

## Atividade em Laboratório Virtual 02

<b>Disciplina</b>	<b>HDM – Desenvolvimento de Soluções com MapReduce utilizando Hadoop</b>
-------------------	--

### Objetivos

A segunda atividade prática em laboratório possui como objetivos principais:

- ✓ Analisar o código-fonte que foi compilado e executado na Atividade 01.
- ✓ Realizar alterações nesse código.
- ✓ Compilar o código alterado.
- ✓ Executar o novo programa.
- ✓ Comparar os resultados.

Ao final desta atividade o aluno deverá ser capaz de alterar, compilar e executar um programa no Hadoop/MapReduce.

#### 1. Verificando o código-fonte que foi compilado:

Primeiramente iremos verificar o código-fonte que foi compilado na Atividade 01. Para isso, após realizar o login na máquina virtual (*login: igt i senha: igt i*) abra o terminal do Linux (Figura 1).

**Figura 1 – Tela inicial da máquina virtual.**

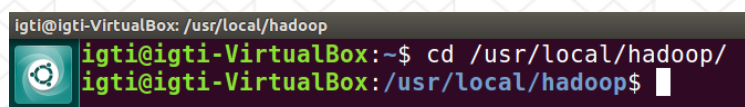


A ferramenta Hadoop já foi previamente instalada no ambiente virtual e encontra-se no diretório `/usr/local/hadoop`. Sendo assim, devemos ir até o diretório de instalação do Hadoop utilizando o seguinte comando:

```
cd /usr/local/hadoop
```

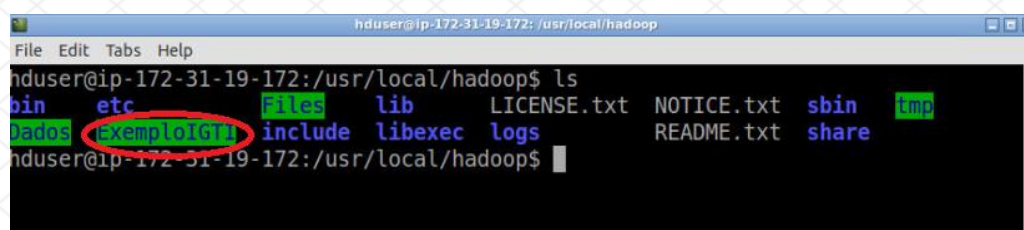
Após a digitação do comando acima, teremos a seguinte tela apresentada na Figura 2:

**Figura 2 – Diretório de instalação do Hadoop.**



Nesse momento você se encontra no diretório de instalação do Hadoop. O código-fonte que iremos verificar encontra-se dentro da pasta `ExemploIGTI`. Para verificar a existência dessa pasta, execute no *prompt* o comando `ls`, conforme apresentado na Figura 3:

**Figura 3 – Conteúdo do diretório Hadoop.**





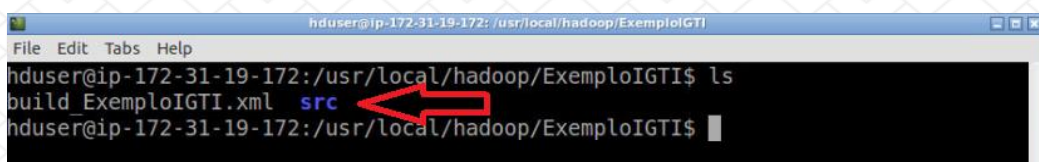
Dentro do diretório ExemploIGTI encontraremos um arquivo chamado build\_ExemploIGTI.xml. Esse arquivo possui as informações para compilação do nosso projeto. Além desse arquivo podemos encontrar o diretório src. Dentro desse diretório encontra-se o nosso código-fonte.

Vá para o diretório ExemploIGTI. Para isso execute o seguinte comando:

```
cd /usr/local/hadoop/ExemploIGTI
```

Liste o conteúdo do diretório ExemploIGTI com o comando ls. O conteúdo do diretório é apresentado pela Figura 4:

**Figura 4 – Conteúdo do diretório ExemploIGTI.**

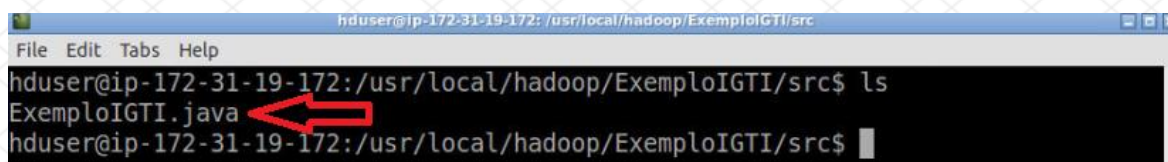


Vamos para o diretório src. Caso você já esteja no diretório ExemploIGTI, basta digitar cd src, caso contrário digite o comando abaixo:

```
cd /usr/local/hadoop/ExemploIGTI/src
```

