COE 593 – COE Application

Welcome to the Midterm Exam
Friday December 03, 2015

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 **110 minutes** to complete the **5** 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: Lambdas and Streams (**15 minutes**) [20 points]

1) Which one of the following functional interfaces contains a method called `apply` that takes a `T` parameter and returns a value of type `R`?
   a. `Consumer`
   b. `Function`
   c. `Supplier`
   d. `BinaryOperator`

2) Which one of the following generic functional interfaces contains a method called `get` that takes no arguments and produces a value of type `T`? Note that this interface is often used to create a collection in which a stream operation's results are placed.
   a. `Function`
   b. `Consumer`
   c. `BinaryOperator`
   d. `Supplier`

3) Which one of the following statements about lambdas is false?
   a. A lambda that receives two `int`s, x and y, and returns their sum can be written as follows:
      ```
      (int x, int y) -> {return x + y;}
      ```
   b. A lambda's parameter types can be omitted as in:
      ```
      (x, y) -> {return x + y;}
      ```
      In this case, the parameter and return types are determined by the lambda's context
   c. When a lambda's parameter list contains only one parameter, the parentheses may be omitted, as in:
      ```
      value -> System.out.printf("%d ", value)
      ```
   d. None of the above is false

4) Which of the following intermediate operations results in a stream containing only the unique elements?
   a. `unique`
   b. `filter`
   c. `limit`
   d. None of the above

5) Which of the following intermediate operations results in a stream in which each element of the original stream is transformed to a new value? The new stream will have the same number of elements as the original stream.
   a. `reduce`
   b. `limit`
   c. `filter`
   d. **None of the above**

6) Which of the following methods can be used to generate a `Stream<String>` from an array of `String`s?
   a. `Arrays.stream`
   b. `Stream.of`
   c. Both of the above
   d. None of the above

7) Which one of the following statements is false about `Predicate`?
   a. `Stream` method filter receives a `Predicate` and results in a stream of objects that match the `Predicate`
   b. `Predicate` method `test` returns a `boolean` indicating whether the argument satisfies a condition
   c. Both of the above
   d. None of the above

8) Which of the following is a method reference?
   a. `String::toUpperCase`
   b. `Math::sqrt`
   c. Both of the above
   d. None of the above

9) Which of the following methods of the `Comparator` interface reverses an existing `Comparator`'s ordering?
   a. `invert`
   b. `descending`
   c. `reverse`
   d. None of the above

10) Which of the following methods of `Collectors` returns a `Collector` that counts the number of objects in a given classification, rather than collecting them into a list?
   a. `counter`
   b. `counting`
   c. `count`
   d. None of the above

Problem 2: Miscellaneous (**15 minutes**) [20 points]

1) Which of the following classes provides static methods for common file and directory manipulations, including methods for copying files, creating and deleting files and directories, getting information about files and directories, and reading the contents of files and directories?
    a. `File`
    b. `DirectoryStream`
    c. `Paths`
    d. **None of the above**

2) Which of the following classes enable input and output of entire objects to or from a file?
    A. `SerializedInputStream`
    B. `SerializedOutputStream`
    C. `ObjectInputStream`
    D. `ObjectOutputStream`
    E. `Scanner`
    F. `Formatter`
    a. A and B
    b. **C and D**
    c. C, D, E, and F
    d. E and F

3) What interface must a class implement to indicate that objects of the class can be output and input as a stream of bytes?
    a. `Serialize`
    b. `Synchronize`
    c. `Deserialize`
    d. **None of the above**

4) Which of the following statement is true about `Buffer`s of the `java.nio` package?
    a. When in read mode, the `limit` property of a Buffer matches its `capacity`
    b. **When in write mode, the `position` property of a Buffer represents the position that immediately follows the last unread data.**
    c. Both of the above
    d. None of the above

5) `Channel` of the `java.nio` package differs from a stream in that
    a. it is unidirectional
    b. **it always reads from and writes to a Buffer**
    c. Both of the above
    d. None of the above

6) Which of the following does not match the regular expression "`\\d{2, 3}`"?
   a. 12
   b. 123
   c. Both of the above
   d. None of the above

7) What output does the following code fragment produce?

```
String searchObject = "xxfooxxxxfoo";
Pattern pattern = Pattern.compile(".*?foo");
Matcher matcher = pattern.matcher(searchObject);
while(matcher.find())
      System.out.println(matcher.group());
```

   a. **xxfoo**
      xxxxfoo
   b. xxfooxxxxfoo
   c. No output
   d. None of the above

