**Exercise 1: ……./5**

Index is Out Of Bounds ….. 1

Arithmetic Exception: Denominator = zero ….. 1

Exception: Parameter with opposite sign ….. 1

Enter a number: ….. 1

Result: 10 ….. 1

**Exercise 2:**

public interface PersistentData { ….. 1 **……./2**

public void Open() throws Exception; ….. 1

}

public abstract class File implements PersistentData { …..1+1 **……./8**

private String name;

private int size;

private String date;

File(String name, int size, String date) { ….. 2

this.name = name;

this.size = size;

this.date = date;

}

File(File f) { ….. 2

this.name = f.name;

this.size = f.size;

this.date = f.date;

}

public int getSize() {

return size; ….. 1

}

public String getDate() {

return date; ….. 1

}

}

public class TextFile extends File { ….. 1 **……./10**

public String content;

public int nbOfWords;

TextFile(String name, int size, String date, String content, int nbOfWords) { **……./2**

super(name, size, date); ….. 1

this.content = content; …..0.5

this.nbOfWords = nbOfWords; …..0.5

}

TextFile(TextFile t) { **……./2**

super(t); ….. 1

this.content = t.content; …..0.5

this.nbOfWords = t.nbOfWords; …..0.5

}

public void Open() throws Exception { ….. 1+1 **……./5**

if(getSize()/1024 > 2048) ….. 1

throw new Exception("Too Large File"); ….. 1

System.out.println(content); ….. 1

}

}

**Exercise 3:**

public class Folder { **……./45**

public String name;

File[] arrFiles; ….. 1

int nbFls; ….. 1

Folder[] subFolders; ….. 1

int nbSF; ….. 1

Folder(String name, int size) { **……./4**

this.name = name;

arrFiles = new File[size]; ….. 1

nbFls = 0; ….. 1

subFolders = new Folder[100]; ….. 1

nbSF = 0; ….. 1

}

Folder(Folder f) { **……./15**

this.name = f.name;

arrFiles = new File[f.arrFiles.length]; ….. 1

nbFls = f.nbFls; ….. 1

for(int i=0;i<nbFls;i++) { ….. 1

if (f.arrFiles[i] instanceof TextFile) ….. 1

arrFiles[i] = new TextFile((TextFile)f.arrFiles[i]); …..3

else

arrFiles[i] = new ExecFile((ExecFile)f.arrFiles[i]); ….. 3

}

subFolders = new Folder[f.subFolders.length]; ….. 1

nbSF = f.nbSF; ….. 1

for(int j=0; j<nbSF; j++) { ….. 1

subFolders[j] = new Folder(f.subFolders[j]); ….. 2

}

}

public boolean addSubFolder(Folder f) { **……./4**

if(nbSF < subFolders.length) { ….. 1

subFolders[nbSF++] = new Folder(f); ….. 2

return true; ….. 0.5

}

return false; ….. 0.5

}

public int openAllTextFiles(String d, int n) { **……./12**

int openedFiles=0; ….. 1

for(int i=0;i<nbFls; i++) { ….. 1

try { ….. 1

if(arrFiles[i] instanceof TextFile) { ….. 1

if( (arrFiles[i].getDate().equals(d)) && ….. 1

(((TextFile)arrFiles[i]).getSize())>=n) { ….. 2

arrFiles[i].Open(); ….. 1

openedFiles++; ….. 1

}

}

}

catch(Exception e) { ….. 1

System.out.println(e.getMessage()); ….. 1

}

}

return openedFiles; ….. 1

}

public int getFolderSize() { **……./6**

int totalFolderSize = 0; ….. 1

for(int i=0; i<nbFls;i++) { ….. 1

totalFolderSize += arrFiles[i].getSize(); ….. 1

}

for(int j=0;j<nbSF;j++) { ….. 1

totalFolderSize += subFolders[j].getFolderSize(); ….. 1

}

return totalFolderSize; ….. 1

}

}