args) {
        (...)

        RentalService rentalService = new RentalService(pricePerHour, pricePerDay, new BrazilTaxService());
```

A red arrow points from the line 'new BrazilTaxService()' in the first code block down to the declaration 'private TaxService taxService;' in the second code block. The word 'upcasting' is written in red next to the arrow.

```
class RentalService {
    private TaxService taxService;

    public RentalService(double pricePerHour, double pricePerDay, TaxService taxService) {
        this.pricePerHour = pricePerHour;
        this.pricePerDay = pricePerDay;
        this.taxService = taxService;
    }
}
```

19

Inversão de controle

- **Inversão de controle**

Padrão de desenvolvimento que consiste em retirar da classe a responsabilidade de instanciar suas dependências.

- **Injeção de dependência**

É uma forma de realizar a inversão de controle: um componente externo instancia a dependência, que é então injetada no objeto "pai". Pode ser implementada de várias formas:

- Construtor
- Classe de instanciação (builder / factory)
- Container / framework

20

Exercício de fixação

educandoweb.com.br

Prof. Dr. Nelio Alves

21

Uma empresa deseja automatizar o processamento de seus contratos. O processamento de um contrato consiste em gerar as parcelas a serem pagas para aquele contrato, com base no número de meses desejado.

A empresa utiliza um serviço de pagamento online para realizar o pagamento das parcelas. Os serviços de pagamento online tipicamente cobram um juro mensal, bem como uma taxa por pagamento. Por enquanto, o serviço contratado pela empresa é o do Paypal, que aplica **juros simples** de 1% a cada parcela, mais uma **taxa** de pagamento de 2%.

Fazer um programa para ler os dados de um contrato (número do contrato, data do contrato, e valor total do contrato). Em seguida, o programa deve ler o número de meses para parcelamento do contrato, e daí gerar os registros de parcelas a serem pagas (data e valor), sendo a primeira parcela a ser paga um mês após a data do contrato, a segunda parcela dois meses após o contrato e assim por diante. Mostrar os dados das parcelas na tela.

Veja exemplo na próxima página.

22

Example:

Entre os dados do contrato:

Numero: **8028**

Data (dd/MM/yyyy): **25/06/2018**

Valor do contrato: **600.00**

Entre com o numero de parcelas: **3**

Parcelas:

25/07/2018 - 206.04

25/08/2018 - 208.08

25/09/2018 - 210.12

Cálculos (1% juro simples mensal + 2% taxa de pagamento):

Parcela #1:

$$200 + 1\% * 1 = 202$$

$$202 + 2\% = 206.04$$

Parcela #2:

$$200 + 1\% * 2 = 204$$

$$204 + 2\% = 208.08$$

Parcela #3:

$$200 + 1\% * 3 = 206$$

$$206 + 2\% = 210.12$$

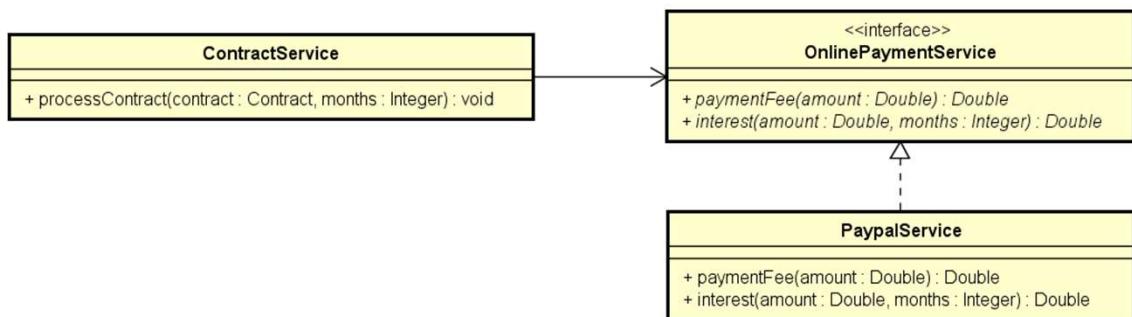
23

Domain layer design (entities)



24

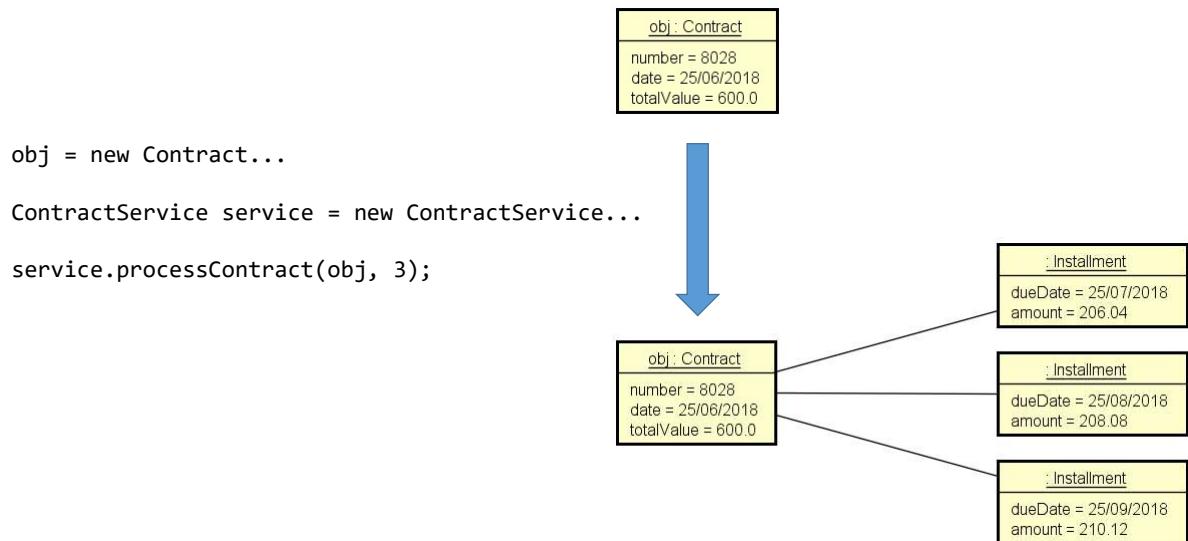
Service layer design



(exemplos explicativos nas próximas páginas)

25

O que o método processContract deve fazer?



26

Entendendo melhor a interface proposta:

PaypalService
+ paymentFee(amount: Double) : Double
+ interest(amount: Double, months: Integer) : Double

```
OnlinePaymentService service = new PaypalService...

double result = service.interest(200, 1);
(result = 2)

double result = service.interest(200, 2);
(result = 4)

double result = service.interest(200, 3);
(result = 6)

double result = service.paymentFee(202);
(result = 4.04)

double result = service.paymentFee(204);
(result = 4.08)