8) What output does the following code fragment produce?

```
String birthdays = "Chadi's birthday is: 12/04/2014\n"+
                   "Chen's birthday is: 23/05/1976\n"+
                   "Wissam's birthday is: 01/05/1989";
Pattern pattern = Pattern.compile("C.*\\d\\d[-/]0[1-35-9][-
/]\\d\\d");
Matcher matcher = pattern.matcher(birthdays);
while(matcher.find())
   System.out.println(matcher.group());
```

   a. `Chadi's birthday is: 12/04/2014`
   b. `Chadi's birthday is: 12/04/2014`
      `Chen's birthday is: 23/05/1976`
   c. **`Chen's birthday is: 23/05/1976`**
   d. None of the above

9) Which of the following Java `String`s represents the regular expression `, \s*`?
   a. `"\, \s\*"`
   b. `", \\s\*"`
   c. `",\\s*"`
   d. None of the above

10) Which of the following symbols can be used to make a "quantifier" greedy in a regular expression?
   a. +
   b. ?
   c. *
   d. **None of the above**

Problem 3: IO streams (**30 minutes**) [20 points]

Design and implement a Java program that copies the content of a text file called "srcFile.txt" into a target file called "trgtFile1.txt" using first the `BufferedReader` and `BufferedWriter` classes of the `java.io` package. Your program should then copy the content of "srcFile.txt" into another target file called "trgtFile2.txt" using the `FileChannel` and `ByteBuffer` classes of the `java.nio` package.

Solution using BufferedReader and BufferedWriter:

```
import java.io.*;
import java.nio.file.*;
public class FileManagement {
      public static void main(String[] args) {
            Path srcPath = Paths.get("srcFile.txt");
            Path copyPath = Paths.get("copyFile.txt");
            try {
                  BufferedReader br =
                  Files.newBufferedReader(srcPath);
                  BufferedWriter bw =
                  Files.newBufferedWriter(copyPath);
                  String line;
                  while((line = br.readLine()) != null)
                        bw.write(line+"\n");
                  br.close();
                  bw.close();
            } catch (IOException e) {e.printStackTrace();}}}
```

Solution using FileChannel and ByteBuffer:

```
import java.io.*; import java.nio.channels.FileChannel;
import java.nio.ByteBuffer; import java.nio.file.*;
public class ChannelBasedFileMgt {
      public static void main(String[] args) throws Exception{
            Path srcPath = Paths.get("srcFile.txt");
            Path copyPath = Paths.get("copyFile.txt");
            FileChannel srcChannel = FileChannel.open(srcPath);
            FileChannel copyChannel = FileChannel.open(copyPath,
                              StandardOpenOption.WRITE,
                              StandardOpenOption.CREATE);
            ByteBuffer buffer = ByteBuffer.allocate(1024);
            int readBytes = srcChannel.read(buffer);
            while(readBytes != -1) {
                        buffer.flip();
                        copyChannel.write(buffer);
                        buffer.clear();
                        readBytes = srcChannel.read(buffer);
            }

            srcChannel.close();
            copyChannel.close();

      }

}
```

Problem 4: JDOM (**25 minutes)** [20 points]

Consider the following XML and JSON documents that show credit client details:

```
<clients>
        <client>
                <client-name>Steve</client-name>
                <account-number>100</account-number>
                <client-balance>-345.67</client-balance>
        </client>
        <client>
                <client-name>Sam</client-name>
                <account-number>200</account-number>
                <client-balance>-42.16</client-balance>
        </client>
        <client>
                <client-name>Bob</client-name>
                <account-number>300</account-number>
                <client-balance>224.62</client-balance>
        </client>
</clients>
```

```
[
        {“client-name”:“Steve”,
         “account-number”:100,
         “client-balance”:345.67
        },
        {“client-name”:“Sam”,
         “account-number”:200,
         “client-balance”:42.16
        },
        {“client-name”:“Bob”,
         “account-number”:300,
         “client-balance”:224.62
        }
]
```

The information given on the left hand side is stored in an **XML file** called “clients.xml” while the information on the right hand side is stored in a **JSON file** called “clients.json”. Both files reside inside **folders** named “XML” and “JSON” respectively on the “http://www.wissamfawaz.com” **webserver**. So, “clients.xml” can be referenced through the following URL: “http://www.wissamfawaz.com/XML/clients.xml” while “clients.json” can be referenced via: “http://www.wissamfawaz.com/XML/clients.json”. These xml and JSON files might be used in an accounts receivable system to keep track of the amounts owed to a company by its credit clients.