Liste o conteúdo do diretório src com o comando ls. Perceba que dentro do diretório src temos o arquivo ExemploIGTI.java. Esse é o arquivo que contém nosso código-fonte que foi compilado na Atividade 01 e que iremos estudar melhor agora. A Figura 5 apresenta o conteúdo do diretório src.

**Figura 5 – Conteúdo do diretório src.**



Nesse momento vamos visualizar o conteúdo do arquivo ExemploIGTI.java. Para isso, digite o seguinte comando no *prompt* (você precisa estar no diretório src):

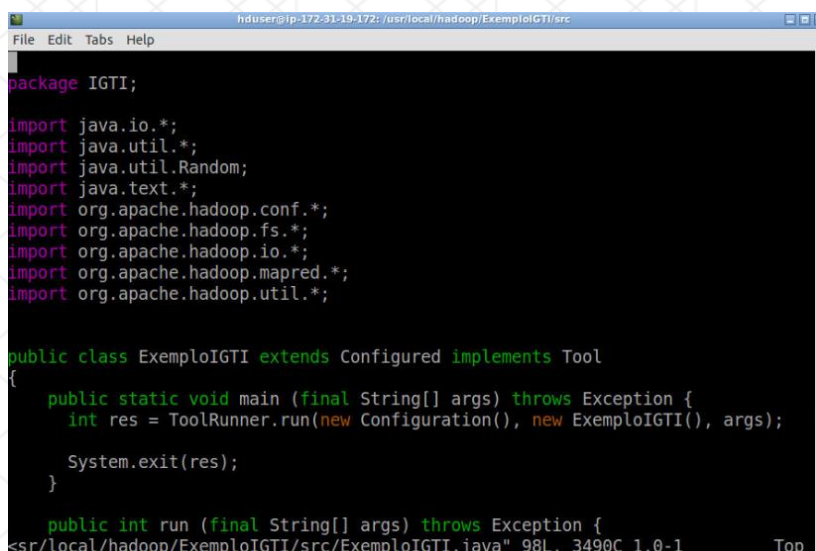
```
vim ExemploIGTI.java
```

ou

```
vim /usr/local/hadoop/ExemploIGTI/src/ExemploIGTI.java
```

Nesse momento a tela com o código-fonte da aplicação será exibida, conforme a Figura 6:

**Figura 6 – Código-fonte da aplicação.**



```
package IGTI;

import java.io.*;
import java.util.*;
import java.util.Random;
import java.text.*;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class ExemploIGTI extends Configured implements Tool
{
    public static void main (final String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new ExemploIGTI(), args);

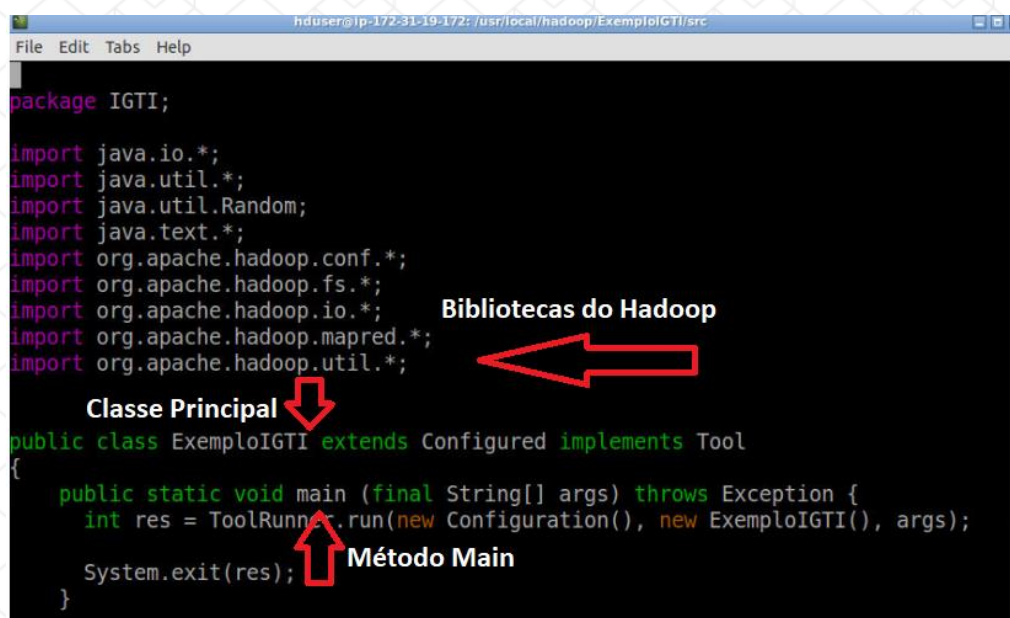
        System.exit(res);
    }

    public int run (final String[] args) throws Exception {

