



Treinamento

Java Básico

Aula 1

an NTT DATA Company



Java Básico

| Introdução

- O que é Java ?

É uma Linguagem de programação Orientada a objetos, portável entre diferentes plataformas e sistemas operacionais.

Foi criada na década de 90 pela empresa Sun Microsystems.

Atualmente pertence a empresa Oracle que comprou por US\$ 7,4 bilhões.

Atualmente é a linguagem mais popular utilizada no mundo.
Fonte: <https://computerworld.com.br/2019/01/08/python-mantem-ascensao-e-ganha-ainda-mais-popularidade/>



ORACLE





Características e vantagens da linguagem Java:

- Suporte à orientação a objetos;
- Portabilidade;
- Segurança;
- Linguagem Simples;
- Alta Performance;
- Dinamismo;
- Retrocompatível
- Interpretada (o compilador pode executar os bytecodes do Java diretamente em qualquer máquina);
- Distribuído;
- Independente de plataforma;
- Tipada (detecta os tipos de variáveis quando declaradas);



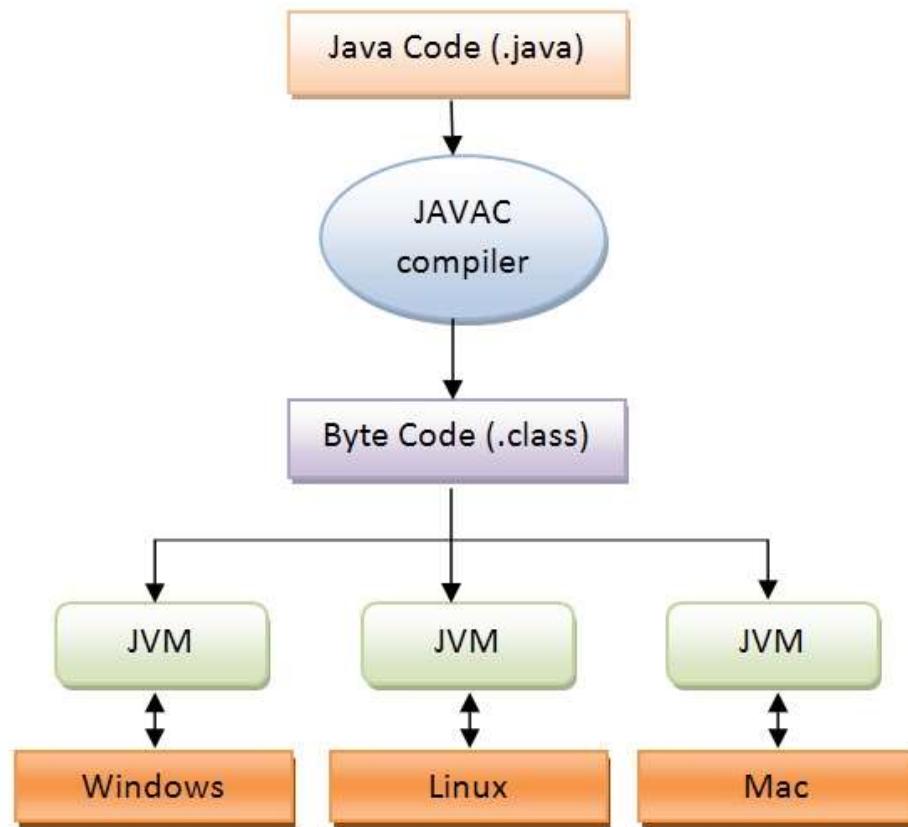
Máquina Virtual Java (Java Virtual Machine - JVM)

É um programa que carrega e executa os aplicativos Java, convertendo os bytecodes em código executável de máquina.

Permite que os programas escritos em Java possam funcionar em qualquer plataforma que possua uma versão da JVM.



- Como funciona a JVM ?



Java Básico

| Introdução



- JDK e JRE

Edições:

- Java Standard Edition (Java SE)
- Java Enterprise Edition (Java EE)
- Java Mobile Edition (Java ME)

Java Development Kit (JDK)

- Ferramentas de desenvolvimento
jdk-8u211-windows-x64.exe

Java Runtime Environment (JRE)

- Necessária para rodar programas Java (bytecodes compilados);
- É a única parte da plataforma Java que os clientes precisam instalar;



- Plataforma de Desenvolvimento

A popularidade da linguagem Java fez com que muitas empresas desenvolvessem ferramentas para facilitar desenvolvimento de aplicações. Estas ferramentas também são conhecidas como IDE (Ambiente de Desenvolvimento Integrado), que embutem uma série de recursos para dar produtividade. Todavia, cada uma delas tem suas próprias particularidades e algumas características semelhantes.

As principais ferramentas do mercado são:

Eclipse (<https://www.eclipse.org/>)

IntelliJ (<https://www.jetbrains.com/idea/>)

NetBeans (<https://netbeans.org/>)

...



- Diferenças entre Java e JavaScript

Em primeiro lugar, o Javascript não é uma linguagem de programação propriamente dita. Ou seja, ela não é compilada e nem executada posteriormente. No mundo dos desenvolvedores ela recebe o nome de linguagem de scripting, pois ela é executada pelo navegador e depende do mesmo para funcionar.

Já os programas feitos com Java são independentes, ou seja, não dependem da plataforma para funcionar. São compilados pela máquina virtual Java para posteriormente entrar em funcionamento.



- Mão na massa, vamos fazer a configuração no Windows.

1. Instalar a JDK.

<https://download.oracle.com/otn/java/jdk/8u211-b12/478a62b7d4e34b78b671c754eaaf38ab/jdk-8u211-windows-x64.exe>

2. Instalar a IDE Eclipse

<https://www.eclipse.org/downloads/download.php?file=/oomph/epp/2019-03/R/eclipse-inst-win64.exe>

3. Configurar a variável de ambiente do Java.

Java Básico

Introdução



- Configurando a variável de ambiente

Variáveis de Ambiente

Variáveis de usuário para dhumbert

Variável	Valor
OneDrive	C:\Users\dhumbert\OneDrive - everis
OneDriveCommercial	C:\Users\dhumbert\OneDrive - everis
Path	C:\Users\dhumbert\AppData\Local\Microsoft\WindowsApps;;C:\U...
TEMP	C:\Users\dhumbert\AppData\Local\Temp
TMP	C:\Users\dhumbert\AppData\Local\Temp

Variáveis do sistema

Variável	Valor
asl.log	Destination=file
ComSpec	C:\WINDOWS\system32\cmd.exe
DEFLOGDIR	C:\ProgramData\McAfee\Endpoint Security\Logs
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_161
JAVA_HOME_MDONE_DESK...	C:\Program Files\Java\jdk1.8.0_161
M2_HOME	C:\dev\apps\apache-maven-3.5.2
NUMBER OF PROCESSORS	4

Novo... **Editar...** **Excluir**

OK **Cancelar**

Editar o path e adicionar a variável criada.

Editar a variável de ambiente

C:\ProgramData\DockerDesktop\version-bin	Novo
C:\Program Files\Docker\Docker\Resources\bin	Editar
C:\Program Files (x86)\Common Files\Oracle\Java\javapath	Procurar...
C:\ProgramData\Oracle\Java\javapath	Excluir
%SystemRoot%\system32	Mover para Cima
%SystemRoot%\System32\Wbem	Mover para baixo
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\	Editar texto...
C:\Program Files\Git\cmd	
C:\Program Files\PutTY\	
C:\Program Files\TortoiseSVN\bin	
%M2_HOME%\bin	
%JAVA_HOME%\bin	
C:\Program Files\nodejs\	

OK **Cancelar**



- Validando a configuração da variável de ambiente
 - Abrir o cmd
 - Executar o seguinte comando:
 - java -version

```
Microsoft Windows [versão 10.0.16299.1087]
(c) 2017 Microsoft Corporation. Todos os direitos reservados.

C:\Users\dhumbert>java -version
java version "1.8.0_181"
Java(TM) SE Runtime Environment (build 1.8.0_181-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.181-b13, mixed mode)

C:\Users\dhumbert>
```



- Criando seu primeiro programa Java.

Abrir o notepad e copiar o texto exatamente igual o trecho abaixo:

```
public class Teste {  
    public static void main (String [] args) {  
        System.out.println("Meu primeiro exercicio de programacao, obrigado Everis");  
    }  
}
```

Salvar o arquivo com o nome Teste.java



- Criando seu primeiro programa Java.

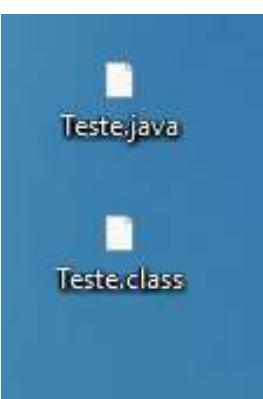
Após salvar o seu arquivo, abra o cmd e acesse a pasta onde foi salvo o arquivo e execute o seguinte comando:

Prompt de Comando

```
C:\Users\dhumbert\Desktop>javac Teste.java  
C:\Users\dhumbert\Desktop>
```



- Criando seu primeiro programa Java.



Execute o comando abaixo e veja o resultado do seu primeiro programa.

```
Prompt de Comando
C:\Users\dhumbert\Desktop>javac Teste.java
C:\Users\dhumbert\Desktop>java Teste
Meu primeiro exercicio de programacao, obrigado Everis
C:\Users\dhumbert\Desktop>
```



Toda aplicação (não toda classe!) deve possuir um método `public static void main(String[] args) {...}`

A assinatura do método contém três modificadores:

- `public` - pode ser invocado por qualquer objeto;
- `static` - método de classe (pode ser acessado globalmente);
- `void` - o método não retorna valor;
- `System` – um nome de Classe;
- `out` – um nome de Objeto – saída de dados;
- `println()` ou `print()` – uma chamada de um método;

Quando o interpretador executa a aplicação, começa por chamar o método `main`, que, por sua vez, chama os outros métodos existentes na aplicação.



Números Primitivos

Tipo	Descrição	Bits	Intervalo	Exemplo
short	Números Inteiros (intervalo reduzido)	16	-32.768 a 32.767	short a; a = 4;
int	Números Inteiros	32	-2^{31} a $2^{31}-1$	int x; x = 4;
long	Números Inteiros (intervalo maior)	64	-2^{63} a $2^{63}-1$	long y; y = 4L;
float	Números fracionários	32	$\pm 1.5 \times 10^{45}$ a $\pm 3.4 \times 10^{38}$	float w; w = 0.4f;
double	Números fracionários (maior precisão)	64	$\pm 5.0 \times 10^{-324}$ a $\pm 1.7 \times 10^{308}$	double z; z = 0.4d;



Outros Primitivos

Tipo	Descrição	Bits	Intervalo	Exemplo
String (objeto)	Sequência de Caracteres	16 por caractere	Não aplicável	String str; str = "Olá!";
char	Caractere Único	16	Caractere Unicode 0 a 65.535	char letra; letra = 'x';
boolean	Valor Lógico	8	true ou false	boolean algo; algo = true;



Declaração de Variáveis

Variável é um espaço reservado na memória do computador para armazenar um tipo de dado.

Devem receber nomes para poderem ser referenciadas e modificadas quando necessário.

Toda variável tem: um tipo de dado, nome e um valor

```
String nome = "Danilo Humberto";
```



```
int a; /* declara uma variável do tipo int */  
float b; /* declara uma variável do tipo float */  
double c; /* declara uma variável do tipo double */  
int d, e; /* declara duas variáveis do tipo int */  
int f = 10; /* declaração e inicialização da variável */
```



Convenção de nomes

Nome de Classes:

Cada palavra do nome da classe começa com letra maiúscula;
Todas as palavras são unidas;

Ex.:

JTextField

Integer

Math

ActionEvent

Aluno

everis

an NTT DATA Company



Convenção de nomes

Métodos:

Começam com letra minúscula e devem apresentar os caracteres “(“ e “)”;

Havendo mais do que uma palavra para identificar o nome de um método, cada uma delas terá sua primeira letra em maiúsculo a partir da segunda.

