

Practical no 1

Aim: Write a program to demonstrate bitwise operation.

Code:

```
public class Test {  
  
    public static void main(String args[]) {  
        int a = 60;  
        int b = 13;  
        int c = 0;  
  
        c = a & b;  
        System.out.println("a & b = " + c );  
  
        c = a | b;  
        System.out.println("a | b = " + c );  
  
        c = a ^ b;  
        System.out.println("a ^ b = " + c );  
  
        c = ~a;  
        System.out.println("~a = " + c );  
  
        c = a << 2;  
        System.out.println("a << 2 = " + c );  
    }  
}
```

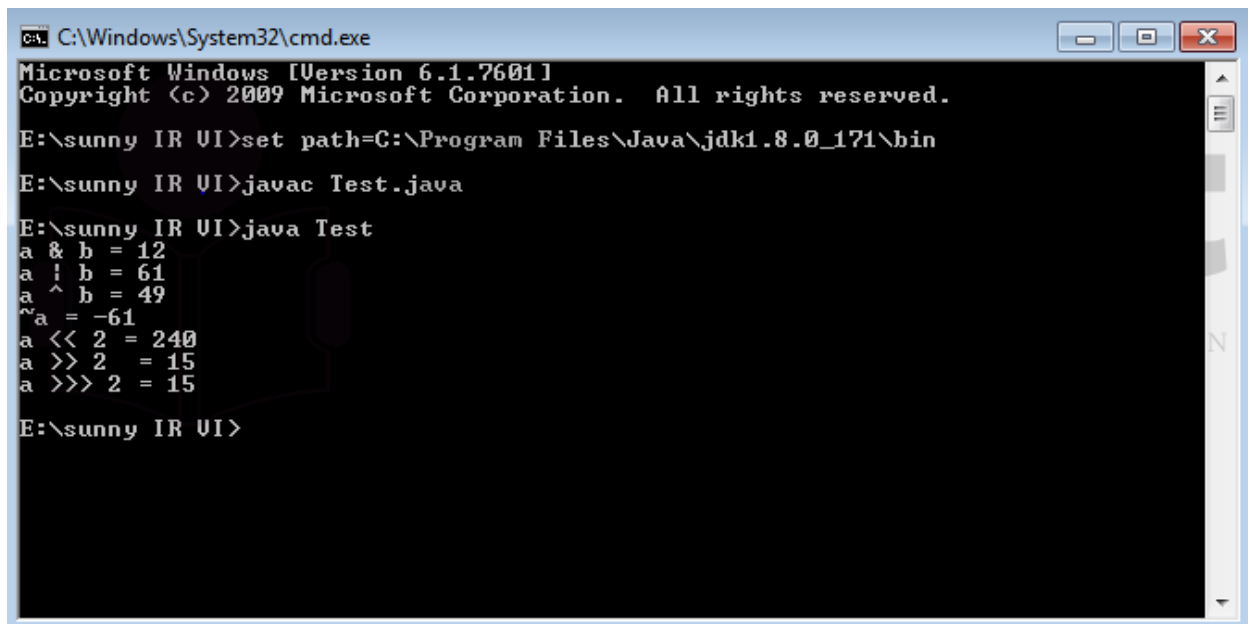
```

c = a >> 2;
System.out.println("a >> 2 = " + c);

c = a >>> 2;    /* 15 = 0000 1111 */
System.out.println("a >>> 2 = " + c);
}
}

```

Output:



```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

E:\sunny IR UI>set path=C:\Program Files\Java\jdk1.8.0_171\bin
E:\sunny IR UI>javac Test.java
E:\sunny IR UI>java Test
a & b = 12
a | b = 61
a ^ b = 49
~a = -61
a << 2 = 240
a >> 2 = 15
a >>> 2 = 15
E:\sunny IR UI>

```

Practical No.2

Aim:- Implement Page Rank Algorithm.

Code:-

```
import java.util.*;
import java.io.*;
public class PageRank {

    public int path[][] = new int[10][10];
    public double pagerank[] = new double[10];

    public void calc(double totalNodes){

        double InitialPageRank;
        double OutgoingLinks=0;
        double DampingFactor = 0.85;
        double TempPageRank[] = new double[10];

        int ExternalNodeNumber;
        int InternalNodeNumber;
        int k=1; // For Traversing
        int ITERATION_STEP=1;

        InitialPageRank = 1/totalNodes;

        System.out.printf(" Total Number of Nodes :"+totalNodes+"\t Initial PageRank of
        All Nodes :"+InitialPageRank+"\n");
```

```

for(k=1;k<=totalNodes;k++)
{
    this.pagerank[k]=InitialPageRank;
}
System.out.printf("\n Initial PageRank Values , 0th Step \n");
for(k=1;k<=totalNodes;k++)
{
    System.out.printf(" Page Rank of "+k+" is :\t"+this.pagerank[k]+" \n");
}
while(ITERATION_STEP<=2) // Iterations
{
    for(k=1;k<=totalNodes;k++)
    {
        TempPageRank[k]=this.pagerank[k];
        this.pagerank[k]=0;
    }

    for(InternalNodeNumber=1;InternalNodeNumber<=totalNodes;InternalNodeNum
ber++)
    {
        for(ExternalNodeNumber=1;ExternalNodeNumber<=totalNodes;ExternalNodeNu
mber++)
        {
            if(this.path[ExternalNodeNumber][InternalNodeNumber] == 1)

```

```

{
    k=1;

    OutgoingLinks=0; // Count the Number of Outgoing Links for each
    ExternalNodeNumber

    while(k<=totalNodes)
    {
        if(this.path[ExternalNodeNumber][k] == 1 )
        {
            OutgoingLinks=OutgoingLinks+1; // Counter for Outgoing Links
        }
        k=k+1;
    }

    this.pagerank[InternalNodeNumber]+=TempPageRank[ExternalNodeNumber]*(1/
    OutgoingLinks);
}
}

System.out.printf("\n After "+ITERATION_STEP+"th Step \n");

for(k=1;k<=totalNodes;k++)

    System.out.printf(" Page Rank of "+k+" is :\t"+this.pagerank[k]+" \n");

    ITERATION_STEP = ITERATION_STEP+1;
}

for(k=1;k<=totalNodes;k++)
{

    this.pagerank[k]=(1-DampingFactor)+ DampingFactor*this.pagerank[k];

```

```

    }
    System.out.printf("\n Final Page Rank : \n");
    for(k=1;k<=totalNodes;k++)
    {
        System.out.printf(" Page Rank of "+k+" is :\t"+this.pagerank[k]+" \n");
    }
}

public static void main(String args[])
{
    int nodes,i,j,cost;
    Scanner in = new Scanner(System.in);
    System.out.println("Enter the Number of WebPages \n");
    nodes = in.nextInt();
    PageRank p = new PageRank();
    System.out.println("Enter the Adjacency Matrix with 1->PATH & 0->NO
PATH Between two WebPages: \n");
    for(i=1;i<=nodes;i++)
        for(j=1;j<=nodes;j++)
        {
            p.path[i][j]=in.nextInt();
            if(j==i)
                p.path[i][j]=0;
        }
    p.calc(nodes);
}

```

}

Output:-

```
C:\Windows\system32\cmd.exe

D:\IR\IR>set path="C:\Program Files\Java\jdk1.8.0_171\bin"
D:\IR\IR>javac PageRank.java
D:\IR\IR>java PageRank
Enter the Number of WebPages
4
Enter the Adjacency Matrix with 1->PATH & 0->NO PATH Between two WebPages:
0 1 0 1
1 2 3 0
2 1 0 1
0 2 1 1
Total Number of Nodes :4.0      Initial PageRank of All Nodes :0.25

Initial PageRank Values , 0th Step
Page Rank of 1 is :      0.25
Page Rank of 2 is :      0.25
Page Rank of 3 is :      0.25
Page Rank of 4 is :      0.25

After 1th Step
Page Rank of 1 is :      0.25
Page Rank of 2 is :      0.25
Page Rank of 3 is :      0.25
Page Rank of 4 is :      0.25

After 2th Step
Page Rank of 1 is :      0.25
Page Rank of 2 is :      0.25
Page Rank of 3 is :      0.25
Page Rank of 4 is :      0.25

Final Page Rank :
Page Rank of 1 is :      0.36250000000000004
Page Rank of 2 is :      0.36250000000000004
Page Rank of 3 is :      0.36250000000000004
Page Rank of 4 is :      0.36250000000000004
```

Practical No.3

Aim:- Implement Dynamic programming algorithm for computing the edit distance between