1. Using the JSON parser, write a Java program that outputs the total amount of money the clients owe the company by summing up the “client-balance” values pertaining to all the clients.

```
import java.io.InputStream;
import java.net.URI;
import java.net.URL;
import org.json.JSONArray;
import org.json.JSONObject;
import org.json.JSONTokener;
public class ClientDetailsParsing {
     public static void main(String[] args) {
      URL url =                                    ;//Incomplete
      InputStream is = url.openStream();
      JSONTokener tokener = new JSONTokener(is);
     JSONArray clients = new JSONArray(tokener);
     double amount=0; JSONObject client; double balance;
     for(int i=0; i < clients.length(); i++) {
          client = clients.getJSONObject(i);
          balance = client.getDouble(“client-balance”);
          amount+=balance;}
```

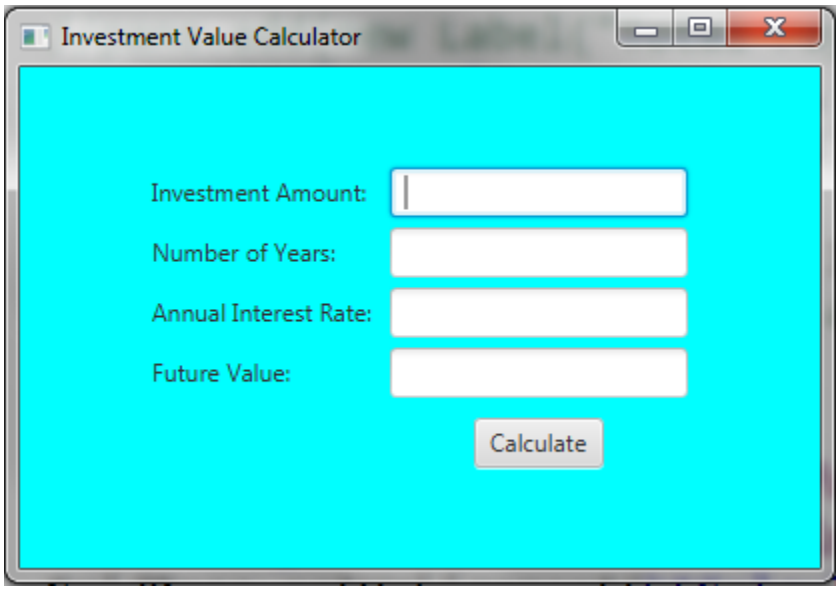
```
     System.out.println("Total amount:"+amount);
     } // end main method
} // end ClientDetails class
```

2. Using the **JDOM parser**, write a Java program that:
    a. outputs the number of clients having a negative "client-balance", and then
    b. outputs the "client-name" of the client having the largest "client-balance" associated with him.

```
import java.io.InputStream;
import java.net.URL;
import java.util.List;
import org.jdom2.Document;
import org.jdom2.Element;
import org.jdom2.input.SAXBuilder;
import org.jdom2.filter.ElementFilter;
import org.jdom2.util.IteratorIterable;

public class ParsingCreditClientInformation {
   public static void main(String[] args)throws Exception {
     URL url =                                    ;//Incomplete
     InputStream is = url.openStream();
     SAXBuilder builder = new SAXBuilder();
     Document doc = builder.build(is);
     Element root = doc.getRootElement();
     IteratorIterable<Element> clientsIterator =
          root.getDescendants(new ElementFilter("client"));
     int nbNeg=0; Element client;
     double largest = Double.MIN_VALUE;
     String nameLargest; double clientBalance;
     while(clientsIterator.hasNext()) {
          client = clientsIterator.next();
          clientBalance=Double.parseDouble(
          client.getChildText("client-balance"));
          if(clientBalance < 0)
               nbNeg++;
          if(clientBalance > largest) {
               nameLargest=
               client.getChildText("client-name");
               largest = clientBalance;
          }}
     System.out.println("Nb of negatives:"+nbNeg);
     System.out.println("Name:" + nameLargest);
     } // end main method
} // end ParsingCreditClientInformation class
```

Problem 5: Java-FX (**25 minutes)** [20 points]



Write a Java-FX application that calculates the future value of an investment at a given interest rate for a specified number of years. The formula for the calculation is:

$$futureValue = investmentAmount \times (1 + monthlyInterestRate)^{years \times 12}$$