Ex.:

```
getText();
setText("Programar em Java é bom!!!");
copy(Text);
somaArea(A, B);
```



Convenção de nomes

Variáveis primitivas e variáveis de referência (de Objetos):

Seguem as mesmas regras para definição de nomes de Métodos.

Ex: (primitiva):

```
int x, y;  
double d1;  
char ch, ch2;
```

Ex: (referência):

```
JTextField jtfValor;  
String nomePessoa;  
Funcionario funcionario1, funcionario2;  
Endereco endereco;
```



Convenção de nomes

Todas as letras em maiúsculo.

Ex.:

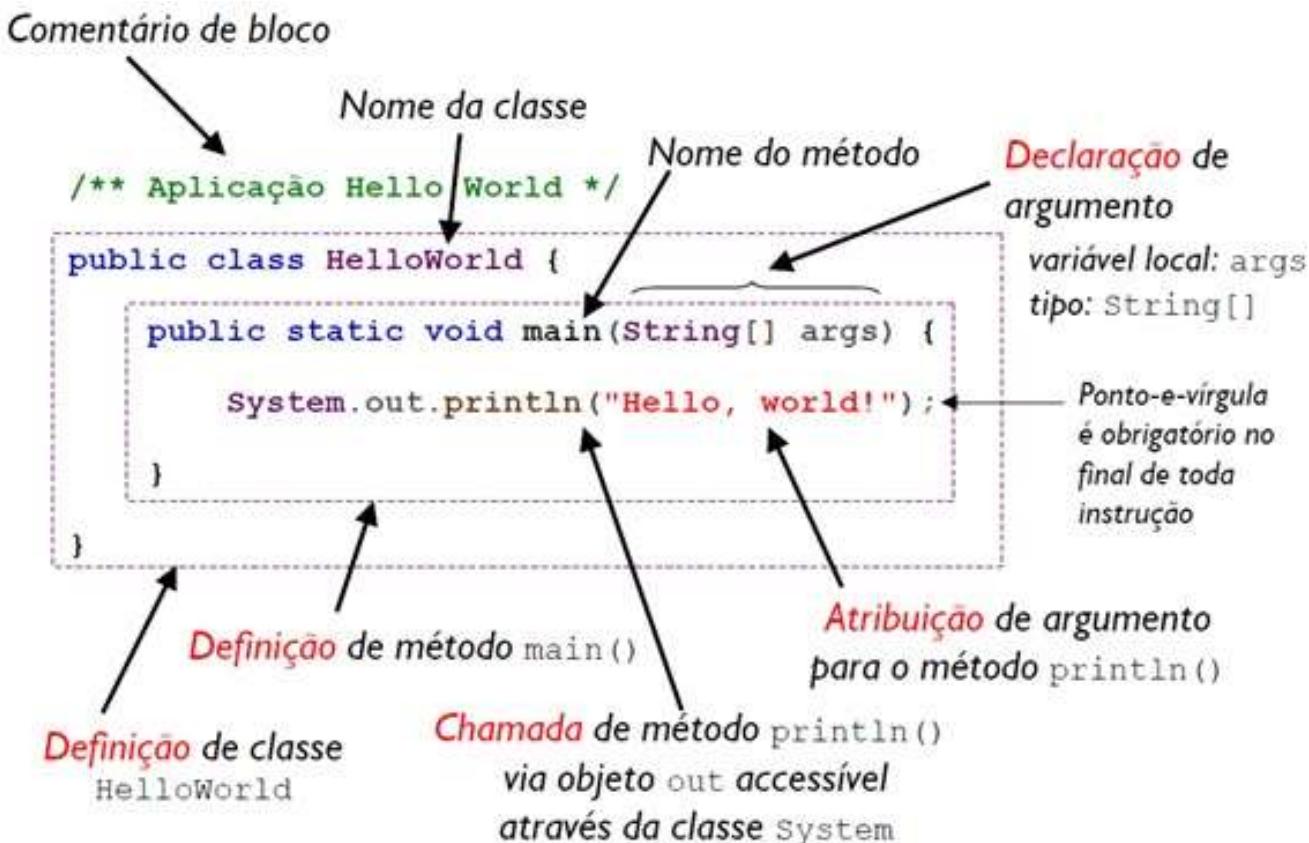
- PI
- DEFAULT_KEYMAP
- LEFT

Literal:

- 10
- 342.34
- 'A'
- "Linguagem de Programação Java"



Operação	Operador	Expressão algébrica	Expressão Java
Adição	+	$a + 1$	<code>a +1</code>
Subtração	-	$b - 2$	<code>b -2</code>
Multiplicação	*	$c m$	<code>c * m</code>
Divisão	/	d / e	<code>d / e</code>
Resto	%	$f \text{ mod } g$	<code>f % g</code>





- Criar três variáveis inteiros e atribuir valores a elas.
- Somar as três variáveis e imprimir o resultado.



- Utilizando IDE para desenvolvimento



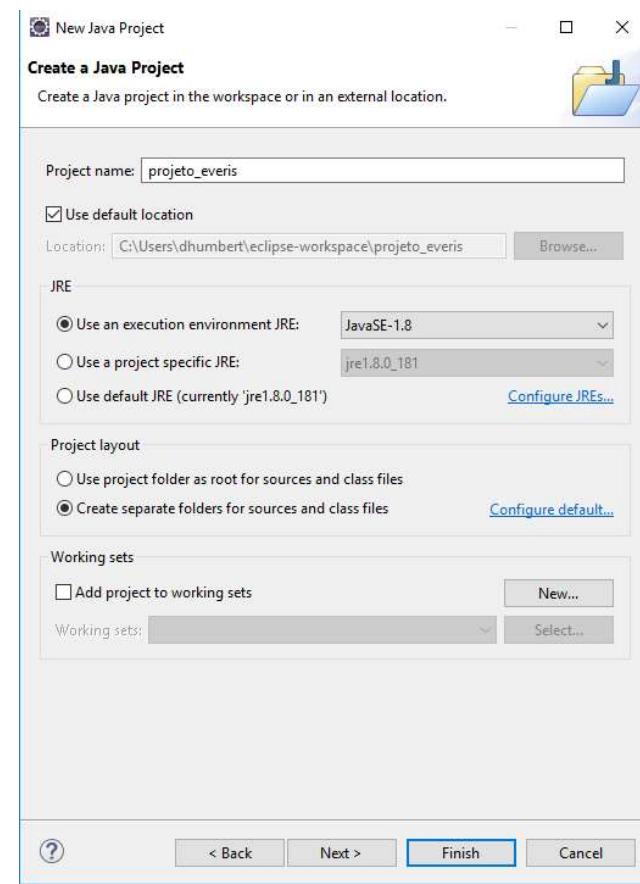
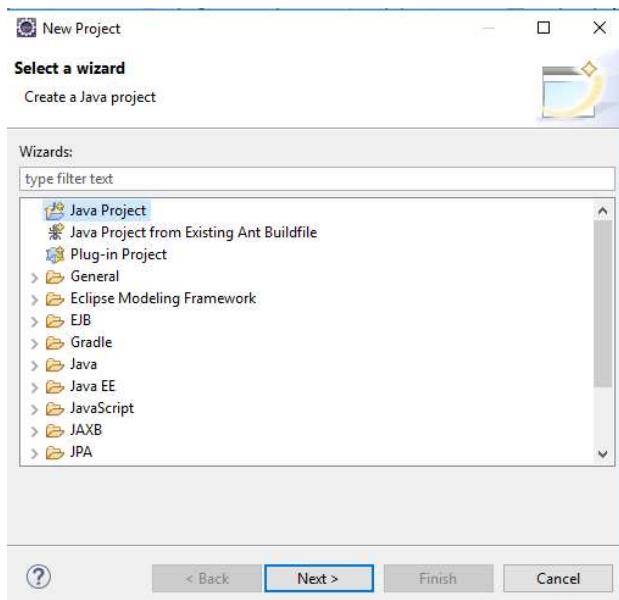
Java Básico