double result = service.paymentFee(206);
(result = 4.12)
```

27

Repositório Github

<https://github.com/acenelio/interfaces4-java>

28

Herdar vs. cumprir contrato

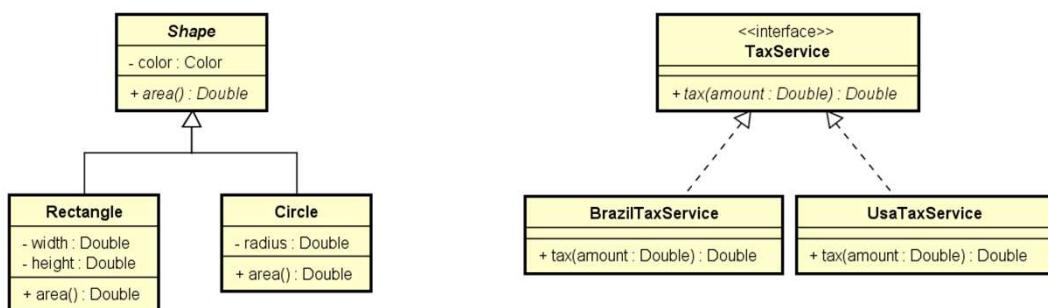
educandoweb.com.br

Prof. Dr. Nelio Alves

29

Aspectos em comum entre herança e interfaces

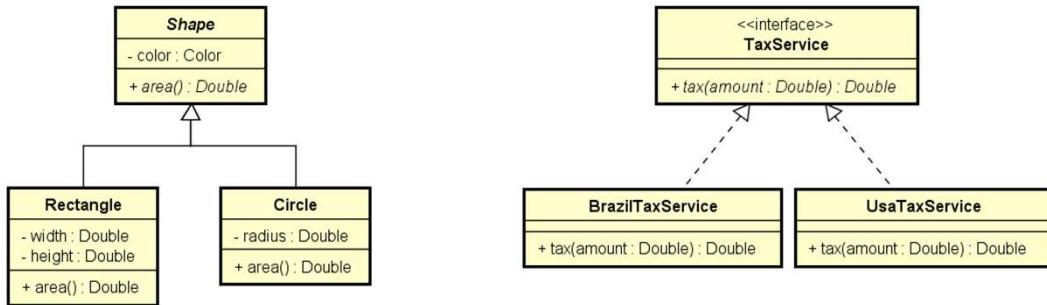
- Relação é-um
- Generalização/especialização
- Polimorfismo



30

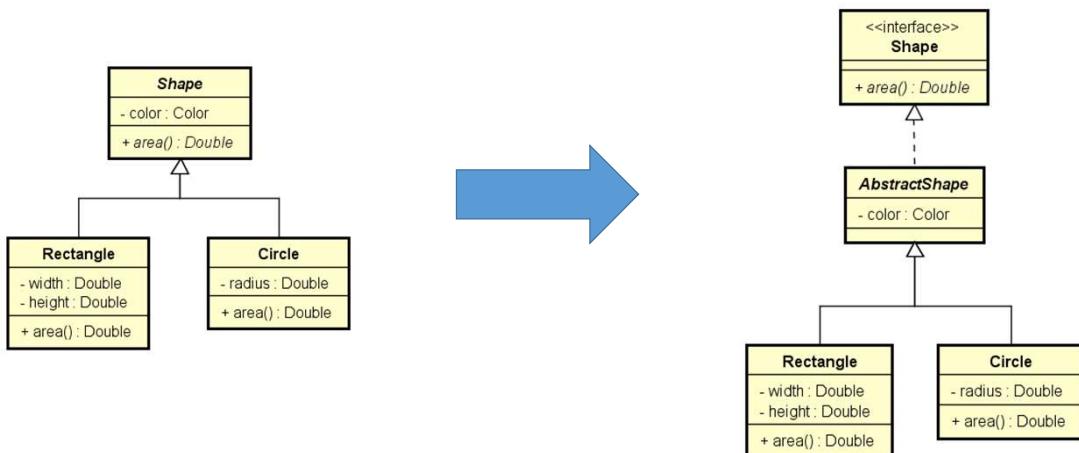
Diferença fundamental

- Herança => reuso
- Interface => contrato a ser cumprido



31

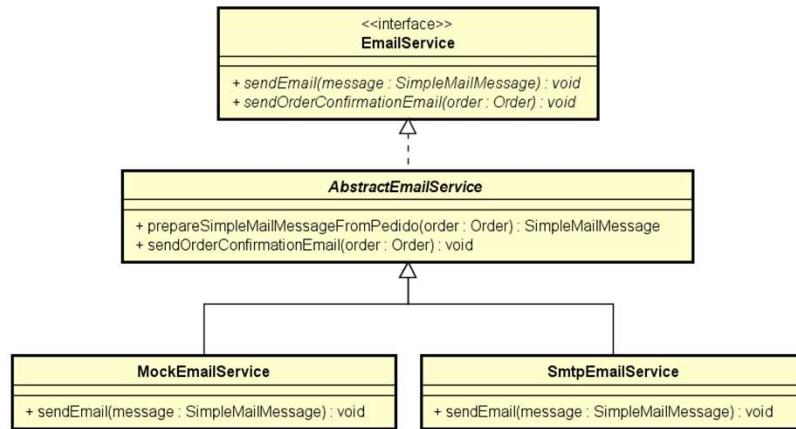
E se eu precisar implementar Shape como interface, porém também quiser definir uma estrutura comum reutilizável para todas figuras?



<https://github.com/acenelio/interfaces2-java>

32

Outro exemplo



<https://github.com/acenelio/springboot2-ionic-backend/tree/master/src/main/java/com/nelioalves/cursomc/services>

33

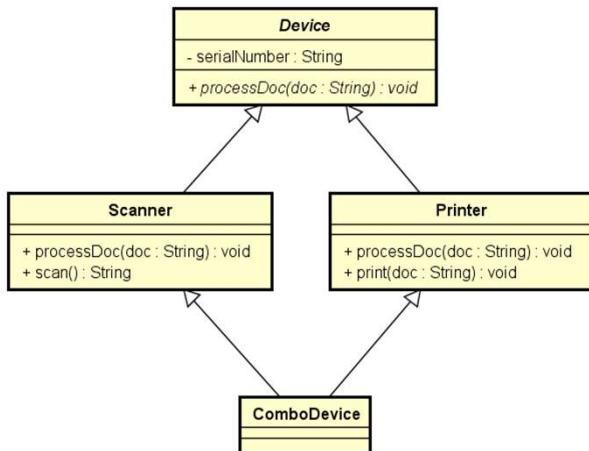
Herança múltipla e o problema do diamante

educandoweb.com.br

Prof. Dr. Nelio Alves

34

Problema do diamante

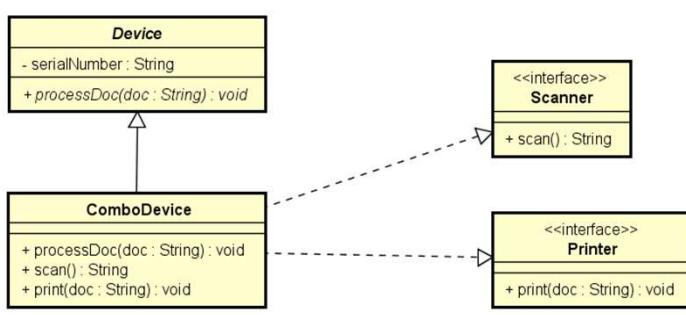


A herança múltipla pode gerar o problema do diamante: uma ambiguidade causada pela existência do mesmo método em mais de uma superclasse.

Herança múltipla não é permitida na maioria das linguagens!

35

Porém, uma classe pode implementar mais de uma interface



ATENÇÃO:

Isso NÃO é herança múltipla, pois NÃO HÁ REUSO na relação entre `ComboDevice` e as interfaces `Scanner` e `Printer`.

`ComboDevice` não herda, mas sim implementa as interfaces (cumpre o contrato).

<https://github.com/acenelio/interfaces3-java>

36

Interface Comparable

educandoweb.com.br

Prof. Dr. Nelio Alves

37

Interface Comparable

<https://docs.oracle.com/javase/10/docs/api/java/lang/Comparable.html>

```
public interface Comparable<T> {  
    int compareTo(T o);  
}
```

38

Problema motivador

Faça um programa para ler um arquivo contendo nomes de pessoas (um nome por linha), armazenando-os em uma lista. Depois, ordenar os dados dessa lista e mostra-los ordenadamente na tela. Nota: o caminho do arquivo pode ser informado "*hardcode*".

```
Maria Brown  
Alex Green  
Bob Grey  
Anna White  
Alex Black  
Eduardo Rose  
Willian Red  
Marta Blue  
Alex Brown
```

39

```
package application;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.List;  
  
public class Program {  
  
    public static void main(String[] args) {  
  
        List<String> list = new ArrayList<>();  
        String path = "C:\\temp\\in.txt";  
  
        try (BufferedReader br = new BufferedReader(new FileReader(path))) {  
  
            String name = br.readLine();  
            while (name != null) {  
                list.add(name);  
                name = br.readLine();  
            }  
            Collections.sort(list);  
            for (String s : list) {  
                System.out.println(s);  
            }  
        } catch (IOException e) {  
            System.out.println("Error: " + e.getMessage());  
        }  
    }  
}
```

40

Outro problema

Faça um programa para ler um arquivo contendo funcionários (nome e salário) no formato .csv, armazenando-os em uma lista. Depois, ordenar a lista por nome e mostrar o resultado na tela. Nota: o caminho do arquivo pode ser informado "hardcode".

```
Maria Brown,4300.00  
Alex Green,3100.00  
Bob Grey,3100.00  
Anna White,3500.00  
Alex Black,2450.00  
Eduardo Rose,4390.00  
Willian Red,2900.00  
Marta Blue,6100.00  
Alex Brown,5000.00
```

41

Interface Comparable

```
public interface Comparable<T> {  
    int compareTo(T o);  
}
```

```
System.out.println("maria".compareTo("alex"));  
System.out.println("alex".compareTo("maria"));  
System.out.println("maria".compareTo("maria"));
```

Output:

```
12  
-12  
0
```

<https://docs.oracle.com/javase/10/docs/api/java/lang/Comparable.html>

Method compareTo:

Parameters:

o - the object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

42

```

package application;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import entities.Employee;

public class Program {

    public static void main(String[] args) {
        List<Employee> list = new ArrayList<>();
        String path = "C:\\temp\\in.txt";

        try (BufferedReader br = new BufferedReader(new FileReader(path))) {
            String employeeCsv = br.readLine();
            while (employeeCsv != null) {
                String[] fields = employeeCsv.split(",");
                list.add(new Employee(fields[0], Double.parseDouble(fields[1])));
                employeeCsv = br.readLine();
            }
            Collections.sort(list);
            for (Employee emp : list) {
                System.out.println(emp.getName() + ", " + emp.getSalary());
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}

```

43

```

package entities;

public class Employee implements Comparable<Employee> {

    private String name;
    private Double salary;

    public Employee(String name, Double salary) {
        this.name = name;
        this.salary = salary;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Double getSalary() {
        return salary;
    }

    public void setSalary(Double salary) {
        this.salary = salary;
    }

    @Override
    public int compareTo(Employee other) {
        return name.compareTo(other.getName());
    }
}

```

44

Default methods

educandoweb.com.br

Prof. Dr. Nelio Alves

45

Default methods (defender methods)

- A partir do Java 8, interfaces podem conter métodos concretos.
- A intenção básica é prover implementação padrão para métodos, de modo a evitar:
 - 1) repetição de implementação em toda classe que implemente a interface
 - 2) a necessidade de se criar classes abstratas para prover reuso da implementação
- Outras vantagens:
 - Manter a retrocompatibilidade com sistemas existentes
 - Permitir que "interfaces funcionais" (que devem conter apenas um método) possam prover outras operações padrão reutilizáveis

46

Problema exemplo

Fazer um programa para ler uma quantia e a duração em meses de um empréstimo. Informar o valor a ser pago depois de decorrido o prazo do empréstimo, conforme regras de juros do Brasil. A regra de cálculo de juros do Brasil é juro composto padrão de 2% ao mês.

Veja o exemplo.

47

Exemplo

Quantia: **200.00**

Meses: **3**

Pagamento apos 3 meses:

212.24

BrazilInterestService

- interestRate : double

+ payment(amount : double, months : int) : double

Calculos:

$$\text{Payment} = 200 * 1.02 * 1.02 * 1.02 = 200 * 1.02^3 = 212.2416$$

$$\text{Payment} = \text{amount} * (1 + \text{interestRate} / 100)^N$$

<https://github.com/acenelio/interfaces5-java>

48

E se houvesse outro serviço de juros de outro país?

Quantia: **200.00**

Meses: **3**

Pagamento apos 3 meses:
206.06

UsaInterestService

- interestRate : double

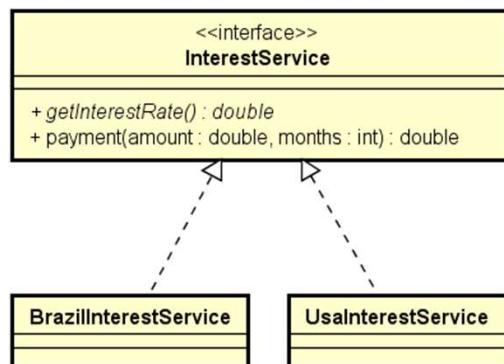
+ payment(amount : double, months : int) : double

Cálculos: $\text{Payment} = 200 * 1.01 * 1.01 * 1.01 = 200 * 1.01^3 = 206.0602$

$\text{Payment} = \text{amount} * (1 + \text{interestRate} / 100)^N$

<https://github.com/acenelio/interfaces5-java>

49



50

Considerações importantes

- Sim: agora as interfaces podem prover reuso
- Sim: agora temos uma forma de herança múltipla
 - Mas o compilador reclama se houver mais de um método com a mesma assinatura, obrigando a sobrescrevê-lo
- Interfaces ainda são bem diferentes de classes abstratas. Interfaces não possuem recursos tais como construtores e atributos.

Curso Programação Orientada a Objetos com Java

Capítulo: Tratamento de exceções

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

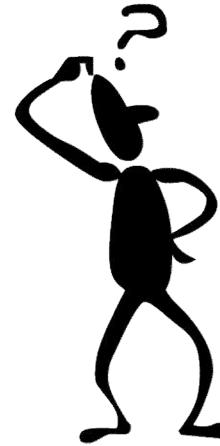
Discussão inicial sobre exceções

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Atenção: você não vai compreender
tudo desta aula ainda.

Mas tudo ficará claro com os
exemplos práticos nas próximas aulas
😊

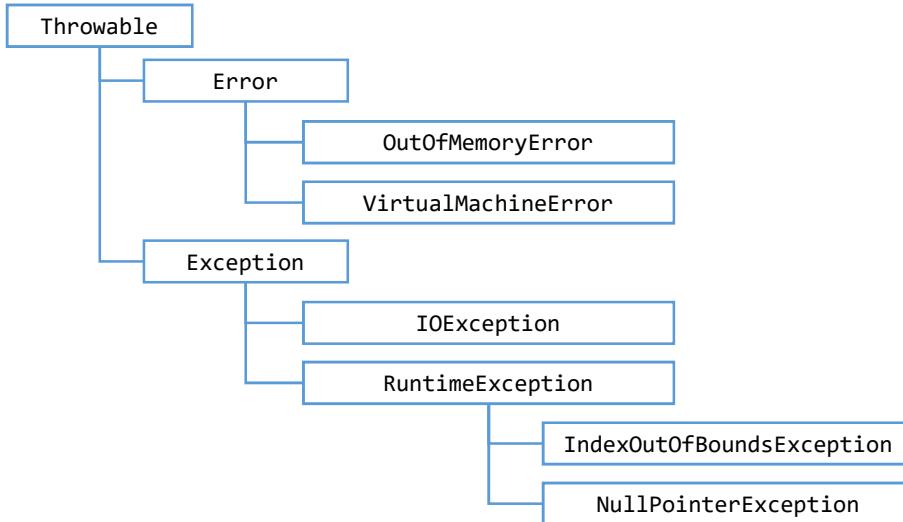


Exceções

- Uma exceção é qualquer condição de erro ou comportamento inesperado encontrado por um programa **em execução**
- Em Java, uma exceção é um objeto herdado da classe:
 - `java.lang.Exception` - o compilador obriga a tratar ou propagar
 - `java.lang.RuntimeException` - o compilador não obriga a tratar ou propagar
- Quando lançada, uma exceção é propagada na pilha de chamadas de métodos em execução, até que seja capturada (tratada) ou o programa seja encerrado

Hierarquia de exceções do Java

<https://docs.oracle.com/javase/10/docs/api/java/lang/package-tree.html>



Por que exceções?

- O modelo de tratamento de exceções permite que erros sejam tratados de forma consistente e flexível, usando boas práticas
- Vantagens:
 - Delega a lógica do erro para a classe responsável por conhecer as regras que podem ocasionar o erro
 - Trata de forma organizada (inclusive hierárquica) exceções de tipos diferentes
 - A exceção pode carregar dados quaisquer

Estrutura try-catch

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Estrutura try-catch

- Bloco try
 - Contém o código que representa a execução normal do trecho de código que **pode** acarretar em uma exceção
- Bloco catch
 - Contém o código a ser executado caso uma exceção ocorra
 - Deve ser especificado o tipo da exceção a ser tratada (upcasting é permitido)
- Demo

Sintaxe

