

BIG DATA



Disciplina: Aplicações de Big Data com Hadoop

Tema da Aula: MapReduce

Coordenação:

Prof. Dr. Adolpho Walter
Pimazzi Canton

Profa. Dra. Alessandra de
Ávila Montini

Profa. Rosangela de Fátima Pereira

Junho de 2016

Curriculum

Formação

- Mestrado em Engenharia de Computação pela Escola Politécnica da Universidade de São Paulo (Poli-USP) ([em andamento](#))
- Especialização em Tecnologia Java pela Universidade Tecnológica Federal do Paraná (UTFPR) (2011)
- Tecnologia em Análise e Desenvolvimento de Sistemas pela UTFPR ([2011](#))
- Bacharelado em Administração de Empresas pela Universidade Estadual do Norte do Paraná (UENP) (2007)

Experiência

- Professora de Big Data Analytics em empresas e programas de MBA - FIA ([2013 - atual](#))
- Pesquisadora no Laboratório de Arquitetura e Redes de Computadores (LARC) – USP ([2013 - atual](#))
- Professora de cursos de engenharia na UTFPR ([2011 -2012](#))
- Analista de sistemas na BSI Tecnologia ([2009-2010](#))

LinkedIn: <https://br.linkedin.com/pub/rosangela-de-fatima-pereira/68/a10/b56>

Apaixonada por Big Data!

Objetivo da Aula

Apresentar ao aluno fundamentos da biblioteca
MapReduce

Apresentar exemplos práticos da implementação e
execução de uma aplicação MapReduce

Conteúdo da Aula

- Revisão da aula anterior
- MapReduce
 - Fundamentos de Java
 - Biblioteca Hadoop
- MapReduce na Prática

Conteúdo da Aula

- Revisão da aula anterior
- MapReduce
 - Fundamentos de Java
 - Biblioteca Hadoop
- MapReduce na Prática

Aula anterior



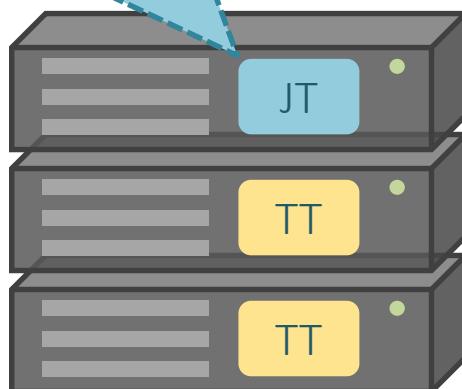
Aula anterior

1. Paradigma de programação distribuída
2. Engine de execução de aplicações distribuídas
3. Implementado em Java
4. Executa programas implementados em Java, Python, Ruby e C++

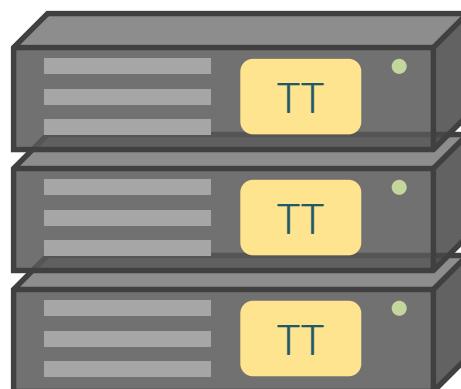
Aula anterior

JobTracker (JT)

- N o mestre
- Gerenciador de tarefas MapReduce



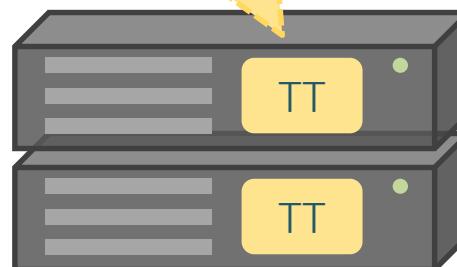
Rack 1



Rack 2

TaskTracker (TT)

- Executam as tarefas MapReduce



Rack 3

Aula anterior

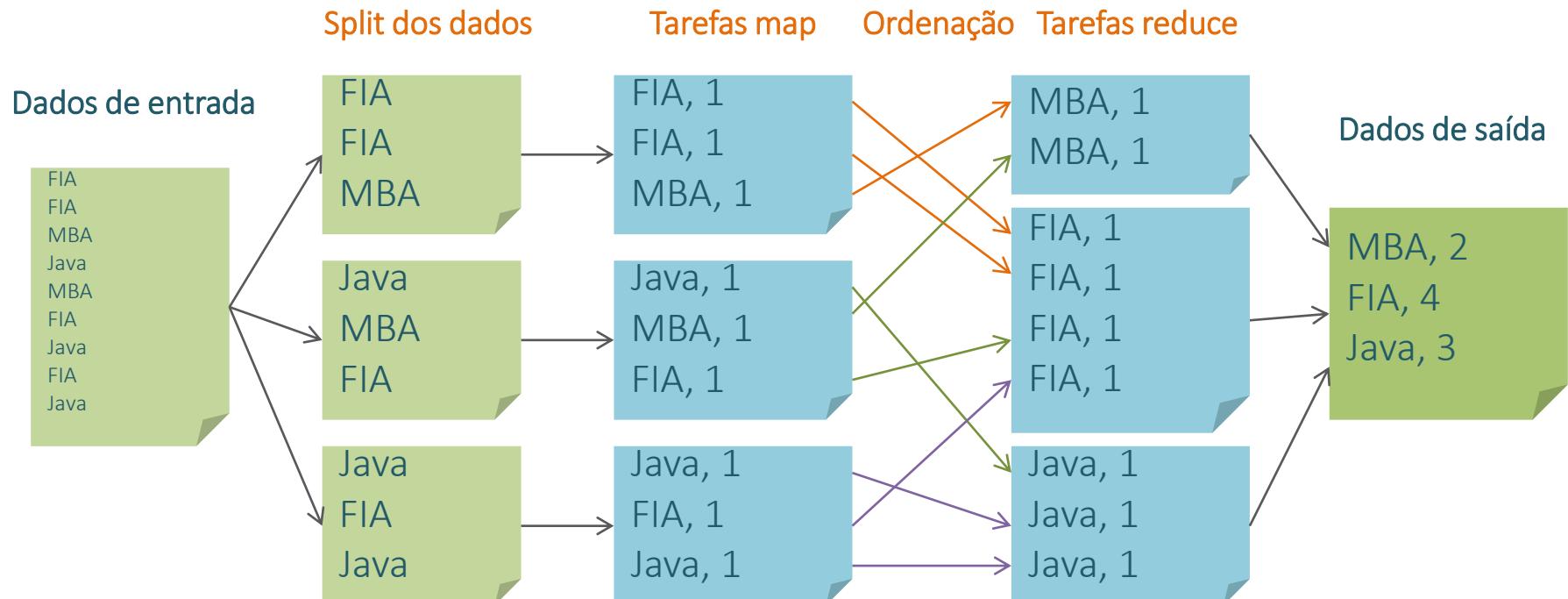
Map

Atua **exclusivamente** sobre um conjunto de entrada com chaves e valores, produzindo uma lista de chaves e valores

Reduce

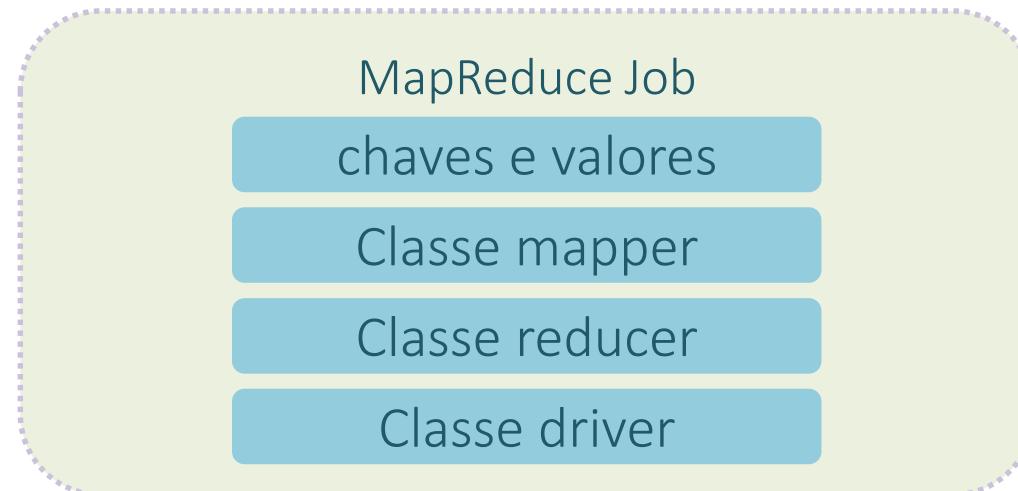
Atua sobre os valores intermediários produzidos pelo map para, normalmente, agrupar os valores e produzir uma saída

Aula anterior



Aula anterior

Responsabilidade do **desenvolvedor**



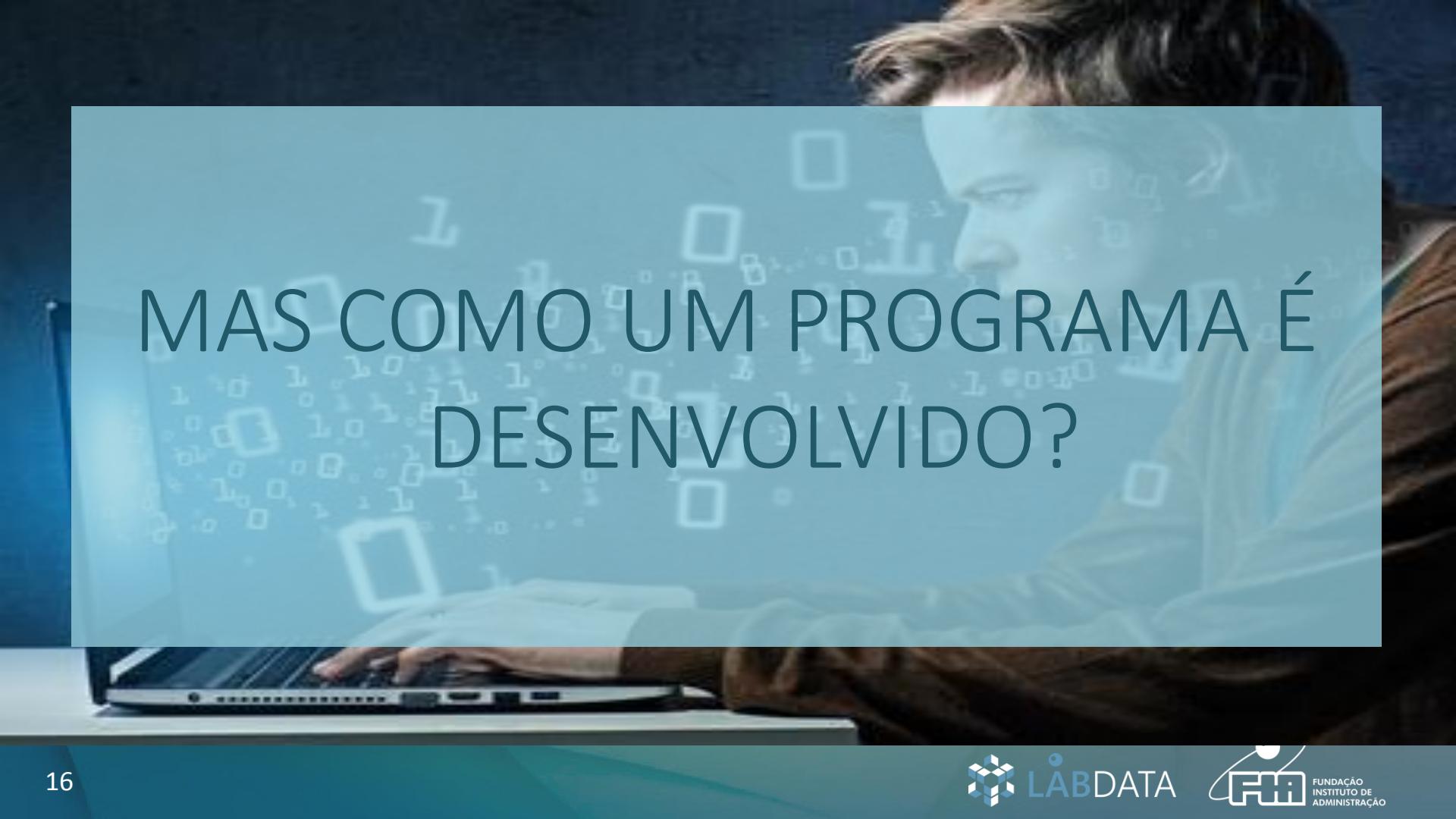
Conteúdo da Aula

- Revisão da aula anterior
- MapReduce
 - Fundamentos de Java
 - Biblioteca Hadoop
- MapReduce na Prática

ONDE A PROGRAMAÇÃO É UTILIZADA?