| Prática



- Criando um projeto

- File > New > Project

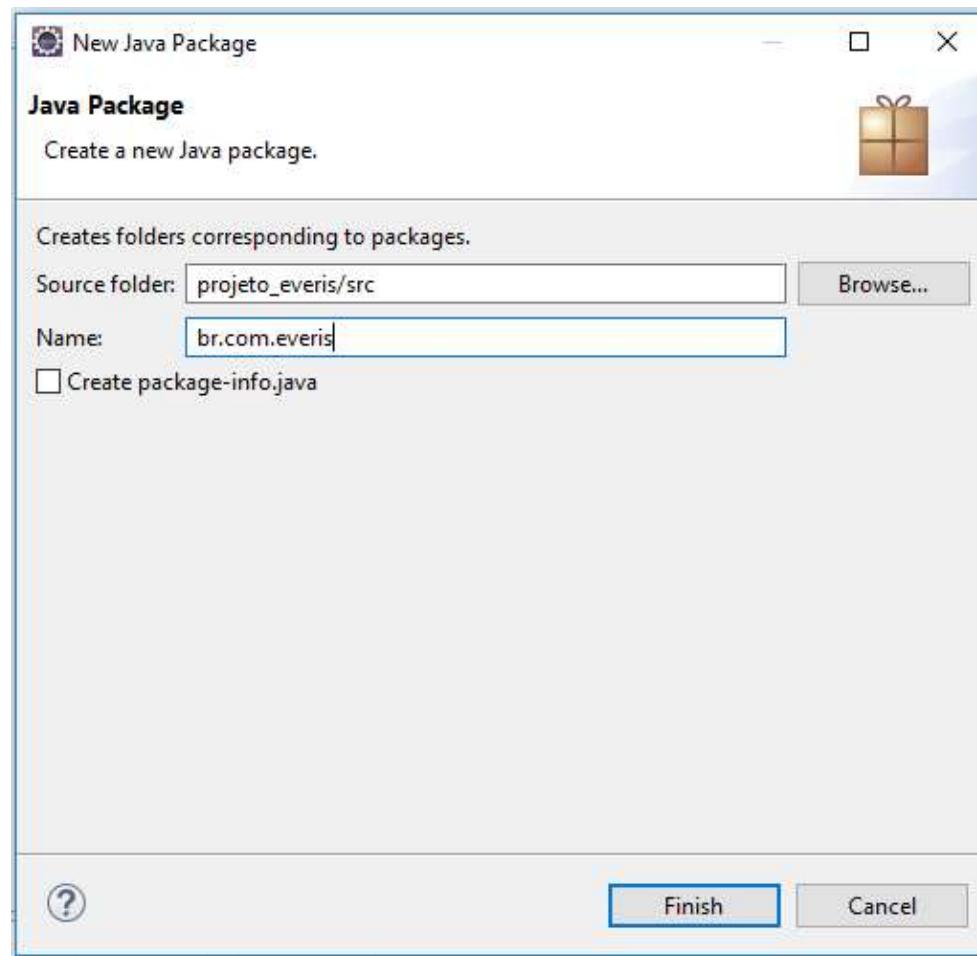


Java Básico

| Prática



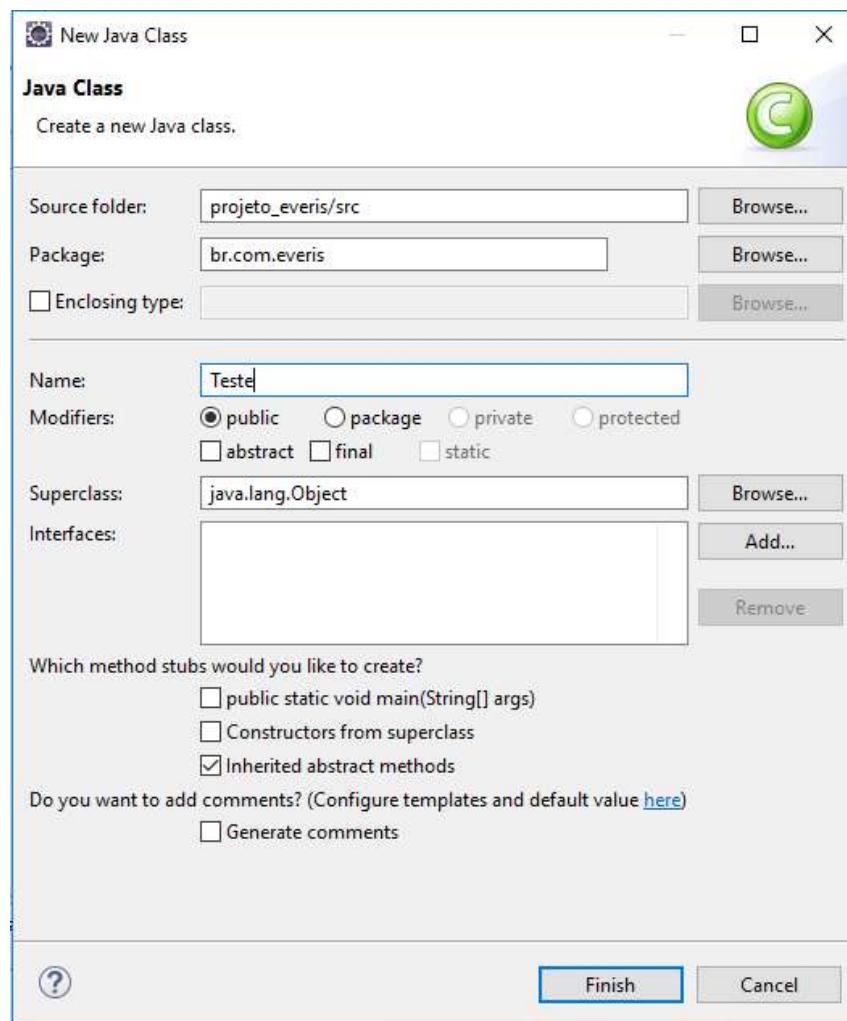
- Criando um pacote



Java Básico

- Criando uma Classe

| Prática



an NTT DATA Company

everis



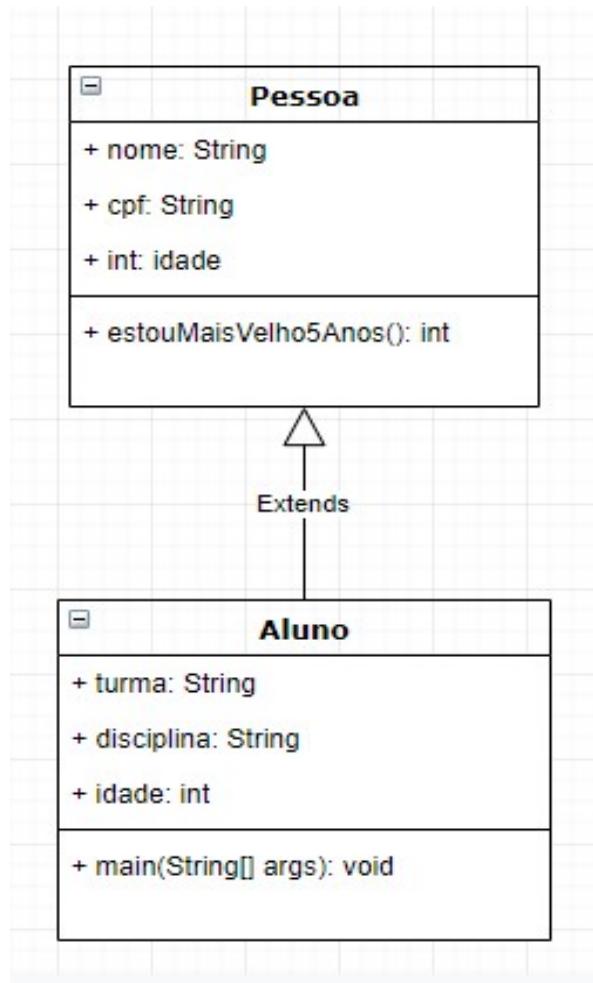
- Replicar código criado no notepad ++ para executarmos pela IDE

```
public class Teste {  
    public static void main (String [] args) {  
        System.out.println("Meu primeiro exercicio de programacao, obrigado Everis");  
    }  
}
```

Exercício

Criar a classe conforme modelo ao lado.

Criar um main na classe Aluno, instanciar a classe e chamar o método estouMaisVelho5Anos.





Treinamento

Java Básico

Aula 2

everis

an NTT DATA Company



- Seguindo o exemplo abaixo:
 1. Use o dialog para atribuir valores a todos os atributos da classe Aluno.java
 2. Exiba o retorno usando o método `toString()` e utilizando o `ShowMessageDialog`.
 3. A partir da classe Aluno.java chame o método `estouMaisVelho5Anos()` da classe Pessoa e exiba o retorno com o `ShowMessageDialog`.

```
J Teste.java X
1 package br.com.everis;
2
3 import javax.swing.JOptionPane;
4
5 public class Teste {
6     public static void main(String[] args) {
7
8         String nome = JOptionPane.showInputDialog("Qual o seu nome?");
9
10        if (nome == null || nome.equals("")) {
11            JOptionPane.showMessageDialog(null, "Que pena você não respondeu a pergunta", "Ops", JOptionPane.ERROR_MESSAGE);
12            return;
13        }
14
15        JOptionPane.showMessageDialog(null, "Seu nome é: " + nome, "Alerta", JOptionPane.INFORMATION_MESSAGE);
16    }
17}
```

Java Básico

Principais Características

- Simples
- Orientada a objetos
- Familiar
- Robusta
- Segura
- Portável

Paradigmas

- OO
- Imperativo
- Refletivo
- Genérico

| Java – Características

Sistema de Tipos

- Forte;
- Estática;
- Segura;

Versões

- 8;
- 11;
- 12;

Extensões

- java;
- jar/war/ear;
- class;

Compiladores

- OpenJDK (javac, sjavac);
- GNU Compiler for Java (GCJ);
- Eclipse Compiler for Java (ECJ);

an NTT DATA Company



POO



Programação Orientada à Objetos

Objetivo:

Aproximar mundo digital do mundo real!

Objeto:

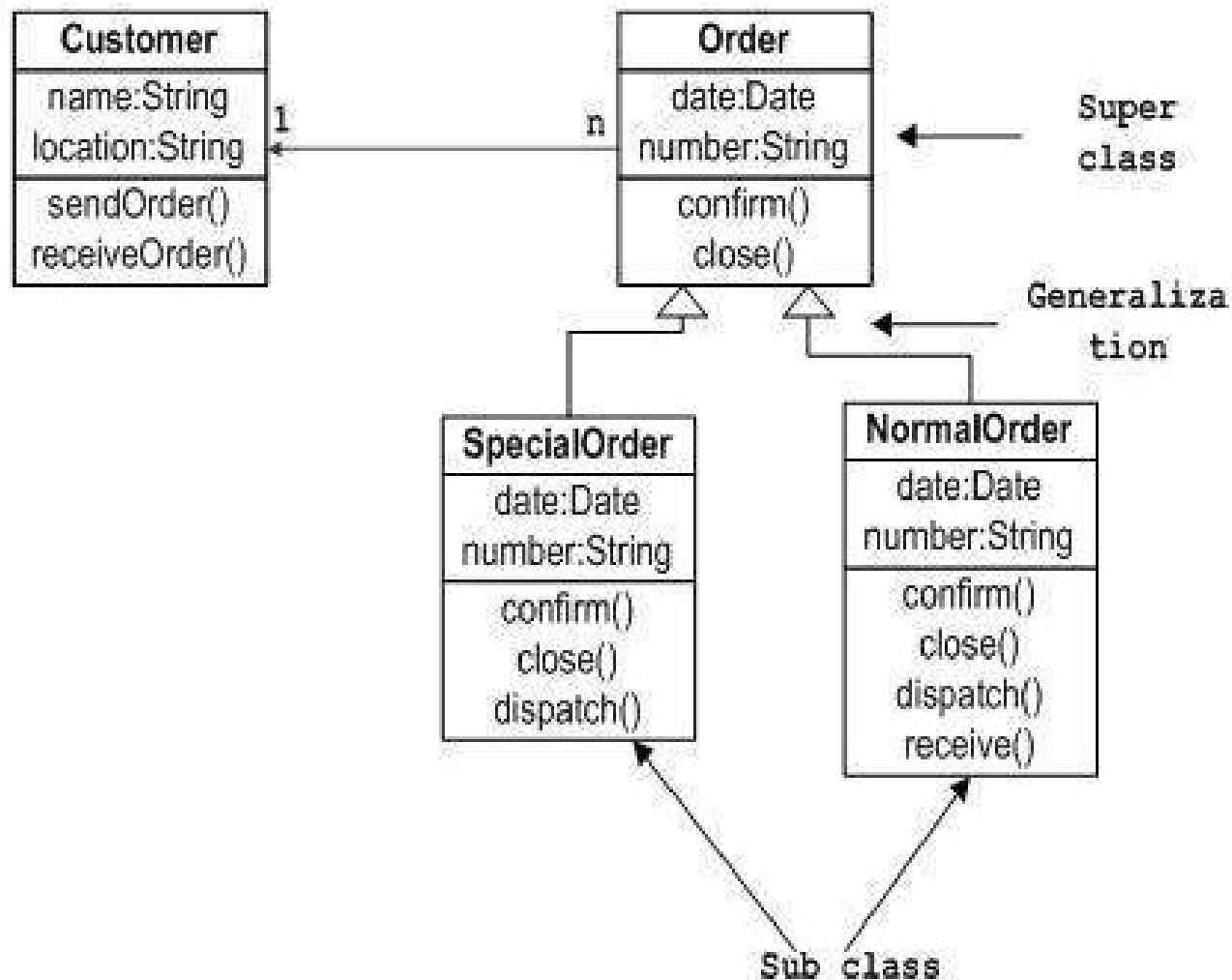
Coisa material ou abstrata que pode ser percebida pelos sentidos e descrita por meio das suas características, comportamento e estado atual.

Criador:

Alan Key (*SmallTalk*, 1969)

Java Básico

| POO – Diagrama de Classes



Java Básico

| POO – Classes vs. Objetos

Classe

Define métodos e atributos comuns que serão compartilhados por um objeto.



Objeto

Instância de uma classe.



everis

an NTT DATA Company

Java Básico

Conceitos Básicos

Classes

São receitas de um objeto, aonde têm características e comportamentos, permitindo assim armazenar propriedades e métodos dentro dela.

Métodos

São as ações que os objetos podem exercer quando solicitados, onde podem interagir e se comunicarem com outros objetos.

Atributos

São as características de um objeto, essas características também são conhecidas como variáveis.

Pacotes

São conjuntos de classes localizadas na mesma estrutura hierárquica de diretórios.