```
try {
}
catch (ExceptionType e) {
}
catch (ExceptionType e) {
}
catch (ExceptionType e) {
}
```

```
package application;

import java.util.InputMismatchException;
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        try {
            String[] vect = sc.nextLine().split(" ");
            int position = sc.nextInt();
            System.out.println(vect[position]);
        }
        catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Invalid position!");
        }
        catch (InputMismatchException e) {
            System.out.println("Input error");
        }

        System.out.println("End of program");
        sc.close();
    }
}
```

Pilha de chamadas de métodos

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

```
package application;

import java.util.InputMismatchException;
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        method1();
        System.out.println("End of program");
    }

    public static void method1() {
        System.out.println("****METHOD1 START****");
        method2();
        System.out.println("****METHOD1 END****");
    }

    public static void method2() {
        System.out.println("****METHOD2 START****");
        Scanner sc = new Scanner(System.in);

        try {
            String[] vect = sc.nextLine().split(" ");
            int position = sc.nextInt();
            System.out.println(vect[position]);
        }
        catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Invalid position!");
            e.printStackTrace();
            sc.next();
        }
        catch (InputMismatchException e) {
            System.out.println("Input error");
        }
        sc.close();
        System.out.println("****METHOD2 END****");
    }
}
```

Bloco finally

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Bloco finally

- É um bloco que contém código a ser executado independentemente de ter ocorrido ou não uma exceção.
- Exemplo clássico: fechar um arquivo, conexão de banco de dados, ou outro recurso específico ao final do processamento.

Sintaxe:

```
try {  
}  
catch (ExceptionType e) {  
}  
finally {  
}
```

```
package application;

import java.io.File;
import java.io.IOException;
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {

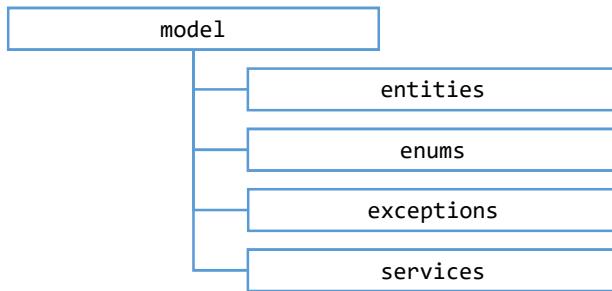
        File file = new File("C:\\\\temp\\\\in.txt");
        Scanner sc = null;
        try {
            sc = new Scanner(file);
            while (sc.hasNextLine()) {
                System.out.println(sc.nextLine());
            }
        } catch (IOException e) {
            System.out.println("Error opening file: " + e.getMessage());
        } finally {
            if (sc != null) {
                sc.close();
            }
        }
    }
}
```

Criando exceções personalizadas

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Sugestão de pacotes "model"



Problema exemplo

Fazer um programa para ler os dados de uma reserva de hotel (número do quarto, data de entrada e data de saída) e mostrar os dados da reserva, inclusive sua duração em dias. Em seguida, ler novas datas de entrada e saída, atualizar a reserva, e mostrar novamente a reserva com os dados atualizados. O programa não deve aceitar dados inválidos para a reserva, conforme as seguintes regras:

- Alterações de reserva só podem ocorrer para datas futuras
- A data de saída deve ser maior que a data de entrada

Reservation
- roomNumber : Integer - checkin : Date - checkout : Date
+ duration() : Integer + updateDates(checkin : Date, checkout : Date) : void

Examples

Room number: **8021**

Check-in date (dd/MM/yyyy): **23/09/2019**

Check-out date (dd/MM/yyyy): **26/09/2019**

Reservation: Room 8021, check-in: 23/09/2019, check-out: 26/09/2019, 3 nights

Enter data to update the reservation:

Check-in date (dd/MM/yyyy): **24/09/2019**

Check-out date (dd/MM/yyyy): **29/09/2019**

Reservation: Room 8021, check-in: 24/09/2019, check-out: 29/09/2019, 5 nights

Room number: **8021**

Check-in date (dd/MM/yyyy): **23/09/2019**

Check-out date (dd/MM/yyyy): **21/09/2019**

Error in reservation: Check-out date must be after check-in date

Examples

Room number: **8021**

Check-in date (dd/MM/yyyy): **23/09/2019**

Check-out date (dd/MM/yyyy): **26/09/2019**

Reservation: Room 8021, check-in: 23/09/2019, check-out: 26/09/2019, 3 nights

Enter data to update the reservation:

Check-in date (dd/MM/yyyy): **24/09/2015**

Check-out date (dd/MM/yyyy): **29/09/2015**

Error in reservation: Reservation dates for update must be future dates

Room number: **8021**

Check-in date (dd/MM/yyyy): **23/09/2019**

Check-out date (dd/MM/yyyy): **26/09/2019**

Reservation: Room 8021, check-in: 23/09/2019, check-out: 26/09/2019, 3 nights

Enter data to update the reservation:

Check-in date (dd/MM/yyyy): **24/09/2020**

Check-out date (dd/MM/yyyy): **22/09/2020**

Error in reservation: Check-out date must be after check-in date

Resumo da aula

- Solução 1 (muito ruim): lógica de validação no programa principal
 - Lógica de validação não delegada à reserva
- Solução 2 (ruim): método retornando string
 - A semântica da operação é prejudicada
 - Retornar string não tem nada a ver com atualização de reserva
 - E se a operação tivesse que retornar um string?
 - Ainda não é possível tratar exceções em construtores
 - Ainda não há auxílio do compilador: o programador deve "lembra" de verificar se houve erro
 - A lógica fica estruturada em condicionais aninhadas
- Solução 3 (boa): tratamento de exceções

<https://github.com/acenelio/exceptions1-java>

Resumo da aula

- Cláusula throws: propaga a exceção ao invés de trata-la
- Cláusula throw: lança a exceção / "corta" o método
- Exception: compilador obriga a tratar ou propagar
- RuntimeException: compilador não obriga
- O modelo de tratamento de exceções permite que erros sejam tratados de forma consistente e flexível, usando boas práticas
- Vantagens:
 - Lógica delegada
 - Construtores podem ter tratamento de exceções
 - Possibilidade de auxílio do compilador (Exception)
 - Código mais simples. Não há aninhamento de condicionais: a qualquer momento que uma exceção for disparada, a execução é interrompida e cai no bloco catch correspondente.
 - É possível capturar inclusive outras exceções de sistema

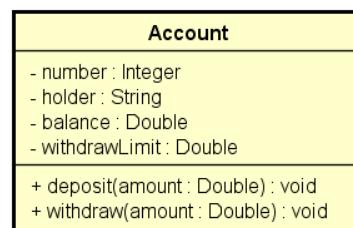
Exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Exercício de fixação

Fazer um programa para ler os dados de uma conta bancária e depois realizar um saque nesta conta bancária, mostrando o novo saldo. Um saque não pode ocorrer ou se não houver saldo na conta, ou se o valor do saque for superior ao limite de saque da conta. Implemente a conta bancária conforme projeto abaixo:



Examples

```
Enter account data
Number: 8021
Holder: Bob Brown
Initial balance: 500.00
Withdraw limit: 300.00
```

```
Enter amount for withdraw: 100.00
New balance: 400.00
```

```
Enter account data
Number: 8021
Holder: Bob Brown
Initial balance: 500.00
Withdraw limit: 300.00
```

```
Enter amount for withdraw: 400.00
Withdraw error: The amount exceeds withdraw limit
```

Examples

```
Enter account data
Number: 8021
Holder: Bob Brown
Initial balance: 500.00
Withdraw limit: 300.00
```