Onde a programação é utilizada





MAS COMO UM PROGRAMA É DESENVOLVIDO?

Programa

Conjunto de instruções (algoritmo)
que um computador segue para
realizar uma tarefa

Programador

Profissional que cria algoritmos
para o computador realizar uma
tarefa



Problema

Linguagem humana

```
a = 2 + 2;  
b = a + 5;  
c = a + b;
```



Linguagem de máquina

```
01110011 01100101 01110010 01  
01100101 01110010 00100000 01  
01101000 01100001 01110100 00  
01100100 01101001 01110011 01  
01110010 01101001 01100010 01  
01110100 01100101 01110011 00  
01100001 01101110 01111001 00  
01101001 01101110 01100011 01  
01101101 01101001 01101110 01  
00100000 01101101 01100101 01  
01110011 01100001 01100111 01  
01110011 00100000 01110100 01  
00100000 01100001 01101100 01  
00001101 00001010 00100000 00
```

Solução

Linguagem humana

```
a = 2 + 2;  
b = a + 5;  
c = a + b;
```



INTERPRETADOR



Linguagem de máquina

```
01110011 01100101 01110010 01  
01100101 01110010 00100000 01  
01101000 01100001 01110100 00  
01100100 01101001 01110011 01  
01110010 01101001 01100010 01  
01110100 01100101 01110011 00  
01100001 01101110 01111001 00  
01101001 01101110 01100011 01  
01101101 01101001 01101110 01  
00100000 01101101 01100101 01  
01110011 01100001 01100111 01  
01110011 00100000 01110100 01  
00100000 01100001 01101100 01  
00001101 00001010 00100000 00
```

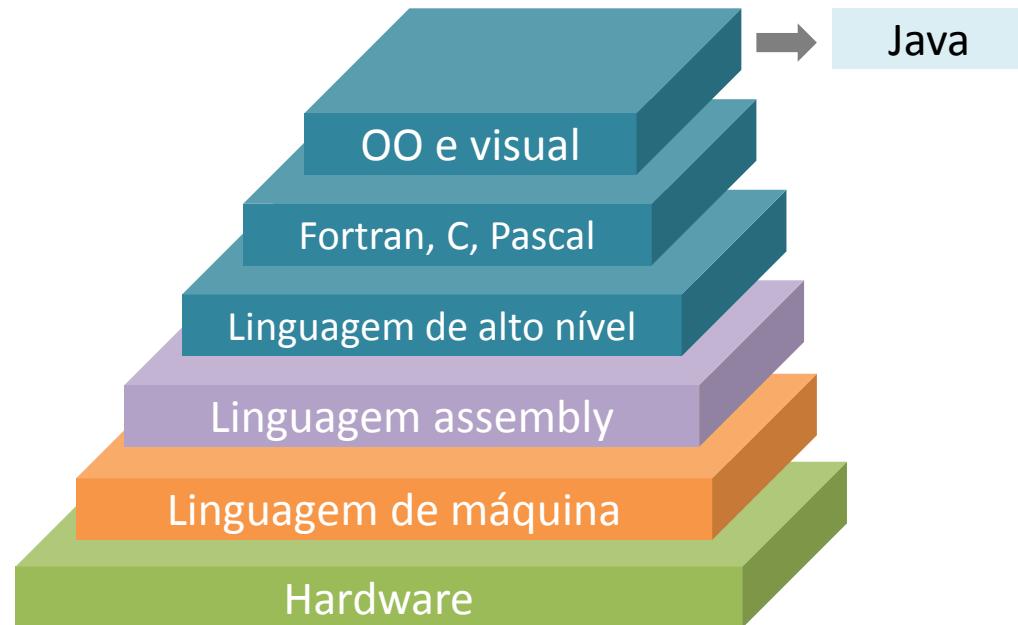
Linguagens de programação

Criadas para **facilitar** a tarefa de programação.
Permite criar algoritmos com uma linguagem
mais compreensível por humanos e faz a
tradução do código para linguagem de máquina.

Linguagens de programação



Linguagens de programação





História

Java - História

- Criada em **1991** por James Gosling
- Lançada pela primeira vez pela *Sun Microsystems* em 1995
- Inicialmente criada para ser utilizada em dispositivos embarcados
- Adquirida pela *Oracle Corporation* 2010

<https://www.oracle.com/java/>



Java - História

- Sintaxe baseada na linguagem de programação C
- Segue o paradigma de orientação a objetos
- Utilizada para desenvolvimento Web, aplicações móveis e Desktop



POR QUE APRENDER JAVA?

Java - Motivação

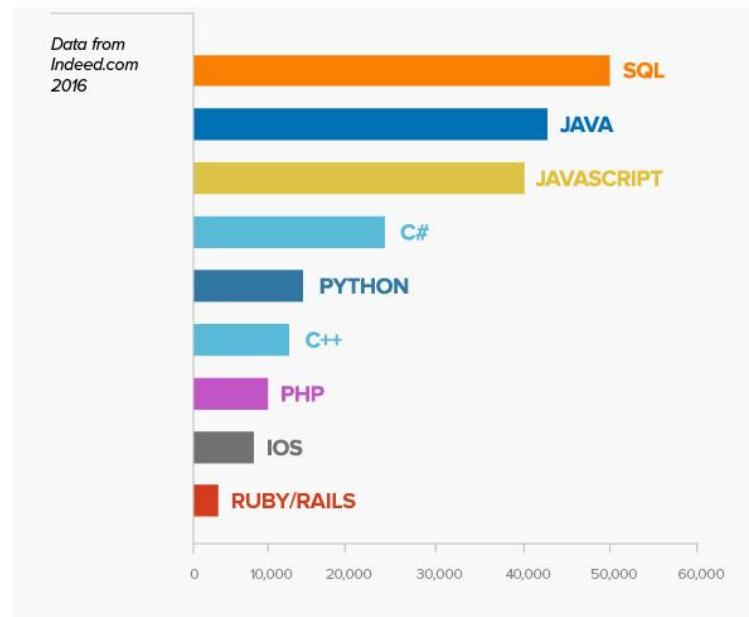
Linguagem mais utilizada no mundo

May 2016	May 2015	Change	Programming Language	Ratings	Change
1	1		Java	20.956%	+4.09%
2	2		C	13.223%	-3.62%
3	3		C++	6.698%	-1.18%
4	5	▲	C#	4.481%	-0.78%
5	6	▲	Python	3.789%	+0.06%
6	9	▲	PHP	2.992%	+0.27%
7	7		JavaScript	2.340%	-0.79%
8	15	▲	Ruby	2.338%	+1.07%
9	11	▲	Perl	2.326%	+0.51%
10	8	▼	Visual Basic .NET	2.325%	-0.64%

Fonte: Índice TIOBE. http://www.tiobe.com/tiobe_index?page=index

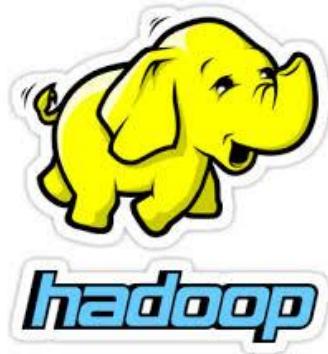
Java - Motivação

Segunda linguagem mais requisitada pelas empresas



Java - Motivação

Onde a linguagem Java é utilizada



SONY



Java - Motivação

Estatísticas sobre Java

- **9 milhões** de desenvolvedores de Java em todo o mundo
- **3 bilhões** de telefones celulares executam o Java
- **100%** dos players de disco Blu-ray vêm equipados com o Java
- **125 milhões** de aparelhos de TV executam o Java

Fonte: https://java.com/pt_BR/about/

Java - Motivação

Um pouco do universo Java



Struts

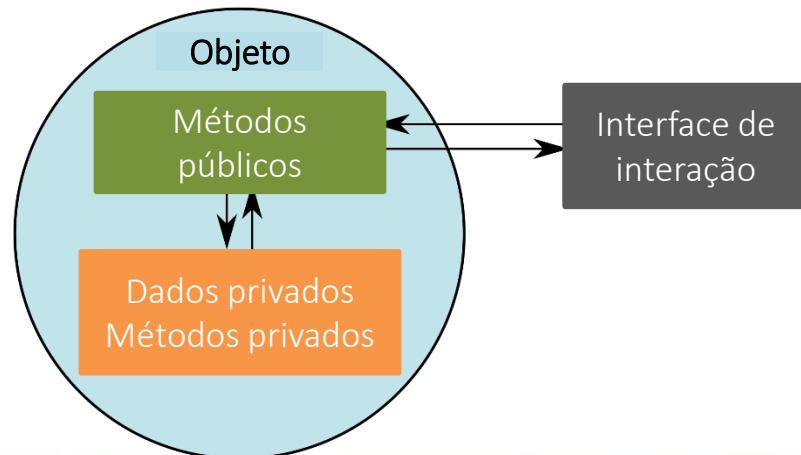


QUAIS AS CARACTERÍSTICAS DA LINGUAGEM JAVA?

Java – Características

Java permite o **encapsulamento** dos dados

- Esconde detalhes da implementação de um objeto
- O código cliente pode usar apenas a interface para a operação



Java – Características

É fácil compreender a organização e fluxo
de um código de **20** linhas?

Java – Características

É fácil compreender a organização e fluxo
de um código de **20.000** linhas?

Java – Características

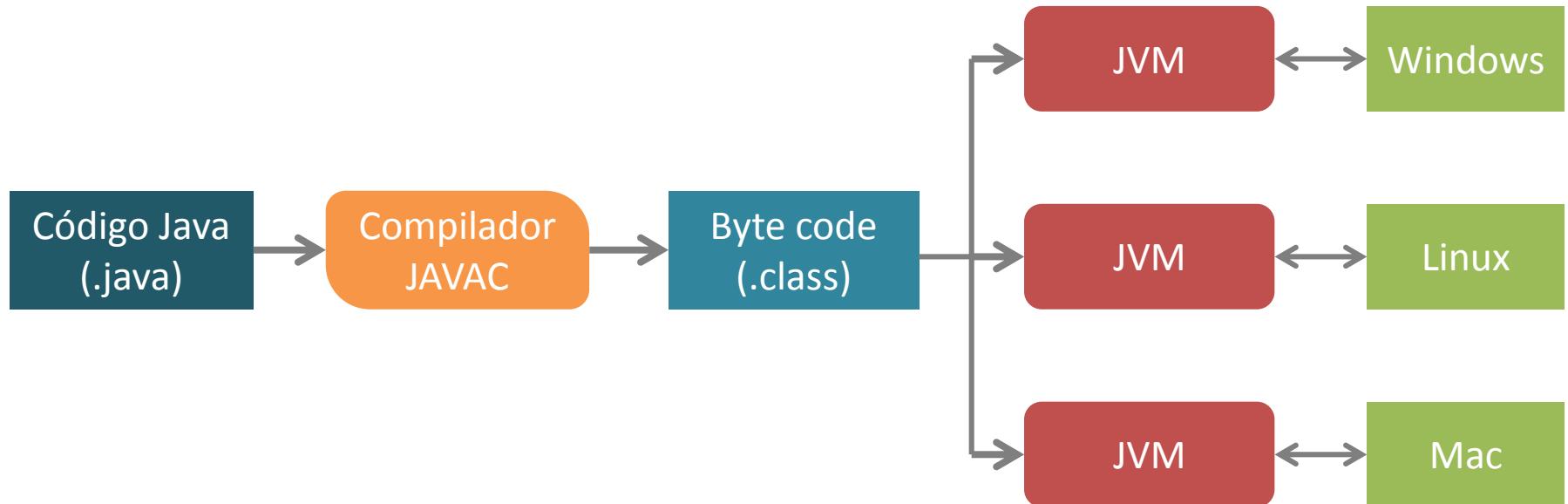
- Permite que o código fique bem organizado
- A manutenção é mais fácil e menos custosa
- Permite a alteração do código sem alterar o código do cliente
- Permite executar o mesmo código em diversos sistemas operacionais

Java – Características

JVM - Java Virtual Machine

- Manipula todos os detalhes de baixo nível para que o programador possa apenas escrever um programa e executá-lo em uma variedade de sistemas operacionais.
- Auxilia o gerenciamento de memória e fornece informações de depuração, entre outras coisas.