Use the graphical user interface shown above. In particular, the text fields for the investment amount, number of years, and annual interest rate are used to gather the input data from the user. Then, the output future amount is calculated based on the user-supplied data and displayed in the appropriate text field when the user clicks the "Calculate" button. A program skeleton is provided next and your job is to simply complete the implementation of the program as per the guidelines conveyed through the comments that are highlighted in bold.

```
import javafx.application.Application ;
import javafx.geometry.HPos ;
import javafx.geometry.Pos ;
import javafx.scene.Scene ;
import javafx.scene.control.Button ;
import javafx.scene.control.Label ;
import javafx.scene.control.TextField ;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.GridPane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;

public class Main extends Application {
      // Instance variables representing GUI components
      // should be included below
      private TextField tfInvestmentAmount;
      private TextField tfNumberOfYears;
      private TextField tfAnnualInterestRate;
      private TextField tfFutureValue;
      private Button btCalculate;
      // start method for creating pane, scene, nodes, and stage
      @Override
      public void start(Stage primaryStage) {
            GridPane pane = new GridPane();
            Scene scene = new Scene(pane, 400, 250);
            pane.setBackground(new Background(
                  new BackgroundFill(Color.CYAN, null, null)));
            pane.setHgap(5);
            pane.setVgap(5);
            pane.setAlignment(Pos.CENTER);
            pane.add(new Label("Investment Amount: "), 0, 0);
            pane.add(new Label("Number of Years: "), 0, 1);
            pane.add(new Label("Annual Interest Rate: "), 0, 2);
            pane.add(new Label("Future Value: "), 0, 3);
            pane.add(tfAnnualInterestRate, 1, 0);
            pane.add(tfNumberOfYears, 1, 1);
            pane.add(tfLoanAmount, 1, 2);
            pane.add(tfMonthlyPayment, 1, 3);
            pane.add(btCalculate, 1, 5);

            tfFutureValue.setEditable(false);
            GridPane.setHalignment(btCalculate, HPos.CENTER);
            btCalculate.setOnAction(e->calculateFutureValue());

            primaryStage.setScene(scene);
            primaryStage.setTitle(“Future Value Calculator”);
            primaryStage.show();


      }
```

```
	// Method for calculating and displaying future value
	private void calculateFutureValue() {
		double monthlyInterest =
	Double.parseDouble(tfAnnualInterestRate.getText())/1200;
		int nbOfYrs =
	Integer.parseInt(tfNumberOfYears.getText());
		double amount =
	Double.parseDouble(tfInvestmentAmount.getText());

		double futureValue =
		amount*Math.pow(1+monthlyInterest, nbOfYrs*12);

		tfFutureValue.setText(futureValue+ "");









	}

	public static void main(String[] args) {
		launch(args);
	}
}
```

<table>
<tr>
<td>

org.jdom2.Document

- Element getRootElement()

org.jdom2.Element

- List<Element> getChildren(String)
- IteratorIterable<Element> getDescendants(ElementFilter)
- String getText()

String getChildText(String)

</td>
<td>

org.jdom2.filter.ElementFilter

- ElementFilter(String)

org.jdom2.util.IteratorIterable

- boolean hasNext()
- Element next()

java.util.List<Element>

- int size()
- Element get(int index)

</td>
</tr>
<tr>
<td>

JSONArray

JSONArray(JSONTokener)
int length()
JSONObject getJSONObject(int i)
JSONArray getJSONArray(int i)
double getDouble(int i)
int getInt(int i)
String getString(int i)

</td>
<td>

JSONObject

JSONObject(JSONTokener)
JSONArray getJSONArray(String key)
JSONObject getJSONObject(String key)
String getString(String key)
double getDouble(String key)
int getInt(String key)

</td>
</tr>
<tr>
<td>

java.io.BufferedReader

- BufferedReader(InputStreamReader)
- String readLine()

java.io.BufferedWriter

- BufferedWriter(FileWriter)
- void write(String s, int off, int len)

java.io.BufferedOutputStream

- BufferedOutputStream(OutputStream)
- void write(byte[] data, int off, int len)

java.io.File

- File(String)

</td>
<td>

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.FileOutputStream

- FileOutputStream(File)

java.io.FileInputStream

- FileInputStream(File)

</td>
</tr>
<tr>
<td>

java.nio.file.Files

- static BufferedReader newBufferedReader(Path)
- static BufferedWriter newBufferedWriter(Path)
- static InputStream newInputStream(Path)
- static OutputStream newOutputStream(Path)

</td>
<td>

java.nio.file.Paths

- static Path get(String)

java.nio.file.Path

- Path getFileName()
- File toFile()

java.lang.String

- String[] split(String)
- byte[] getBytes
- boolean matches(String)

</td>
</tr>
<tr>
<td>

java.nio.channels.FileChannel

- FileChannel open(Path)
- int read(ByteBuffer)
- int write(ByteBuffer)
- void close()

</td>
<td>

java.nio.ByteBuffer

- ByteBuffer allocate(int)
- void flip()
- void clear()

</td>
</tr>
</table>

GridPane GridPane() void add(Node, int colIndex, int rowIndex) void setHalignment(Node, HPos.Right|HPos.LEFT) void setHgap(int) void setVgap(int)	**Button** Button(String) void setOnAction(EventHandler<ActionEvent>) **EventHandler<T extends Event>** void handle(T event) **Scene** Scene(Pane, double width, double height)
TextField TextField() void setEditable(boolean) String getText() void setText(String)	**Stage** void setResizable(boolean) void setScene(Scene) void setTitle(String) void show()