```
Enter amount for withdraw: 800.00
Withdraw error: The amount exceeds withdraw limit
```

```
Enter account data
Number: 8021
Holder: Bob Brown
Initial balance: 200.00
Withdraw limit: 300.00
```

```
Enter amount for withdraw: 250.00
Withdraw error: Not enough balance
```

<https://github.com/acenelio/exceptions2-java>

Curso Programação Orientada a Objetos com Java

Capítulo: Generics, Set, Map

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Introdução aos Generics

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Generics

- Generics permitem que **classes, interfaces e métodos** possam ser parametrizados por tipo. Seus benefícios são:
 - Reuso
 - Type safety
 - Performance
- Uso comum: coleções

```
List<String> list = new ArrayList<>();  
list.add("Maria");  
String name = list.get(0);
```

Problema motivador 1 (reuso)

Deseja-se fazer um programa que leia uma quantidade N, e depois N números inteiros. Ao final, imprima esses números de forma organizada conforme exemplo. Em seguida, informar qual foi o primeiro valor informado.

How many values? **3**

10

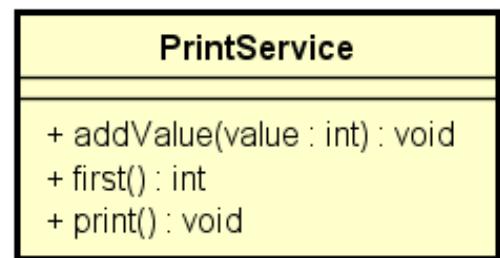
8

23

[10, 8, 23]

First: 10

Criar um serviço de impressão:



Problema motivador 2 (type safety & performance)

Deseja-se fazer um programa que leia uma quantidade N, e depois N nomes de pessoas. Ao final, imprima esses números de forma organizada conforme exemplo. Em seguida, informar qual foi o primeiro valor informado.

How many values? **3**

10

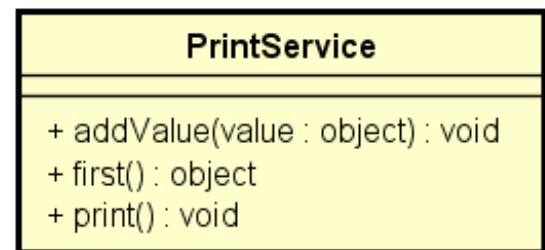
8

23

[10, 8, 23]

First: 10

Criar um serviço de impressão:



Solução com generics

Deseja-se fazer um programa que leia uma quantidade N, e depois N números inteiros. Ao final, imprima esses números de forma organizada conforme exemplo. Em seguida, informar qual foi o primeiro valor informado.

How many values? **3**

10

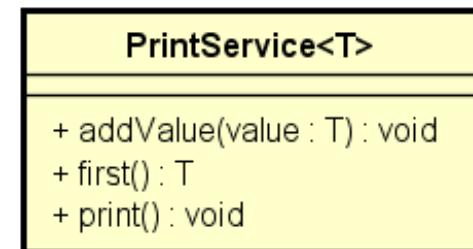
8

23

[10, 8, 23]

First: 10

Criar um serviço de impressão:



<https://github.com/acenelio/generics1-java>

Genéricos delimitados

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Problema

Uma empresa de consultoria deseja avaliar a performance de produtos, funcionários, dentre outras coisas. Um dos cálculos que ela precisa é encontrar o maior dentre um conjunto de elementos. Fazer um programa que leia um conjunto de produtos a partir de um arquivo, conforme exemplo, e depois mostre o mais caro deles.

Computer, 890.50

IPhone X, 910.00

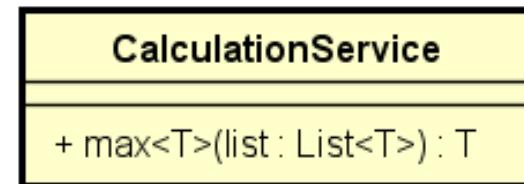
Tablet, 550.00

Most expensive:

IPhone, 910.00

<https://github.com/acenelio/generics2-java>

Criar um serviço de cálculo:



Nota: Java possui:
Collections.max(list)

```
package services;

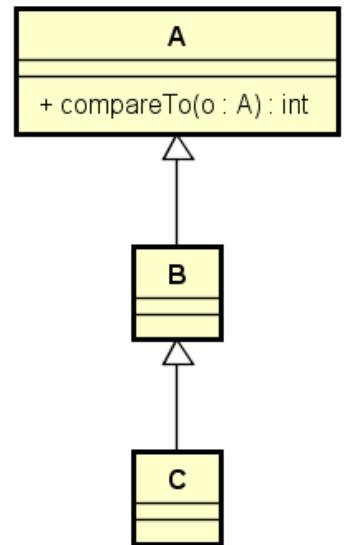
import java.util.List;

public class CalculationService {

    public static <T extends Comparable<T>> T max(List<T> list) {
        if (list.isEmpty()) {
            throw new IllegalStateException("List can't be empty");
        }
        T max = list.get(0);
        for (T item : list) {
            if (item.compareTo(max) > 0) {
                max = item;
            }
        }
        return max;
    }
}
```

Versão alternativa (completa)

```
public static <T extends Comparable<? super T>> T max(List<T> list) {  
    if (list.isEmpty()) {  
        throw new IllegalStateException("List can't be empty");  
    }  
    T max = list.get(0);  
    for (T item : list) {  
        if (item.compareTo(max) > 0) {  
            max = item;  
        }  
    }  
    return max;  
}
```



Tipos curinga (wildcard types)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Generics são invariantes

List<Object> não é o supertipo de qualquer tipo de lista:

```
List<Object> myObjs = new ArrayList<Object>();  
List<Integer> myNumbers = new ArrayList<Integer>();  
myObjs = myNumbers; // erro de compilação
```

O supertipo de qualquer tipo de lista é List<?>. Este é um tipo curinga:

```
List<?> myObjs = new ArrayList<Object>();  
List<Integer> myNumbers = new ArrayList<Integer>();  
myObjs = myNumbers;
```

Com tipos curinga podemos fazer métodos que recebem um genérico de "qualquer tipo":

```
package application;

import java.util.Arrays;
import java.util.List;

public class Program {

    public static void main(String[] args) {
        List<Integer> myInts = Arrays.asList(5, 2, 10);
        printList(myInts);
    }

    public static void printList(List<?> list) {
        for (Object obj : list) {
            System.out.println(obj);
        }
    }
}
```

Porém não é possível adicionar dados a uma coleção de tipo curinga

```
package application;

import java.util.ArrayList;
import java.util.List;

public class Program {

    public static void main(String[] args) {

        List<?> list = new ArrayList<Integer>();
        list.add(3); // erro de compilação
    }
}
```

O compilador não sabe qual é o tipo específico do qual a lista foi instanciada.

Curingas delimitados (bounded wildcards)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Problema 1

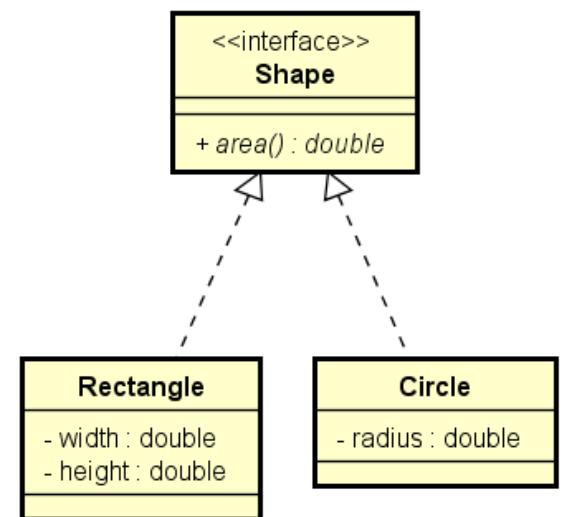
Vamos fazer um método para retornar a soma das áreas de uma lista de figuras.

Nota 1: soluções impróprias:

```
public double totalArea(List<Shape> list)  
  
public double totalArea(List<?> list)
```

Nota 2: não conseguiremos adicionar elementos na lista do método

<https://github.com/acenelio/generics4-java>



Problema 2 (princípio *get/put*)

Vamos fazer um método que copia os elementos de uma lista para uma outra lista que pode ser mais genérica que a primeira.

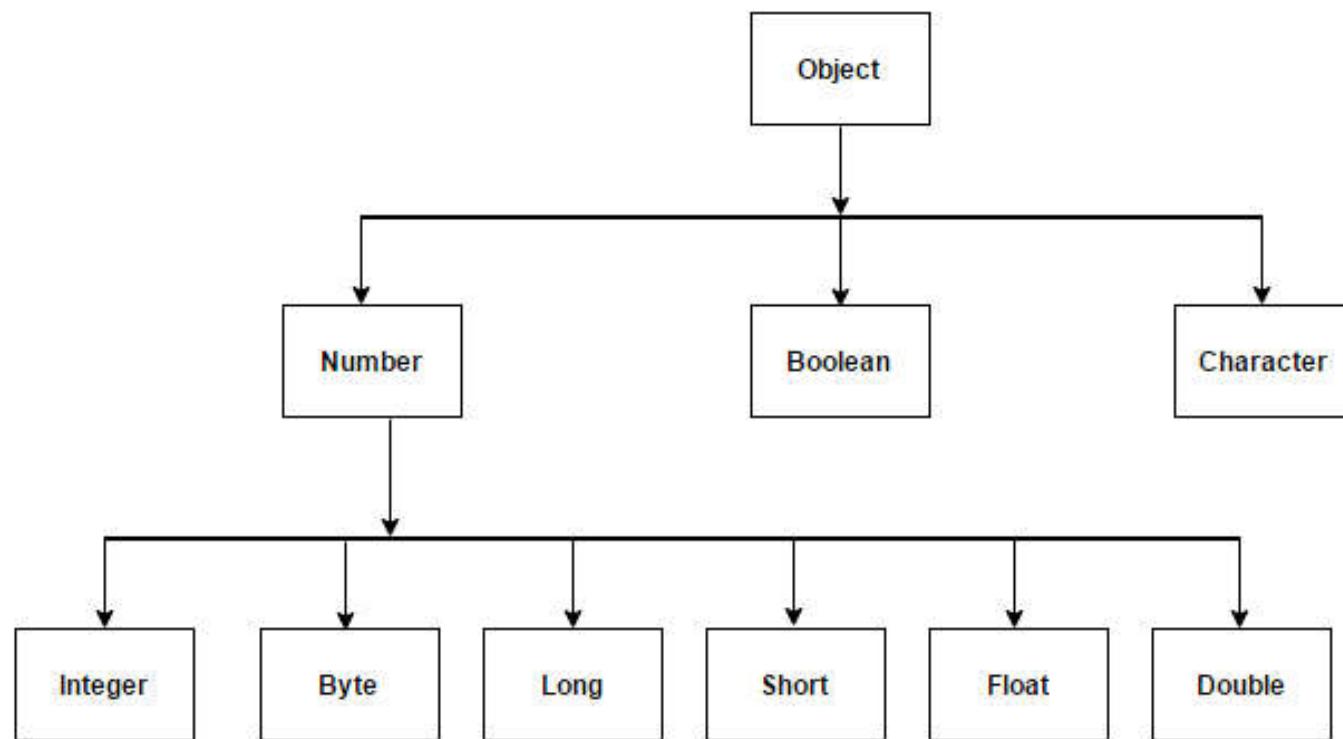
```
List<Integer> myInts = Arrays.asList(1, 2, 3, 4);  
List<Double> myDoubles = Arrays.asList(3.14, 6.28);  
List<Object> myObjs = new ArrayList<Object>();
```