Code:

```
public class EditDistanceProblem
{
    public int editDistanceRecursion(String s1,String s2,int m,int n)
    {
        if(m==0)
            return n;
        if(n==0)
            return m;
        if(s1.charAt(m-1)==s2.charAt(n-1))
            return editDistanceRecursion(s1,s2,m-1,n-1);

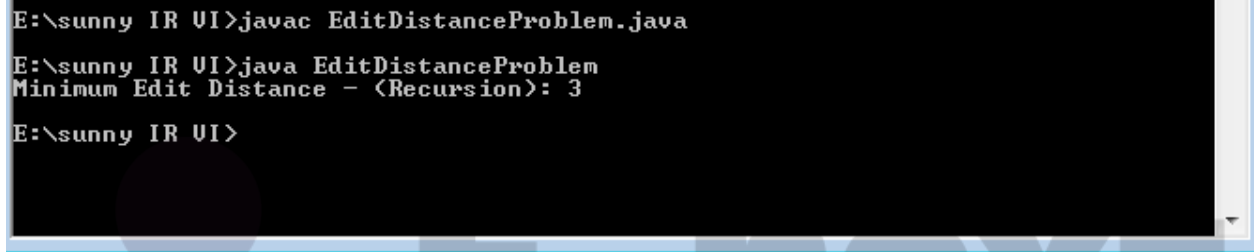
        return 1 + Math.min(editDistanceRecursion(s1, s2, m, n-1 ),
            Math.min(editDistanceRecursion(s1, s2 , m-1 , n ),
                editDistanceRecursion(s1 ,s2 , m-1 , n-1) ) );
    }

    public static void main(String[] args)
    {
        String s1 = "horizon";
```



```
String s2 = "horizontal";  
EditDistanceProblem ed = new EditDistanceProblem();  
System.out.println("Minimum Edit Distance - (Recursion): " +  
    ed.editDistanceRecursion(s1,s2,s1.length(),s2.length() ) );  
}  
}
```

Output:



```
E:\sunny IR UI>javac EditDistanceProblem.java  
E:\sunny IR UI>java EditDistanceProblem  
Minimum Edit Distance - (Recursion): 3  
E:\sunny IR UI>
```



E-next
THE NEXT LEVEL OF EDUCATION

Practical No.4

Aim:- Write a program to Compute Similarity between two text documents.

Code:-

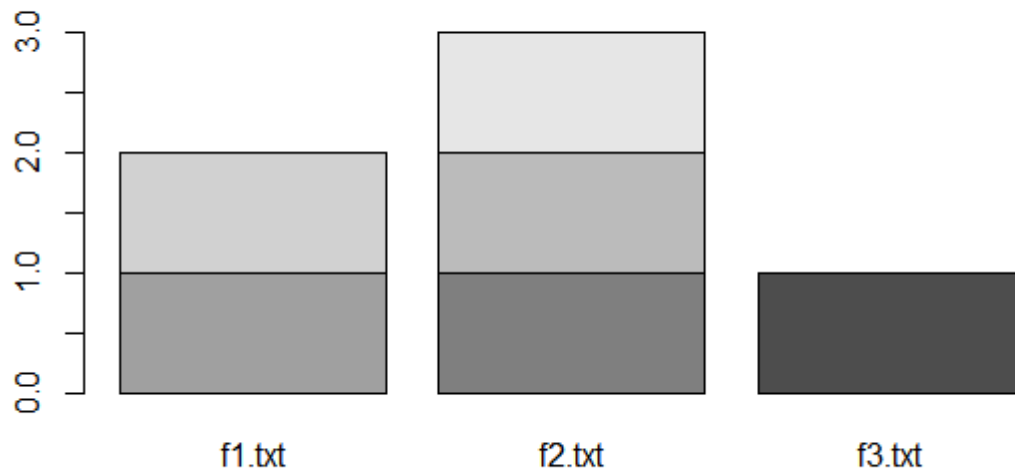
```
> install.packages('tm')
> install.packages('ggplot2')
> install.packages('textreuse')
> install.packages('devtools')
> install.packages('NLP')
> library('tm')
> require('NLP')
> library('tm')
> setwd('C:/r-corpus/')
> my.corpus<-Corpus(DirSource("C:/r-corpus"))
> my.corpus<-tm_map(my.corpus,removeWords,stopwords(kind = "english"))
> my.tdm<-TermDocumentMatrix(my.corpus)
> my.df<-as.data.frame(inspect(my.tdm))
```

Output:-

```
<<TermDocumentMatrix (terms: 6, documents: 3)>>
Non-/sparse entries: 6/12
Sparsity           : 67%
Maximal term length: 7
weighting          : term frequency (tf)
```

Terms	Docs		
	f1.txt	f2.txt	f3.txt
boy	0	0	1
cricket	0	1	0
divesh	1	0	0
like	0	1	0
name	1	0	0
play	0	1	0

```
> barplot(as.matrix(my.tdm))
```



E-next
THE NEXT LEVEL OF EDUCATION

Practical No. 5

Aim: Write a map-reduce program to count the number of occurrences of each alphabetic character in the given dataset. The count for each letter should be case-insensitive (i.e., include both upper-case and lower-case versions of the letter; Ignore non-alphabetic characters).

Steps:

1. Install Java 8: Download Java 8 from the link:

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

a. Set environmental variables:

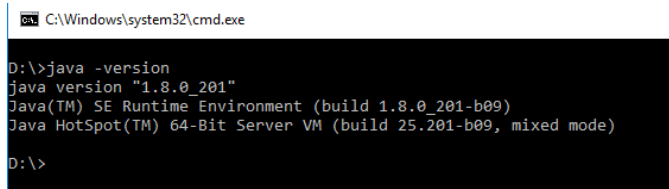
i. User variable:

- Variable: JAVA_HOME
- Value: C:\java

ii. System variable:

- Variable: PATH
- Value: C:\java\bin

b. Check on cmd, see below:



```
C:\Windows\system32\cmd.exe
D:\>java -version
java version "1.8.0_201"
Java(TM) SE Runtime Environment (build 1.8.0_201-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)
D:\>
```

2. Install Eclipse Mars. Download it from the link: <https://eclipse.org/downloads/> and extract it into C drive.

a. Set environmental variables:

i. User variable:

- Variable: ECLIPSE_HOME

- Value: C:\eclipse

ii. System variable:

- Variable: PATH
- Value: C:\eclipse\bin

b. Download “hadoop2x-eclipse-plugin-master.”

You will see three Jar files on the path “hadoop2x-eclipse-plugin-master\release.”

Copy these three jar files and paste them into “C:\eclipse\dropins.”

c. Download “slf4j-1.7.21.”

Copy Jar files from this folder and paste them to “C:\eclipse\plugins”.

3. Download Hadoop-2.6.x: download Hadoop 2.6.x from the link:

<http://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-2.6.2/hadoop-2.6.2.tar.gz>

a. Put extracted Hadoop-2.6.x files into D drive.

Note that do not put these extracted files into C drive, where you installed your Windows.

b. Download “hadoop-common-2.6.0-bin-master” from the link:

<https://github.com/amihalik/hadoop-common-2.6.0-bin/tree/master/bin>.

You will see 11 files there.

Paste all these files into the “bin” folder of Hadoop-2.6.x.

c. Create a “data” folder inside Hadoop-2.6.x, and

also create two more folders in the “data” folder as “data” and “name.”

d. Create a folder to store temporary data during execution of a project, such as “D:\hadoop\temp.”

e. Create a log folder, such as “D:\hadoop\userlog”

f. Go to Hadoop-2.6.x /etc / Hadoop and edit four files:

i. core-site.xml

ii. hdfs-site.xml

iii. mapred.xml

iv. yarn.xml

core-site.xml

```
<?xml version="1.0" encoding="UTF-8"?> <?xml-stylesheet type="text/xsl"
href="configuration.xsl"?> <!-- Licensed under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with the License.
You may obtain a copy of the License at
```

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. See accompanying LICENSE file. -->

```
<!-- Put site-specific property overrides in this file. -->
```

```
<configuration> <property>
```

```
<name>hadoop.tmp.dir</name>
```

```
<value>D:\hadoop\temp</value>
```

```
</property>
```

```
<property>
```

```
<name>fs.default.name</name>
```

```
<value>hdfs://localhost:50071</value>
```

```
</property>
```

```
</configuration>
```

hdfs-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?> <!-- Licensed under
the Apache License, Version 2.0 (the "License"); you may not use this file except
in compliance with the License. You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed
under the License is distributed on an "AS IS" BASIS, WITHOUT
WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and limitations
under the License. See accompanying LICENSE file. -->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property><name>dfs.replication</name><value>1</value></property>
<property>
<name>dfs.namenode.name.dir</name><value>/hadoop2.6.2/data/name</value><
final>true</final></property>
<property><name>dfs.datanode.data.dir</name><value>/hadoop2.6.2/data/data</
value><final>true</final> </property> </configuration>
```

mapred.xml

```
<?xml version="1.0"?> <configuration <property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>mapred.job.tracker</name>
<value>localhost:9001</value>
```

```

</property>
<property>
<name>mapreduce.application.classpath</name>
<value>/hadoop-2.6.2/share/hadoop/mapreduce/*,
/hadoop-2.6.2/share/hadoop/mapreduce/lib/*,
/hadoop-2.6.2/share/hadoop/common/*,
/hadoop-2.6.2/share/hadoop/common/lib/*,
/hadoop-2.6.2/share/hadoop/yarn/*,
/hadoop-2.6.2/share/hadoop/yarn/lib/*,
/hadoop-2.6.2/share/hadoop/hdfs/*,
/hadoop-2.6.2/share/hadoop/hdfs/lib/*,
</value>
</property></configuration>

```

yarn-site.xml

<?xml version="1.0"?> <!-- Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. See accompanying LICENSE file. -->

```

<configuration>  <property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>

```



