COE 312 – Data Structures

Welcome to Exam I
Thursday October 26, 2017

Instructor: Dr. Wissam F. Fawaz

Name: _____

Student ID: _____

Instructions:

1. This exam is **Closed Book**. Please do not forget to write your name and ID on the first page.
2. You have exactly **75 minutes** to complete the 4 required problems.
3. Read each problem carefully. If something appears ambiguous, please write your assumptions.
4. Points allocated to each problem are shown in square brackets.
5. Put your answers in the space provided only. No other spaces will be graded or even looked at.

Good Luck!!

Problem 1: Exception (**15 minutes**) [20 points]

1) Which of the following expressions produces an `ArithmeticException`?
   a. `10/0.0`
   b. `0.0/0`
   c. Both of the above
   d. **None of the above**

2) Which of the following statements produces a `NumberFormatException`? Assume that `scan` is a `Scanner` object that was instantiated properly.
   a. `int value = scan.nextLine();`
   b. **`int value = Integer.parseInt("12.5f");`**
   c. Both of the above
   d. None of the above

3) Which of the following statements produces an `InputMismatchException` if the user enters a `double` value via the keyboard? Assume that `scan` is a `Scanner` object that is attached to the keyboard and that was instantiated properly.
   a. **`int value = scan.nextInt();`**
   b. `String value = scan.nextLine();`
   c. Both of the above
   d. None of the above

4) Which of the following represents a checked exception?
   a. `InputMismatchException`
   b. **`IOException`**
   c. Both of the above
   d. None of the above

5) Consider the following statement:
   `assert (!(value>0 && value<10)):"Invalid value";`
   Which of the following values of the `value` variable causes the above statement to produce an error message?
   a. `10`
   b. `0`
   c. Both of the above
   d. **None of the above**

6) What output does the following code fragment?

```
String str = "Programming is fun";
try {
System.out.print(str.substring(str.length()-3,str.length()));
} catch(StringIndexOutOfBoundsException e) {
      System.out.print("Cannot extract string!");}
System.out.print(", try catch end");
```

   a. `fun`
   b. `Cannot extract string!, try catch end`
   c. **`fun, try catch end`**
   d. None of the above

7) Consider the following method definition:

```
void foo() {
      throw new Exception("Error in foo!");
}
```

Which of the following is true about this definition?

a. **This method definition does not compile correctly**
b. The following method call: `foo();` produces an exception
c. Both of the above are true
d. None of the above is true

8) Which of the following methods is not defined in the `Exception` class?

a. `getMessage()`
b. `printStackTrace()`
c. **`getStackTraceElements()`**
d. None of the above

9) Which of the following causes the compiler to issue an error message?

a. `Exception e = new IOException("Problem occurred");`
b. `RunTimeException e = new ArithmeticException("Problem");`
c. Both of the above
d. **None of the above**

10) Which of the following parent classes can be used to derive and create a new unchecked exception class?

a. `Exception`
b. `Throwable`
c. Both of the above
d. **None of the above**

Problem 2: Polymorphism and IO (**15 minutes**) [20 points]

1) Which of the following statements is true about the relationship between superclass and subclass types?
   a. A subclass reference cannot reference an instance of the superclass
   b. An instance of the subclass can be assigned to a superclass variable
   c. **Both of the above are true**
   d. None of the above is true

2) Which of the following interfaces must be implemented by a class that is designed to respond to the events generated by a `Timer` object?
   a. `ActionEventHandler`
   b. `ActionEventListener`
   c. `ActionHandler`
   d. **None of the above**

3) Assume that `Hourly` and `Executive` are both derived from the `Employee` superclass, which in turn is derived from the `StaffMember` abstract superclass. Which of the following statements causes a compile-time error to occur? Assume that all of the previously mentioned classes are equipped with default constructors.
   a. `StaffMember s = new Employee();`
   b. **`Hourly h = new Executive();`**
   c. `StaffMember sm = new Hourly();`
   d. Both (b) and (c)

4) Consider the class hierarchy from the previous question. Assume now that `StaffMember` defines an abstract method called `pay()` with the following signature: `abstract public void pay();`
   Suppose that all the descendants of `StaffMember` redefine the inherited `pay` method. Which of the following statements calls the pay method as defined in the `Hourly` class?
   a. `Employee e = new Hourly(); e.pay();`
   b. `StaffMember sm = new Hourly(); e.pay();`
   c. **Both of the above**
   d. None of the above

5) Consider the class hierarchy defined in question (3) again. Assume now that `Hourly` defines a method called `addHours(int)` and `Executive` defines a method called `awardBonus(double)`. Which of the following statements causes a compile-time error to occur?
   a. `Employee e = new Hourly(); ((Hourly) e).addHours(3.5);`
   b. `Employee e = new Executive(); e.awardBonus(400);`
   c. **Both of the above**
   d. None of the above

6) Which of the following interfaces defined in the NIO package can be used to decide if a given directory/file should be accepted or filtered?
   a. `FileFilter<Path>`
   b. `Filter<File>`
   c. `Filter<DirectoryStream>`
   d. **None of the above**