```
copy(myInts, myObjs);
```

```
copy(myDoubles, myObjs);
```

<https://stackoverflow.com/questions/1368166/what-is-a-difference-between-super-e-and-extends-e>

Java wrapper types (próximos exemplos)



Princípio get/put - covariância

```
List<Integer> intList = new ArrayList<Integer>();  
intList.add(10);  
intList.add(5);  
  
List<? extends Number> list = intList;  
  
Number x = list.get(0);  
  
list.add(20); // erro de compilacao
```

get - OK

put - ERROR

Princípio get/put - contravariância

```
List<Object> myObjs = new ArrayList<Object>();  
myObjs.add("Maria");  
myObjs.add("Alex");  
  
List<? super Number> myNums = myObjs;  
  
myNums.add(10);  
myNums.add(3.14);  
  
Number x = myNums.get(0); // erro de compilacao
```

get - ERROR

put - OK

```
package application;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class Program {

    public static void main(String[] args) {

        List<Integer> myInts = Arrays.asList(1, 2, 3, 4);
        List<Double> myDoubles = Arrays.asList(3.14, 6.28);
        List<Object> myObjs = new ArrayList<Object>();

        copy(myInts, myObjs);
        printList(myObjs);
        copy(myDoubles, myObjs);
        printList(myObjs);
    }

    public static void copy(List<? extends Number> source, List<? super Number> destiny) {
        for(Number number : source) {
            destiny.add(number);
        }
    }

    public static void printList(List<?> list) {
        for (Object obj : list) {
            System.out.print(obj + " ");
        }
        System.out.println();
    }
}
```

hashCode e equals

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

hashCode e equals

- São operações da classe Object utilizadas para comparar se um objeto é igual a outro
- equals: lento, resposta 100%
- hashCode: rápido, porém resposta positiva não é 100%
- Tipos comuns (String, Date, Integer, Double, etc.) já possuem implementação para essas operações. Classes personalizadas precisam sobrepor-las.

Equals

Método que compara se o objeto é igual a outro, retornando true ou false.

```
String a = "Maria";  
String b = "Alex";
```

```
System.out.println(a.equals(b));
```

HashCode

Método que retorna um número inteiro representando um código gerado a partir das informações do objeto

```
String a = "Maria";
String b = "Alex";

System.out.println(a.hashCode());
System.out.println(b.hashCode());
```

Regra de ouro do hashCode

- Se o hashCode de dois objetos for diferente, então os dois objetos são diferentes

Isso nunca
acontece:



- Se o código de dois objetos for igual, **muito provavelmente** os objetos são iguais (pode haver colisão)

HashCode e Equals personalizados

```
public class Client {  
  
    private String name;  
    private String email;  
}
```

Set

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Set<T>

- Representa um conjunto de elementos (similar ao da Álgebra)
 - Não admite repetições
 - Elementos não possuem posição
 - Acesso, inserção e remoção de elementos são rápidos
 - Oferece operações eficientes de conjunto: interseção, união, diferença.
 - Principais implementações:
 - **HashSet** - mais rápido (operações O(1) em tabela hash) e não ordenado
 - **TreeSet** - mais lento (operações O(log(n)) em árvore rubro-negra) e ordenado pelo compareTo do objeto (ou Comparator)
 - **LinkedHashSet** - velocidade intermediária e elementos na ordem em que são adicionados
- <https://docs.oracle.com/javase/10/docs/api/java/util/Set.html>

Alguns métodos importantes

- `add(obj)`, `remove(obj)`, `contains(obj)`
 - Baseado em `equals` e `hashCode`
 - Se `equals` e `hashCode` não existir, é usada comparação de ponteiros
- `clear()`
- `size()`
- `removeIf(predicate)`
- `addAll(other)` - **união**: adiciona no conjunto os elementos do outro conjunto, sem repetição
- `retainAll(other)` - **interseção**: remove do conjunto os elementos não contidos em `other`
- `removeAll(other)` - **diferença**: remove do conjunto os elementos contidos em `other`

Demo 1

```
package application;

import java.util.HashSet;
import java.util.Set;

import Entities.Product;

public class Program {

    public static void main(String[] args) {

        Set<String> set = new HashSet<>();

        set.add("TV");
        set.add("Notebook");
        set.add("Tablet");

        System.out.println(set.contains("Notebook"));

        for (String p : set) {
            System.out.println(p);
        }
    }
}
```

Demo 2

```
package application;

import java.util.Arrays;
import java.util.Set;
import java.util.TreeSet;

public class Program {

    public static void main(String[] args) {

        Set<Integer> a = new TreeSet<>(Arrays.asList(0,2,4,5,6,8,10));
        Set<Integer> b = new TreeSet<>(Arrays.asList(5,6,7,8,9,10));

        //union
        Set<Integer> c = new TreeSet<>(a);
        c.addAll(b);
        System.out.println(c);

        //intersection
        Set<Integer> d = new TreeSet<>(a);
        d.retainAll(b);
        System.out.println(d);

        //difference
        Set<Integer> e = new TreeSet<>(a);
        e.removeAll(b);
        System.out.println(e);
    }
}
```

Como Set testa igualdade?

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Como as coleções Hash testam igualdade?

- Se hashCode e equals estiverem implementados:
 - Primeiro hashCode. Se der igual, usa equals para confirmar.
 - Lembre-se: String, Integer, Double, etc. já possuem equals e hashCode
- Se hashCode e equals **NÃO** estiverem implementados:
 - Compara as referências (ponteiros) dos objetos.

Demo

```
package application;

import java.util.HashSet;
import java.util.Set;

import Entities.Product;

public class Program {

    public static void main(String[] args) {

        Set<Product> set = new HashSet<>();

        set.add(new Product("TV", 900.0));
        set.add(new Product("Notebook", 1200.0));
        set.add(new Product("Tablet", 400.0));

        Product prod = new Product("Notebook", 1200.0);

        System.out.println(set.contains(prod));
    }
}
```

```
package entities;

public class Product {

    private String name;
    private Double price;

    public Product(String name, Double price) {
        this.name = name;
        this.price = price;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Double getPrice() {
        return price;
    }

    public void setPrice(Double price) {
        this.price = price;
    }
}
```

Como TreeSet compara os elementos?

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Recordando as implementações

- Principais implementações:
 - **HashSet** - mais rápido (operações $O(1)$ em tabela hash) e não ordenado
 - **TreeSet** - mais lento (operações $O(\log(n))$ em árvore rubro-negra) e ordenado pelo `compareTo` do objeto (ou `Comparator`)
 - **LinkedHashSet** - velocidade intermediária e elementos na ordem em que são adicionados

Demo

```
package application;

import java.util.Set;
import java.util.TreeSet;

import Entities.Product;

public class Program {

    public static void main(String[] args) {

        Set<Product> set = new TreeSet<>();

        set.add(new Product("TV", 900.0));
        set.add(new Product("Notebook", 1200.0));
        set.add(new Product("Tablet", 400.0));

        for (Product p : set) {
            System.out.println(p);
        }
    }
}
```

```
package entities;

public class Product implements Comparable<Product> {

    private String name;
    private Double price;

    public Product(String name, Double price) {
        this.name = name;
        this.price = price;
    }

    // (... get / set / hashCode / equals)

    @Override
    public String toString() {
        return "Product [name=" + name + ", price=" + price + "]";
    }

    @Override
    public int compareTo(Product other) {
        return name.toUpperCase().compareTo(other.getName().toUpperCase());
    }
}
```

Exercício resolvido (Set)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Problema exemplo

Um site de internet registra um log de acessos dos usuários. Um registro de log consiste no nome de usuário (apenas uma palavra) e o instante em que o usuário acessou o site no padrão ISO 8601, separados por espaço, conforme exemplo. Fazer um programa que leia o log de acessos a partir de um arquivo, e daí informe quantos usuários distintos acessaram o site.

Example

input file:

```
amanda 2018-08-26T20:45:08Z
alex86 2018-08-26T21:49:37Z
bobbrown 2018-08-27T03:19:13Z
amanda 2018-08-27T08:11:00Z
jeniffer3 2018-08-27T09:19:24Z
alex86 2018-08-27T22:39:52Z
amanda 2018-08-28T07:42:19Z
```

Execution:

```
Enter file full path: c:\temp\in.txt
```

```
Total users: 4
```

<https://github.com/acenelio/set1-java>

Exercício proposto (Set)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Em um portal de cursos online, cada usuário possui um código único, representado por um número inteiro.

Cada instrutor do portal pode ter vários cursos, sendo que um mesmo aluno pode se matricular em quantos cursos quiser. Assim, o número total de alunos de um instrutor não é simplesmente a soma dos alunos de todos os cursos que ele possui, pois pode haver alunos repetidos em mais de um curso.

O instrutor Alex possui três cursos A, B e C, e deseja saber seu número total de alunos.

Seu programa deve ler os alunos dos cursos A, B e C do instrutor Alex, depois mostrar a quantidade total e alunos dele, conforme exemplo.

<https://github.com/acenelio/set2-java>

Example:

How many students for course A? **3**

21

35

22

How many students for course B? **2**

21

50

How many students for course C? **3**

42

35

13

Total students: 6

Map

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Map<K,V>

- <https://docs.oracle.com/javase/10/docs/api/java/util/Map.html>
- É uma coleção de pares chave / valor
 - Não admite repetições do objeto chave
 - Os elementos são indexados pelo objeto chave (não possuem posição)
 - Acesso, inserção e remoção de elementos são rápidos
- Uso comum: cookies, local storage, qualquer modelo chave-valor
- Principais implementações:
 - **HashMap** - mais rápido (operações O(1) em tabela hash) e não ordenado
 - **TreeMap** - mais lento (operações O(log(n)) em árvore rubro-negra) e ordenado pelo compareTo do objeto (ou Comparator)
 - **LinkedHashMap** - velocidade intermediária e elementos na ordem em que são adicionados

Alguns métodos importantes

- `put(key, value)`, `remove(key)`, `containsKey(key)`, `get(key)`
 - Baseado em `equals` e `hashCode`
 - Se `equals` e `hashCode` não existir, é usada comparação de ponteiros
- `clear()`
- `size()`
- `keySet()` - retorna um `Set<K>`
- `values()` - retornaa um `Collection<V>`

Demo 1