Constructor

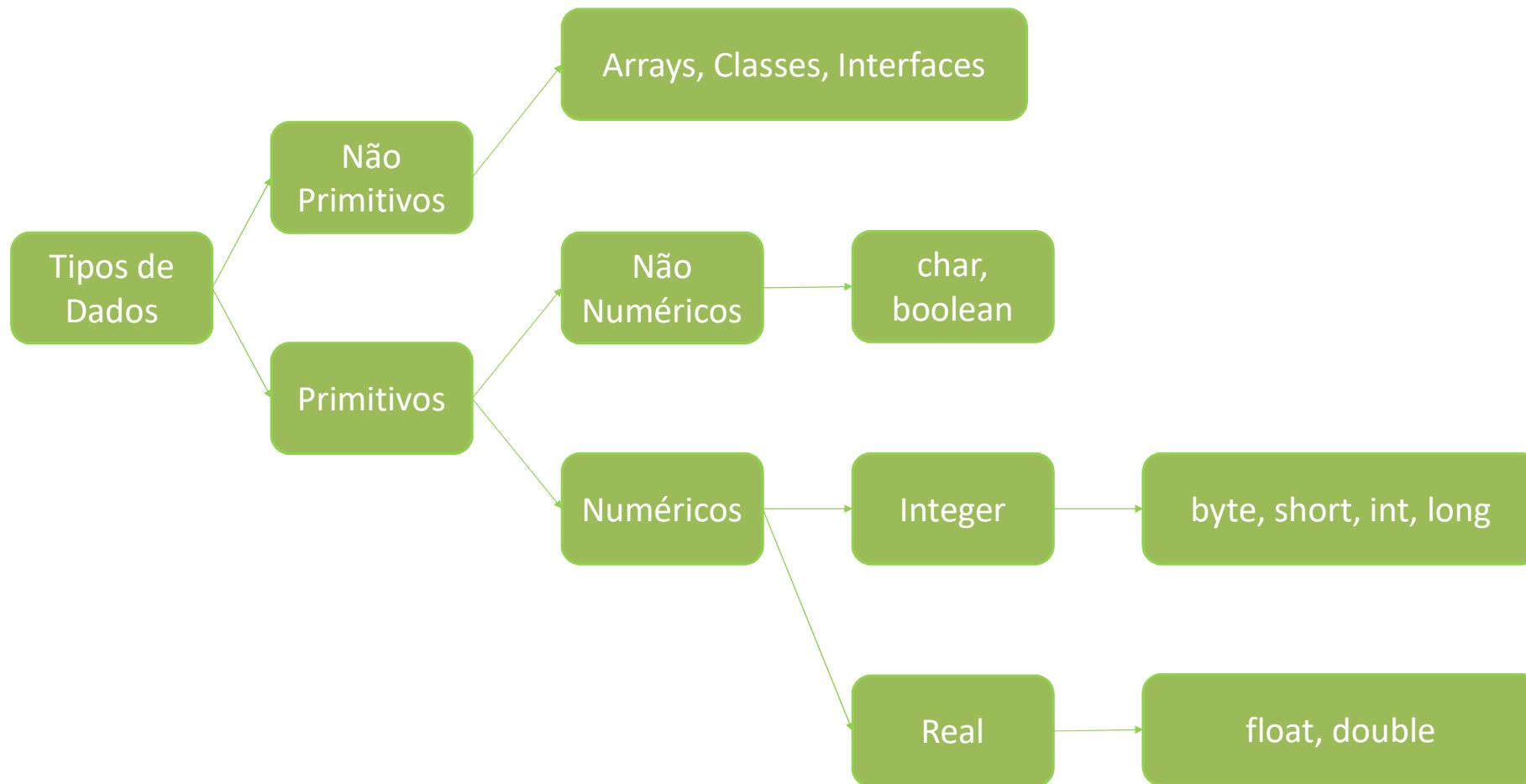
É um método chamado assim que uma nova instância do objeto for criada. Tal método geralmente é responsável pela alocação de recursos necessários ao funcionamento do objeto além da definição inicial das variáveis de estado (atributos).

Getter e Setter

Eles são usados como facilitadores em casos de atributos que deve ser acessados de forma simples e direta para pegar o valor de um atributo (get) ou para alterar o valor deste atributo (set).

Eles são chamados também de acessores e mutadores.

Name	Convention
class name	should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread etc.
interface name	should start with uppercase letter and be an adjective e.g. Runnable, Remote, ActionListener etc.
method name	should start with lowercase letter and be a verb e.g. actionPerformed(), main(), print(), println() etc.
variable name	should start with lowercase letter e.g. firstName, orderNumber etc.
package name	should be in lowercase letter e.g. java, lang, sql, util etc.
constants name	should be in uppercase letter. e.g. RED, YELLOW, MAX_PRIORITY etc.



Como inicializar um objeto?

```
Objeto o = new Object();
```

O que o “new” faz?

Ele invoca o método construtor do objeto em questão.

O que acontece se eu não inicializar o objeto?

O objeto fica com o valor inicial de acordo com o seu tipo de dado.

Valores iniciais

- boolean: false
- byte: 0
- short: 0
- int: 0
- long: 0L
- char: \u0000
- float: 0.0f
- double: 0.0d
- object: null

IMPORTANTE!

Todo “objeto” cujo tipo não seja primitivo deve ser inicializado antes do uso.
Caso contrário, ao tentar acessá-lo, ocorrerá um erro!

Java Básico

| Literais

Literais numéricos

```
int decimalValue = 26;
```

```
int hexadecimalValue = 0x1A;
```

```
int binaryValue = 0b11010;
```

```
double doubleValue = 123.4;
```

```
float floatValue = 123.4f;
```

Literais de caracter e string

```
char charValue = 'a';
```

```
String stringValue = "abc\n";
```

Literal de classe

```
Class classValue = String.class;
```

everis

an NTT DATA Company

Modificadores de Visibilidade

- + • **public:** Uma declaração com o modificador *public* pode ser acessada de qualquer lugar e por qualquer entidade que possa visualizar a classe a que ela pertence.
- • **private:** Os membros da classe definidos como não podem ser acessados ou usados por nenhuma outra classe. Esse modificador não se aplica às classes, somente para seus métodos e atributos. Esses atributos e métodos também não podem ser visualizados pelas classes herdadas.
- # • **protected:** O modificador *protected* torna o membro acessível às classes do mesmo pacote ou através de herança, seus membros herdados não são acessíveis a outras classes fora do pacote em que foram declarados.
- ~ • **default (padrão):** A classe e/ou seus membros são acessíveis somente por classes do mesmo pacote, na sua declaração não é definido nenhum tipo de modificador, sendo este identificado pelo compilador.

Modificadores de controle de acesso

default – visível para o pacote

private – visível apenas para a classe

public – visível para o mundo

protected – visível para o pacote e para as subclasses

São utilizados para classes, variáveis e métodos

Outros modificadores de classe

final – inibe especialização

abstract – inibe instanciação da classe – deve ser especializada

Outros modificadores de atributos e métodos

final – inibe sobrescrita (método), inibe alteração (atributo)

static – atributos e métodos recebem escopo de classe

abstract – método sem implementação, deve ser sobreescrito por classe filha

transient – atributos e métodos não são serializados

synchronized – método acessado de forma exclusiva por múltiplas threads

Java Básico

| Java – Operadores

Operador de Atribuição

- =

Operadores Aritméticos

- +
- -
- *
- /
- %

Operadores Unários

- +
- -
- ++
- --
- !

Operadores de Igualdade e Relacionais

- ==
- !=
- >
- >=
- <
- <=

Operadores Condicionais

- &&
- ||
- ?:

Operador de Comparação de Tipo

- instanceof

an NTT DATA Company



Java Básico

Declaracões

Variáveis

```
modifier Type variable;  
modifier Type variable = expression;
```

Métodos

```
modifier void method( ... ){  
    //ação  
}
```

```
modifier Type method( ... ){  
    //ação  
    return object;  
}
```

```
public ClassName( ... ){  
    //construtor  
}
```

Constantes

```
static final Type VARIABLE = expression;
```

Classe

```
public class ClassName {  
    //atributos  
  
    //construtor  
  
    //métodos  
}
```

Cast

```
Type variable = (Type)otherVariable;
```

A linguagem de programação Java possui **52** palavras reservadas. Seguem abaixo algumas delas:

Retorno de um Método

- **void**: indica que o método não tem retorno

Modificadores de classes, variáveis ou métodos

- **final**: impossibilita que uma classe seja estendida, que um método seja sobreescrito ou que uma variável seja reinicializada
- **static**: faz um método ou variável pertencer à classe ao invés de às instâncias

Variáveis de Referência

- **super**: refere-se a superclasse imediata
- **this**: refere-se a instância atual do objeto

Controle de Pacotes

- **import**: importa pacotes ou classes para dentro do código
- **package**: especifica a que pacote todas as classes de um arquivo pertencem

Como executar meu programa Java!

Definindo a classe “MainClass”

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

Definindo o método “main”

Definindo o argumento “args” para o método “main”

Método que imprime a frase “Hello World”

Java Básico

| Escopo de variáveis

```
public class Example {  
    public int x; ————— Variável de instância  
    private static final int CONSTANTE = 10; ————— Variável de classe
```

Método

```
    public add(int y) { ————— Parâmetro  
        Variável local ← int z = x + y;  
        return z;  
    }  
}
```

Acessando variável de instância

```
Example example = new Example();  
example.x;
```

Acessando variável de classe

```
int c = Example.CONSTANTE;
```

everis

an NTT DATA Company

Java Básico

| Java – Wrappers

Definição

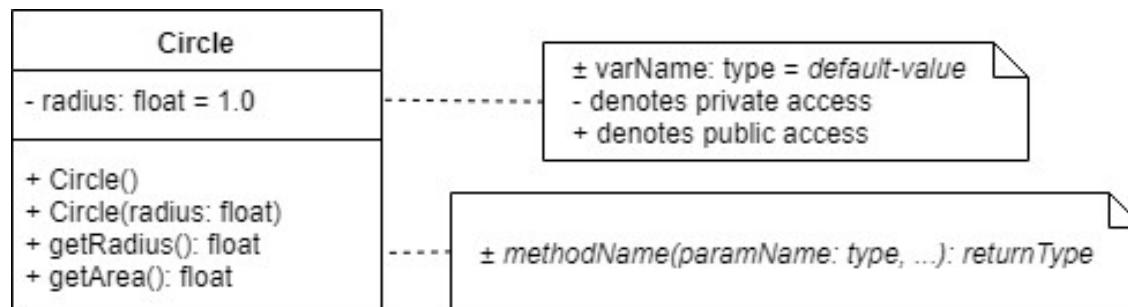
Classes especiais que possuem métodos capazes de fazer conversões em variáveis primitivas e também de **encapsular tipos primitivos para serem trabalhados como objetos**.

Tipo Primitivo	Classe Wrapper	Argumentos do Construtor
boolean	Boolean	boolean ou String
byte	Byte	byte ou String
char	Character	char
int	Integer	int ou String
float	Float	float, double ou String
double	Double	double ou String
long	Long	long ou String
short	Short	short ou String

String: é um “wrapper” de um vetor de “char”.

Exercício

Passo 1: crie uma classe Java chamada *Circle*, conforme o diagrama abaixo:



Passo 2: Adicione um método *main* à classe *Circle*:

- Deve criar duas instâncias de *Circle*;
- Devem ser utilizados os dois constructors;
- Deve imprimir no console os valores de *radius* e *area* para ambos os círculos;

Passo 3:

- Adicione um atributo *color* do tipo *String* (*default "red"*), com *getter*;
- Adicionar um novo constructor que deve receber ambos os valores de atributos.

everis

an NTT DATA Company

Abstração

Os objetos armazenam dados e possuem interface definida que determina o seu comportamento.

Encapsulamento

É a técnica utilizada para esconder uma ideia, ou seja, não expor detalhes internos para o usuário, tornando partes do sistema mais independentes possível.

Herança

A herança é um mecanismo da OO que permite criar novas classes a partir de classes já existentes, aproveitando-se das características existentes na classe a ser estendida. Com a herança é possível criar subclasses, a partir de classes bases, superclasses.