Java – Características



Java – Características

JRE – *Java Runtime Environment*

- Componente necessário para executar uma aplicação Java

JDK – *Java Development Kit*

- Conjunto de ferramentas necessárias para realizar o desenvolvimento de aplicações java e inclui a JRE e ferramentas de programação, como:
 - javac – compilador
 - javadoc – ferramenta para geração de documentação

ONDE POSSO IMPLEMENTAR MEU CÓDIGO EM JAVA?

Java na Prática

Que ferramenta utilizar para programar em Java?

- Editor de texto

- Bloco de notas, Gedit, Notepad++



- IDE - *Integrated Development Environment* ou Ambiente de

Desenvolvimento Integrado

- NetBeans, Eclipse, IntelliJ IDEA



Java na Prática

Software: [VMware](#)

Máquina virtual: [Cloudera-training-analyst](#)

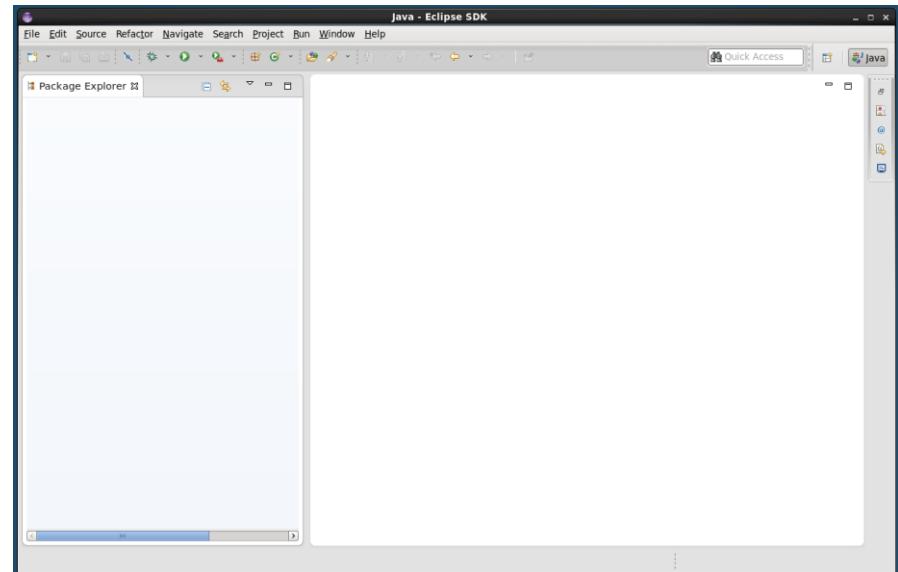


cloudera

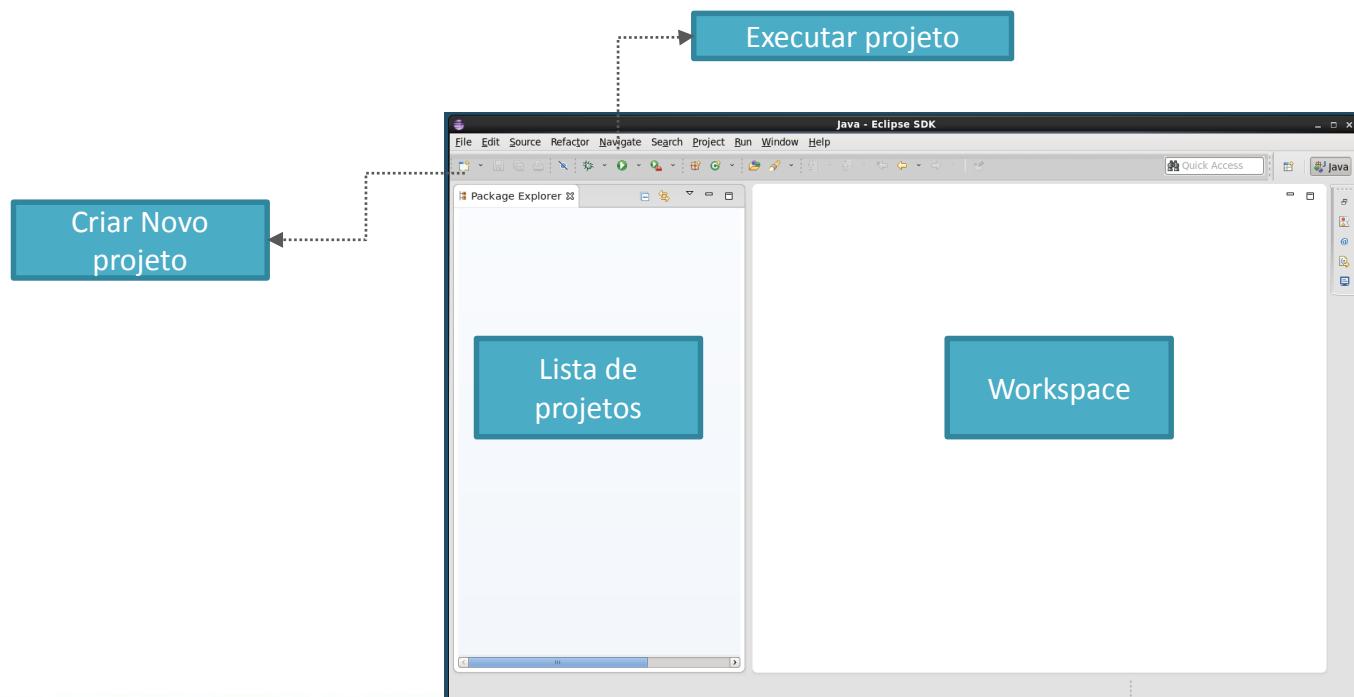
Java na Prática

Desenvolvendo a primeira aplicação Java

- Abra a IDE Eclipse

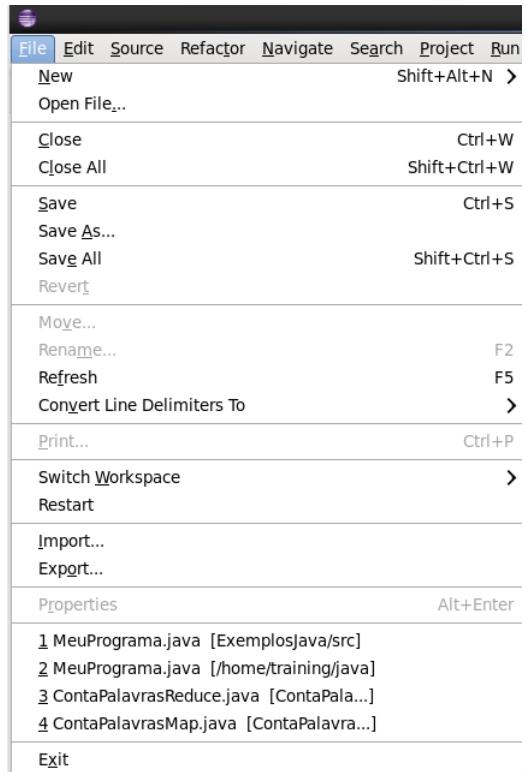


Java na Prática



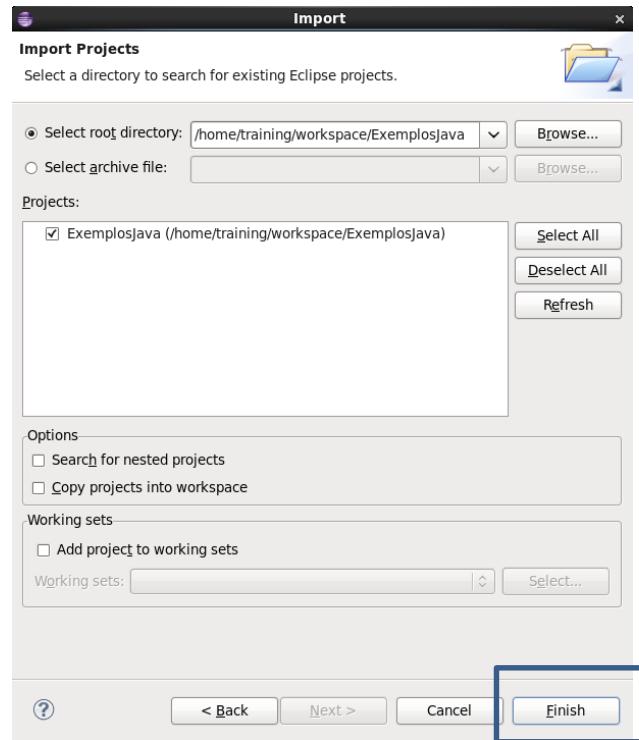
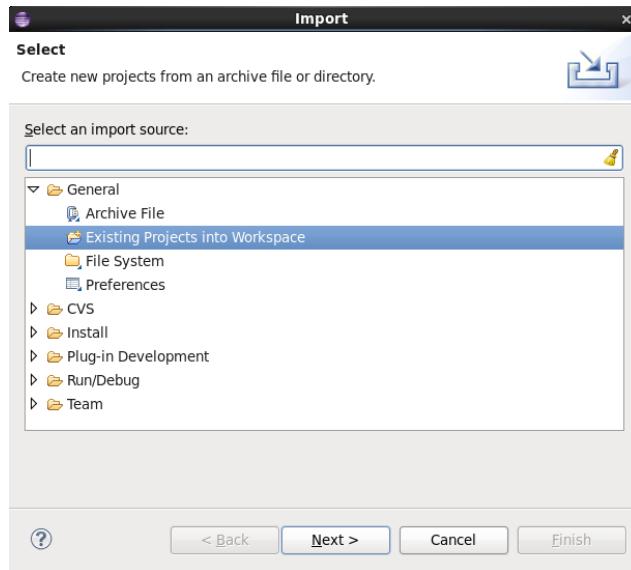
Java na Prática

