Zadatak1

```
hadoop namenode -format
./sbin/start-dfs.sh
hadoop fs -mkdir -p /user/rovkp
hadoop fs -ls /user/rovkp
wget http://svn.tel.fer.hr/gutenberg.zip
hadoop fs -put gutenberg.zip /user/rovkp
./bin/hdfs fsck /user/rovkp/gutenberg.zip -files -blocks -locations

Status: HEALTHY
 Total size:    158790992 B
 Total dirs:    0
 Total files:   1
 Total symlinks:        0
 Total blocks (validated):  2 (avg. block size 79395496 B)
 Minimally replicated blocks:   2 (100.0 %)
 Over-replicated blocks:    0 (0.0 %)
 Under-replicated blocks:   0 (0.0 %)
 Mis-replicated blocks:     0 (0.0 %)
 Default replication factor:    1
 Average block replication: 1.0
 Corrupt blocks:        0
 Missing replicas:      0 (0.0 %)
 Number of data-nodes:      1
 Number of racks:       1
FSCK ended at Thu Mar 16 11:06:53 CET 2017 in 0 milliseconds
The filesystem under path '/user/rovkp/gutenberg.zip' is HEALTHY

2 bloka, replikacijski faktor 1. Relativno je mala datoteka, moze se cijela
cachat u memoriju. HDFS je za vece datoteke

hadoop fs -get /user/rovkp/gutenberg.zip  /tmp
md5sum gutenberg.zip
e3fc0eb2c51e0290c9b85fd2a7cee071  gutenberg.zip
md5sum /tmp/gutenberg.zip
e3fc0eb2c51e0290c9b85fd2a7cee071  /tmp/gutenberg.zip
```

Zadatak 2

```
import java.io.BufferedWriter;
import java.io.IOException;
import java.nio.charset.Charset;
import java.nio.file.*;
import java.nio.file.attribute.BasicFileAttributes;
import java.util.List;

/**
```

```
 - Koja je veličina konačne datoteke gutenberg_books.txt?
 391 MB
 - Koliko je ukupno redaka pročitano?
 8481553

 - U slučaju kad bi tu datoteku pohranili na HDFS s veličinom blokova 128 MB i
faktorom replikacije 3, koliko bi
 se ukupno blokova stvorilo na HDFS-u?
 3(faktor replikacije) x 4 (ceil(391/128)) = 12 blokova
 - Koliko vremena se izvodio program? Oko 5 sekundi
 Kakvo bi bilo očekivano vrijeme izvođenja kada bi se taj program izvršavao
 na Hadoopovom grozdu, uz pohranu na HDFS? Ovisi o clusteru...za ovako male
podatke imamo overhead komunikacije i
 map-reduce job..Ajmo reci uz dovoljni broj masina 1s.
 */
public class Zadatak2 {
    public static void main(String [] args) throws Exception{
        Charset charset = Charset.forName("ISO-8859-1");
        Path start = Paths.get("/tmp/gg");
        BufferedWriter bw =
Files.newBufferedWriter(Paths.get("/tmp/gg/gutenberg_books.txt"), charset);
        final int[] num_lines = {0};
        Files.walkFileTree(start, new SimpleFileVisitor<Path>() {
            @Override
            public FileVisitResult visitFile(Path file, BasicFileAttributes
attrs)
                    throws IOException
            {
                try {
                    List<String> lines = Files.readAllLines(file, charset);
                    for (String line: lines)
                        bw.write(line);
                        bw.newLine();
                    num_lines[0] += lines.size();
                }
                catch (Exception ex) {
                    System.out.println(file);
                    System.err.println(ex);
                }
                return FileVisitResult.CONTINUE;
            }
            @Override
            public FileVisitResult postVisitDirectory(Path dir, IOException e)
                    throws IOException
            {
                    return FileVisitResult.CONTINUE;
            }
        });

        bw.flush();
        bw.close();
        System.out.println("Total lines " + num_lines[0]);
```

```
    }

}
```

Zadatak 3

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.LocalFileSystem;
import org.apache.hadoop.fs.Path;


import java.util.Arrays;

public class Zadatak3 {
    public static void main(String [] args) throws Exception{
        Configuration conf = new Configuration ();
        try (FileSystem hdfs = FileSystem.get(conf);
             LocalFileSystem lfs = LocalFileSystem.getLocal(conf)){
            for (Path p: Arrays.asList(
                    new Path("/user/lpp/gutenberg.zip"),
                    new Path("/usr/lib/hadoop/gutenberg.zip")
            )) {
                System.out.println("hdfs " + p + " " + hdfs.exists(p));
                System.out.println("local fs " + p + " " + lfs.exists(p));
            }
        }
    }
}
```