```

<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
<name>yarn.nodemanager.log-dirs</name>
<value>D:\hadoop\userlog</value><final>true</final>
</property>
<property><name>yarn.nodemanager.local-
dirs</name><value>D:\hadoop\temp\nm-localdir</value></property>
<property>
<name>yarn.nodemanager.delete.debug-delay-sec</name>
<value>600</value></property>
<property><name>yarn.application.classpath</name>
<value>/hadoop-2.6.2/,/hadoop-
2.6.2/share/hadoop/common*/,/hadoop2.6.2/share/hadoop/common/lib*/,/hadoop-
2.6.2/share/hadoop/hdfs*/,/hadoop2.6.2/share/hadoop/hdfs/lib*/,/hadoop-
2.6.2/share/hadoop/mapreduce*/,/hadoop2.6.2/share/hadoop/mapreduce/lib*/,/hado
op-2.6.2/share/hadoop/yarn*/,/hadoop2.6.2/share/hadoop/yarn/lib*</value>
</property></configuration>

```

g. Go to the location: “Hadoop-2.6.0/etc/hadoop,” and edit “hadoop-env.cmd” by writing set JAVA_HOME= **C:\Progra~1\Java\jdk1.8.0_201**

h. Set environmental variables:

Do: My computer ☐ Properties ☐ Advance system settings ☐ Advanced ☐
Environmental variables

i. User variables:

- Variable: HADOOP_HOME
- Value: D:\hadoop-2.6.0

ii. System variable

- Variable: Path
- Value: D:\hadoop-2.6.2\bin D:\hadoop-2.6.2\sbin D:\hadoop-2.6.2\share\hadoop\common* D:\hadoop-2.6.2\share\hadoop\hdfs D:\hadoop-2.6.2\share\hadoop\hdfs\lib* D:\hadoop-2.6.2\share\hadoop\hdfs* D:\hadoop-2.6.2\share\hadoop\yarn\lib* D:\hadoop-2.6.2\share\hadoop\yarn* D:\hadoop-2.6.2\share\hadoop\mapreduce\lib* D:\hadoop-2.6.2\share\hadoop\mapreduce* D:\hadoop-2.6.2\share\hadoop\common\lib*

i. Check on cmd; see below