```

Um programa desenvolvido para ser executado no Hadoop/MapReduce segue a mesma estrutura de um programa Java comum, como você pode perceber na sequência de figuras abaixo:

**Figura 7 – Estrutura de um programa MapReduce.**



```
package IGTI;

import java.io.*;
import java.util.*;
import java.util.Random;
import java.text.*;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class ExemploIGTI extends Configured implements Tool
{
    public static void main (final String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new ExemploIGTI(), args);

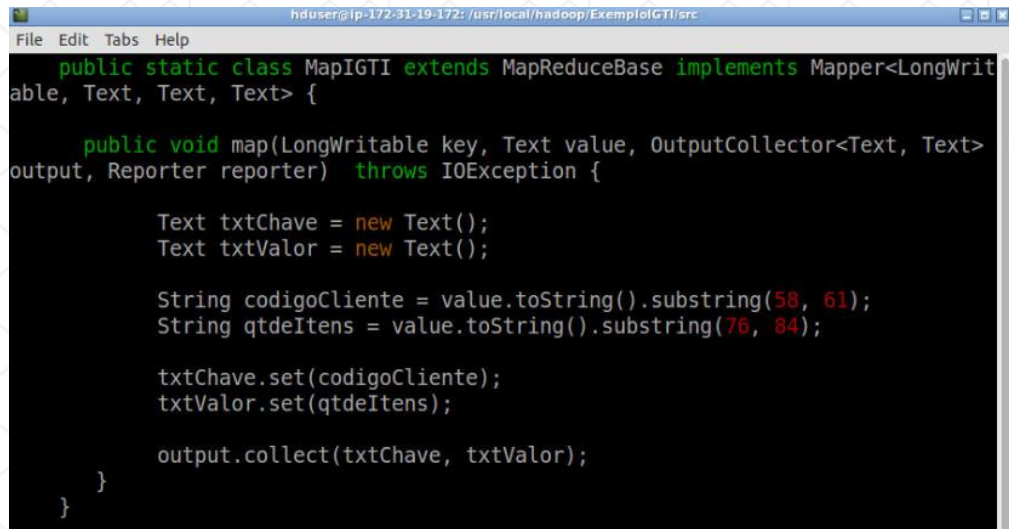
        System.exit(res);
    }

    public int run (final String[] args) throws Exception {

```

Na linha 53 do arquivo temos a definição da classe MapIGTI que é a responsável por manter o método Map do nosso modelo MapReduce. Na linha 55 temos o método Map implementado. Veja na Figura 8:



**Figura 8 – Implementação do método Map.**

```
public static class MapIGTI extends MapReduceBase implements Mapper<LongWritable, Text, Text, Text> {

    public void map(LongWritable key, Text value, OutputCollector<Text, Text>
output, Reporter reporter) throws IOException {

        Text txtChave = new Text();
        Text txtValor = new Text();

        String codigoCliente = value.toString().substring(58, 61);
        String qtdeItens = value.toString().substring(76, 84);

        txtChave.set(codigoCliente);
        txtValor.set(qtdeItens);

        output.collect(txtChave, txtValor);

    }
}
```

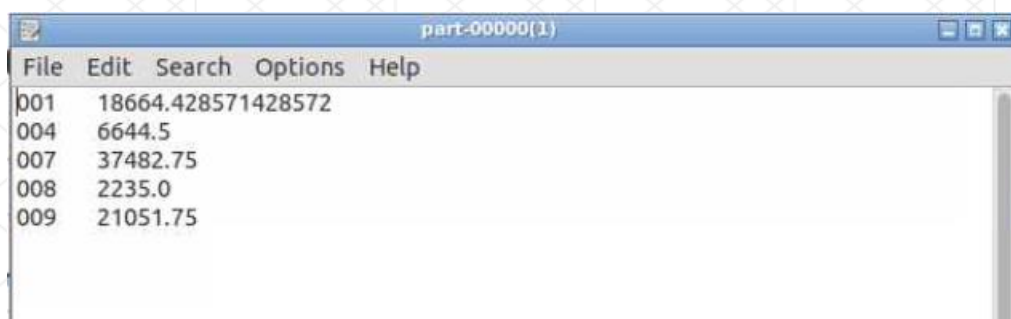
O que o método Map está fazendo? Está lendo um arquivo do HDFS, linha a linha e retirando o “código do cliente” (Posições 58 até 61) e a quantidade de itens que esse cliente comprou (Posições 76 até 84). Em seguida, esse conteúdo retirado é enviado para a função Reduce.