7) Consider the task of downloading the text that make up some online webpage. Assume that a `URL` object called `url` is created to represent that remote webpage. Which of the following statements correctly instantiates a `BufferedReader` object called `br` that can be used to perform the said task?
   a. `BufferedReader br = new BufferedReader(url.openStream());`
   b.
```
InputStream is = ur.openStream();
BufferedReader br =
new BufferedReader(new ReaderInputStream(is));
```
   c.
```
InputStream is = ur.openStream();
BufferedReader br =
new BufferedReader(new InputStreamReader(is));
```
   d. **None of the above**

8) Which of the following statements correctly creates a `BufferedInputStream` object that is attached to a remote webpage supplying binary content? Assume that the webpage is represented by a `URL` object called `url`.
   a.
```
InputStream is = url.openStream();
BufferedInputStream bis =
new BufferedInputStream(new InputStreamReader(is));
```
   b.
```
BufferedInputStream bis =
new BufferedInputStream(url.openStream());
```
   c. Both of the above
   d. None of the above

9) Which of the following methods of the `File` class accepts as a parameter a `FileFilter` object?
   a. `list`
   b. `listFile`
   c. `listFilteredFiles`
   d. **None of the above**

10) Which of the following represents the data type of the output produced by the method mentioned in the previous question?
   a. **`File[]`**
   b. `String[]`
   c. `Path[]`
   d. None of the above

Problem 3: I/O streams (**20 minutes**) [30 points]

You are given two text files called "textFile1.txt" and "textFile2.txt", respectively. Moreover, you are given two binary files called "binaryFile1.jpg" and "binaryFile2.jpg", respectively. Design and implement a Java program that:

1. Exchanges the content of "textFile1.txt" with that of "textFile2.txt". Be careful, as you are required to swap the contents of the two text files, meaning that "textFile1.txt" shall receive the original content of "textFile2.txt" and "textFile2.txt" shall receive the initial content of "textFile1.txt".
2. Then, your program should the same for the binary files exchanging their contents as well.

Assume in both cases that the source and the target files belong to the same directory as your Java project.

```
import java.io.*;
import java.util.Scanner;
public class SwappingFiles {
	public static void main(String[] args) throws IOException {
		File textFile1 = new File("textFile1.txt");
		File textFile2 = new File("textFile2.txt");
		File tmpTextFile = new File("tmpFile.txt");
		copyTextFile(textFile1, tmpTextFile);
		copyTextFile(textFile2, textFile1);
		copyTextFile(tmpTextFile, textFile2);
		File binaryFile1 = new File("binaryFile1.jpg");
		File binaryFile2 = new File("binaryFile2.jpg");
		File tempBinaryFile = new File("tmpFile.jpg");
		copyBinaryFile(binaryFile1, tempBinaryFile);
		copyBinaryFile(binaryFile2, binaryFile1);
		copyBinaryFile(tempBinaryFile, binaryFile2);

	}
private static void copyTextFile(File srcFile, File trgtFile) throws
IOException {
		Scanner fileScan = new Scanner(srcFile);
		FileWriter fw = new FileWriter(trgtFile);
		BufferedWriter bw = new BufferedWriter(fw);
		PrintWriter pw = new PrintWriter(bw);
		String line;
		while(fileScan.hasNextLine()) {
			line = fileScan.nextLine();
			pw.println(line);}
		pw.close();
		fileScan.close();}
private static void copyBinaryFile(File srcFile, File trgtFile) throws
IOException {
BufferedInputStream bis = new BufferedInputStream(new
FileInputStream(srcFile));
BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(trgtFile));
		byte[] buffer = new byte[1024];
		int len;

		while((len = bis.read(buffer)) > 0) {
			bos.write(buffer, 0, len);
		}

		bis.close();
		bos.close();
	}
```

```
}
```

Problem 4: XML (**25 minutes)** [30 points]

Consider the following XML-formatted document that shows a list of employees along with their gross and net salaries:

```
<?xml version="1.0"?>
<company>
<employee>
<firstname>Wissam</firstname>
<lastname>Fawaz</lastname>
<salary>
<gross>600</gross>
<net>440.54</net>
</salary>
</employee>
<employee>
<firstname>John</firstname>
<lastname>Stewart</lastname>
<salary>
<gross>400</gross>
<net>378.46</net>
</salary>
</employee>
<employee>
<firstname>Christiano</firstname>
<lastname>Ronaldo</lastname>
<salary>
<gross>300000</gross>
<net>288000</net>
</salary>
</employee>
<employee>
<firstname>Stephen</firstname>
<lastname>Colbert</lastname>
<salary>
<gross>340</gross>
<net>327.5</net>
</salary>
</employee>
</company>
```

The information given above is stored in an online **XML file** called "employees.xml" that resides inside a **folder** named "XML" on the "http://www.wissamfawaz.com" **webserver**. So, "employees.xml" can be referenced through the following URL: "http://www.wissamfawaz.com/XML/employees.xml".