```
D:\hadoop-2.6.2\bin>hadoop version
Hadoop 2.6.2
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r 0cfd050febe4a30b1ee1551dcc527589509fb681
Compiled by jenkins on 2015-10-22T00:42Z
Compiled with protoc 2.5.0
From source with checksum f9ebb94bf5bf9bec892825ede28baca
This command was run using /D:/hadoop-2.6.2/share/hadoop/common/hadoop-common-2.6.2.jar
D:\hadoop-2.6.2\bin>
```

j. Format name-node:

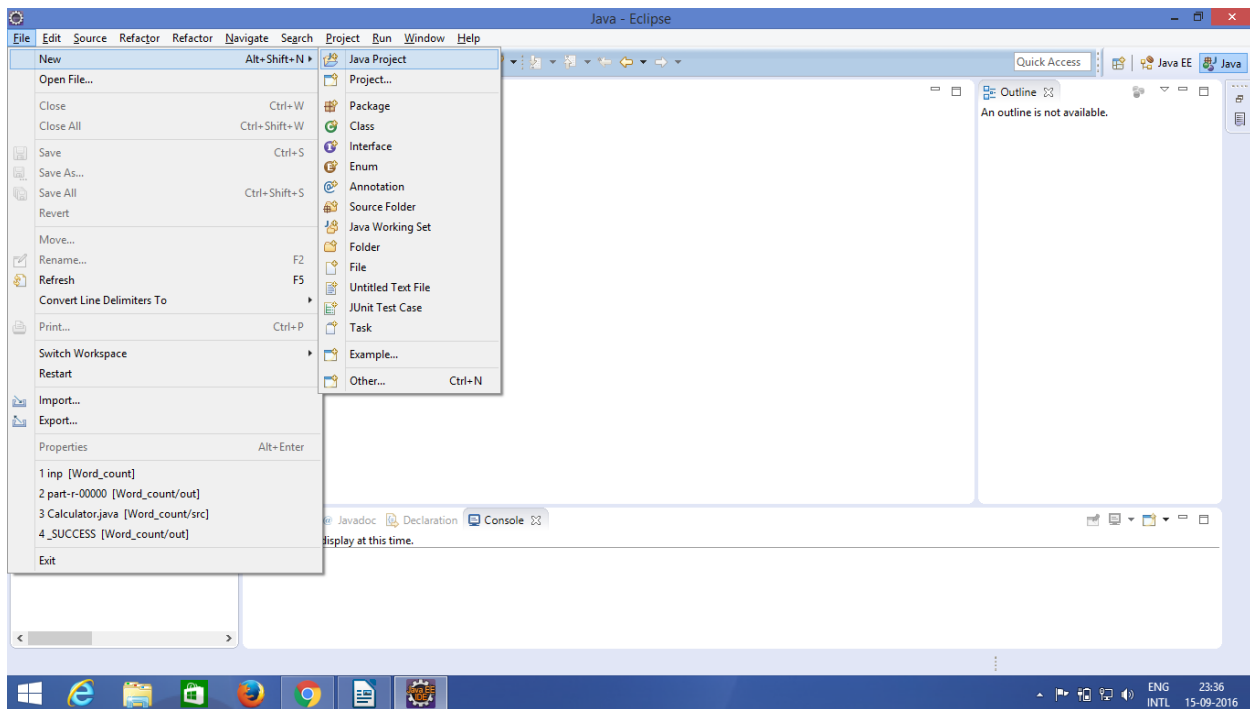
On cmd go to the location “Hadoop-2.6.2/bin” by writing on cmd “cd hadoop-2.6.2.\bin” and then “**hdfs namenode –format**”

k. Start Hadoop. Go to the location: “D:\hadoop-2.6.0\sbin.”

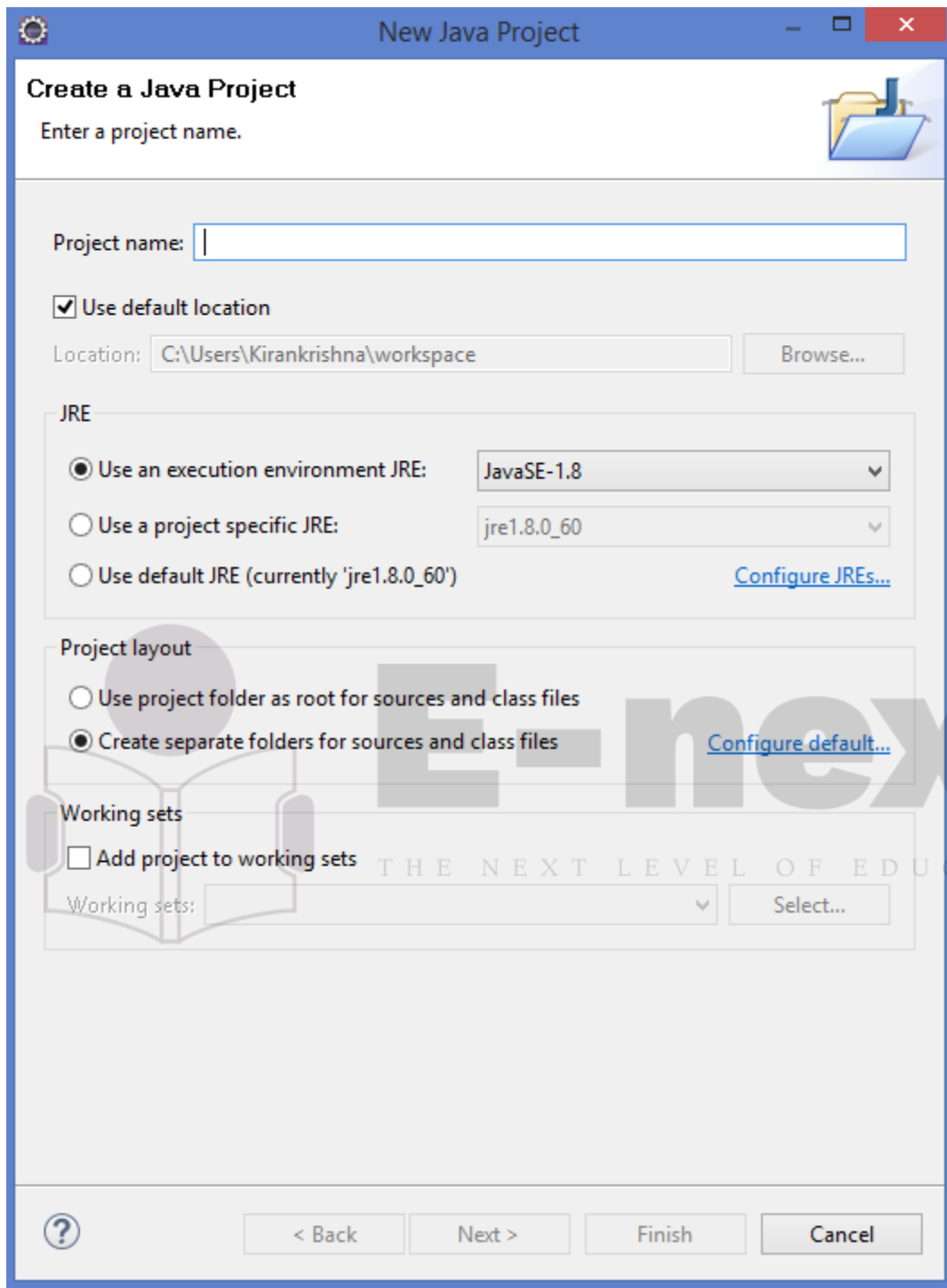
Run the following files as administrator “**start-all.cmd**”

How to create a new MapReduce project in Eclipse

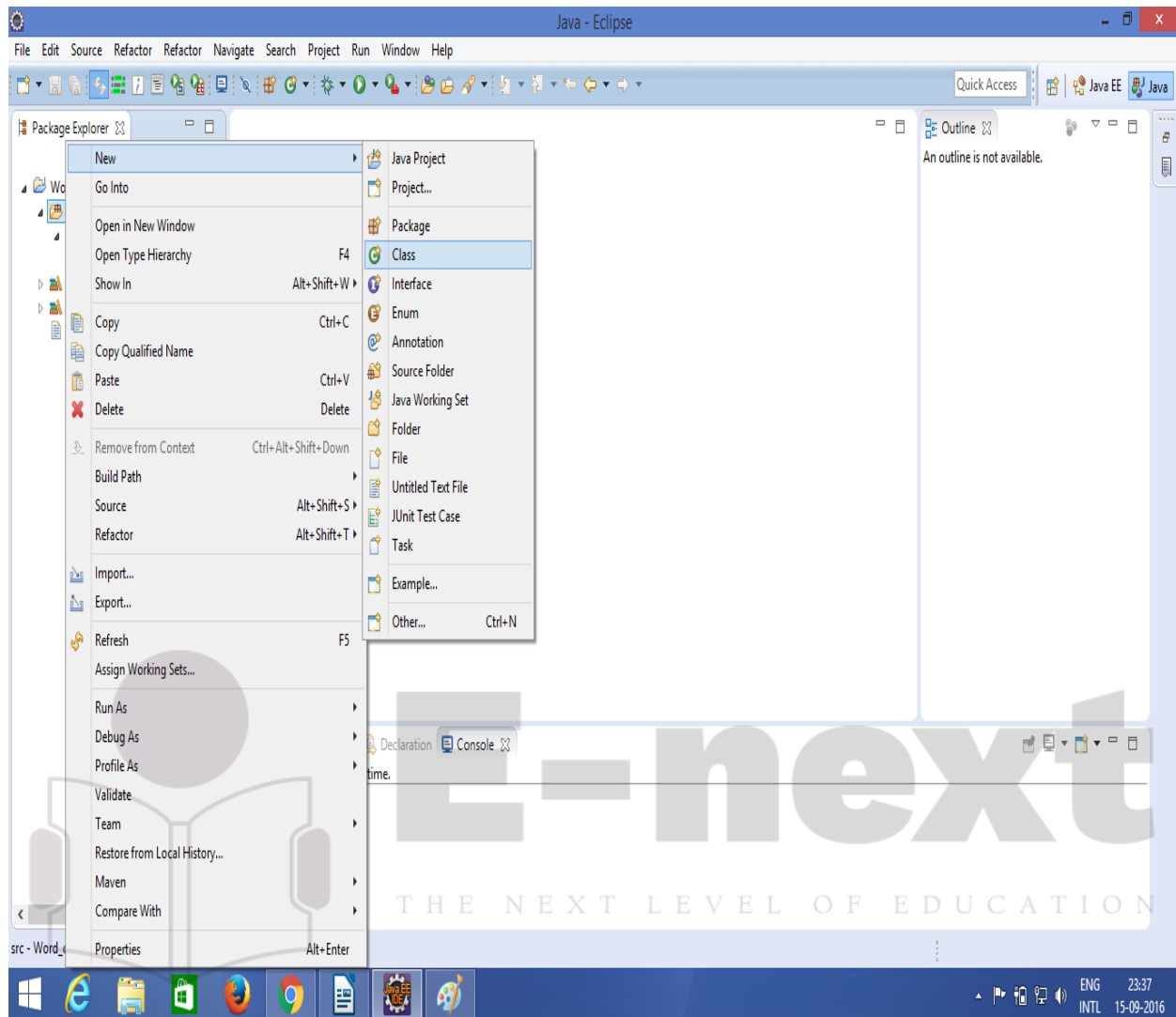
1. Open Ellipse
2. Click File - New Project - Java project



Hereafter clicking on the New Java project, it will ask for the project name as shown in the below screen shot. Give a project name. Here we have given the project name as **Word_count**.

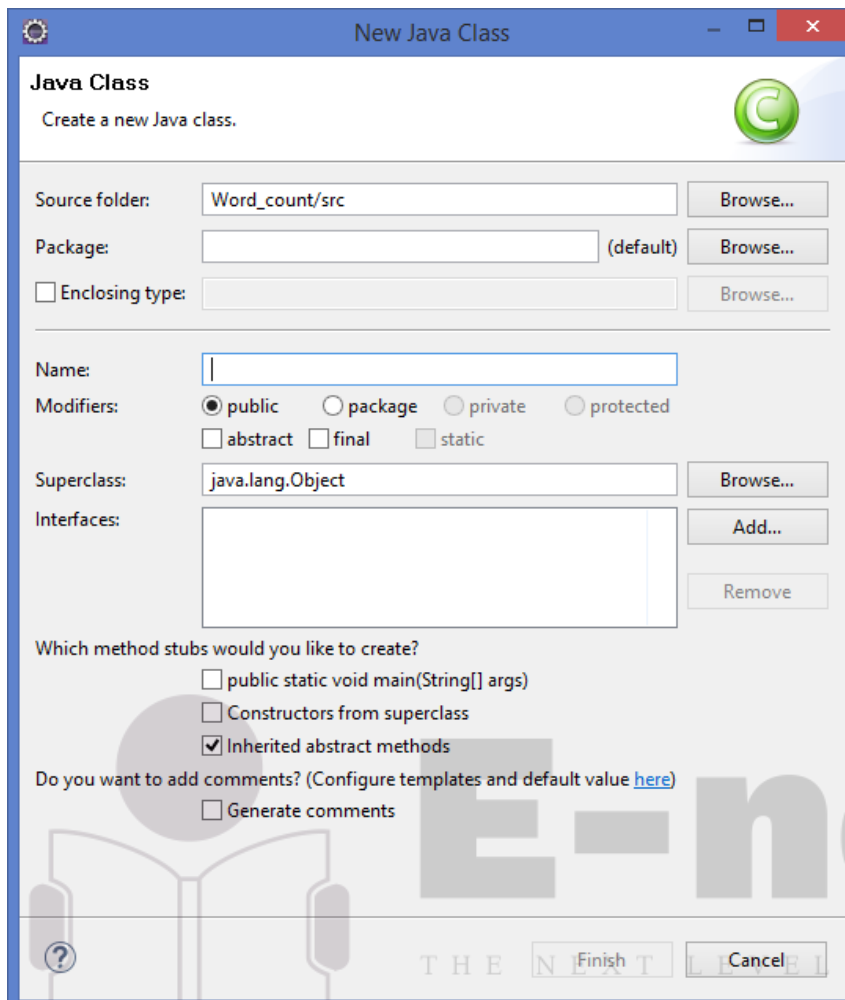


Now after giving the project name, a project will be created with the given name. Click on the project and inside the project, you will find a directory called **src**. Right click and create new class as shown in the below screen shot.



Now you will be prompted with another screen to provide the class name as shown in the below screen shot.

Here, give the class name of your choice. We have given the name as **WordCount**. Inside the **src**, a file with name **WordCount.java** has been created. Click on the file and write the MapReduce code for the word count program



Source Code:

```
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.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCount {
    public static class TokenizerMapper
```

```

    extends Mapper<Object, Text, Text, IntWritable>{
private final static IntWritable one = new IntWritable(1);
private Text word = new Text();
public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
    }
}
}

public static class IntSumReducer
    extends Reducer<Text,IntWritable,Text,IntWritable> {
private IntWritable result = new IntWritable();
public void reduce(Text key, Iterable<IntWritable> values,
                  Context context
                  ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
        sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
}
}