Selecionar opção: File >> Import

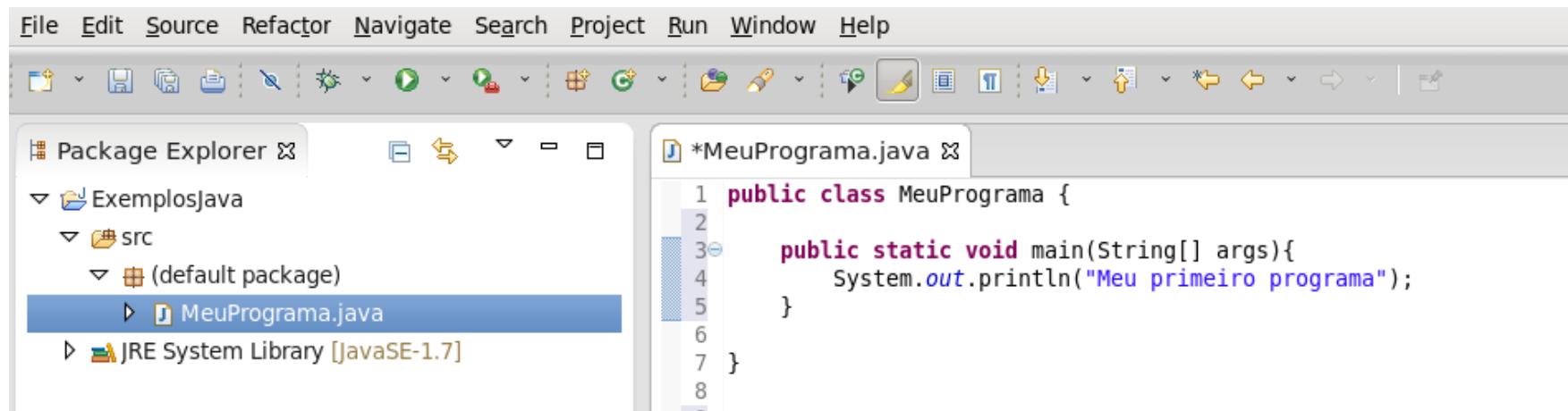


Java na Prática

Selecionar o projeto ExemplosJava



Java na Prática



Java na Prática

Exemplo de um código Java

```
public class MeuPrograma {
```

Palavra reservada para
indicar início de uma classe

Declaração do nome da
classe

}

Chaves indicam início e fim
de um bloco de código

Java na Prática

Exemplo de um código Java

Declaração de método

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

parâmetro do método

nome do método

Java na Prática

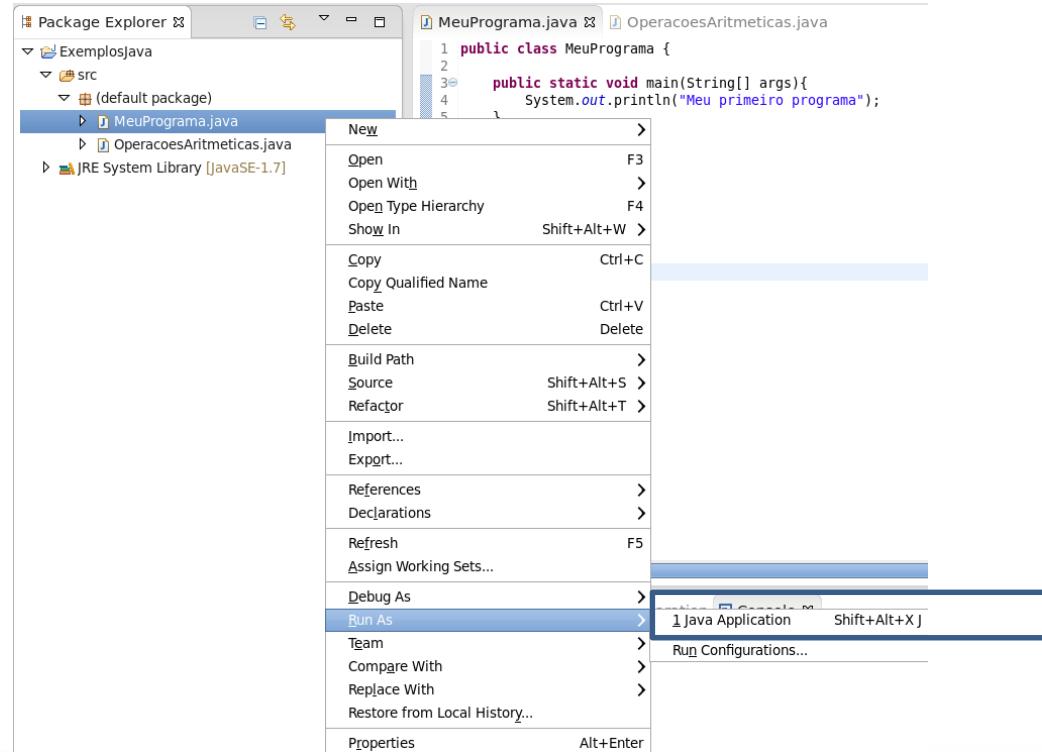
Exemplo de um código Java

```
public class MeuPrograma {  
    public static void main(String[] args) {  
        System.out.println("meu primeiro programa");  
    }  
}
```

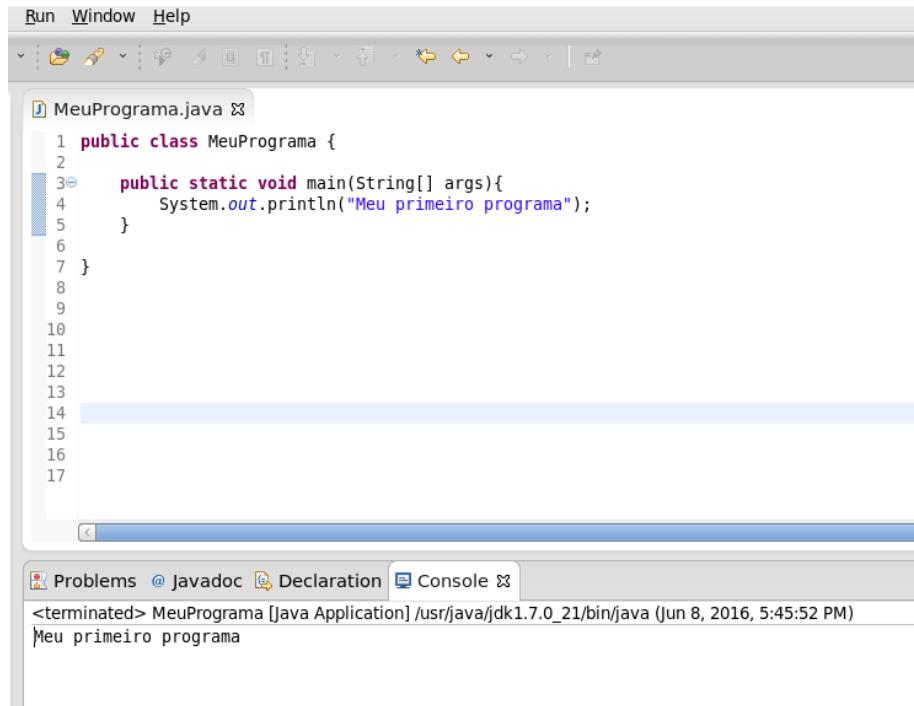
Chamada de método para
apresentar o texto entre aspas

Java na Prática

Clique com o botão direito do mouse sobre a classe MeuPograma.java e selecione a opção “Run As” >> “Java Application”



Java na Prática



The screenshot shows a Java development environment. The code editor window is titled "MeuPrograma.java" and contains the following Java code:

```
1 public class MeuProgramma {
2
3     public static void main(String[] args){
4         System.out.println("Meu primeiro programa");
5     }
6
7 }
```

The terminal window (Console tab) shows the output of the program:

```
<terminated> MeuProgramma [Java Application] /usr/java/jdk1.7.0_21/bin/java (Jun 8, 2016, 5:45:52 PM)
Meu primeiro programa
```

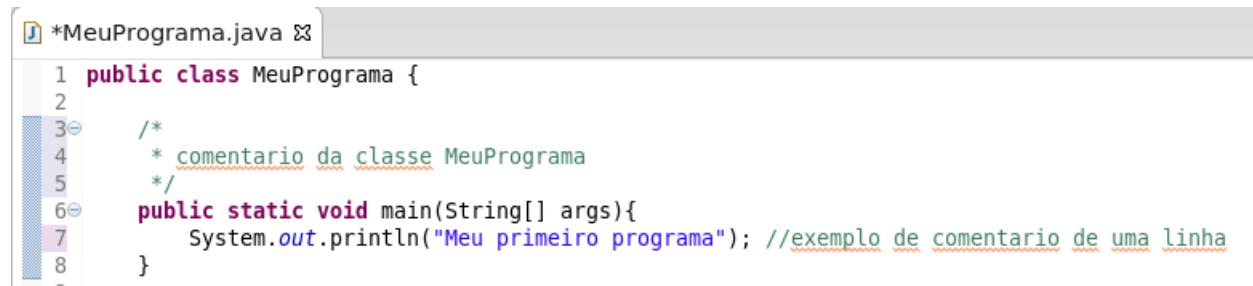
A teal box on the left contains the text "SAÍDA" with an arrow pointing to the terminal window.

Java na Prática

Comentários em Java

```
// comentário de uma linha
```

```
/*
 * comentário em bloco
 */
```



```
*MeuPrograma.java
1 public class MeuPrograma {
2
3     /*
4      * comentário da classe MeuProgramma
5      */
6     public static void main(String[] args){
7         System.out.println("Meu primeiro programa"); //exemplo de comentário de uma linha
8     }
9 }
```

Java na Prática

Variáveis em Java

- int: para inteiros entre -2 bilhões e 2 bilhões
- long: para inteiros entre -9×10^{18} e 9×10^{18}
- float, double: para números de ponto flutuantes
- char: para um único caracter
- boolean: verdadeiro ou falso
- String: para strings caracteres

Java na Prática

Variáveis em Java

- Em Java é preciso definir previamente o tipo de dado que a variável irá armazenar

```
tipoDaVariavel nomeDaVariavel;  
int idade;
```

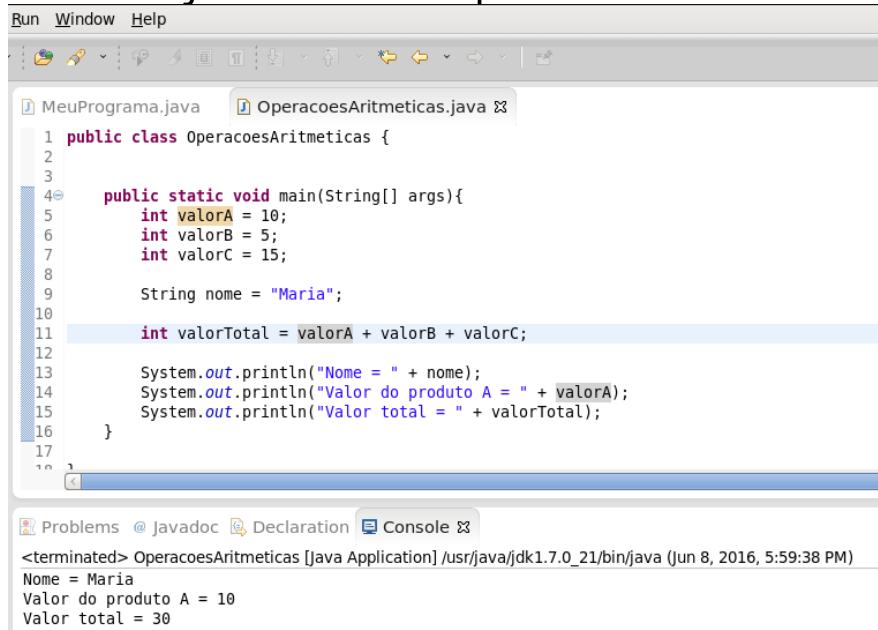
Java na Prática

Operações aritméticas

Operador	Operação
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo
++	Incremento
--	Decremento

Java na Prática

OperacoesAritmeticas.java - Exemplos



The screenshot shows a Java application running in an IDE. The code in `OperacoesAritmeticas.java` is as follows:

```
1 public class OperacoesAritmeticas {  
2  
3  
4    public static void main(String[] args){  
5        int valorA = 10;  
6        int valorB = 5;  
7        int valorC = 15;  
8  
9        String nome = "Maria";  
10  
11        int valorTotal = valorA + valorB + valorC;  
12  
13        System.out.println("Nome = " + nome);  
14        System.out.println("Valor do produto A = " + valorA);  
15        System.out.println("Valor total = " + valorTotal);  
16    }  
17  
18}
```