Java Básico

Java – Herança

O conceito de herança em Java é implementado através da palavra chave **extends**

```
public class Funcionario {  
    private final String nome;  
    private final String cpf;  
    private final double salario;  
  
    public Funcionario(String nome, String cpf, double salario){  
        this.nome = nome;  
        this.cpf = cpf;  
        this.salario = salario;  
    }  
  
    public String getNome() {  
        return nome;  
    }  
  
    public String getCpf() {  
        return cpf;  
    }  
  
    public double getSalario() {  
        return salario;  
    }  
}
```

```
public class Gerente extends Funcionario {  
  
    private int numeroFuncionarios;  
  
    public Gerente(String nome, String cpf, double salario){  
        super(nome, cpf, salario);  
        this.numeroFuncionarios = 0;  
    }  
  
    public int getNumeroFuncionarios() {  
        return numeroFuncionarios;  
    }  
  
    public void setNumeroFuncionarios(int numeroFuncionarios) {  
        this.numeroFuncionarios = numeroFuncionarios;  
    }  
}
```

everis

an NTT DATA Company

Sobrescrita (Override)

Um novo método na classe filha contendo a mesma assinatura e mesmo tipo de retorno do método sobrescrito.

```
public class Gerente extends Funcionario {  
  
    String setor;  
  
    @Override  
    public double calculaBonus() {  
        return this.salario * 0.20;  
    }  
}
```

Sobrecarga (Overload)

Permite métodos de mesmo nome, mas com suas assinaturas diferentes

```
public class Calculadora {  
  
    public int soma(int num1, int num2) {  
        return num1 + num2;  
    }  
  
    public double soma(double num1, double num2) {  
        return num1 + num2;  
    }  
}
```

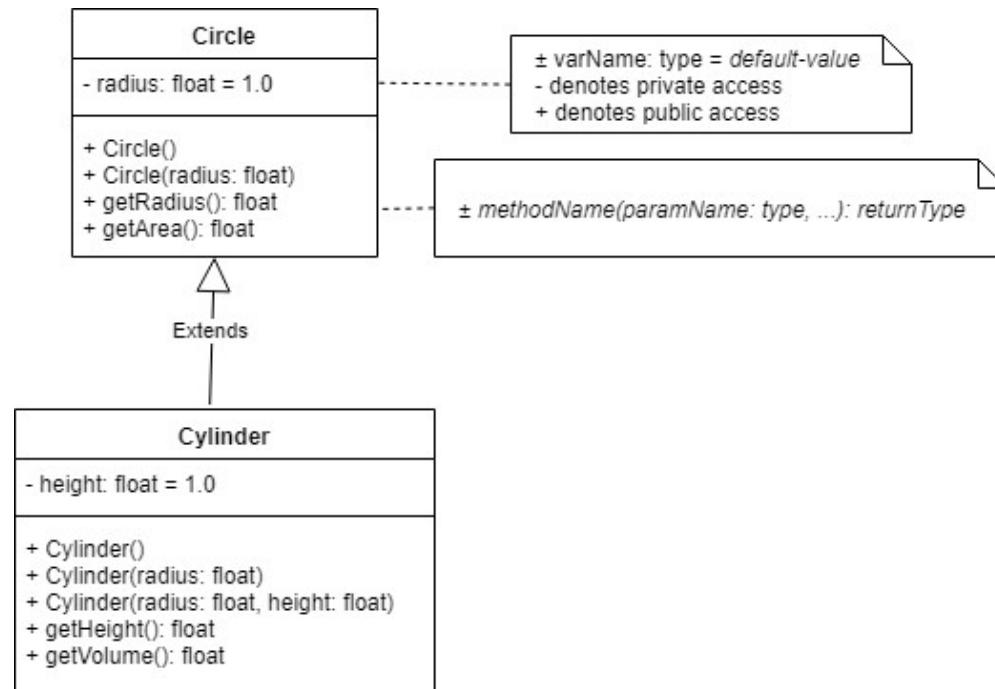
Sobrescrita versus ocultação

```
public class Animal {  
    public void instanceMethod() {  
        System.out.println("The instance method in Animal");  
    }  
  
    public static void classMethod() {  
        System.out.println("The class method in Animal");  
    }  
}  
  
  
public class Cat extends Animal {  
    @Override  
    public void instanceMethod() {  
        super.instanceMethod();  
        System.out.println("The instance method in Cat");  
    }  
  
    public static void classMethod() {  
        System.out.println("The class method in Cat");  
    }  
  
    public static void main(String[] args) {  
        Cat myCat = new Cat();  
        Animal myAnimal = myCat;  
        Animal.classMethod();  
        myAnimal.instanceMethod();  
    }  
}
```

A classe **Cat** sobrescreve o método *instanceMethod* de **Animal** e oculta o método *classMethod* de **Animal**

Exercício

Passo 1: Crie uma classe Java chamada *Cylinder*, conforme o diagrama abaixo:

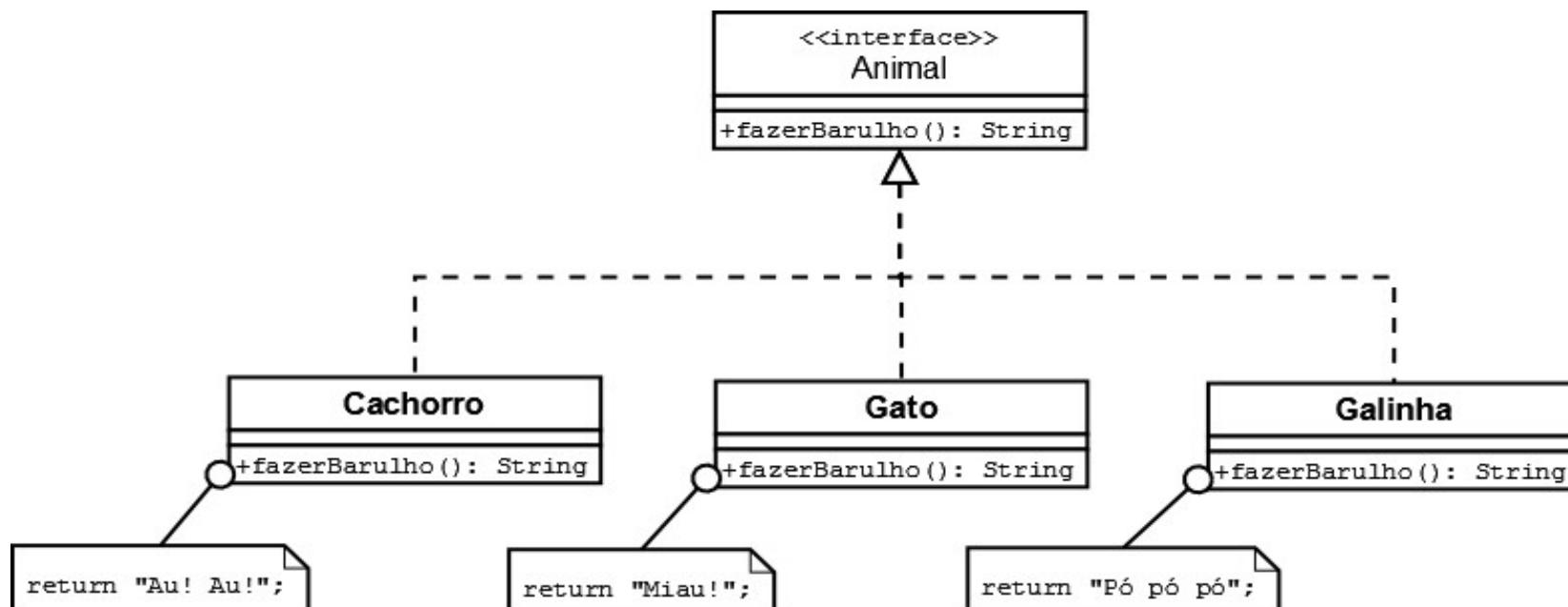


- Passo 2: Na classe *Cylinder*, faça sobrescrita do método *getArea()*
- $2 * \pi * r * (r+h) \rightarrow \text{AREA CYLINDER}$
- $\pi * R^2 \rightarrow \text{AREA CIRCLE}$

Polimorfismo

É a habilidade de diferentes objetos receberem a mesma mensagem e comportar-se de maneira diferente.

Polimorfismo implica em criar uma interface pela qual serão recebidas mensagens determinando o comportamento do objeto.



- Interface é contrato que deve ser seguido por uma implementação
- Especifica o que uma classe deve fazer, mas não como

Interfaces possuem restrições

- constantes
- assinaturas de métodos
- métodos *static*
- métodos *default* (**Java 8+**)

Palavras reservadas

- **interface**: declaração
- **implements**: implementação

Java Básico

Interfaces

```
public interface Animal {  
    void sound();  
    void run();  
}  
  
public class Cat implements Animal {  
    @Override  
    public void sound() {  
        System.out.println("Miau");  
    }  
  
    @Override  
    public void run() {  
        System.out.println("run forrest run");  
    }  
}
```

- **Animal** não pode ser instanciado (via **new()**), desta forma não pode possuir constructor;
- **Cat** pode ser instanciado
- **Cat** deve implementar todos os métodos da interface (caso contrário deveria ser *abstract*)
- Métodos de interface por default são ambos *public* e *abstract*

everis

an NTT DATA Company

Java Básico

Object

equals e hashCode

- Em Java toda classe é descendente direto ou indireto da classe **Object**;
- Desta forma toda classe herda os métodos de instância de Object:
 - **clone()** : cria e retorna uma cópia do objeto;
 - **equals(Object obj)** : indica se outro objeto é igual a este;
 - **hashCode()** : calcula o hash code do objeto;
 - **getClass()** : retorna a classe (runtime) do objeto;
 - **toString()** : retorna uma representação string do objeto;
 - **finalize()** : utilizado pelo garbage collector;
- Pode-se utilizar as implementações default de Object, mas quando se trabalha com Collections é essencial sobrescrever **equals** e **hashCode**;

Java Básico

equals e hashCode

O método equals

- Compara dois objetos e retorna true se forem iguais
- Em Object é implementado como comparação das referências (==)

```
public class Book {  
    private final String isbn;  
    public Book(String isbn){  
        this.isbn = isbn;  
    }  
    public String getISBN() {  
        return this.isbn;  
    }  
    @Override  
    public boolean equals(Object obj) {  
        if (obj instanceof Book) {  
            Book other = (Book) obj;  
            return getISBN().equals(other.getISBN());  
        }  
        return false;  
    }  
  
    public static void main(String[] args) {  
        Book first = new Book("0201914670");  
        Book second = new Book("0201914670");  
        System.out.println(first.equals(second));  
        System.out.println(first == second);  
    }  
}
```

Java Básico

O método hashCode

- Retorna um hash code calculado para o objeto
- Em Object é implementado como o endereço de memória do objeto

```
@Override  
public boolean equals(Object obj) {  
  
    if (obj instanceof Book) {  
        Book other = (Book) obj;  
        return getISBN().equals(other.getISBN());  
    }  
  
    return false;  
}  
  
@Override  
public int hashCode() {  
    return 31 + ((isbn == null) ? 0 : isbn.hashCode());  
}
```

equals e hashCode

- O valor retornado é utilizado como *bucket number* para armazenar elementos de Map/Set em implementações hash (HashMap, HashTable, HashSet);
- Ao fazer *lookup* de um elemento, primeiro é identificado o *bucket* onde ele se encontra através com o *hash code*, depois *equals()* para encontrar o elemento dentro do *bucket*.

everis

an NTT DATA Company



Treinamento

Java Básico

Aula 3

everis

an NTT DATA Company

Definição

Objeto container que mantém um número fixo de objetos do mesmo tipo.

Declaração

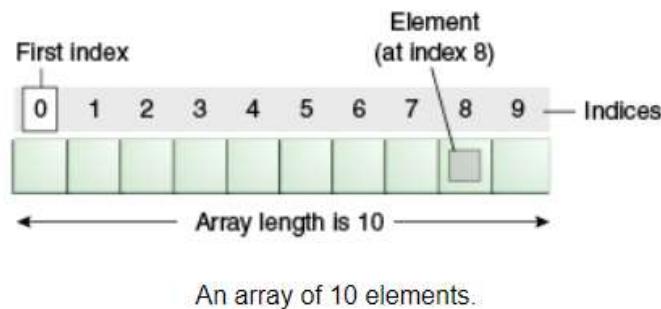
```
int[] array = new int[10];
```

Atribuição

```
array[0] = 100;  
array[1] = 200;  
...  
array[9] = 1000;
```

Acesso

```
x = array[5] + array[1];
```



Exercício

Passo 1: Crie um array de String c/ a quantidade de posições de alunos na sala.

Passo 2: Insira o nome de cada aluno nesse array, seguindo a ordem das posições, começando pelo instrutor.