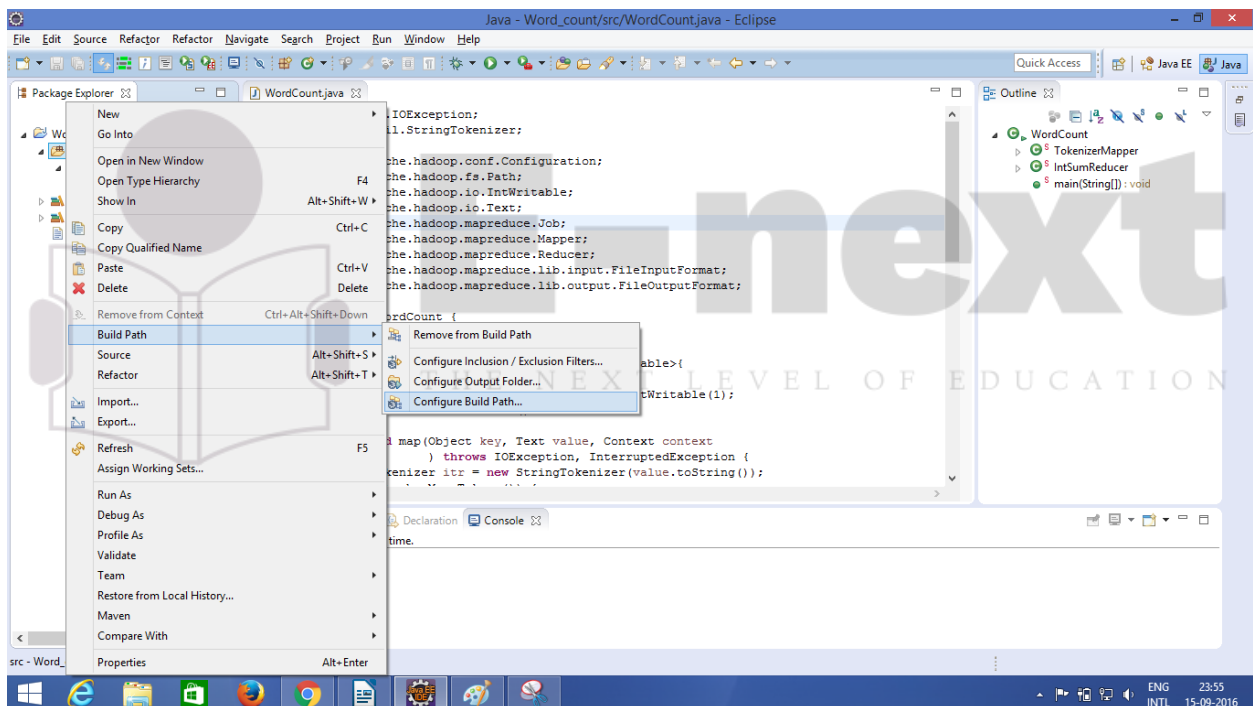
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    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);
}

```