1. Write a Java program that:
    a. retrieves all the gross and net salaries contained in this online xml file. This information should be stored in a text file called "`input.txt`",
    b. Your program should then output the first name and last name of the employees having the highest and lowest gross salaries.

```
import java.io.*;
import java.net.URL;
import java.util.ArrayList;
import javax.xml.parsers.*;
import org.w3c.dom.*;
```

```
public class Salaries {
	public static void main(String[] args) throws Exception {
URL employees = new URL("http://www.wissamfawaz.com/employees.xml");
InputStream employeesAsIS = employees.openStream();
		StringBuilder sb = new StringBuilder();
		ArrayList<String> xmlGrossAL = new ArrayList<>();
		ArrayList<String> xmlFirstNameAL = new ArrayList<>();
		ArrayList<String> xmlLastNameAL = new ArrayList<>();
		File input = new File("input.txt");
		FileWriter fw1 = new FileWriter(input);
		BufferedWriter bw1 = new BufferedWriter(fw1);
		PrintWriter pw1 = new PrintWriter(bw1);
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
DocumentBuilder db = dbf.newDocumentBuilder();
Document doc = db.parse(employeesAsIS);
NodeList salariesNodeList, firstNameNodeList, lastNameNodeList;
Node salaryNode, firstNameNode, lastNameNode;
String xmlGrossStr, xmlNetStr, xmlFirstNameStr, xmlLastNameStr;
		Element salaryElt;
		salariesNodeList = doc.getElementsByTagName("salary");
		firstNameNodeList = doc.getElementsByTagName("firstname");
		lastNameNodeList = doc.getElementsByTagName("lastname");
		for(int i=0; i<salariesNodeList.getLength(); i++) {
			salaryNode = salariesNodeList.item(i);
			salaryElt = (Element) salaryNode;
xmlGrossStr = salaryElt.getElementsByTagName("gross").item(0).getTextContent();
xmlNetStr = salaryElt.getElementsByTagName("net").item(0).getTextContent();
			firstNameNode = firstNameNodeList.item(i);
			xmlFirstNameStr = firstNameNode.getTextContent();
			lastNameNode = lastNameNodeList.item(i);
			xmlLastNameStr = lastNameNode.getTextContent();
			xmlGrossAL.add(xmlGrossStr);
			xmlFirstNameAL.add(xmlFirstNameStr);
			xmlLastNameAL.add(xmlLastNameStr);
			sb.append(xmlGrossStr + "\n");
			sb.append(xmlNetStr + "\n");}
		pw1.print(sb);
		pw1.close();
		double maxGross = Double.MIN_VALUE;
		double minGross = Double.MAX_VALUE;
		int indexMin = -1, indexMax = -1;
		double gross;
		for(int i=0; i<xmlGrossAL.size(); i++) {
			gross = Double.parseDouble(xmlGrossAL.get(i));
			if(gross > maxGross) {
				maxGross = gross;
				indexMax = i;
			}
			if(gross < minGross) {
				minGross = gross;
				indexMin = i;
			}
		}
		System.out.println("First name of employee with max gross: " +
		xmlFirstNameAL.get(indexMax));
		System.out.println("Last name of employee with max gross: " +
		xmlLastNameAL.get(indexMax));
		System.out.println("First name of employee with min gross: " +
		xmlFirstNameAL.get(indexMin));
		System.out.println("Last name of employee with min gross: " +
		xmlLastNameAL.get(indexMin));
	}
}
```

Appendix: Classes and Methods

1. XML parsing related classes along with their associated methods:

DocumentBuilderFactory • static DocumentBuilderFactory newInstance() • DocumentBuilder newDocumentBuilder() **Document** • NodeList getElementsByTagName(String) **Element** • NodeList getElementsByTagName(String)	**DocumentBuilder** • Document parse(InputStream) **NodeList** • int getLength() • Node item(int index) **Node** • String getTextContent() • NodeList getChildNodes()

2. Classes of the `java.io` package:

java.io.File • File(String) • long length() **java.io.FileOutputStream** • FileOutputStream(File) • void write(byte[] buff, int off, int len) • void close() **java.io.FileInputStream** • FileInputStream(File) • int read(byte[]) • void close()	**java.io.FileWriter** • FileWriter(File) **java.io.PrintWriter** • PrintWriter(BufferedWriter) • void print(String) • void println(String) • void close() **java.io.InputStreamReader** • InputStreamReader(InputStream) **java.io.BufferedReader** • BufferedReader(InputStreamReader) • String readLine()

3. Classes of the `java.nio` package:

java.nio.file.Files • static BufferedReader newBufferedReader(Path) • static BufferedWriter newBufferedWriter(Path) • static InputStream newInputStream(Path) • static Path copy(Path src, Path trgt) • static void write(Path trgt, byte[] buff)	**java.nio.file.Paths** • static Path get(String) **java.nio.file.Path** • Path getFileName() • File toFile()

4. `Scanner` and `ArrayList` and some of their related methods.

java.util.Scanner	**java.util.ArrayList<T>**
Scanner(File)	ArrayList<>()
String nextLine()	int size()
boolean hasNext()	T get()
String next()	void add(T)
int nextInt()	boolean contains(T)
double nextDouble()	int indexOf(T)