Passo 3: Liste todos os nomes utilizando o foreach.

Passo 4: Liste somente os nomes das posições pares.

$i \% 2 == 0$

Passo 5: Liste somente os nomes das posições ímpares.

$i \% 2 != 0$

Utilize o JOptionPane onde ele for necessário.

Obs: lembrem-se de utilizar todos os conceitos aplicados.

Java Básico

if / if ... else

```
if(expression) {  
    // do something  
}  
  
if(expression) {  
    // do something!  
} else {  
    // do something else ...  
}  
  
if(expression) {  
    // do something  
} else if(expression2) {  
    // do something different  
} else {  
    // do something else  
}
```

Controle de Fluxo

for

```
for(start; end; increment) {  
    // do something  
}  
  
for(Object o : objects) {  
    // do something  
}
```

continue

Pula para a próxima iteração do loop.

break

Interrompe a execução do loop.

return

Retorna de um método sem executar qualquer código que venha depois desta linha.
Também pode retornar uma variável.

an NTT DATA Company



Java Básico

Controle de Fluxo

while

```
while(expression) {  
    //ação  
}
```

do ... while

```
do {  
    //ação  
} while(expression)
```

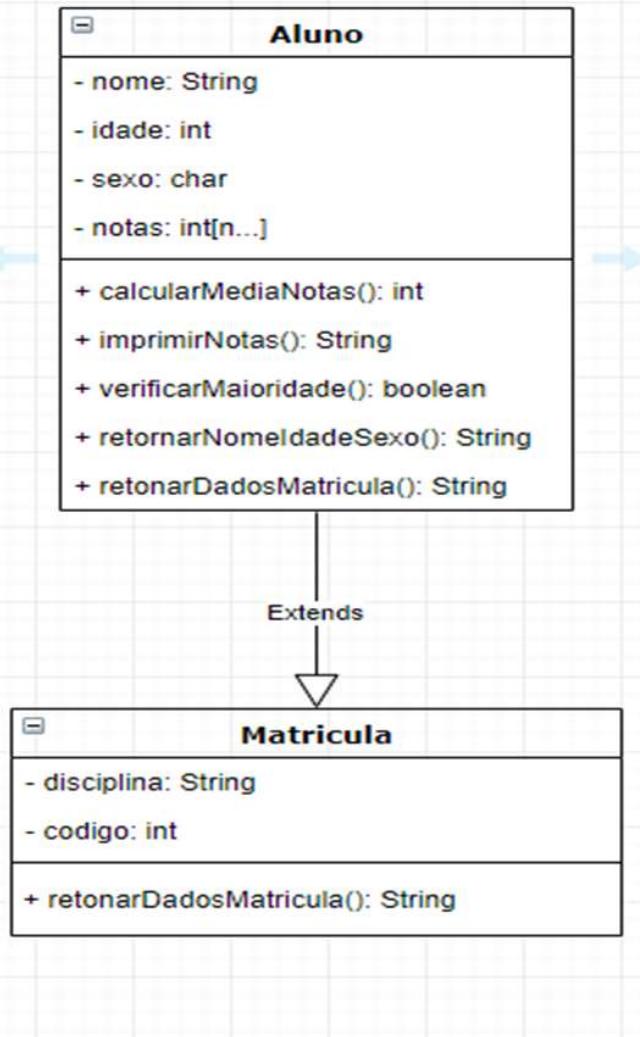
switch ... case

```
switch(variable) {  
    case 1:  
        //ação  
        break;  
    case 2:  
        //ação  
        break;  
    default:  
        //ação  
        break;  
}
```

everis

an NTT DATA Company

Exercício



Passo 1: crie uma classe Java conforme o modelo:
Constructor, Getters e setters deverão ser criados.

Passo 2: crie um método main na classe Aluno que irá
chamar todos os métodos que foram criados.
Utilize o JOptionPane onde ele for necessário.

Obs: lembrem-se de utilizar todos os conceitos
aplicados.

Generics permitem que tipos (classes e interfaces) sejam parâmetros ao se definir outras classes, interfaces e métodos.

Código sem generics

```
List list = new ArrayList();
list.add("Hello");

String string = (String) list.get(0);
```

Código com generics

```
List<String> list = new ArrayList<String>();
list.add("Hello");

String string = list.get(0); // no typecasting
```

Vantagens

- Verificação forte de tipos em tempo de compilação;
- Eliminação de casts;
- Implementação de algoritmos genéricos

Definição de classe genérica

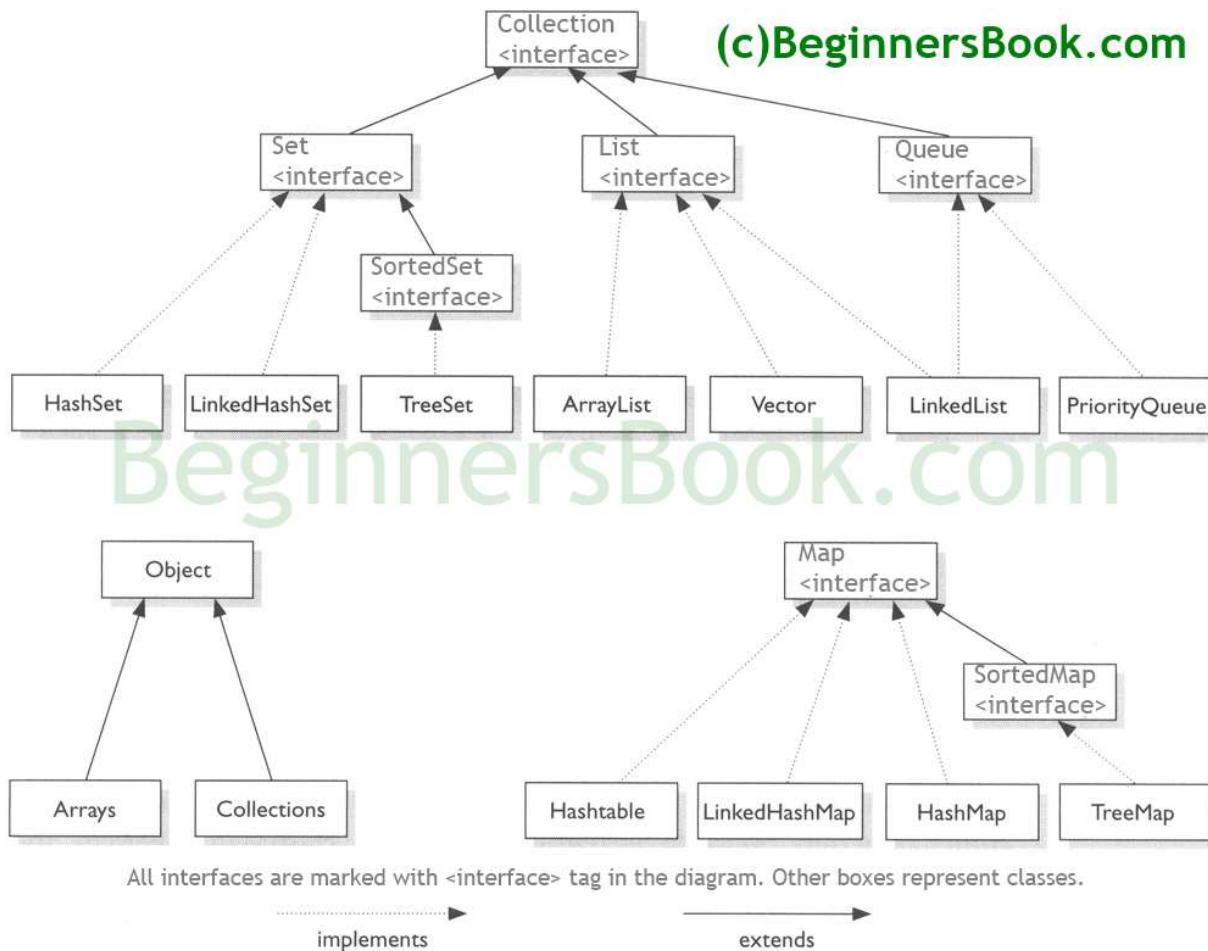
```
class ClassName<T1, T2, ..., Tn> { /* ... */ }
```

```
public class Box<T> {  
    private final T t;  
    public Box(T t){  
        this.t = t;  
    }  
    public T get() {  
        return this.t;  
    }  
    @SuppressWarnings("unused")  
    public static void main(String[] args) {  
        Box<Integer> zeroBoxed = new Box<Integer>(0);  
        Box<Integer> oneBoxed = new Box<>(1); // "the diamond" - Java 7+  
        Box<String> emptyBoxed = new Box<>("");  
        Box<String> nullBoxed = new Box<>(null);  
    }  
}
```

```
public interface Pair<K, V> {  
    K getKey();  
    V getValue();  
}
```

```
public class OrderedPair<K, V>  
implements Pair<K, V>
```

 everis



- **List:** ordenada, permite elementos duplicados
- **Set:** não permite elementos duplicados
- **Queue:** ordenada, fila (FIFO ou prioridade)
- **Deque:** ordenada, fila com dois endpoints (FIFO ou LIFO)
- **Map:** mapeia chaves e valores

Java Básico

Interface Collection

```
Collection<String> strings = new ArrayList<String>();
```

Métodos

```
int size()  
void clear()  
boolean isEmpty()  
boolean contains(Object element)  
boolean add(E element)  
boolean remove(Object element)  
Iterator<E> iterator()
```

for-each

```
for(Object object : collection) {  
    object.doSomething();  
}
```

Iterator

```
public interface Iterator<E> {  
    boolean hasNext();  
    E next();  
    void remove();  
}
```

Bulk

```
boolean containsAll(Collection<E> c)  
boolean addAll(Collection<E> c)  
boolean removeAll(Collection<E> c)  
boolean retainAll(Collection<E> c)  
void clear()
```

Set é uma collection que não permite elementos duplicados.

Implementações

- **HashSet**: mais performática, utiliza hash tables, sem garantia de ordenação;
- **TreeSet**: utiliza red-black tree, ordena elementos baseado em seus valores;
- **LinkedHashSet**: hash table com linked list, elementos ordenados pela ordem de inserção;

add

`boolean add(E element)` – retorna
false se o elemento já existir

criação

```
Set<String> strings = new HashSet<>();
```

everis

an NTT DATA Company

List é uma collection ordenada que permite elementos duplicados.

Implementações

- **ArrayList** – implementada via array interno
- **LinkedList** – duplamente ligada
- **Stack** – LIFO (push/pop)
- **Vector** – synchronized array list, com gerenciamento diferenciado de capacidade

Acesso posicional

```
E get(int index)  
boolean add(int index, E element)  
boolean addAll(int index, Collection<E> c)  
E set(int index, E element)  
E set(int index, E element)
```

Busca

```
int indexOf(Object object)  
int lastIndexOf(Object object)
```

Range

```
List<E> subList(int from, int to)
```

Algoritmos

```
sort  
shuffle  
reverse  
rotate  
binarySearch  
fill  
copy
```

Java Básico

Ordenação de List

Uma lista pode ser ordenada utilizando a classe utilitária *Collections*:

Collections.sort(list);

Para tanto os elementos da lista devem implementar a interface **Comparable** (ordem natural)

Comparable

```
public interface Comparable<T> {  
    int compareTo(T other);  
}
```

Se a ordem natural não poder ser utilizada, devemos construir um **Comparator**
list.sort(comparator);

Comparator

```
public interface Comparator<T> {  
    int compare(T first, T last);  
}
```

Java Básico

Interface Map

Map é uma collection que mapeia chaves e valores. Não permite chave duplicadas; cada chave mapeia no máximo um valor.