```
package application;

import java.util.Map;
import java.util.TreeMap;

public class Program {

    public static void main(String[] args) {

        Map<String, String> cookies = new TreeMap<>();

        cookies.put("username", "maria");
        cookies.put("email", "maria@gmail.com");
        cookies.put("phone", "99771122");

        cookies.remove("email");
        cookies.put("phone", "99771133");

        System.out.println("Contains 'phone' key: " + cookies.containsKey("phone"));
        System.out.println("Phone number: " + cookies.get("phone"));
        System.out.println("Email: " + cookies.get("email"));
        System.out.println("Size: " + cookies.size());

        System.out.println("ALL COOKIES:");
        for (String key : cookies.keySet()) {
            System.out.println(key + ": " + cookies.get(key));
        }
    }
}
```

Demo 2

```
package application;

import java.util.HashMap;
import java.util.Map;

import Entities.Product;

public class Program {

    public static void main(String[] args) {

        Map<Product, Double> stock = new HashMap<>();

        Product p1 = new Product("Tv", 900.0);
        Product p2 = new Product("Notebook", 1200.0);
        Product p3 = new Product("Tablet", 400.0);

        stock.put(p1, 10000.0);
        stock.put(p2, 20000.0);
        stock.put(p3, 15000.0);

        Product ps = new Product("Tv", 900.0);

        System.out.println("Contains 'ps' key: " + stock.containsKey(ps));
    }
}
```

```
package entities;

public class Product {

    private String name;
    private Double price;

    public Product(String name, Double price) {
        this.name = name;
        this.price = price;
    }

    // getters, setters, equals, hashCode
}
```

Exercício proposto (Map)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Na contagem de votos de uma eleição, são gerados vários registros de votação contendo o nome do candidato e a quantidade de votos (formato .csv) que ele obteve em uma urna de votação. Você deve fazer um programa para ler os registros de votação a partir de um arquivo, e daí gerar um relatório consolidado com os totais de cada candidato.

Input file example:

```
Alex Blue,15
Maria Green,22
Bob Brown,21
Alex Blue,30
Bob Brown,15
Maria Green,27
Maria Green,22
Bob Brown,25
Alex Blue,31
```

Execution:

```
Enter file full path: c:\temp\in.txt
Alex Blue: 76
Maria Green: 71
Bob Brown: 61
```

Solução do exercício

<https://github.com/acenelio/map1-java>

Curso Programação Orientada a Objetos com Java

Capítulo: Programação Funcional e Expressões Lambda

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

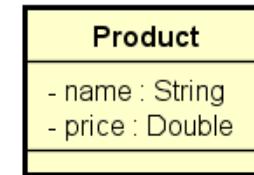
Uma experiência com Comparator

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Problema

- Suponha uma classe Product com os atributos name e price.
- Podemos implementar a comparação de produtos por meio da implementação da interface Comparable<Product>
- Entretanto, desta forma nossa classe não fica fechada para alteração: se o critério de comparação mudar, precisaremos alterar a classe Product.
- Podemos então usar o default method "sort" da interface List:
`default void sort(Comparator<? super E> c)`



Comparator

<https://docs.oracle.com/javase/10/docs/api/java/util/Comparator.html>

Veja o método sort na interface List:

<https://docs.oracle.com/javase/10/docs/api/java/util/List.html>

Resumo da aula

- Comparator objeto de classe separada
- Comparator objeto de classe anônima
- Comparator objeto de expressão lambda com chaves
- Comparator objeto de expressão lambda sem chaves
- Comparator expressão lambda "direto no argumento"

<https://github.com/acenelio/lambda1-java>

Programação funcional e cálculo lambda

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Paradigmas de programação

- **Imperativo** (C, Pascal, Fortran, Cobol)
- **Orientado a objetos** (C++, Object Pascal, Java (< 8), C# (< 3))
- **Funcional** (Haskell, Closure, Clean, Erlang)
- **Lógico** (Prolog)
- **Multiparadigma** (JavaScript, Java (8+), C# (3+), Ruby, Python, Go)

Paradigma funcional de programação

Baseado no formalismo matemático Cálculo Lambda (Church 1930)

	Programação Imperativa	Programação Funcional
Como se descreve algo a ser computado (*)	comandos ("como" - imperativa)	expressões ("o quê" - declarativa)
Funções possuem transparência referencial (ausência de efeitos colaterais)	fraco	forte
Objetos imutáveis (*)	raro	comum
Funções são objetos de primeira ordem	não	sim
Expressividade / código conciso	baixa	alta
Tipagem dinâmica / inferência de tipos	raro	comum

Transparência referencial

Uma função possui transparência referencial se seu resultado for sempre o mesmo para os mesmos dados de entrada. Benefícios: simplicidade e previsibilidade.

```
package application;

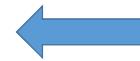
import java.util.Arrays;

public class Program {

    public static int globalValue = 3;

    public static void main(String[] args) {
        int[] vect = new int[] {3, 4, 5};
        changeOddValues(vect);
        System.out.println(Arrays.toString(vect));
    }

    public static void changeOddValues(int[] numbers) {
        for (int i=0; i<numbers.length; i++) {
            if (numbers[i] % 2 != 0) {
                numbers[i] += globalValue;
            }
        }
    }
}
```



Exemplo de função que não é referencialmente transparente

Funções são objetos de primeira ordem (ou primeira classe)

Isso significa que funções podem, por exemplo, serem passadas como parâmetros de métodos, bem como retornadas como resultado de métodos.

```
public class Program {

    public static int compareProducts(Product p1, Product p2) {
        return p1.getPrice().compareTo(p2.getPrice());
    }

    public static void main(String[] args) {
        List<Product> list = new ArrayList<>();

        list.add(new Product("TV", 900.00));
        list.add(new Product("Notebook", 1200.00));
        list.add(new Product("Tablet", 450.00));

        list.sort(Program::compareProducts);

        list.forEach(System.out::println);
    }
}
```

Utilizamos aqui "method references"

Operador ::

Sintaxe:
Classe::método

Tipagem dinâmica / inferência de tipos

```
public static void main(String[] args) {  
    List<Product> list = new ArrayList<>();  
  
    list.add(new Product("TV", 900.00));  
    list.add(new Product("Notebook", 1200.00));  
    list.add(new Product("Tablet", 450.00));  
  
    list.sort((p1, p2) -> p1.getPrice().compareTo(p2.getPrice()));  
  
    list.forEach(System.out::println);  
}
```

Expressividade / código conciso

```
Integer sum = 0;  
for (Integer x : list) {  
    sum += x;  
}
```

VS.

```
Integer sum = list.stream().reduce(0, Integer::sum);
```

O que são "expressões lambda"?

Em programação funcional, expressão lambda corresponde a uma função anônima de primeira classe.

```
public class Program {
    public static int compareProducts(Product p1, Product p2) {
        return p1.getPrice().compareTo(p2.getPrice());
    }

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

        list.sort(Program::compareProducts);

        list.sort((p1, p2) -> p1.getPrice().compareTo(p2.getPrice()));

        (...)
```

Resumo da aula

	Programação Imperativa	Programação Funcional
Como se descreve algo a ser computado (*)	comandos ("como" - imperativa)	expressões ("o quê" - declarativa)
Funções possuem transparência referencial (ausência de efeitos colaterais)	fraco	forte
Objetos imutáveis (*)	raro	comum
Funções são objetos de primeira ordem	não	sim
Expressividade / código conciso	baixa	alta
Tipagem dinâmica / inferência de tipos	raro	comum

Cálculo Lambda = formalismo matemático base da programação funcional

Expressão lambda = função anônima de primeira classe

Interface funcional

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Interface funcional

É uma interface que possui um único método abstrato. Suas implementações serão tratadas como expressões lambda.

```
public class MyComparator implements Comparator<Product> {  
    @Override  
    public int compare(Product p1, Product p2) {  
        return p1.getName().toUpperCase().compareTo(p2.getName().toUpperCase());  
    }  
}
```

```
public static void main(String[] args) {  
    (...)  
    list.sort(new MyComparator());
```

Algumas outras interfaces funcionais comuns

- Predicate
 - <https://docs.oracle.com/javase/8/docs/api/java/util/function/Predicate.html>
- Function
 - <https://docs.oracle.com/javase/8/docs/api/java/util/function/Function.html>
- Consumer
 - <https://docs.oracle.com/javase/8/docs/api/java/util/function/Consumer.html>
 - Nota: ao contrário das outras interfaces funcionais, no caso do Consumer, é esperado ele possa gerar efeitos colaterais

Predicate (exemplo com removelf)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Predicate

<https://docs.oracle.com/javase/10/docs/api/java/util/function/Predicate.html>

```
public interface Predicate<T> {  
    boolean test(T t);  
}
```

Problema exemplo

Fazer um programa que, a partir de uma lista de produtos, remova da lista somente aqueles cujo preço mínimo seja 100.

```
List<Product> list = new ArrayList<>();  
  
list.add(new Product("Tv", 900.00));  
list.add(new Product("Mouse", 50.00));  
list.add(new Product("Tablet", 350.50));  
list.add(new Product("HD Case", 80.90));
```

<https://github.com/acenelio/lambda2-java>

Versões:

- Implementação da interface
- Reference method com método estático
- Reference method com método não estático
- Expressão lambda declarada
- Expressão lambda inline

Consumer (exemplo com forEach)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Consumer

<https://docs.oracle.com/javase/10/docs/api/java/util/function/Consumer.html>

```
public interface Consumer<T> {  
    void accept(T t);  
}
```

Problema exemplo

Fazer um programa que, a partir de uma lista de produtos, aumente o preço dos produtos em 10%.

```
List<Product> list = new ArrayList<>();  
  
list.add(new Product("Tv", 900.00));  
list.add(new Product("Mouse", 50.00));  
list.add(new Product("Tablet", 350.50));  
list.add(new Product("HD Case", 80.90));
```

<https://github.com/acenelio/lambda3-java>

Function (exemplo com map)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Function

<https://docs.oracle.com/javase/10/docs/api/java/util/function/Function.html>

```
public interface Function<T, R> {  
    R apply(T t);  
}
```

Problema exemplo

Fazer um programa que, a partir de uma lista de produtos, gere uma nova lista contendo os nomes dos produtos em caixa alta.