```
}  
}
```

After copying the code save the file. Now you need to add a few dependency files for running this program in Windows.

First, we need to add the jar files that are present in **hadoop-2.6.0/share/hadoop** directory. For that **Right click on src**→**Build path**→**Configure build path** as shown in the below screen shot.



In the **Build Path** select the **Libraries** tab and click on **Add External Jars**. Now browse the path where the **Hadoop-2.6.0** extracted folder is present.

Copy all the Jar files from the locations “D:\hadoop-2.6.0\”

- .\share\hadoop\common\lib
- .\share\hadoop\mapreduce
- .\share\hadoop\mapreduce\lib

- d. share\hadoop\yarn
- e. \share\hadoop\yarn\lib

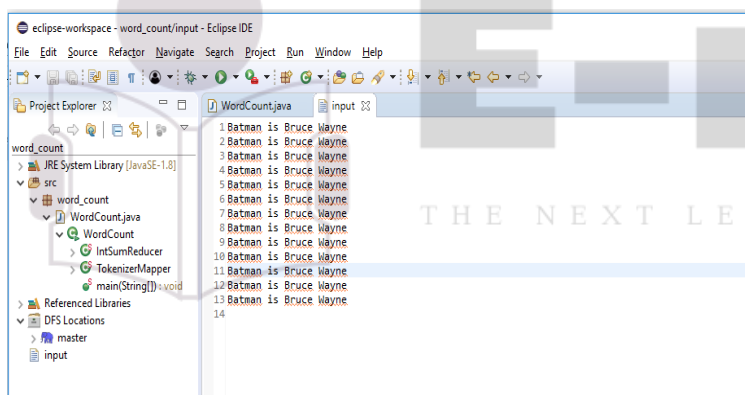
Open the **hadoop-2.6.0/share/hadoop/hdfs/lib** folder and add the **commons-io-2.4.jar** file

Open the **hadoop-2.6.0/share/hadoop/tools/lib** and add the **hadoop-auth-2.6.0.jar** file

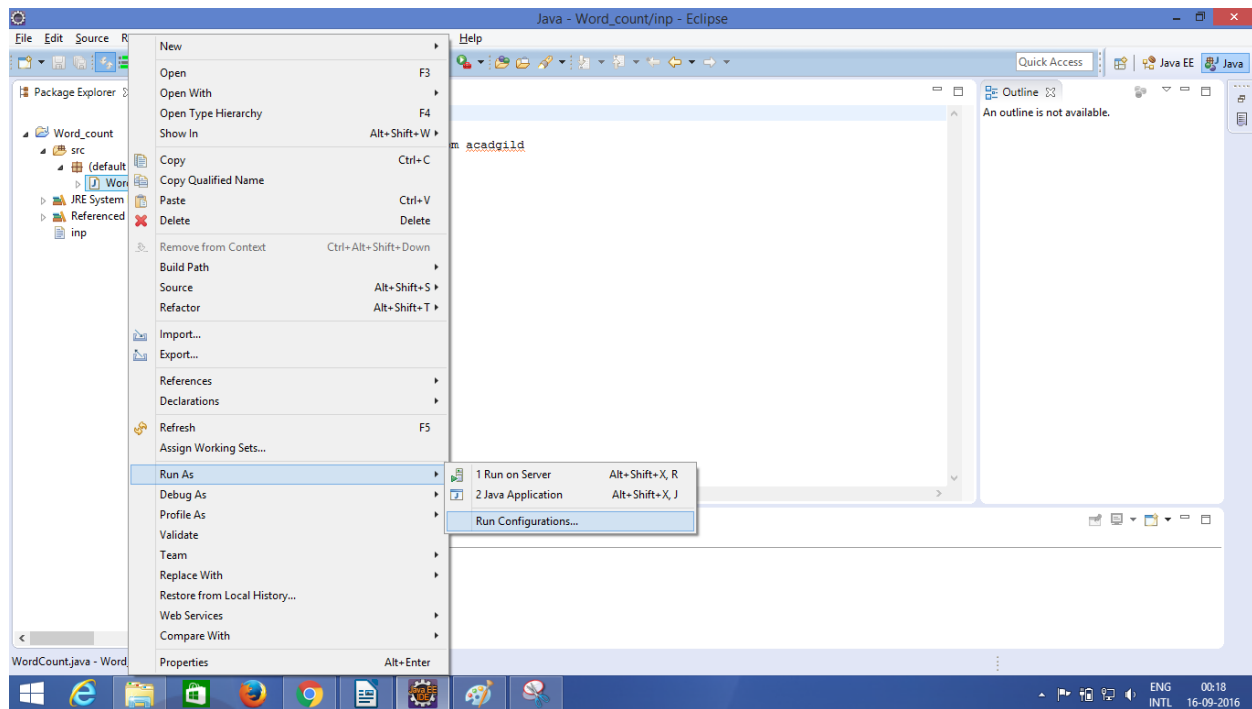
Create bin folder under hadoop-2.6.2/bin
And add **winutils** files(lib and exe)

That's it all the set up required for running your Hadoop application in Windows.
Make sure that your input file is ready.

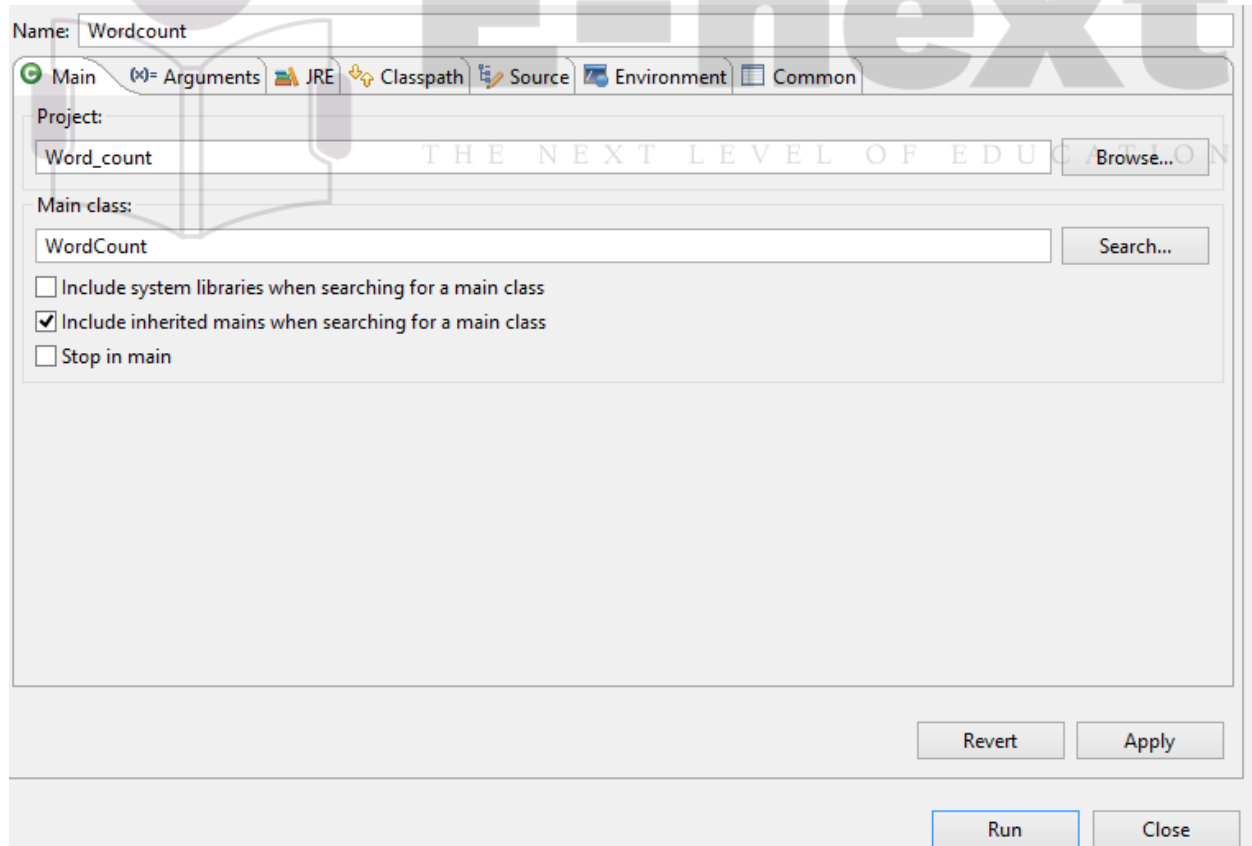
Here we have created our input file in the project directory itself with the name **input** as shown in the below screen shot.



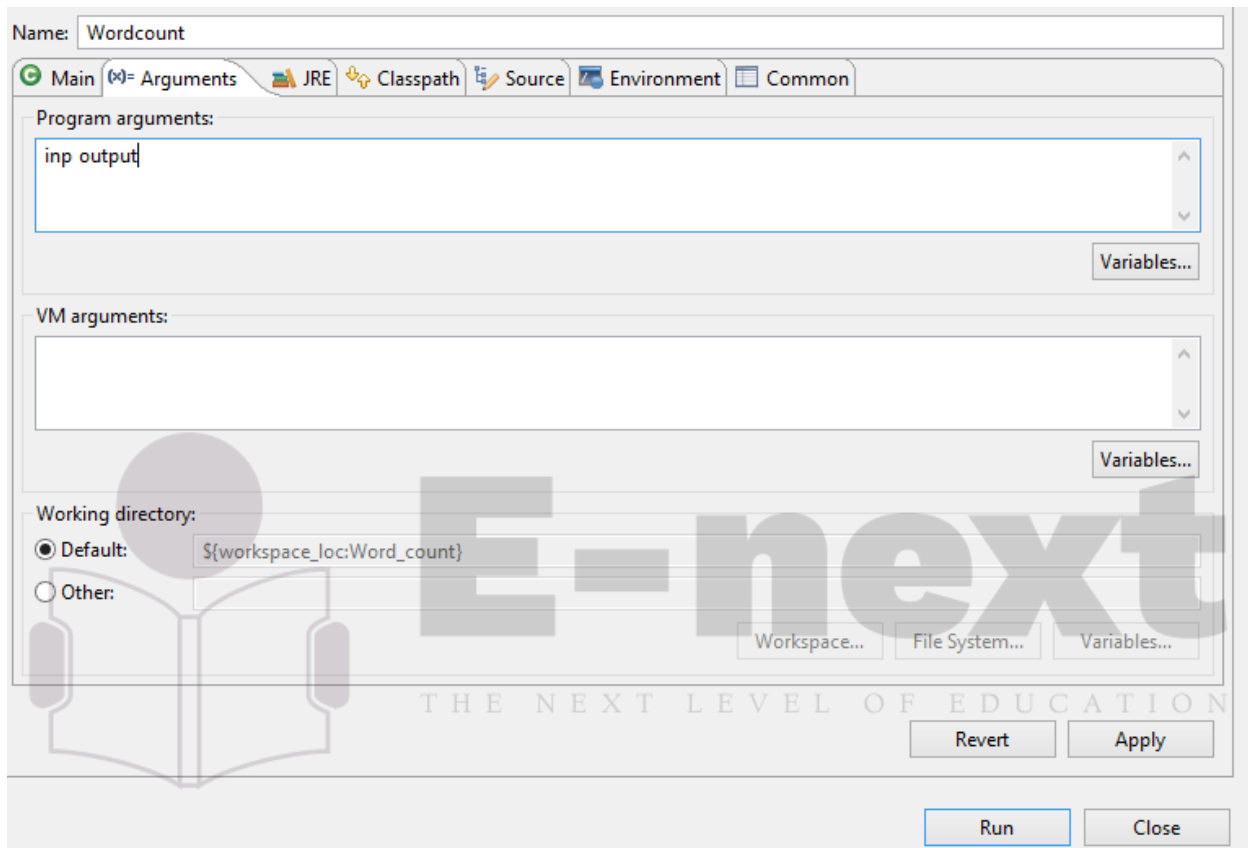
For giving the input and output file paths, **Right click on the main class→Run As→Run configurations** as shown in the below screen shot.



In the **main** select the project name and the class name of the program as shown in the below screen shot.

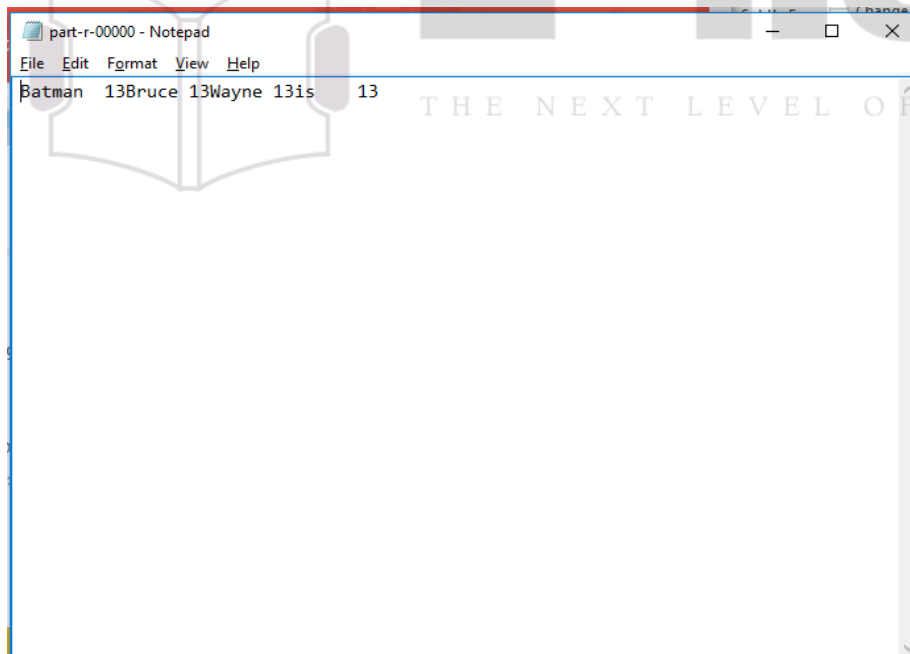
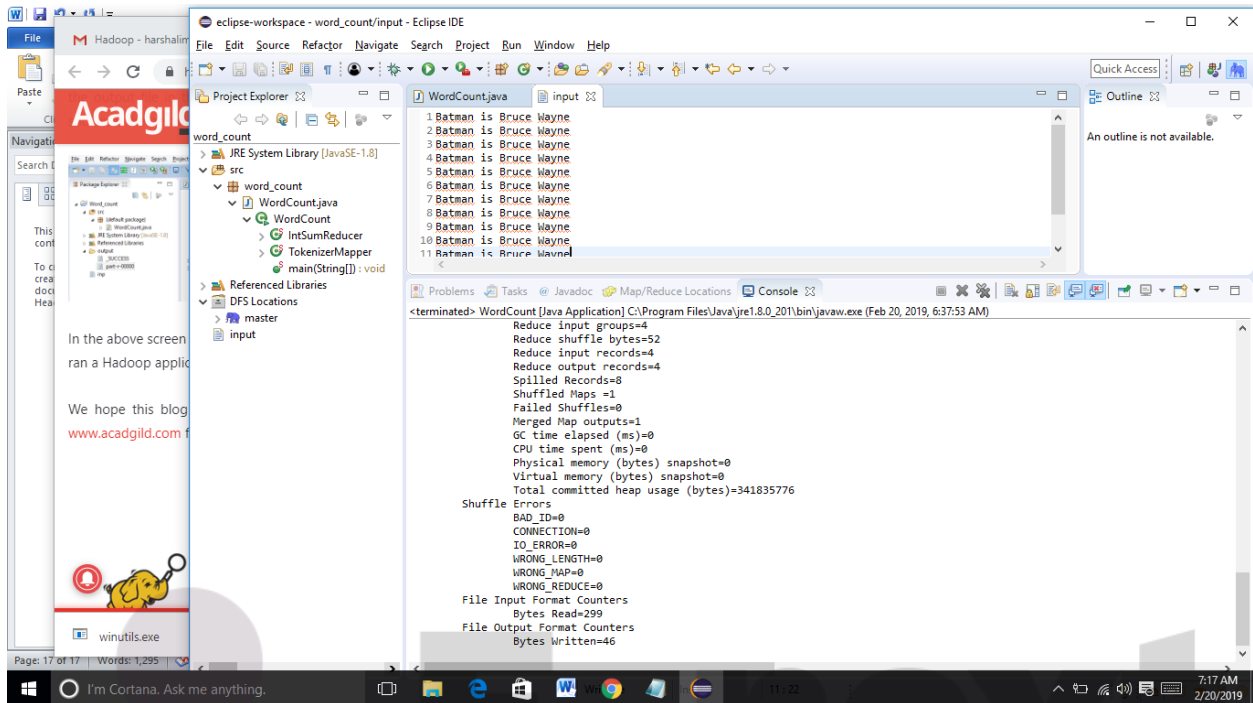


Now move into the **Arguments** tab and provide the **input file path** and the **output file path** as shown in the below screen shot.



Since we have our input file inside the project directory itself, we have just given **inp** as input file path and then a **tab space**. We have given the **output file path** as just **output**. It will create the output directory inside the project directory itself.

Now click on Run. You will see the Eclipse console running



Practical No.6

Aim:- Write a program to Compute Similarity between two text documents.

Code:-

LuceneConstants.java