Implementações

- **HashMap** – implementação baseada em *hash table*, permite valores *null* e chave *null*. Não garante ordem.
- **HashTable** – semelhante ao **HashMap**, porém não permite *nulls*
- **LinkedHashMap** – especialização de **HashMap** que garante ordem.
- **TreeMap** – utiliza *red-black tree*, elementos ordenados pela chave

Métodos

```
V put(K key, V value)
V get(Object key)
Set<K> keySet()
Set<Entry<K,V>> entrySet();
Collection<V> values()
boolean containsKey(Object key)
boolean containsValue(Object value)
```

Exemplos

```
Map<String, String> m1 = new HashMap<>();
m1.put("1", "one");
m1.get("1");

for(Entry<String, String> entry: m1.entrySet()) {
    entry.getKey();
    entry.getValue();
}
```

Exercício

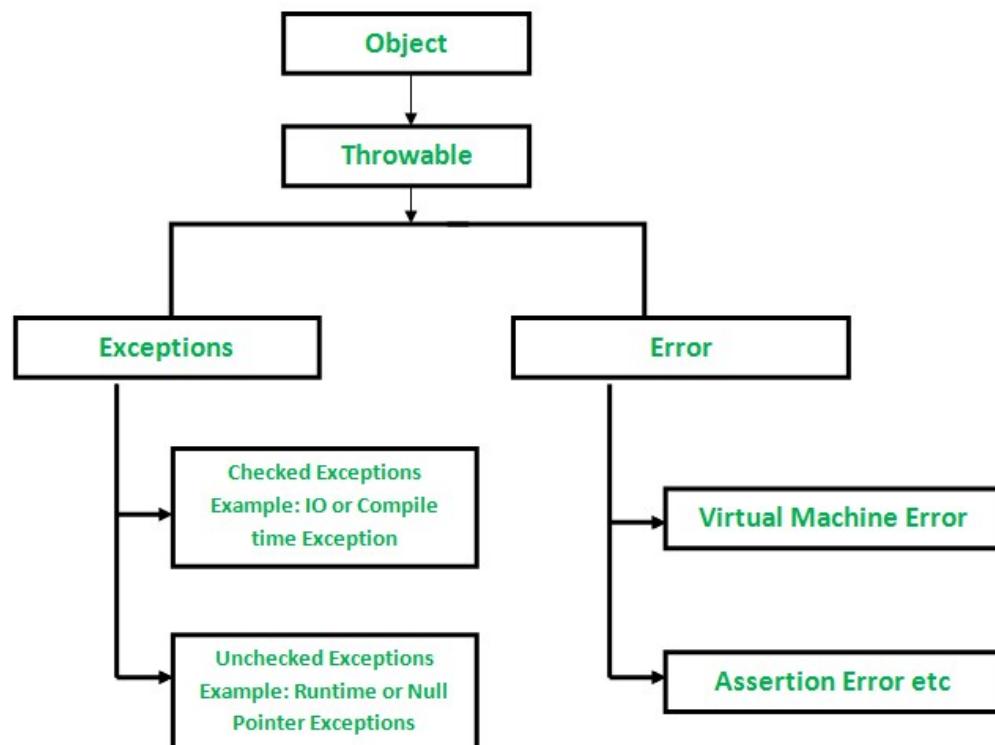
Passo 1: Escreva uma classe Java chamada *MapTest* com método *main*:

- Deverá criar mapa com a implementação *HashMap*;
- Chaves serão objetos *String*;
- Valores serão objetos *Integer*;
- Preencher o mapa com valores de 1 a 10 da seguinte forma:
map.put("um", 1);
- Criar um *loop* imprimindo no console chave e valor para cada entrada;

everis

an NTT DATA Company

Uma exceção é um evento indesejado ou inesperado, que ocorre durante a execução de um programa e que interrompe o fluxo normal de instruções.



- **Error:** Um erro indica um problema sério que um aplicativo razoável não deve tentar capturar.
- **Exception:** Exceção indica condições que um aplicativo razoável pode tentar capturar
 - **Checked:** deve ser tratada ou declarada na assinatura do método
 - **Unchecked:** condições que não podem ser antecipadas e/ou tratadas

Exemplos

```
public void readUserFile(String name)
throws FileNotFoundException {

    File file = new File(name);
    Scanner scanner = new Scanner(file);
    while(scanner.hasNextLine()) {
        System.out.println(scanner.nextLine());
    }
    scanner.close();
}
```

```
public class MyNull {
    public static void main(String[] args) {
        Object o = null;
        System.out.println(o.toString());
    }
}
```

```
public class MyArray {
    public static void main(String[ ] args) {
        int[] myNumbers = {1, 2, 3};
        System.out.println(myNumbers[10]);
    }
}
```

Tratamento de exceções

```
try {  
    //bloco que pode lançar a exceção  
}  
catch(Exception e) {  
    //bloco que será executado caso a exceção ocorra  
}  
finally {  
    //bloco que irá executar após os dois blocos anteriores, independente se  
houve ou não uma exceção  
}
```

Java Básico

Exceptions

```
public void writeList() {  
  
    PrintWriter out = null;  
  
    try {  
        System.out.println("Entering" + " try statement");  
  
        out = new PrintWriter(new FileWriter("OutFile.txt"));  
        for (int i = 0; i < list.size(); i++) {  
            out.println("Value at: " + i + " = " + list.get(i));  
        }  
  
    } catch (IndexOutOfBoundsException e) {  
        System.err.println(  
            "Caught IndexOutOfBoundsException: " + e.getMessage());  
    } catch (IOException e) {  
        System.err.println("Caught IOException: " + e.getMessage());  
    } finally {  
        if (out != null) {  
            System.out.println("Closing PrintWriter");  
            out.close();  
        } else {  
            System.out.println("PrintWriter not open");  
        }  
    }  
}
```

```
public void betterWriteList() {  
  
    try(OutputStreamWriter fw = new FileWriter("OutFile2.txt");  
        PrintWriter out = new PrintWriter(fw)) {  
  
        System.out.println("Entering" + " try statement");  
  
        for (int i = 0; i < list.size(); i++) {  
            out.println("Value at: " + i + " = " + list.get(i));  
        }  
  
    } catch (IndexOutOfBoundsException e) {  
        System.err.println(  
            "Caught IndexOutOfBoundsException: " + e.getMessage());  
    } catch (IOException e) {  
        System.err.println("Caught IOException: " + e.getMessage());  
    }  
}
```

everis

an NTT DATA Company

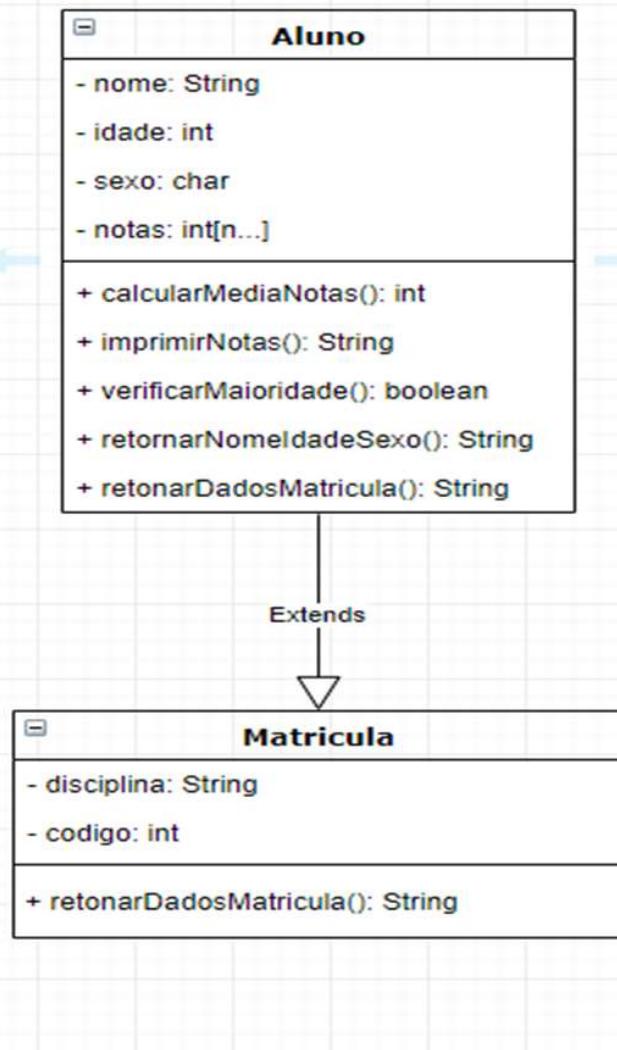
Exceções declaradas pelo método

```
public void unhandledWriteList() throws IOException {  
  
    OutputStreamWriter fw = new FileWriter("OutFile2.txt");  
    PrintWriter out = new PrintWriter(fw);  
  
    ...  
}
```

Lançando exceções explicitamente

```
throw new Exception();  
throw new Exception("Something bad happened");  
  
try {  
    ...  
} catch(IOException ioe) {  
    throw new SampleException("Failed to open file", ioe);  
}
```

Exercício



Passo 1: Escreva uma classe Java conforme o diagrama abaixo:

Passo 2: O método imprimirNotas deverá imprimir os valores e tratar a exception para quando tentar **acessar uma posição do array não existente**. **Lembre-se, deverá forçar para acessar uma posição inexistente.**

Passo 3: Imprimir no retorno da exception o erro lançado.

Passo 4: No finally imprimir o texto: “aprendi a tratar uma exception.”

Passo 5: Criar um método main e fazer o teste.

Homework

- Criar uma classe chamada **Book**, com os atributos: **title** (String), **author** (String), **isbn** (String) e **year** (Integer).
- Implementar o constructor, getters/setters, equals e hashCode;
- Criar uma classe chamada **CSVFileReader** (CSV = Comma Separated Values)
- Deverá ter um método `read(String fileName)` o qual irá ler um arquivo com informações de livros;
- Deverá utilizar **BufferedReader** em conjunto com **FileReader** para ler as linhas do arquivo;
- Para cada linha, gerar um objeto da classe Book e popular seus atributos;
- Inserir cada objeto em um conjunto (**Set**) ordenado por ISBN;
- Iterar sobre o conjunto e imprimir os elementos no console;
- Tratar **FileNotFoundException**, **IOException** e **Exception**. Em cada bloco catch, imprimir uma mensagem apropriada no console.
- Criar um método `main` deverá instanciar um CSVFileReader e executar o método `read` passando um nome de arquivo como parâmetro;
- Criar o arquivo a partir do exemplo e executar o programa

everis

an NTT DATA Company

Homework – Exemplo de arquivo

"DevOps for Digital Leaders","Aruna Ravichandran","978-1484218419",2016

"The Clean Coder: A Code of Conduct for Professional Programmers","Robert C. Martin","978-0137081073",2011

"Cerebus, Volume 1","Dave Sim","978-0919359086",1991

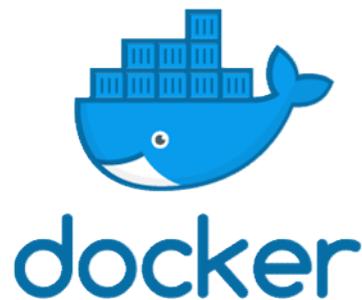
"Blood & Water (Noble Causes, Vol. 4)","Jay Faerber","978-1582405360",2005

everis

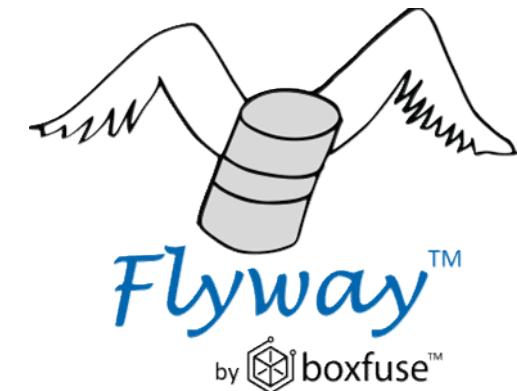
an NTT DATA Company

Java Avançado

| Vamos para o mundo real ?



PostgreSQL



everis

an NTT DATA Company



O que é Docker?