Como o Hadoop sabe qual arquivo buscar? Veja na linha 35 do arquivo que usamos o método `copyFromLocalFile`. Esse método copia um arquivo do sistema de arquivos do sistema operacional para o sistema de arquivos distribuído do Hadoop (HDFS). Veja o conteúdo desse arquivo em `/usr/local/hadoop/Dados/ArquivoBigData.txt`.

A função Reduce está implementada na classe `ReduceIGTI`, a partir da linha 71. O que a função Reduce está fazendo? Ela soma todos os itens comprados por um cliente (linha 80), por meio da variável `acumuladorItens`. Além disso, a função Reduce conta quantas compras o cliente fez e essa contagem é armazenada na variável `contaVendas` (linha 70). Ao final a média é calculada por meio da variável `média`. Por fim, o valor médio de compra de cada cliente é gravado no HDFS, por meio do método `output.collect`, onde `key` é o “código do cliente” e `value` é a média comprada.

A Figura 9 apresenta o resultado da execução do programa (que você já viu na Atividade 01). Procure entender o que realmente foi feito comparando o arquivo de entrada disponível em `/usr/local/hadoop/Dados/ArquivoBigData.txt`, com o arquivo final gerado e gravado no HDFS.

**Figura 9 – Resultado do processamento da Atividade 01.**

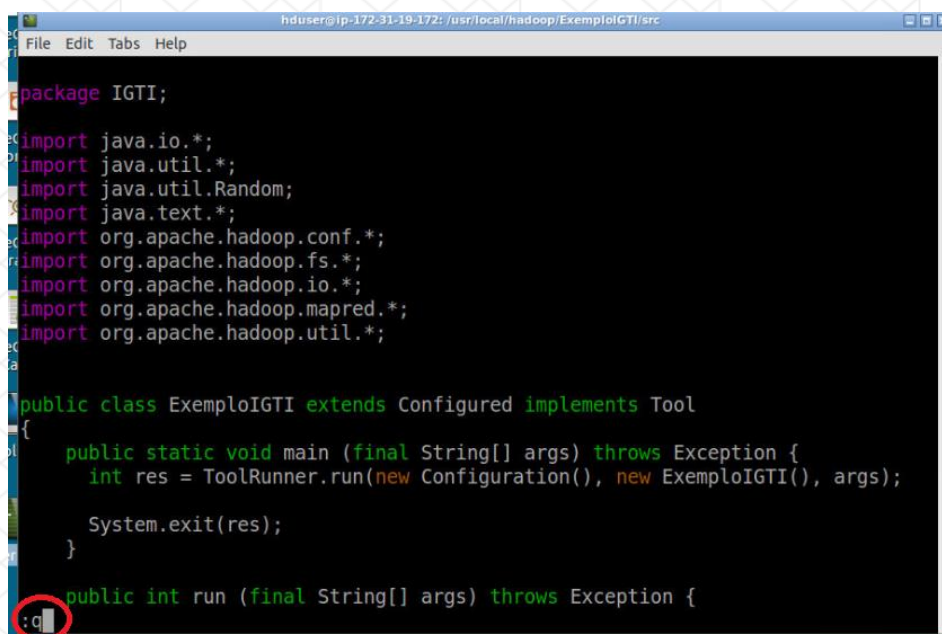


ID	Valor
001	18664.428571428572
004	6644.5
007	37482.75
008	2235.0
009	21051.75

**Atenção:** para sair do editor de texto, digite (Figura 10):

[ESC] :q [ENTER]

**Figura 10 – Saindo do editor Vim.**



```
package IGTI;

import java.io.*;
import java.util.*;
import java.util.Random;
import java.text.*;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class ExemploIGTI extends Configured implements Tool
{
    public static void main (final String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new ExemploIGTI(), args);

        System.exit(res);
    }

    public int run (final String[] args) throws Exception {
```

## 2. Alterando o código fonte:

Na Seção 1 desse documento nós entendemos como funciona a estrutura de um programa Hadoop. Nesse momento vamos alterar o programa que compilamos e executamos na Atividade 01.