```
List<Product> list = new ArrayList<>();  
  
list.add(new Product("Tv", 900.00));  
list.add(new Product("Mouse", 50.00));  
list.add(new Product("Tablet", 350.50));  
list.add(new Product("HD Case", 80.90));
```

<https://github.com/acenelio/lambda4-java>

Nota sobre a função map

- A função "map" (não confunda com a estrutura de dados Map) é uma função que aplica uma função a todos elementos de uma stream.
- Conversões:
 - List para stream: `.stream()`
 - Stream para List: `.collect(Collectors.toList())`

Criando funções que recebem funções como argumento

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Recordando

- `removelf(Predicate)`
- `foreach(Consumer)`
- `map(Function)`

Problema exemplo

Fazer um programa que, a partir de uma lista de produtos, calcule a soma dos preços somente dos produtos cujo nome começa com "T".

```
List<Product> list = new ArrayList<>();  
  
list.add(new Product("Tv", 900.00));  
list.add(new Product("Mouse", 50.00));  
list.add(new Product("Tablet", 350.50));  
list.add(new Product("HD Case", 80.90));
```



1250.50

<https://github.com/acenelio/lambda5-java>

Stream

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Stream

- É uma sequencia de elementos advinda de uma fonte de dados que oferece suporte a "operações agregadas".
 - Fonte de dados: coleção, array, função de iteração, recurso de E/S
- Sugestão de leitura:
<http://www.oracle.com/technetwork/pt/articles/javastreams-api-jav-8-3410098-ptb.html>

Características

- Stream é uma solução para processar sequências de dados de forma:
 - Declarativa (iteração interna: escondida do programador)
 - Parallel-friendly (imutável -> thread safe)
 - Sem efeitos colaterais
 - Sob demanda (lazy evaluation)
- Acesso sequencial (não há índices)
- Single-use: só pode ser "usada" uma vez
- **Pipeline:** operações em streams retornam novas streams. Então é possível criar uma cadeia de operações (fluxo de processamento).

Operações intermediárias e terminais

- O pipeline é composto por zero ou mais operações intermediárias e uma terminal.
- Operação intermediária:
 - Produz uma nova streams (encadeamento)
 - Só executa quando uma operação terminal é invocada (lazy evaluation)
- Operação terminal:
 - Produz um objeto não-stream (coleção ou outro)
 - Determina o fim do processamento da stream

Operações intermediárias

- filter
- map
- flatmap
- peek
- distinct
- sorted
- skip
- limit (*)

* *short-circuit*

Operações terminais

- forEach
- forEachOrdered
- toArray
- reduce
- collect
- min
- max
- count
- anyMatch (*)
- allMatch (*)
- noneMatch (*)
- findFirst (*)
- findAny (*)

* *short-circuit*

Criar uma stream

- Basta chamar o método `stream()` ou `parallelStream()` a partir de qualquer objeto Collection.
<https://docs.oracle.com/javase/10/docs/api/java/util/Collection.html>
- Outras formas de se criar uma stream incluem:
 - `Stream.of`
 - `Stream.ofNullable`
 - `Stream.iterate`

Demo - criação de streams

```
List<Integer> list = Arrays.asList(3, 4, 5, 10, 7);
Stream<Integer> st1 = list.stream();
System.out.println(Arrays.toString(st1.toArray()));

Stream<String> st2 = Stream.of("Maria", "Alex", "Bob");
System.out.println(Arrays.toString(st2.toArray()));

Stream<Integer> st3 = Stream.iterate(0, x -> x + 2);
System.out.println(Arrays.toString(st3.limit(10).toArray()));

Stream<Long> st4 = Stream.iterate(new Long[]{ 0L, 1L }, p->new Long[]{ p[1], p[0]+p[1] }).map(p -> p[0]);
System.out.println(Arrays.toString(st4.limit(10).toArray()));
```

Pipeline (demo)

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Demo - pipeline

```
List<Integer> list = Arrays.asList(3, 4, 5, 10, 7);

Stream<Integer> st1 = list.stream().map(x -> x * 10);
System.out.println(Arrays.toString(st1.toArray()));

int sum = list.stream().reduce(0, (x, y) -> x + y);
System.out.println("Sum = " + sum);

List<Integer> newList = list.stream()
    .filter(x -> x % 2 == 0)
    .map(x -> x * 10)
    .collect(Collectors.toList());
System.out.println(Arrays.toString(newList.toArray()));
```

Exercício resolvido - filter, sorted, map, reduce

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Fazer um programa para ler um conjunto de produtos a partir de um arquivo em formato .csv (suponha que exista pelo menos um produto).

Em seguida mostrar o preço médio dos produtos. Depois, mostrar os nomes, em ordem decrescente, dos produtos que possuem preço inferior ao preço médio.

Veja exemplo na próxima página.

<https://github.com/acenelio/lambda6-java>

Input file:

```
Tv,900.00
Mouse,50.00
Tablet,350.50
HD Case,80.90
Computer,850.00
Monitor,290.00
```

Execution:

```
Enter full file path: c:\temp\in.txt
Average price: 420.23
Tablet
Mouse
Monitor
HD Case
```

<https://github.com/acenelio/lambda6-java>

Exercício de fixação

<http://educandoweb.com.br>

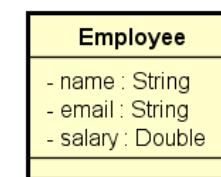
Prof. Dr. Nelio Alves

Fazer um programa para ler os dados (nome, email e salário) de funcionários a partir de um arquivo em formato .csv.

Em seguida mostrar, em ordem alfabética, o email dos funcionários cujo salário seja superior a um dado valor fornecido pelo usuário.

Mostrar também a soma dos salários dos funcionários cujo nome começa com a letra 'M'.

Veja exemplo na próxima página.



<https://github.com/acenelio/lambda7-java>

Input file:

```
Maria,maria@gmail.com,3200.00
Alex,alex@gmail.com,1900.00
Marco,marco@gmail.com,1700.00
Bob,bob@gmail.com,3500.00
Anna,anna@gmail.com,2800.00
```

Execution:

```
Enter full file path: c:\temp\in.txt
Enter salary: 2000.00
Email of people whose salary is more than 2000.00:
anna@gmail.com
bob@gmail.com
maria@gmail.com
Sum of salary of people whose name starts with 'M': 4900.00
```

<https://github.com/acenelio/lambda7-java>

Curso: Programação Orientada a Objetos com Java

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

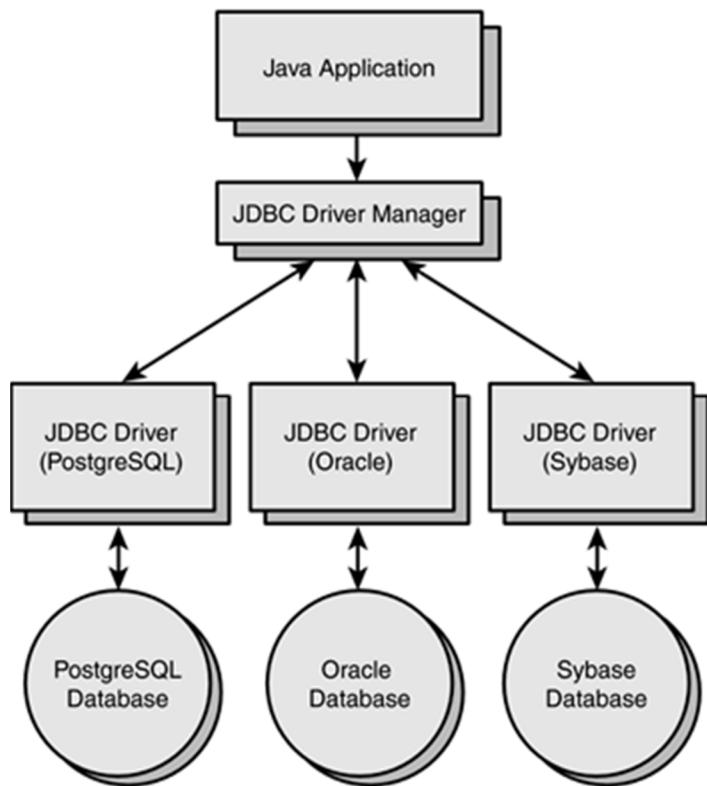
Capítulo: Acesso a banco de dados com JDBC

Objetivo geral:

- Conhecer os principais recursos do JDBC na teoria e prática
- Elaborar a estrutura básica de um projeto com JDBC
- Implementar o padrão DAO manualmente com JDBC

Visão geral do JDBC

- JDBC (Java Database Connectivity): API padrão do Java para acesso a dados
- Páginas oficiais:
 - <https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/>
 - <https://docs.oracle.com/javase/8/docs/api/java/sql/package-summary.html>
- Pacotes: `java.sql` e `javax.sql` (API suplementar para servidores)



Instalação das ferramentas:

- Instalar o MySQL Server e o MySQL Workbench

Preparação do primeiro projeto no Eclipse

Código fonte: <https://github.com/acenelio/jdbc1>

Checklist:

- Usando o MySQL Workbench, crie uma base de dados chamada "coursejdbc"
- Baixar o MySQL Java Connector
- Caso ainda não exista, criar uma User Library contendo o arquivo .jar do driver do MySQL
 - Window -> Preferences -> Java -> Build path -> User Libraries
 - Dê o nome da User Library de MySQLConnector
 - Add external JARs -> (localize o arquivo jar)
- Criar um novo Java Project
 - Acrescentar a User Library MySQLConnector ao projeto
- Na pasta raiz do projeto, criar um arquivo "db.properties" contendo os dados de conexão:

```
user=developer
password=1234567
dburl=jdbc:mysql://localhost:3306/coursejdbc
useSSL=false
```
- No pacote "db", criar uma exceção personalizada DbException
- No pacote "db", criar uma classe DB com métodos estáticos auxiliares
 - Obter e fechar uma conexão com o banco

Demo: recuperar dados

Script SQL: material de apoio ou <https://github.com/acenelio/demo-dao-jdbc/blob/master/database.sql>

Código fonte: <https://github.com/acenelio/jdbc2>

API:

- Statement
- ResultSet
 - first() [move para posição 1, se houver]
 - beforeFirst() [move para posição 0]
 - next() [move para o próximo, retorna false se já estiver no último]
 - absolute(int) [move para a posição dada, lembrando que dados reais começam em 1]

Checklist:

- Usar o script SQL para criar a base de dados "coursejdbc"
- Fazer um pequeno programa para recuperar os departamentos
- Na classe DB, criar métodos auxiliares estáticos para fechar ResultSet e Statement

Atenção: o objeto ResultSet contém os dados armazenados na forma de tabela:

(posição)

	Id	Name
1	1	Computers
2	2	Electronics
3	3	Fashion
4	4	Books

Demo: inserir dados

Código fonte: <https://github.com/acenelio/jdbc3>

API:

- PreparedStatement
- executeUpdate
- Statement.RETURN_GENERATED_KEYS
- getGeneratedKeys

Checklist:

- Inserção simples com preparedStatement
- Inserção com recuperação de Id

Demo: atualizar dados

Código fonte: <https://github.com/acenelio/jdbc4>

Demo: deletar dados

Código fonte: <https://github.com/acenelio/jdbc5>

Checklist:

- Criar DblIntegrityException
- Tratar a exceção de integridade referencial

Demo: transações

Referências: https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.4.0/product-overview/acid.html

Código fonte: <https://github.com/acenelio/jdbc6>

API:

- setAutoCommit(false)
- commit()
- rollback()

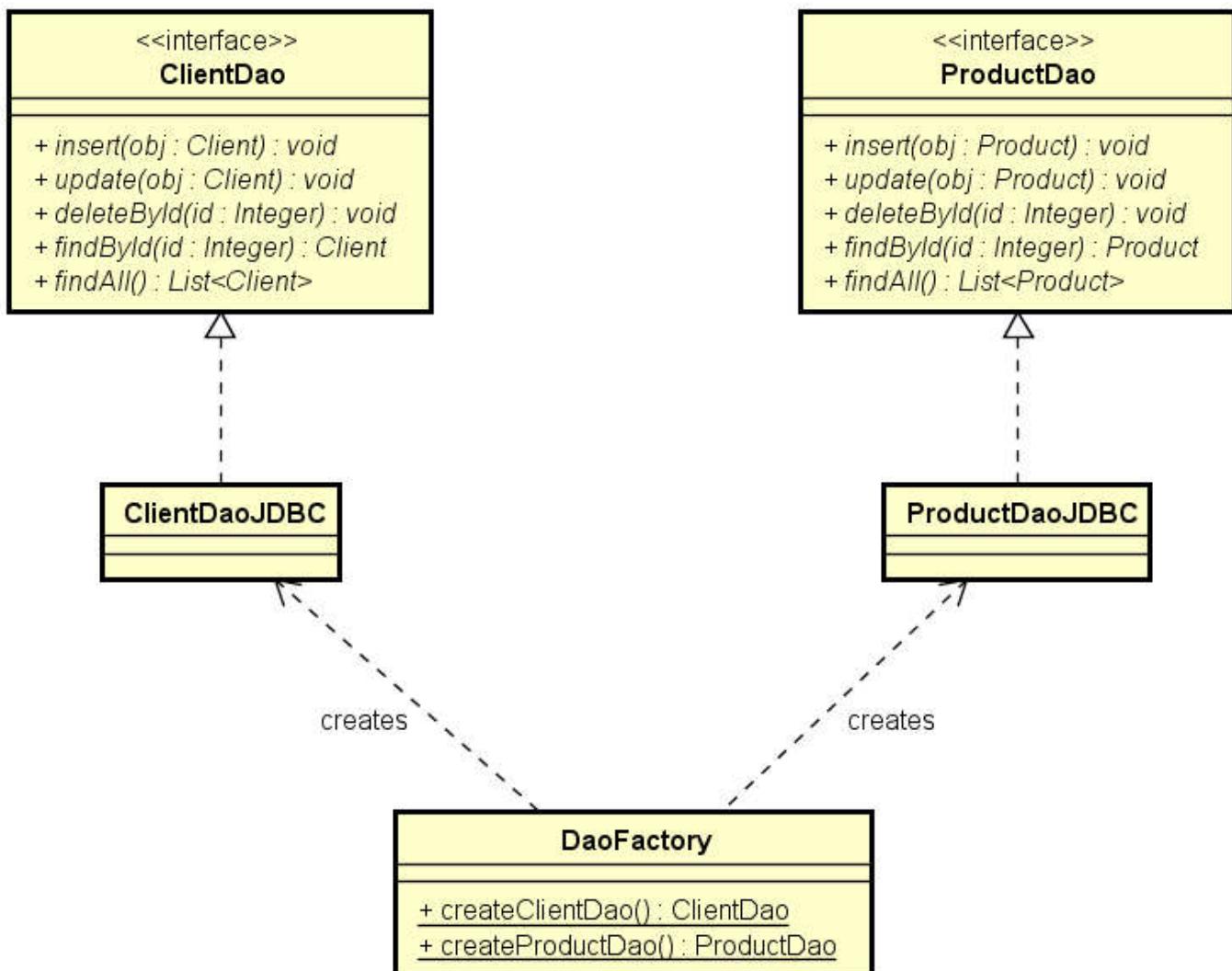
Padrão de projeto DAO (Data Access Object)

Referências:

<https://www.devmedia.com.br/dao-pattern-persistencia-de-dados-utilizando-o-padrao-dao/30999>
<https://www.oracle.com/technetwork/java/dataaccessobject-138824.html>

Ideia geral do padrão DAO:

- Para cada entidade, haverá um objeto **responsável por fazer acesso a dados relacionado a esta entidade**. Por exemplo:
 - Cliente: ClienteDao
 - Produto: ProdutoDao
 - Pedido: PedidoDao
- Cada DAO será definido por uma interface.
- A injeção de dependência pode ser feita por meio do padrão de projeto Factory



Creating project and git repository

Project: <http://github.com/acenelio/demo-dao-jdbc>

Checklist:

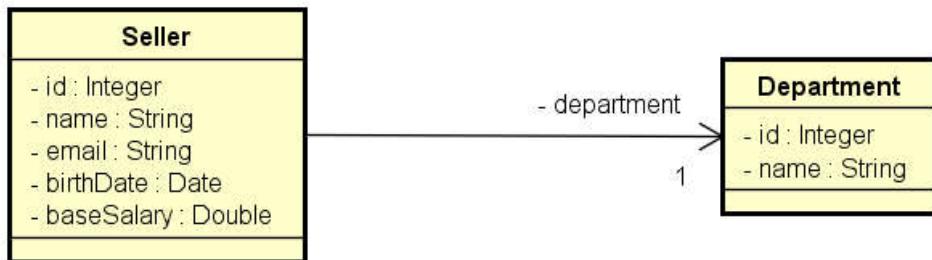
- Github: create a new project
 - **NOTE:** choose `.gitignore` type as Java
- Eclipse: create new java project with MySQLConnector library
 - Copy db package and db.properties from: <https://github.com/acenelio/jdbc5>
- Create local repository and push to Github:

```
git init
git remote add origin https://github.com/acenelio/jdbc-dao-demo.git
git pull origin master
git add .
git commit -m "Project created"
git push -u origin master
```

Department entity class

Entity class checklist:

- Attributes
- Constructors
- Getters/Setters
- hashCode and equals
- toString
- implements Serializable



Seller entity class

(video)

DepartmentDao and SellerDao interfaces

(video)

SellerDaoJDBC and DaoFactory

(video)

findById implementation

SQL Query:

```
SELECT seller.* , department.Name as DepName
FROM seller INNER JOIN department
ON seller.DepartmentId = department.Id
WHERE seller.Id = ?
```

ResultSet (table)

	Id	Name	Email	BirthDate	BaseSalary	DepartmentId	DepName
▶	3	Alex Grey	alex@gmail.com	1988-01-15 00:00:00.000000	2200	1	Computers



associated objects

Reusing instantiation

```
private Seller instantiateSeller(ResultSet rs, Department dep) throws SQLException {
    Seller obj = new Seller();
    obj.setId(rs.getInt("Id"));
    obj.setName(rs.getString("Name"));
    obj.setEmail(rs.getString("Email"));
    obj.setBaseSalary(rs.getDouble("BaseSalary"));
    obj.setBirthDate(rs.getDate("BirthDate"));
    obj.setDepartment(dep);
    return obj;
}

private Department instantiateDepartment(ResultSet rs) throws SQLException {
    Department dep = new Department();
    dep.setId(rs.getInt("DepartmentId"));
    dep.setName(rs.getString("DepName"));
    return dep;
}
```

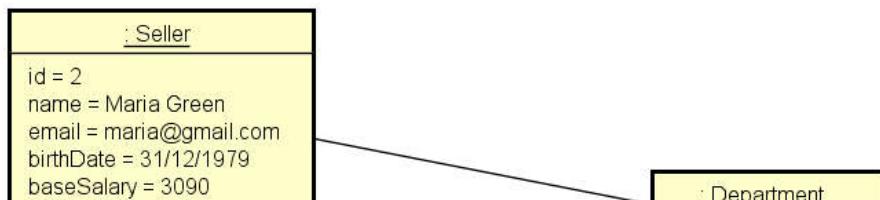
findByDepartment implementation

SQL Query:

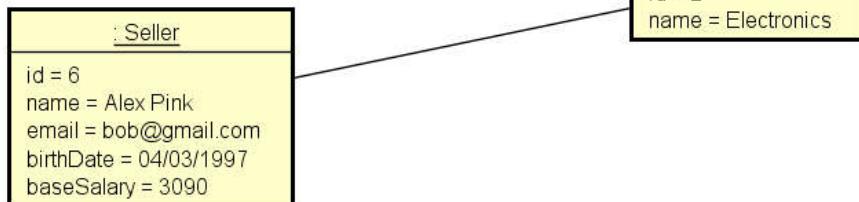
```
SELECT seller.* , department.Name as DepName  
FROM seller INNER JOIN department  
ON seller.DepartmentId = department.Id  
WHERE DepartmentId = ?  
ORDER BY Name
```



INCORRECT



CORRECT



findAll implementation

SQL Query:

```
SELECT seller.* , department.Name as DepName  
FROM seller INNER JOIN department  
ON seller.DepartmentId = department.Id  
ORDER BY Name
```

insert implementation

SQL Query:

```
INSERT INTO seller  
(Name, Email, BirthDate, BaseSalary, DepartmentId)  
VALUES  
(?, ?, ?, ?, ?)
```

update implementation

SQL Query:

```
UPDATE seller  
SET Name = ?, Email = ?, BirthDate = ?, BaseSalary = ?, DepartmentId = ?  
WHERE Id = ?
```

delete implementation

SQL Query:

```
DELETE FROM seller  
WHERE Id = ?
```

DepartmentDao implementation and test

Checklist:

- DepartmentDaoJDBC
- DaoFactory
- Program2

<http://github.com/acenelio/demo-dao-jdbc>