The output in the console is:

```
Nome = Maria  
Valor do produto A = 10  
Valor total = 30
```

Java na Prática

Operadores

&& and

|| ou

! não

> Maior que

>= Maior ou igual a

< menor que

<= menor ou igual a

== igual a

!= não igual a

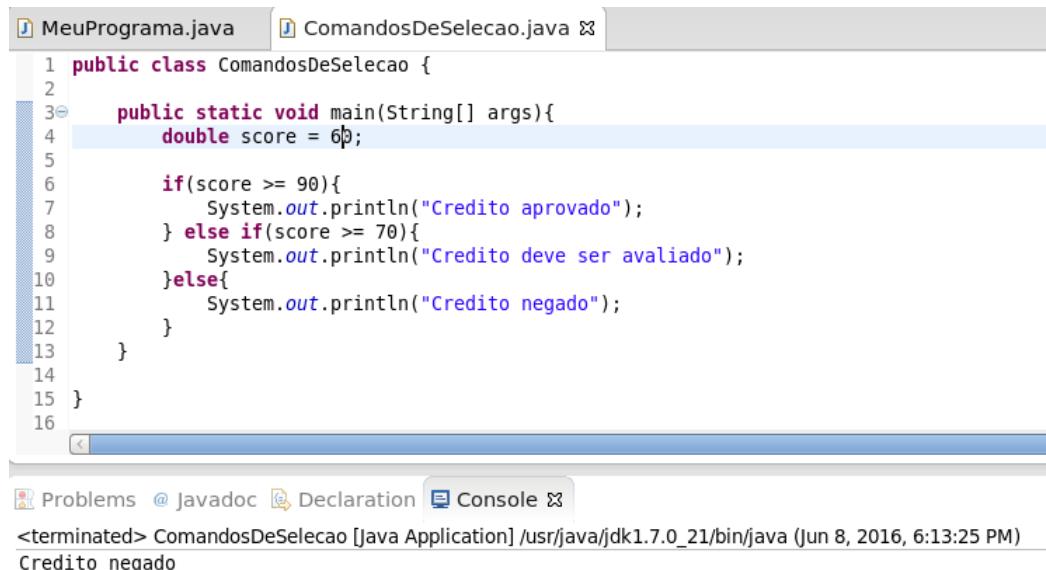
Java na Prática

Comandos de seleção em Java

- if : executa um código se a condição for verdadeira
- else: caminho secundário quando a condição do if for falsa

Java na Prática

ComandosDeSelecao.java – Exemplo



The screenshot shows a Java code editor with two tabs: 'MeuPrograma.java' and 'ComandosDeSelecao.java'. The 'ComandosDeSelecao.java' tab is active, displaying the following code:

```
1 public class ComandosDeSelecao {
2
3     public static void main(String[] args){
4         double score = 60;
5
6         if(score >= 90){
7             System.out.println("Credito aprovado");
8         } else if(score >= 70){
9             System.out.println("Credito deve ser avaliado");
10        }else{
11            System.out.println("Credito negado");
12        }
13    }
14
15 }
16
```

Below the code editor is a 'Console' tab showing the output of the program:

```
<terminated> ComandosDeSelecao [Java Application] /usr/java/jdk1.7.0_21/bin/java (Jun 8, 2016, 6:13:25 PM)
Credito negado
```

Linguagem Java

Laços de repetição

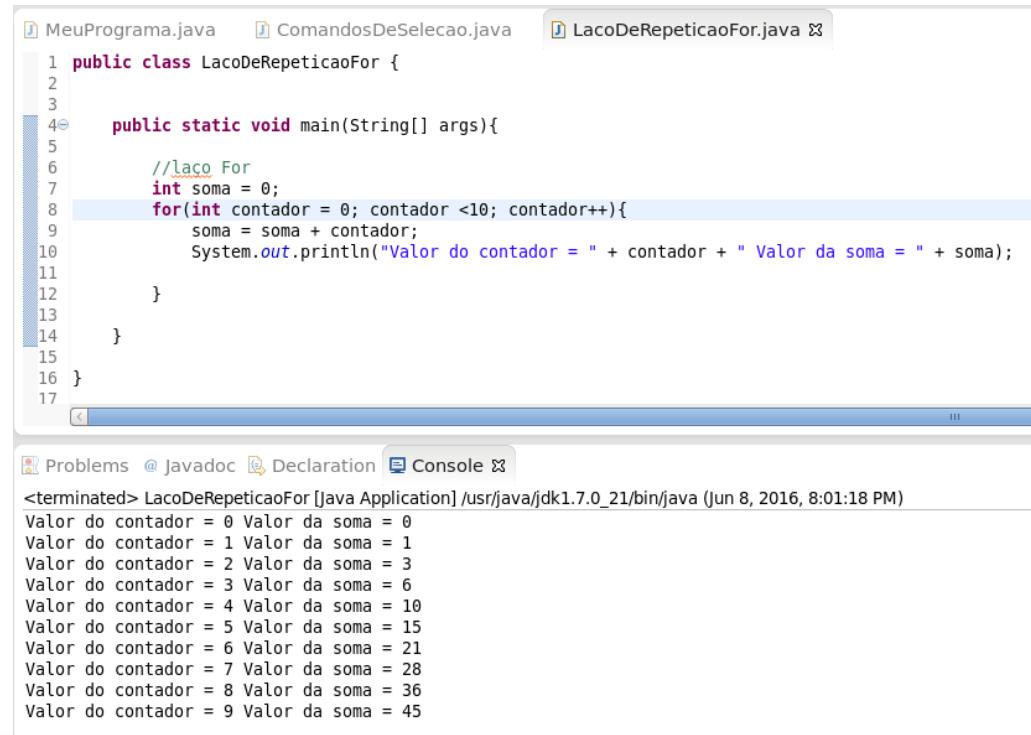
Laços de repetição

Java aceita 3 principais tipos de laços de repetição

- For
- Do
- Do-While

Laços de repetição

LacoDeRepeticaoFor.java Exemplo



The screenshot shows an IDE interface with three tabs at the top: 'MeuPrograma.java', 'ComandosDeSelecao.java', and 'LacoDeRepeticaoFor.java'. The 'LacoDeRepeticaoFor.java' tab is active, displaying the following Java code:

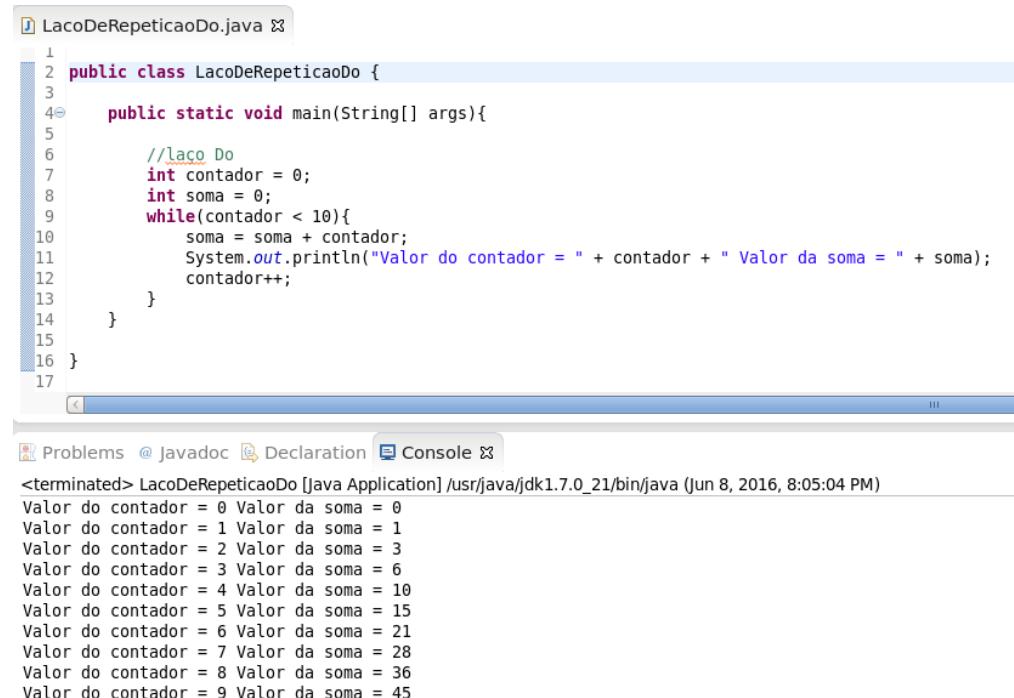
```
1 public class LacoDeRepeticaoFor {
2
3
4     public static void main(String[] args){
5
6         //laco For
7         int soma = 0;
8         for(int contador = 0; contador <10; contador++){
9             soma = soma + contador;
10            System.out.println("Valor do contador = " + contador + " Valor da soma = " + soma);
11        }
12    }
13
14 }
15
16 }
```

Below the code editor is a 'Console' tab showing the execution output:

```
<terminated> LacoDeRepeticaoFor [Java Application] /usr/java/jdk1.7.0_21/bin/java (Jun 8, 2016, 8:01:18 PM)
Valor do contador = 0 Valor da soma = 0
Valor do contador = 1 Valor da soma = 1
Valor do contador = 2 Valor da soma = 3
Valor do contador = 3 Valor da soma = 6
Valor do contador = 4 Valor da soma = 10
Valor do contador = 5 Valor da soma = 15
Valor do contador = 6 Valor da soma = 21
Valor do contador = 7 Valor da soma = 28
Valor do contador = 8 Valor da soma = 36
Valor do contador = 9 Valor da soma = 45
```

Laços de repetição

LacoDeRepeticaoDo.java
Exemplo



The screenshot shows an IDE interface with a Java code editor and a terminal window. The code editor displays a file named 'LacoDeRepeticaoDo.java' containing the following Java code:

```
1  public class LacoDeRepeticaoDo {
2
3
4    public static void main(String[] args){
5
6        //laco Do
7        int contador = 0;
8        int soma = 0;
9        while(contador < 10){
10            soma = soma + contador;
11            System.out.println("Valor do contador = " + contador + " Valor da soma = " + soma);
12            contador++;
13        }
14    }
15
16 }
```

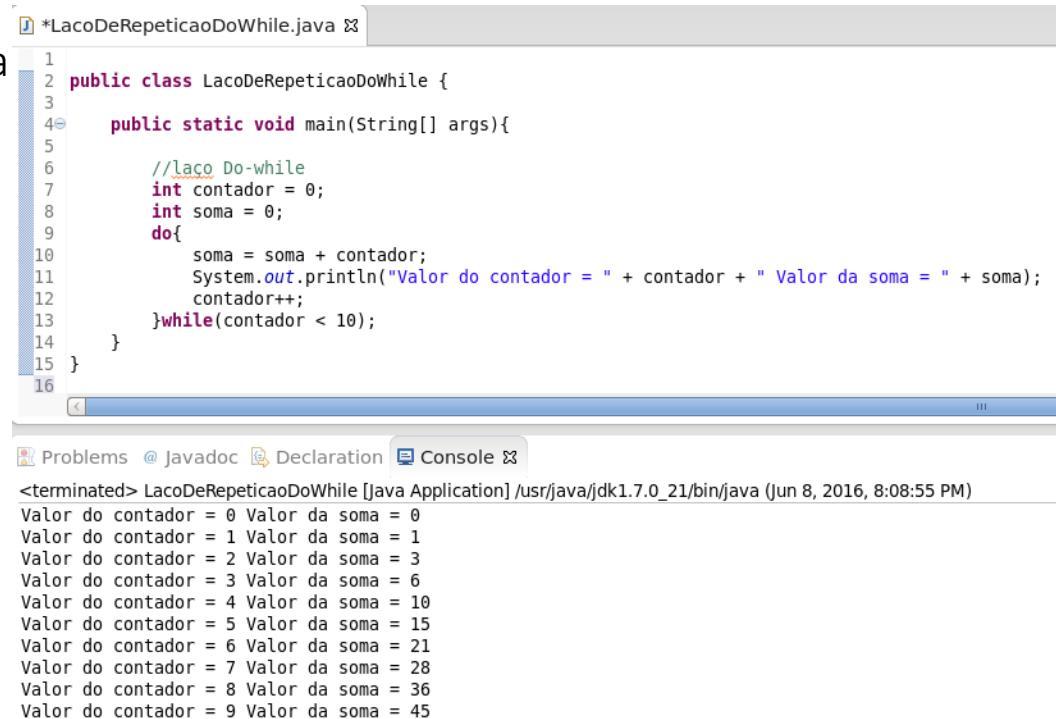
The terminal window below shows the execution output of the code, which prints the value of the counter and the sum for each iteration from 0 to 9:

```
<terminated> LacoDeRepeticaoDo [Java Application] /usr/java/jdk1.7.0_21/bin/java (Jun 8, 2016, 8:05:04 PM)
Valor do contador = 0 Valor da soma = 0
Valor do contador = 1 Valor da soma = 1
Valor do contador = 2 Valor da soma = 3
Valor do contador = 3 Valor da soma = 6
Valor do contador = 4 Valor da soma = 10
Valor do contador = 5 Valor da soma = 15
Valor do contador = 6 Valor da soma = 21
Valor do contador = 7 Valor da soma = 28
Valor do contador = 8 Valor da soma = 36
Valor do contador = 9 Valor da soma = 45
```