```
package com.tutorialspoint.lucene;

public class LuceneConstants {
    public static final String CONTENTS = "contents";
    public static final String FILE_NAME = "filename";
    public static final String FILE_PATH = "filepath";
    public static final int MAX_SEARCH = 10;
}
```

TextFileFilter.java:-

```
package com.tutorialspoint.lucene;

import java.io.File;

import java.io.FileFilter;

public class TextFileFilter implements FileFilter {

    @Override

    public boolean accept(File pathname) {

        return pathname.getName().toLowerCase().endsWith(".txt");

    }

}
```

```
}
```

Indexer.java:-

```
package com.tutorialspoint.lucene;

import java.io.File;

import java.io.FileFilter;

import java.io.FileReader;

import java.io.IOException;

import org.apache.lucene.analysis.standard.StandardAnalyzer;

import org.apache.lucene.document.Document;

import org.apache.lucene.document.Field;

import org.apache.lucene.index.CorruptIndexException;

import org.apache.lucene.index.IndexWriter;

import org.apache.lucene.store.Directory;

import org.apache.lucene.store.FSDirectory;

import org.apache.lucene.util.Version;

public class Indexer {

    private IndexWriter writer;

    public Indexer(String indexDirectoryPath) throws IOException {

        Directory indexDirectory =

            FSDirectory.open(new File(indexDirectoryPath));
```

```
writer = new IndexWriter(indexDirectory,
    new StandardAnalyzer(Version.LUCENE_36),true,
    IndexWriter.MaxFieldLength.UNLIMITED);
}

public void close() throws CorruptIndexException, IOException {
    writer.close();
}

private Document getDocument(File file) throws IOException {
    Document document = new Document();
    Field contentField = new Field(LuceneConstants.CONTENTES, new
    FileReader(file));
    Field fileNameField = new Field(LuceneConstants.FILE_NAME,
        file.getName(),Field.Store.YES,Field.Index.NOT_ANALYZED);
    Field filePathField = new Field(LuceneConstants.FILE_PATH,
        file.getCanonicalPath(),Field.Store.YES,Field.Index.NOT_ANALYZED);
    document.add(contentField);
    document.add(fileNameField);
    document.add(filePathField);
    return document;
}
```

```
}
```

```
private void indexFile(File file) throws IOException {  
    System.out.println("Indexing "+file.getCanonicalPath());  
    Document document = getDocument(file);  
    writer.addDocument(document);  
}
```

```
public int createIndex(String dataDirPath, FileFilter filter)
```

```
throws IOException {
```

```
File[] files = new File(dataDirPath).listFiles();
```

```
for (File file : files) {
```

```
    if(!file.isDirectory()
```

```
        && !file.isHidden()
```

```
        && file.exists()
```

```
        && file.canRead()
```

```
        && filter.accept(file)
```

```
    ){
```

```
        indexFile(file);
```

```
}
```



```

    }

    return writer.numDocs();

}

}

```

Searcher.java:-

package com.tutorialspoint.lucene;

import java.io.File;

import java.io.IOException;

import org.apache.lucene.analysis.standard.StandardAnalyzer;

import org.apache.lucene.document.Document;

import org.apache.lucene.index.CorruptIndexException;

import org.apache.lucene.queryParser.ParseException;

import org.apache.lucene.queryParser.QueryParser;

import org.apache.lucene.search.IndexSearcher;

import org.apache.lucene.search.Query;

import org.apache.lucene.search.ScoreDoc;

import org.apache.lucene.search.TopDocs;

import org.apache.lucene.store.Directory;

import org.apache.lucene.store.FSDirectory;

import org.apache.lucene.util.Version;

public class Searcher {

IndexSearcher **indexSearcher**;

QueryParser **queryParser**;

Query **query**;

public Searcher(String **indexDirectoryPath**)

throws IOException {

Directory **indexDirectory** =

FSDirectory.open(**new** File(**indexDirectoryPath**));

indexSearcher = **new** IndexSearcher(indexDirectory);

```

    queryParser = new QueryParser(Version.LUCENE_36,
        LuceneConstants.CONTENTS,
        new StandardAnalyzer(Version.LUCENE_36));
}

public TopDocs search( String searchQuery)
    throws IOException, ParseException {
    query = queryParser.parse(searchQuery);
    return indexSearcher.search(query, LuceneConstants.MAX_SEARCH);
}

public Document getDocument(ScoreDoc scoreDoc)
    throws CorruptIndexException, IOException {
    return indexSearcher.doc(scoreDoc.doc);
}

public void close() throws IOException {
    indexSearcher.close();
}
}

```

LuceneTester.java:-

```
package com.tutorialspoint.lucene;
```

```
import java.io.IOException;
```

```
import org.apache.lucene.document.Document;
```

```
import org.apache.lucene.queryParser.ParseException;
```

```
import org.apache.lucene.search.ScoreDoc;
```

```
import org.apache.lucene.search.TopDocs;
```

```
public class LuceneTester {
```

```
    String indexDir = "D:\\Lucene\\Index";
```

```
    String dataDir = "D:\\Lucene\\Data";
```

```
    Indexer indexer;
```

```
    Searcher searcher;
```

```

public static void main(String[] args) {
    LuceneTester tester;
    try {
        tester = new LuceneTester();
        tester.createIndex();
        tester.search("Mohan");
    } catch (IOException e) {
        e.printStackTrace();
    } catch (ParseException e) {
        e.printStackTrace();
    }
}

```

```

private void createIndex() throws IOException {
    indexer = new Indexer(indexDir);
    int numIndexed;
    long startTime = System.currentTimeMillis();
    numIndexed = indexer.createIndex(dataDir, new TextFileFilter());
    long endTime = System.currentTimeMillis();
    indexer.close();
    System.out.println(numIndexed+" File indexed, time taken: "
        +(endTime-startTime)+" ms");
}

```

```

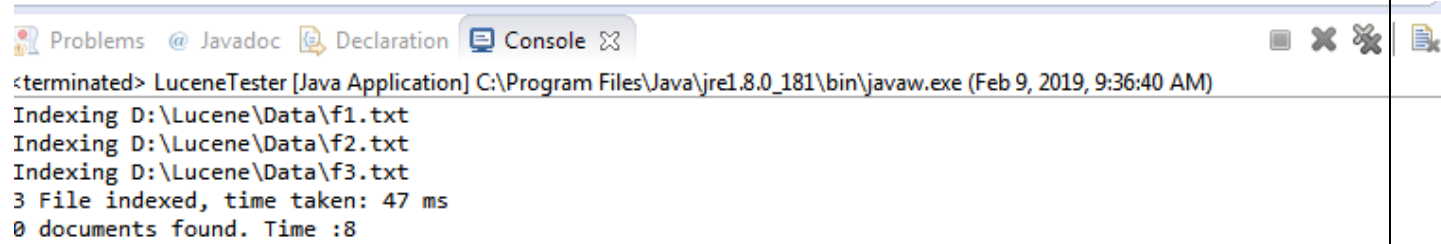
private void search(String searchQuery) throws IOException, ParseException {
    searcher = new Searcher(indexDir);
    long startTime = System.currentTimeMillis();
    TopDocs hits = searcher.search(searchQuery);
    long endTime = System.currentTimeMillis();

    System.out.println(hits.totalHits +
        " documents found. Time : " + (endTime - startTime));
    for(ScoreDoc scoreDoc : hits.scoreDocs) {
        Document doc = searcher.getDocument(scoreDoc);
        System.out.println("File: "
            + doc.get(LuceneConstants.FILE_PATH));
    }
    searcher.close();
}

```

```
}
```

Output:-



```
<terminated> LuceneTester [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Feb 9, 2019, 9:36:40 AM)
Indexing D:\Lucene\Data\f1.txt
Indexing D:\Lucene\Data\f2.txt
Indexing D:\Lucene\Data\f3.txt
3 File indexed, time taken: 47 ms
0 documents found. Time :8
```



E-next
THE NEXT LEVEL OF EDUCATION

Practical No.7

Aim:- Write a program for Pre-processing of a Text Document: stop word removal.

Stopwords1.py:-

```
>>> import nltk  
>>> from nltk.corpus import stopwords  
>>> set(stopwords.words('english'))
```

Output:-