O objetivo dessa alteração é, ao invés de buscar a média dos itens vendidos por cliente, buscar a maior venda realizada por cliente.

Para realizar essa alteração, vamos trabalhar na função Reduce. A função Map continuará fazendo o mesmo trabalho, que é “recortar” o “código do cliente” e a



“quantidade de itens” do arquivo original.

Altere a função Reduce para que ela consiga identificar a maior venda de cada cliente. A Figura 12 apresenta essa alteração (destacada no quadro azul).

Para entrar no modo de edição, pressione a tecla “e”.

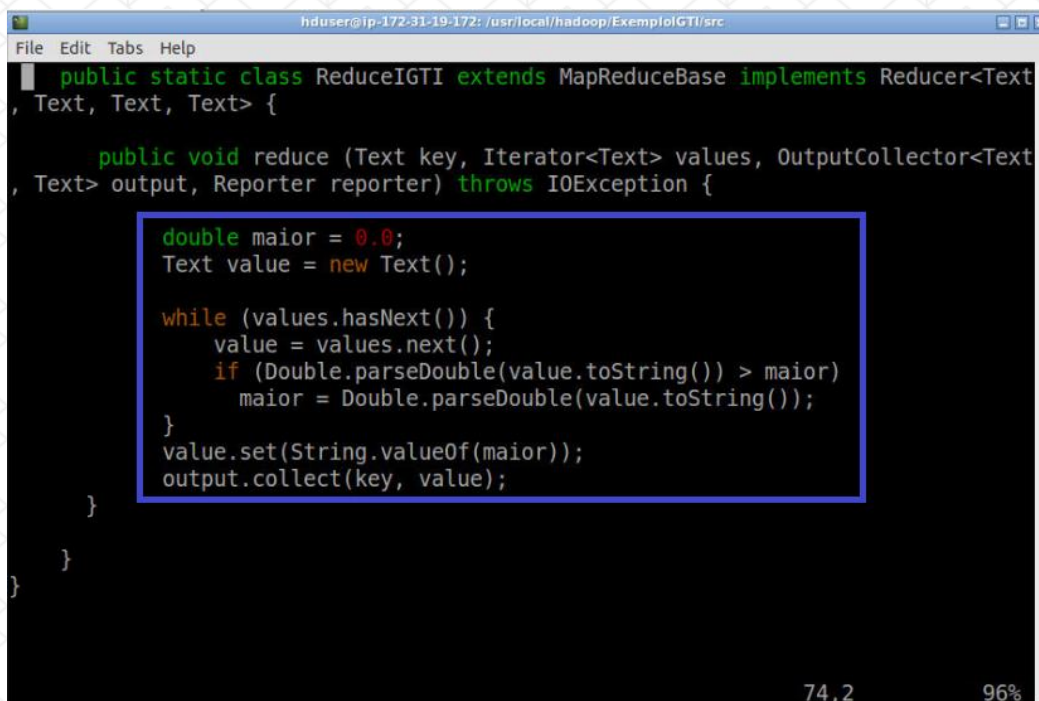
Quando estiver pronto para realizar a alteração, digite a letra “a”. Isso fará com que o arquivo fique em modo de edição. Veja se a palavra INSERT encontra-se presente no canto inferior esquerdo, conforme a Figura 11:

**Figura 11 – Colocando o arquivo em modo de inserção.**



O código da função Reduce ficará no seguinte formato:

**Figura 12 – Novo código da função Reduce.**



**Atenção:** para sair do editor de textos salvando as alterações, digite o seguinte comando:

[ESC] :wq [ENTER]

### **3. Próximos passos:**

Daqui para frente repita exatamente a Atividade 01, porém agora você vai realizá-la com o novo código que você acabou de alterar.

Os próximos passos (previstos na Atividade 01) envolvem as tarefas de formatar o HDFS, iniciar os serviços do Hadoop, compilar o código e executá-lo.

Ao final da nova execução (maior quantidade), compare os resultados com a execução que você fez anteriormente na Atividade 01 (média).

Essa atividade é pontuada e você deverá preencher o documento anexo a ela (Relatório de Atividade Prática Prévia).

### **4. Conclusão:**

Nessa atividade prática realizamos a alteração do código-fonte, compilação e execução de um programa Hadoop.