Laços de repetição

LacoDeRepeticaoDoWhile.java

Exemplo



The screenshot shows an IDE interface with a code editor and a terminal window. The code editor displays a Java file named `LacoDeRepeticaoDoWhile.java` containing the following code:

```
1  public class LacoDeRepeticaoDoWhile {  
2      public static void main(String[] args){  
3          //laco Do-while  
4          int contador = 0;  
5          int soma = 0;  
6          do{  
7              soma = soma + contador;  
8              System.out.println("Valor do contador = " + contador + " Valor da soma = " + soma);  
9              contador++;  
10         }while(contador < 10);  
11     }  
12 }  
13  
14  
15  
16
```

The terminal window below the code editor shows the output of the program, which is a sequence of lines starting with "Valor do contador = 0" and "Valor da soma = 0", followed by "Valor do contador = 1" and "Valor da soma = 1", and so on, up to "Valor do contador = 9" and "Valor da soma = 45".

```
<terminated> LacoDeRepeticaoDoWhile [Java Application] /usr/java/jdk1.7.0_21/bin/java (Jun 8, 2016, 8:08:55 PM)  
Valor do contador = 0 Valor da soma = 0  
Valor do contador = 1 Valor da soma = 1  
Valor do contador = 2 Valor da soma = 3  
Valor do contador = 3 Valor da soma = 6  
Valor do contador = 4 Valor da soma = 10  
Valor do contador = 5 Valor da soma = 15  
Valor do contador = 6 Valor da soma = 21  
Valor do contador = 7 Valor da soma = 28  
Valor do contador = 8 Valor da soma = 36  
Valor do contador = 9 Valor da soma = 45
```

Linguagem Java

Classes

Classes

A palavra classe vem da taxonomia da biologia.

Todos os seres vivos de uma mesma **classe** biológica têm uma série de **atributos e comportamentos** em comum, mas não são iguais, podem variar nos valores desses atributos e como realizam esses comportamentos.

Classes

Estrutura de uma classe Java

ESTRUTURA

```
<modificador> class <nome_classe>{  
    [declaração_dos_atributos]  
    [declaração_dos_métodos]  
}
```

EXEMPLO

```
public class Cliente{  
    String nomeCliente;  
    public void cadastrarCliente();  
}
```

Classes

Exemplos da estrutura da declaração de um método

```
char meuMetodo(String s)           // recebe uma string, retorna um char
void meuMetodo (int x)            // recebe um int, não retorna nada
List<String> meuMetodo ()        // não recebe nada, retorna uma lista
int meuMetodo (int x, double y)  // recebe um int e um double, retorna um int
```

Classes

Criação da classe Conta.java

```
J *Conta.java ✘
1  public class Conta {
2
3      public int numero;
4      public String nomeCliente;
5      public double saldo;
6      public double limite;
7
8
9      public void sacarDinheiro(double quantidade){
10         double novoSaldo = this.saldo - quantidade;
11         this.saldo = novoSaldo;
12     }
13
14      public void depositarDinheiro(double quantidade){
15         this.saldo = this.saldo + quantidade;
16     }
17
18 }
19
```

Classes

Objetos são instâncias de classes, o que se pode construir a partir de uma classe.

RECEITINHA:
BOLO NA CANECA

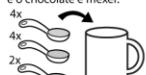
INGREDIENTES:

- 4 colheres de sopa de farinha
- 4 colheres de sopa de açúcar
- 2 colheres de sopa de chocolate em pó
- 1 ovo pequeno
- 3 colheres de sopa de leite
- 3 colheres de sopa de óleo



MODO DE PREPARO

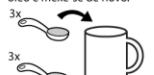
1. Juntar a farinha e o açúcar e o chocolate e mexer.



2. Adicionar o ovo (recheio!) e mexer com um garfo.



3. Por fim junta-se o leite e o óleo e mexe-se de novo.



4. Levar ao microondas em potência máxima durante três minutos. Prontinho!



Classe Bolo



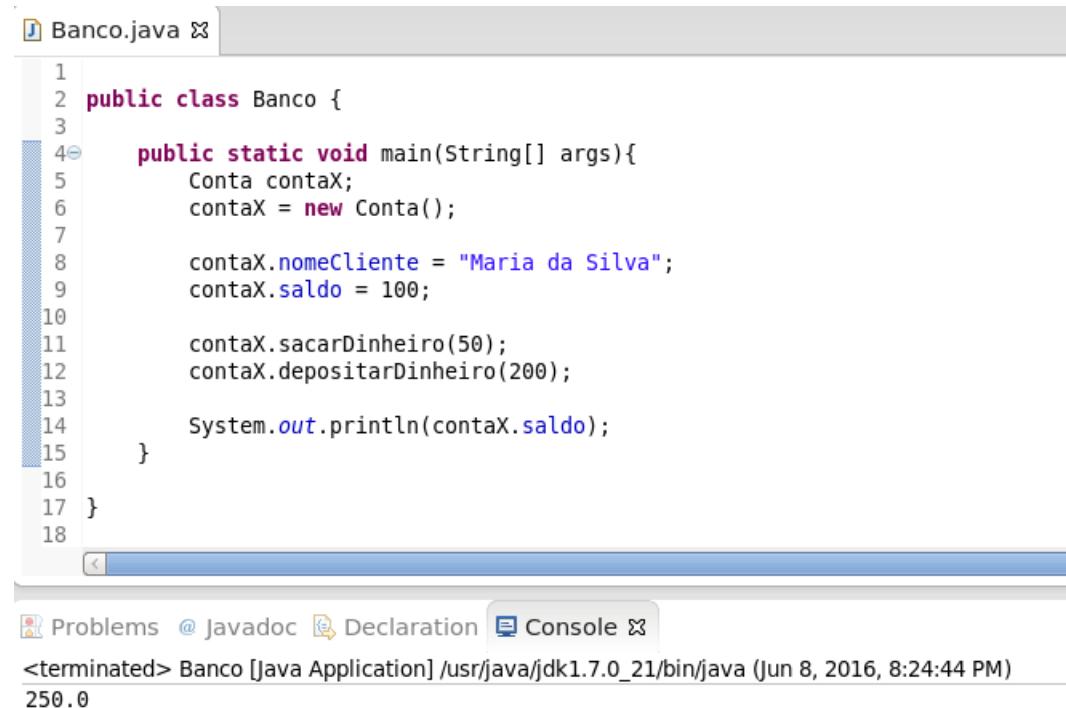
Objeto bolo1



Objeto bolo2

Classes

Criação do objeto Banco.java
Criação do objeto Cliente



The screenshot shows an IDE interface with the following components:

- Code Editor:** The tab "Banco.java" is selected. The code is a Java application that creates a bank account, deposits 200 units, and then prints the balance. The code is as follows:

```
1  public class Banco {  
2      public static void main(String[] args){  
3          Conta contaX;  
4          contaX = new Conta();  
5  
6          contaX.nomeCliente = "Maria da Silva";  
7          contaX.saldo = 100;  
8  
9          contaX.sacarDinheiro(50);  
10         contaX.depositarDinheiro(200);  
11  
12         System.out.println(contaX.saldo);  
13     }  
14 }  
15  
16  
17 }  
18 }
```

- Output Console:** The output shows the application has terminated and the command used to run it: <terminated> Banco [Java Application] /usr/java/jdk1.7.0_21/bin/java (Jun 8, 2016, 8:24:44 PM). The output itself is 250.0.

Conteúdo da Aula

- Revisão da aula anterior
- MapReduce
 - Fundamentos de Java
 - Biblioteca Hadoop
- MapReduce na Prática

Biblioteca Hadoop

Conjunto de classes que podem ser herdadas para utilização de funcionalidades pré-definidas

O conjunto de classe é dividido nos seguintes componentes:

- **Hadoop Common**: funcionalidades comuns para todas as aplicações Hadoop
- **Hadoop Distributed File System (HDFS)**: classes para manipulação do Sistema de arquivos do Hadoop
- **Hadoop MapReduce**: classes para o processamento das aplicações MapReduce
- **Hadoop YARN**: classes para o gerenciamento de jobs e recursos do cluster

Implementação MapReduce

Os seguintes passos serão executados:

1. Implementar as classes da aplicação
2. Gerar um arquivo JAR da aplicação
3. Executar o JAR no ambiente Hadoop
4. Visualizar o resultado da aplicação

Implementação MapReduce

Serão implementadas 3 classes:

ContaPalavrasMap.java

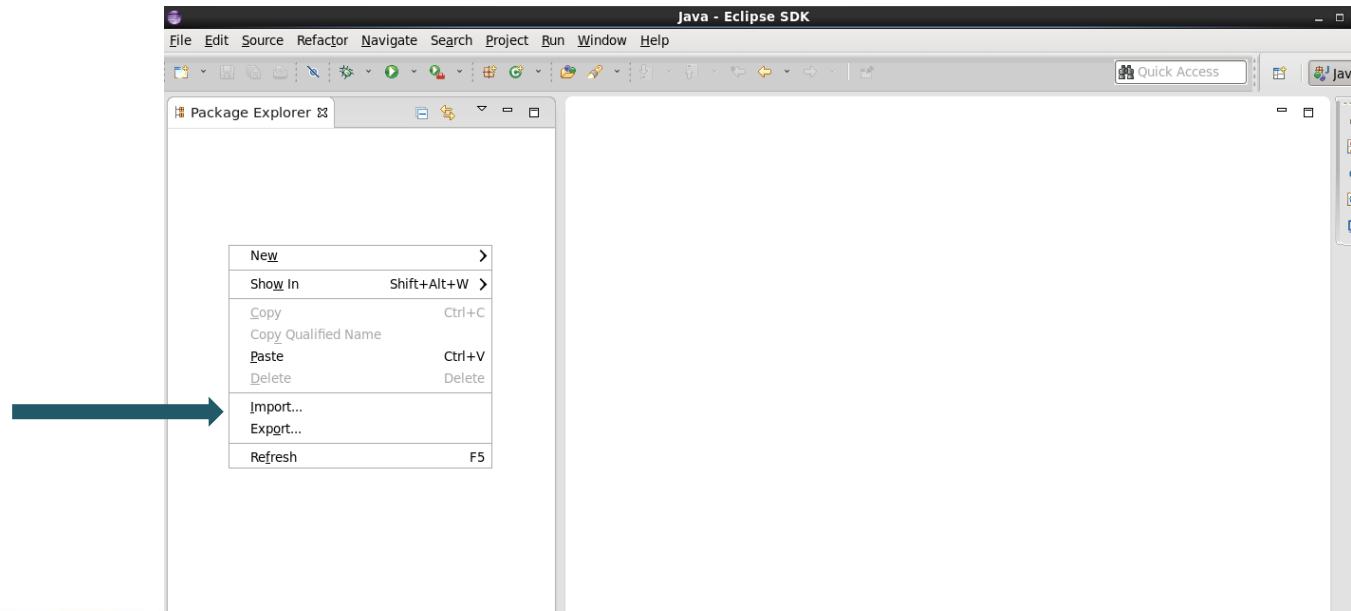
ContaPalavrasReduce.java

ContaPalavrasDriver.java

- Para facilitar o desenvolvimento das classes foi criado um projeto inicial chamado **ContaPalavras** contendo a estrutura-base das classes.