```
{'their', 'has', 'am', 'against', 'our', 'than', 'been', 'an', 'shouldn't', 'd',  
 'each', 'didn', 'shan', 'won', 'out', 'hasn't', 'ourselves', 'down', 'but', 'he  
re', 'there', 'them', 'shan't', 'in', 'herself', 'these', 'your', 'ma', 'as', 'h  
aving', 'didn't', 'isn't', 'some', 'during', 're', 'and', 'she's', 'to', 't', 'a  
re', 'which', 'by', 'o', 'those', 'couldn', 'haven', 'until', 'can', 'its', 'onc  
e', 'why', 'after', 'the', 'with', 'wasn', 'won't', 'doing', 'few', 'should', 'm  
ustn't', 'through', 'i', 'nor', 'hasn', 'is', 'when', 'me', 'myself', 'weren't',  
 'this', 'you'd', 'yourself', 'both', 'hadn', 'about', 'of', 'will', 'do', 'then  
, 'they', 'before', 'y', 'only', 'had', 'all', 'it's', 'off', 'above', 'up', 't  
heirs', 'between', 'll', 'over', 'wouldn', 'being', 'same', 'aren', 'did', 'a',  
 'any', 's', 'how', 'weren', 'just', 've', 'because', 'again', 'don', 'him', 'him  
self', 'we', 'more', 'she', 'aren't', 'wasn't', 'for', 'you've', 'what', 'itself  
, 'hers', 'have', 'you'll', 'who', 'too', 'under', 'mightn', 'doesn', 'it', 'in  
to', 'so', 'should've', 'mightn't', 'does', 'were', 'mustn', 'shouldn', 'other',  
 'no', 'he', 'yourselves', 'doesn't', 'you', 'themselves', 'below', 'or', 'on',  
 'needn', 'wouldn't', 'that', 'hadn't', 'that'll', 'from', 'don't', 'ain', 'such',  
 , 'further', 'where', 'needn't', 'whom', 'very', 'my', 'haven't', 'own', 'you're  
, 'while', 'yours', 'ours', 'was', 'm', 'her', 'isn', 'if', 'not', 'his', 'coul  
dn't', 'now', 'at', 'be', 'most'}
```

Stopwords1.py:-

```
from nltk.corpus import stopwords  
  
from nltk.tokenize import word_tokenize
```

```
example_sent = "This is a sample sentence, showing off the stop words filtration."
```

```
stop_words = set(stopwords.words('english'))
```

```
word_tokens = word_tokenize(example_sent)
```

```
filtered_sentence = [w for w in word_tokens if not w in stop_words]
```

```
filtered_sentence = []
```

```
for w in word_tokens:
```

```
    if w not in stop_words:
```

```
        filtered_sentence.append(w)
```

```
print(word_tokens)
```

```
print(filtered_sentence)
```

Output:-

```
===== RESTART: D:/IR/IR/stopwords2.py =====  
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off', 'the', 'stop',  
'words', 'filtration', '.']  
['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words', 'filtration', '.']  
>>>
```

Practical No.8

Aim:- Write a program for tkinter.

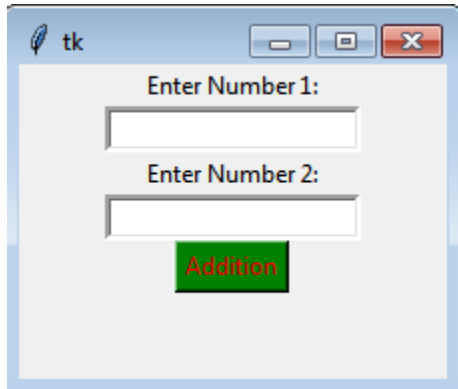
Code:

```
from tkinter import *
root=Tk()
l1=Label(root,text="Enter Number 1:")
l1.pack()
t1=Entry(root,bd="3")
t1.pack()
l2=Label(root,text="Enter Number 2:")
l2.pack()
t2=Entry(root,bd="3")
t2.pack()

def addNumber():
    a=int(t1.get())
    b=int(t2.get())
    c=a+b
    print("Addition of two NOS:",C)

b1=Button(root,text="Addition",fg="red",bg="green",command=addNumber)
b1.pack()
root.mainloop()
```

Output:-



E-next
THE NEXT LEVEL OF EDUCATION

Practical No.9

Aim: Write a program to implement simple web crawler.

Code:

```
import java.net.*;
import java.io.*;

public class Crawler{
    public static void main(String[] args) throws Exception{
        String urls[] = new String[1000];


        String url = "https://www.cricbuzz.com/live-cricket-scores/20307/aus-vs-ind-3rd-odi-india-tour-of-australia-2018-19";
        int i=0,j=0,tmp=0,total=0, MAX = 1000;
        int start=0, end=0;
        String webpage = Web.getWeb(url);
        end = webpage.indexOf("<body");
        for(i=total;i<MAX; i++, total++){
            start = webpage.indexOf("http://", end);
            if(start == -1){
                start = 0;
                end = 0;
                try{
                    webpage = Web.getWeb(urls[j++]);
                }catch(Exception e){
                    System.out.println("*****");
                    System.out.println(urls[j-1]);
                }
            }
        }
    }
}
```

```
        System.out.println("Exception caught \n"+e);
    }

    /*logic to fetch urls out of body of webpage only */
    end = webpage.indexOf("<body");
    if(end == -1){
        end = start = 0;
        continue;
    }
}
end = webpage.indexOf("\"", start);
tmp = webpage.indexOf("'", start);
if(tmp < end && tmp != -1){
    end = tmp;
}
url = webpage.substring(start, end);
urls[i] = url;
System.out.println(urls[i]);
}
System.out.println("Total URLS Fetched are " + total);
}
}
```

```
/*This class contains a static function which will fetch the webpage  
of the given url and return as a string */
```

```
class Web{  
    public static String getWeb(String address)throws Exception{  
        String webpage = "";  
        String inputLine = "";  
        URL url = new URL(address);  
        BufferedReader in = new BufferedReader(  
            new InputStreamReader(url.openStream()));  
        while ((inputLine = in.readLine()) != null)  
            webpage += inputLine;  
        in.close();  
        return webpage;  
    }  
}
```



Output:

conn built
<http://www.mit.edu>
<http://mit.edu/site/?ref=mithomepage>
<http://web.mit.edu/aboutmit>
<http://web.mit.edu/institute-events/visitor/>
<http://whereis.mit.edu>
<http://web.mit.edu/officesdir/>
<http://mitstory.mit.edu/>
<http://web.mit.edu/admissions/>
<http://web.mit.edu/admissions/graduate/>
<http://web.mit.edu/sfs/>
<http://web.mit.edu/education>
<http://web.mit.edu/education/>
<http://ocw.mit.edu/>
<http://odl.mit.edu/mitx/>
<http://web.mit.edu/research/>
<http://www.ll.mit.edu/>
<http://libraries.mit.edu/>
<http://web.mit.edu/community/>
<http://resources.mit.edu/>
<http://web.mit.edu/faculty/>
<http://web.mit.edu/staff/>
<http://alum.mit.edu/>
<http://web.mit.edu/life>
<http://arts.mit.edu/>
<http://web.mit.edu/athletics/www/>
<http://connect.mit.edu/>
<http://web.mit.edu/initiatives/>
<http://mitei.mit.edu>
<http://ki.mit.edu>
<http://diversity.mit.edu/>
<http://global.mit.edu/>
<http://web.mit.edu/impact/>
<http://web.mit.edu/industry/>
<http://web.mit.edu/mitpsc/>
<http://web.mit.edu/commencement/2014/>

E-next

THE NEXT LEVEL OF EDUCATION

Practical No.10

Aim:- Write a program to parse XML text, generate Web graph and compute topic specific page rank.

emp.xml:-

```
<?xml version="1.0" encoding="UTF-8"?>
<employee>
  <fname>Divesh</fname>
  <lname>Saurabh</lname>
  <home>Thane</home>
  <expertise name="SQL"/>
  <expertise name="Python"/>
  <expertise name="Testing"/>
  <expertise name="Business"/>
</employee>
```

emp.py:-

```
import xml.dom.minidom

def main():
    doc=xml.dom.minidom.parse("emp.xml");
    print(doc.nodeName)
    print(doc.firstChild.tagName)

if __name__=="__main__":
```

```
main()
```

```
===== RESTART: C:/Users/Student/Desktop/emp.py =====  
#document  
employee  
>>>
```

emp1.py:-

```
import xml.dom.minidom
```

```
def main():
```

```
    doc = xml.dom.minidom.parse("emp.xml");
```

```
    print (doc.nodeName)
```

```
    print (doc.firstChild.tagName)
```

```
    expertise = doc.getElementsByTagName("expertise")
```

```
    print ("%d expertise:" % expertise.length)
```

```
    for skill in expertise:
```

```
        print (skill.getAttribute("name"))
```

```
    newexpertise = doc.createElement("expertise")
```

```
    newexpertise.setAttribute("name", "BigData")
```

```
    doc.firstChild.appendChild(newexpertise)
```

```
    print (" ")
```

```
    expertise = doc.getElementsByTagName("expertise")
```

```
    print ("%d expertise:" % expertise.length)
```

```
    for skill in expertise:
```

```
        print (skill.getAttribute("name"))
```

```
if __name__ == "__main__":
```

```
    main();
```

Output:

```
===== RESTART: C:/Users/Student/Desktop/empl.py =====  
#document  
employee  
4 expertise:  
SQL  
Python  
Testing  
Business  
  
5 expertise:  
SQL  
Python  
Testing  
Business  
BigData
```



E-next

THE NEXT LEVEL OF EDUCATION