De forma resumida, o Docker é uma plataforma de código aberto, desenvolvido na linguagem Go e criada pelo próprio Google. Por ser de alto desempenho, o software garante maior facilidade na criação e administração de ambientes isolados, garantindo a rápida disponibilização de programas para o usuário final.

Quais são as funcionalidades do Docker?

O Docker tem como objetivo criar, testar e implementar aplicações em um ambiente separado da máquina original, chamado de container. Dessa forma, o desenvolvedor consegue empacotar o software de maneira padronizada. Isso ocorre porque a plataforma disponibiliza funções básicas para sua execução, como: código, bibliotecas, runtime e [ferramentas](#) do sistema.

Quais são os benefícios do Docker?

A grande vantagem no uso da plataforma é a rapidez em que o software pode ser disponibilizado — em uma frequência até 7 vezes mais rápida do que a virtualização convencional.

Outro benefício oferecido pela plataforma é a possibilidade de configurar diferentes ambientes de forma rápida, além de diminuir o número de incompatibilidades entre os sistemas disponíveis.

Java Básico

Maven

O que é Maven?



Apache Maven, ou Maven, é uma ferramenta de automação de compilação utilizada primariamente em projetos Java. Ela é similar à ferramenta Ant, mas é baseada em conceitos e trabalhos diferentes em um modo diferente. Também é utilizada para construir e gerenciar projetos escritos em C#, Ruby, Scala e outras linguagens. O projeto Maven é hospedado pela Apache Software Foundation, que fazia parte do antigo Projeto Jakarta.

O Maven utiliza um arquivo XML (POM) para descrever o projeto de software sendo construído, suas dependências sobre módulos e componentes externos, a ordem de compilação, diretórios e plug-ins necessários. Ele vem com objetivos pré-definidos para realizar certas tarefas bem definidas como compilação de código e seu empacotamento.

O Maven baixa bibliotecas Java e seus plug-ins dinamicamente de um ou mais repositórios, como o Maven 2 Central Repository, e armazena-os em uma área de cache local.[2] Este cache local de artefatos baixados pode também ser atualizado com artefatos criados por projetos locais. Re却tórios públicos podem também ser atualizados.

O que é PostgreSQL?

PostgreSQL é um dos SGBDs (Sistema Gerenciador de Bancos de Dados) de código aberto mais avançados, contando com recursos como:

Consultas complexas

Chaves estrangeiras

Integridade transacional

Controle de concorrência multi-versão

Suporte ao modelo híbrido objeto-relacional

Facilidade de Acesso

Gatilhos

Visões

Linguagem Procedural em várias linguagens (PL/pgSQL, PL/Python, PL/Java, PL/Perl) para

Procedimentos armazenados

Indexação por texto

Estrutura para guardar dados Georreferenciados PostGIS





O que é SpringBoot?

O Spring Boot é um projeto da Spring que veio para facilitar o processo de configuração e publicação de nossas aplicações. A intenção é ter o seu projeto rodando o mais rápido possível e sem complicações.

Ele consegue isso favorecendo a convenção sobre a configuração. Basta que você diga pra ele quais módulos deseja utilizar (WEB, Template, Persistência, Segurança, etc.) que ele vai reconhecer e configurar.

Você escolhe os módulos que deseja através dos starters que inclui no pom.xml do seu projeto. Eles, basicamente, são dependências que agrupam outras dependências. Inclusive, como temos esse grupo de dependências representadas pelo starter, nosso pom.xml acaba por ficar mais organizado.

Apesar do Spring Boot, através da convenção, já deixar tudo configurado, nada impede que você crie as suas customizações caso sejam necessárias.

O maior benefício do Spring Boot é que ele nos deixa mais livres para pensarmos nas regras de negócio da nossa aplicação.



O que é Flyway?

Flyway é uma dentre as várias ferramentas que se propõem a trazer ordem e organização para os scripts SQL que são executados no banco de dados, funcionando como um controle de versão do mesmo.

Quais problemas ele se propõe a sanar?

Sincronizar o banco de dados com a versão da aplicação;

Saber quais scripts SQL foram executados ou não;

Automatizar a execução dos scripts;

Criar um banco de dados do zero;

Permite criar um rollback de mudanças no banco de dados (útil em casos raros).

Java Avançado ?

| Vamos para o mundo real ?

Criando um projeto base

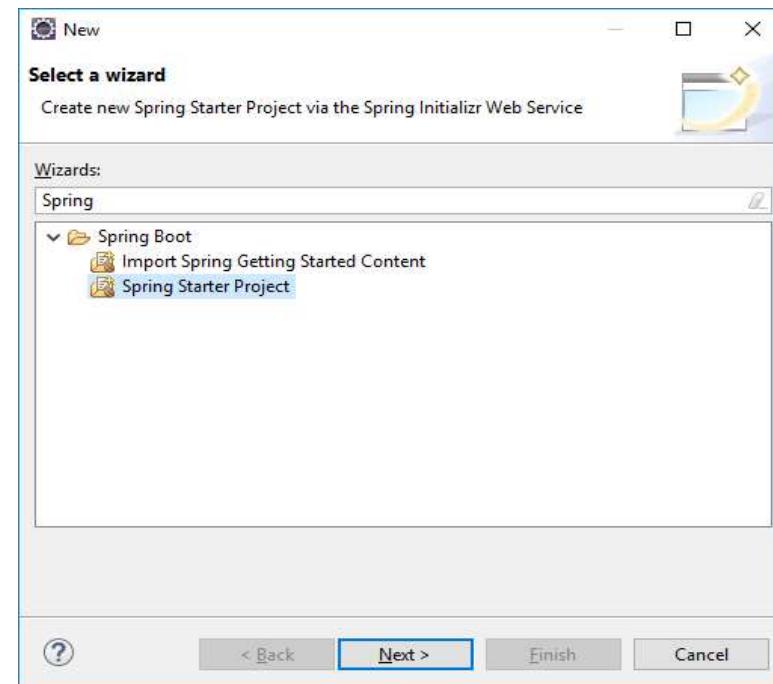
Forma 1

<https://start.spring.io/>

The screenshot shows the Spring Initializr web application. It has several tabs at the top: Project (selected), Maven Project, Gradle Project; Language (selected Java); Spring Boot (selected 2.1.6); and Project Metadata (Group: br.com.everis, Artifact: app-everis). Below these are sections for Options (Name: app-everis) and Description (Demonstração de uma aplicação simples utilizando serviços RES'). A large green button at the bottom right says 'Generate Project'.

Forma 2: Eclipse

FILE > NEW > OTHER > SPRING STARTER PROJECT

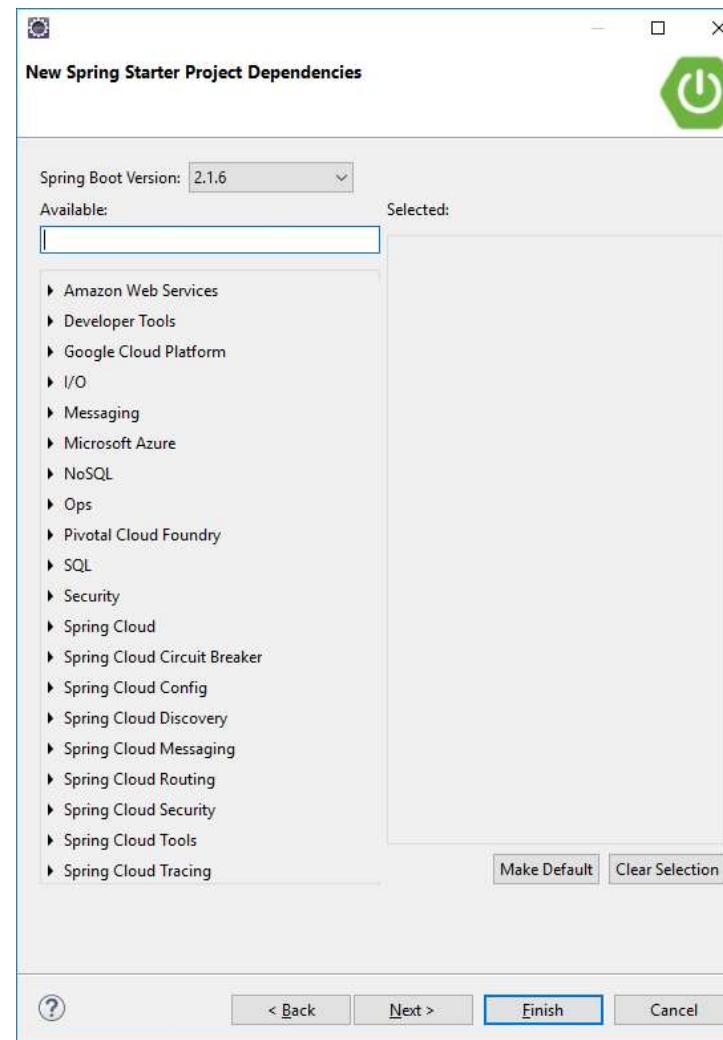
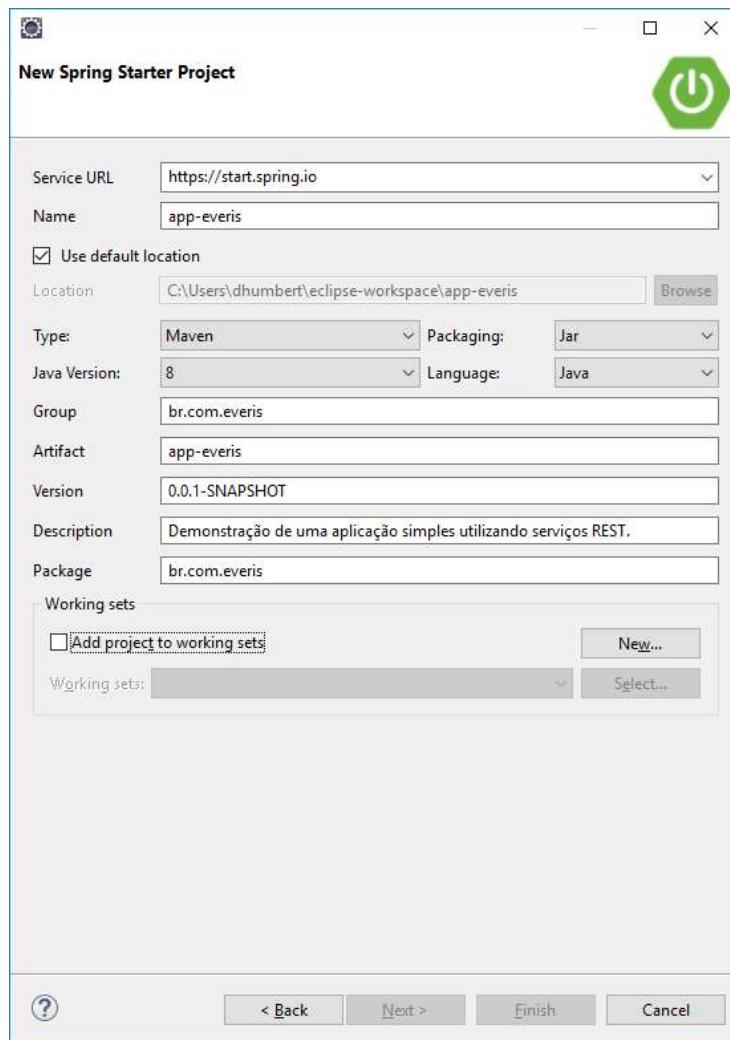


everis

an NTT DATA Company

Java Avançado ?

| Vamos para o mundo real ?



Java Avançado ?

| Vamos para o mundo real ?

<https://github.com/danilohumberto/app-spring-rest>



DÚVIDAS / PERGUNTAS ?

everis

an NTT DATA Company

<https://pt.surveymonkey.com/r/GC85R3D>

OBRIGADO

Danilo Humberto Oliveira

danilo.Humberto.oliveira@everis.com

danilohumm@gmail.com

<https://www.linkedin.com/in/danilo-humberto-97250a20>

everis

an NTT DATA Company