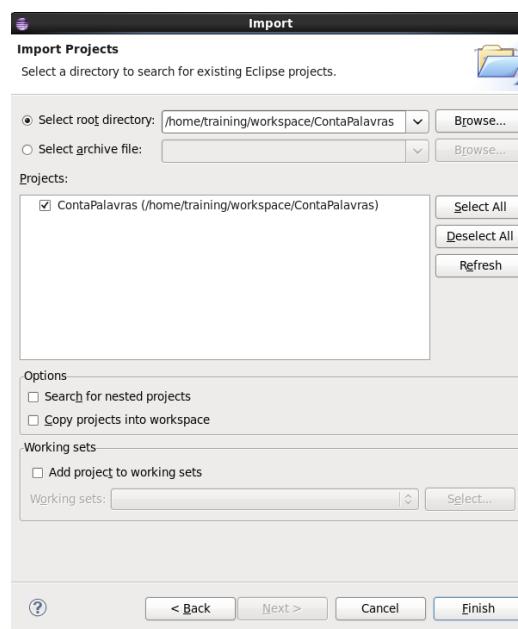
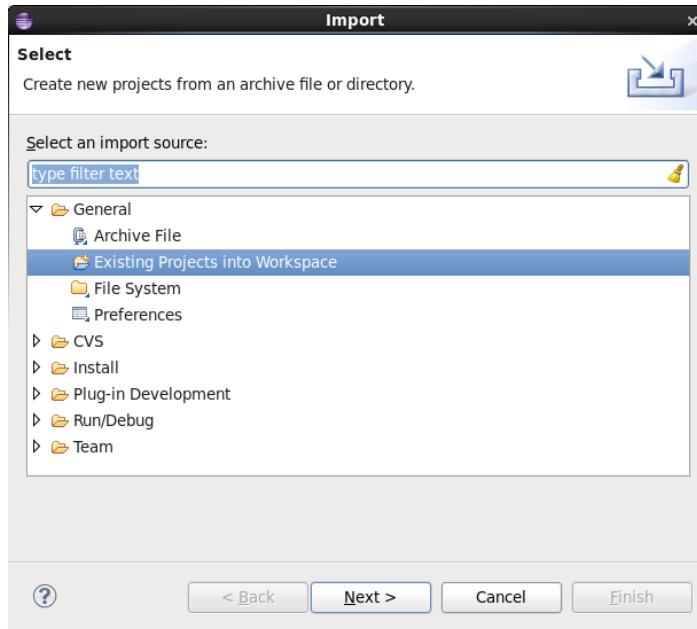
Implementação MapReduce

Acessar o projeto no Eclipse pela opção Import

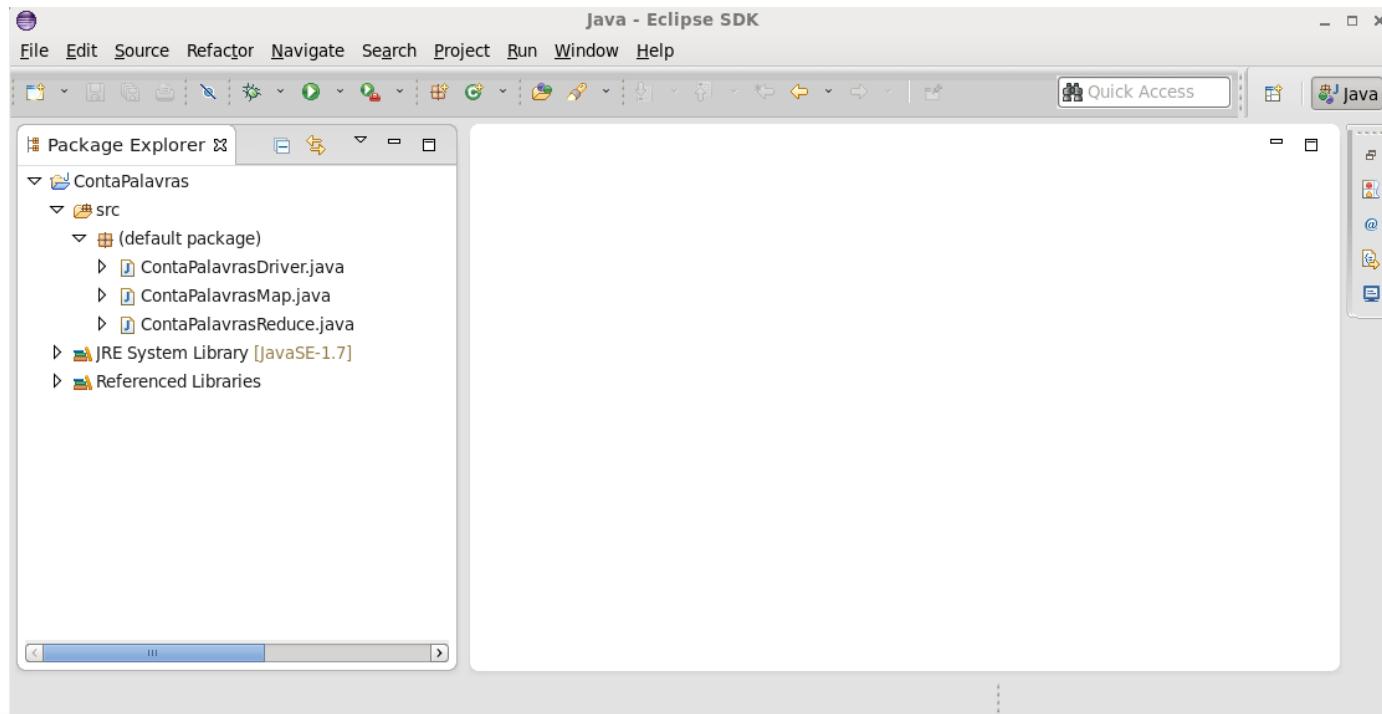


Implementação MapReduce

Selecionar o projeto ContaPalavras



Implementação MapReduce



Implementação MapReduce

Classe ContaPalavrasDriver

Implementação MapReduce

A classe ContaPalavrasDriver possui a seguinte estrutura:

Importação de classes da biblioteca Hadoop

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

Implementação MapReduce

Declaração da Classe e declaração do método

```
public class ContaPalavrasDriver {  
    public static void main(String[] args) throws Exception {  
    }  
}
```

Implementação MapReduce

Criação de um objeto Job

```
public static void main(String[] args) throws Exception {  
    Configuration conf = new Configuration();  
    Job job = Job.getInstance(conf, "contapalavrasdriver");  
  
    }  
}
```

Implementação MapReduce

Configuração do Job

```
public static void main(String[] args) throws Exception {  
    Configuration conf = new Configuration();  
    Job job = Job.getInstance(conf, "contapalavrasdriver");  
    job.setJarByClass(ContaPalavrasDriver.class);  
    job.setMapperClass(ContaPalavrasMap.class);  
    job.setReducerClass(ContaPalavrasReduce.class);  
    job.setOutputKeyClass(Text.class);  
    job.setOutputValueClass(IntWritable.class);  
}  
}
```

Implementação MapReduce

Definição dos arquivos de entrada e de saída

```
public static void main(String[] args) throws Exception {  
    Configuration conf = new Configuration();  
    Job job = Job.getInstance(conf, "contapalavrasdriver");  
    ...  
    job.setOutputValueClass(IntWritable.class);  
    FileInputFormat.addInputPath(job, new Path(args[0]));  
    FileOutputFormat.setOutputPath(job, new Path(args[1]));  
    ...  
}
```

Implementação MapReduce

Definição dos arquivos de entrada e de saída

```
public static void main(String[] args) throws Exception {  
    Configuration conf = new Configuration();  
    Job job = Job.getInstance(conf, "contapalavrasdriver");  
    ...  
    job.setOutputValueClass(IntWritable.class);  
    FileInputFormat.addInputPath(job, new Path(args[0]));  
    FileOutputFormat.setOutputPath(job, new Path(args[1]));  
    System.exit(job.waitForCompletion(true) ? 0 : 1);  
}
```

Implementação MapReduce

Classe ContaPalavrasMap

Implementação MapReduce

Como os dados são enviados às tarefas Map?

- Especificado por um objeto *InputFormat*
- O formato de envio deve ser especificado na classe driver
- Deve ser especificado a localização dos dados de entrada
- Esse objeto determina como os dados de entrada serão divididos

Implementação MapReduce

Exemplos de objetos do tipo InputFormat

- **TextInputFormat**
 - Default, lê cada linha terminada com “\n” como sendo um valor
- **FileInputFormat**
 - Classe abstrata usada para InputFormats baseados em arquivos
- **KeyValueTextInputFormat**
 - Determina linhas por meio de um separador (tab por padrão)

Implementação MapReduce

Como definir as chaves e valores na classe?

- Chaves e valores são objetos Java
- As chaves são objetos que implementam a interface **WritableComparable**
- Os valores são objetos que implementam a interface **Writable**
 - Exemplos: IntWritable, LongWritable, FloatWritable, Text...

Implementação MapReduce

A classe ContaPalavrasMap possui a seguinte estrutura:

Importação de classes da biblioteca Hadoop

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
```

Implementação MapReduce

Declaração da Classe

```
public class ContaPalavrasMap extends Mapper<Object, Text, Text, IntWritable>{  
}  
}
```

Implementação MapReduce

Declaração das variáveis utilizadas

```
public class ContaPalavrasMap extends Mapper<Object, Text, Text, IntWritable>{  
    private final static IntWritable numeroum = new IntWritable(1);  
  
}
```

Implementação MapReduce

Declaração do método map

```
public class ContaPalavrasMap extends Mapper<Object, Text, Text, IntWritable>{  
    private final static IntWritable numeroum = new IntWritable(1);  
    public void map(Object key, Text value, Context context) throws IOException,  
        InterruptedException {  
    }  
}
```

Implementação MapReduce

Declaração do código do método map

```
public class ContaPalavrasMap extends Mapper<Object, Text, Text, IntWritable>{
    private final static IntWritable numeroum = new IntWritable(1);
    public void map(Object key, Text value, Context context) throws IOException,
        InterruptedException {
        StringTokenizer itr = new StringTokenizer(value.toString().replaceAll("[^a-zA-Z ]", "")).toLowerCase();
        while (itr.hasMoreTokens()) {
            context.write(new Text(itr.nextToken()), numeroum);
        }
    }
}
```

Implementação MapReduce

Classe ContaPalavrasReduce

Implementação MapReduce

A classe ContaPalavrasReduce possui a seguinte estrutura:

Importação de classes da biblioteca Hadoop

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
```

Implementação MapReduce

Declaração da classe

```
public class ContaPalavrasReduce extends Reducer<Text,IntWritable,Text,IntWritable> {  
}  
}
```

Implementação MapReduce

Declaração do método reduce

```
public class ContaPalavrasReduce extends Reducer<Text, IntWritable, Text, IntWritable> {  
  
    public void reduce(Text key, Iterable<IntWritable> values, Context context  
        ) throws IOException, InterruptedException {  
  
    }  
}
```

Implementação MapReduce

Declaração do código do método reduce

```
public class ContaPalavrasReduce extends Reducer<Text, IntWritable, Text, IntWritable> {  
    public void reduce(Text key, Iterable<IntWritable> values, Context context  
        ) throws IOException, InterruptedException {  
        int soma = 0;  
        for (IntWritable val : values) {  
            soma += val.get();  
        }  
        context.write(key, new IntWritable(soma));  
    }  
}
```

Implementação MapReduce

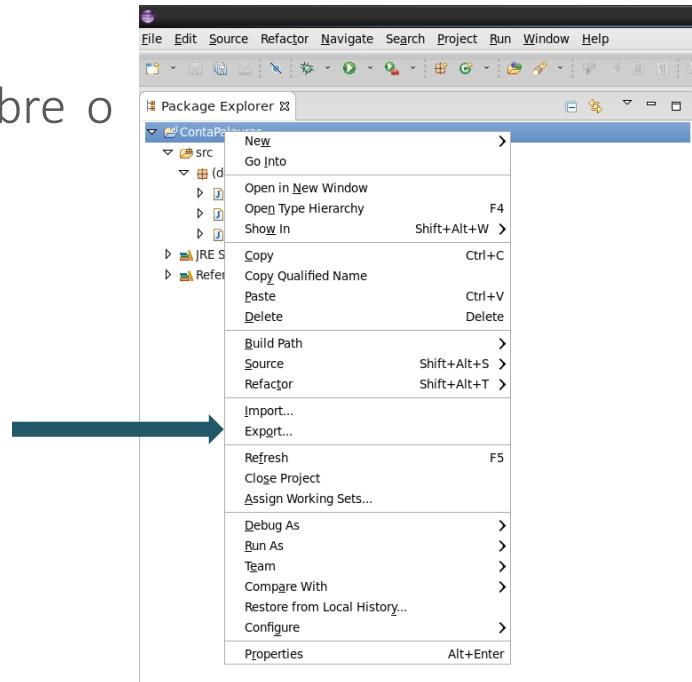
Os seguintes passos serão executados:

1. Implementar as classes da aplicação
2. Gerar um arquivo JAR da aplicação
3. Executar o JAR no ambiente Hadoop
4. Visualizar o resultado da aplicação

Implementação MapReduce

Selecionar a opção export

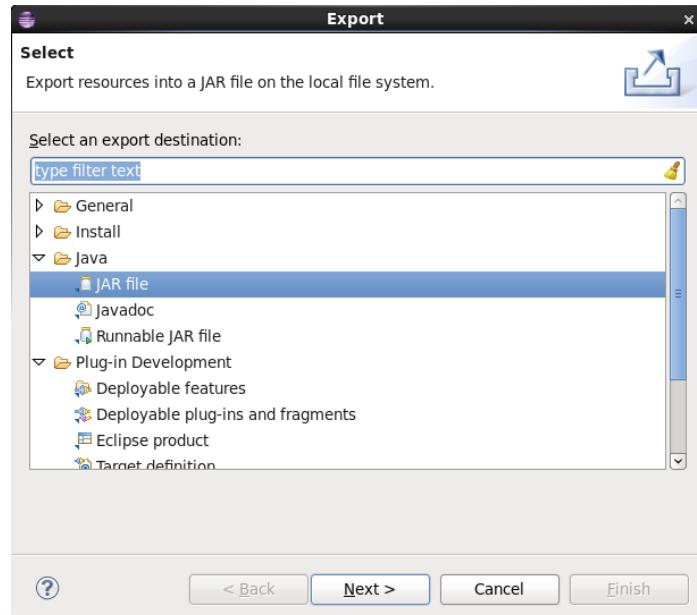
- Clique com o botão direito do mouse sobre o nome do projeto



Implementação MapReduce

Selecionar a opção JAR file

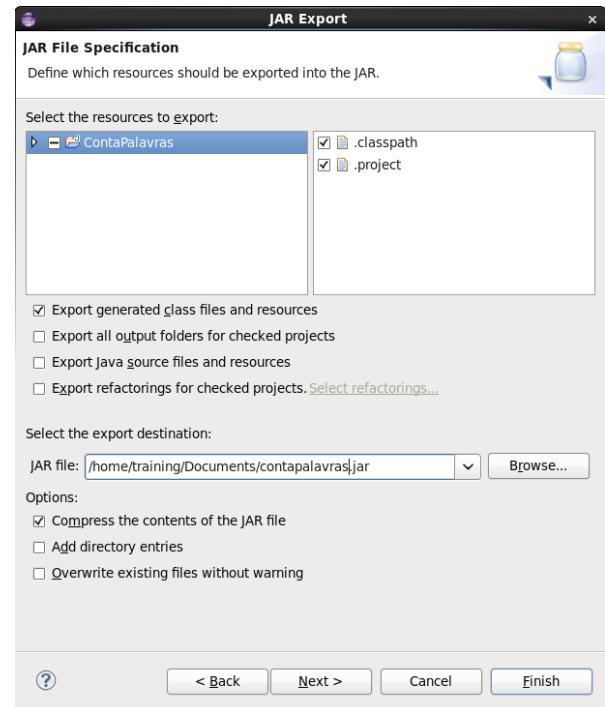
- Após seleção, pressione o botão Next



Implementação MapReduce

Definir a localização do JAR

- Definir o seguinte caminho:
/home/training/Documents/contapalavras.jar
- Pressione o botão Finish



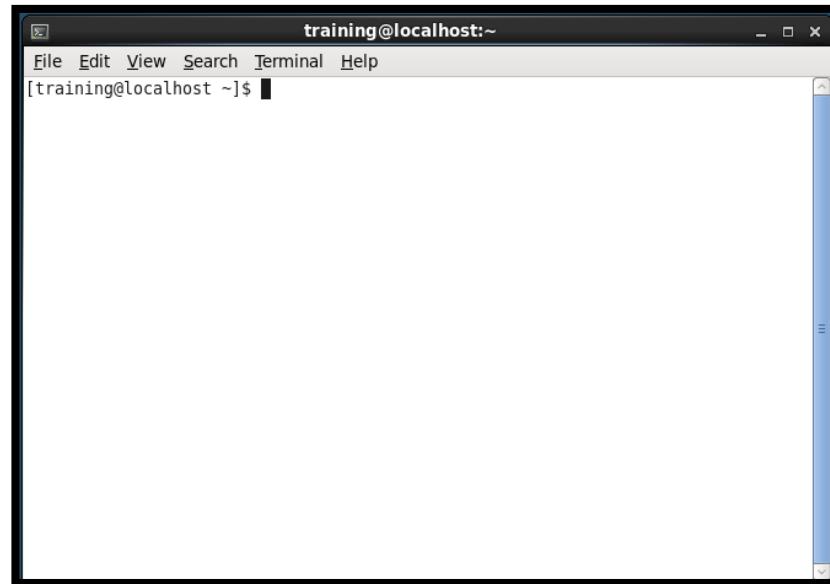
Implementação MapReduce

Os seguintes passos serão executados:

1. Implementar as classes da aplicação
2. Gerar um arquivo JAR da aplicação
3. Executar o JAR no ambiente Hadoop
4. Visualizar o resultado da aplicação

Implementação MapReduce

Abrir um terminal



Implementação MapReduce

Base de dados:

Mensagens da rede social Twitter sobre os temas: Big Data, data visualization, data privacy e Internet of Things

Exemplo

How can data visualization be used in the sales process? Learn how it can yield great insights. <http://t.co/Lq2lcpe7d1>

RT @Mona_Mourshed: "When Big Data Meets the Blackboard" <http://t.co/f3lr4yQW9A> via @TheAtlantic

Big data: reduce privacy risks - REPUTATION PROTECT <http://t.co/DrAb6LzFJJ>

Absolute privacy in handling #mhealth data. Patient security comes first."

RT @jose_garde: RT @MarshaCollier The Internet of Things and the Currency of Privacy <http://t.co/eXwmluX8sL>

Implementação MapReduce

Criar um diretório no HDFS como o nome de input_cp.

```
[training@localhost ~]$ hadoop fs -mkdir input_cp
```

Implementação MapReduce

Enviar o arquivo base_tw.txt para o diretório input_cp.

```
[training@localhost ~]$ hadoop fs -put ~/bases/base_tw.txt input_cp
```

Implementação MapReduce

Executar o jar contapalavras.jar, passando como parâmetro o diretório de entrada input_cp e um diretório de saída output_cp.

```
[training@localhost ~]$ hadoop jar ~/Documents/contapalavras.jar  
ContaPalavrasDriver input_cp output_cp
```

```
16/04/14 05:30:50 WARN mapred.JobClient: Use GenericOptionsParser for parsing the  
arguments. Applications should implement Tool for the same.
```

```
16/04/14 05:30:53 INFO input.FileInputFormat: Total input paths to process : 1
```

```
16/04/14 05:30:55 INFO mapred.JobClient: Running job: job_201512191358_0059
```

```
16/04/14 05:30:56 INFO mapred.JobClient: map 0% reduce 0%
```

```
16/04/14 05:31:52 INFO mapred.JobClient: map 100% reduce 0%
```

```
...
```

Implementação MapReduce

Os seguintes passos serão executados:

1. Implementar as classes da aplicação
2. Gerar um arquivo JAR da aplicação
3. Executar o JAR no ambiente Hadoop
4. Visualizar o resultado da aplicação

Implementação MapReduce

Visualizar o diretório de saída da aplicação

```
#Listar arquivos do diretório resultado
```

```
[training@localhost ~]$ hadoop fs -ls output_cp
```

```
Found 3 items
```

```
-rw-rw-rw- 1 training supergroup      0 2016-04-14 05:32 output_cp/_SUCCESS
drwxrwxrwx - training supergroup      0 2016-04-14 05:30 output_cp/_logs
-rw-rw-rw- 1 training supergroup 187407 2016-04-14 05:32 output_cp/part-r-00000
```

Implementação MapReduce

Visualizar o resultado da aplicação

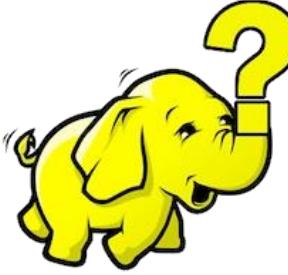
```
#Visualizar arquivo part-r-00000
[training@localhost ~]$ hadoop fs -cat output_cp/part-r-00000
Data      1
DataScience 3
DataScientist"      10
Facebook   1
NewsForBloggers  1
Python     1
...
...
```

Implementação MapReduce

Considerações

- O projeto criado pode ser utilizado como base para outras aplicações
- O mesmo código pode ser executado em um cluster com milhares de máquinas
- É possível criar uma aplicação com múltiplas iterações Map e Reduce
- Em caso de erros, verificar os registros de log do Hadoop
- Documentação do Hadoop: <http://hadoop.apache.org/>

Perguntas



rpereira@larc.usp.br

Referências Bibliográficas

WHITE, Tom. **Hadoop: The definitive guide.** " O'Reilly Media, Inc.", 2012.

DEAN, Jeffrey; GHEMAWAT, Sanjay. **MapReduce: simplified data processing on large clusters.** Communications of the ACM, v. 51, n. 1, p. 107-113, 2008.

GHEMAWAT, Sanjay; GOBIOFF, Howard; LEUNG, Shun-Tak. **The Google file system.** In: ACM SIGOPS Operating Systems Review. ACM, 2003. p. 29-43.

VAVILAPALLI, Vinod Kumar et al. **Apache hadoop yarn: Yet another resource negotiator.** In: Proceedings of the 4th annual Symposium on Cloud Computing. ACM, 2013. p. 5.

SHVACHKO, Konstantin et al. **The hadoop distributed file system.** In: Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. IEEE, 2010. p. 1-10.

Referências Bibliográficas

- GOLDMAN, Alfredo et al. **Apache Hadoop: conceitos teóricos e práticos, evolução e novas possibilidades**. XXXI Jornadas de atualizações em informática, p. 88-136, 2012.
- VENNER, Jason. **Pro Hadoop**. Apress, 2009.