COE 312 – Data Structures

Welcome to Exam I
Tuesday November 13, 2012

Instructor: 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 **90 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: Polymorphism (**15 minutes**) [18 points]

1. Consider the following Java application:

```
public class Polymorphism {
   interface Area {public void area();}
   interface Volume {public void vol();}

   class Shape {
      public void draw(){System.out.println("Shape draw");}
   }
   class TwoDimensionalShape extends Shape implements Area {
      public void area(){System.out.println("2D shape area");}
      public void draw(){System.out.println("2D shape draw");}
   }
   class Rectangle extends TwoDimensionalShape {
      public void area(){System.out.println("Rectangle area");}
   }
   class ThreeDimensionalShape extends Shape
   implements Area, Volume {
      public void area(){System.out.println("3D shape area");}
      public void vol(){System.out.println("3D shape volume");}
      public void draw(){System.out.println("3D shape draw");}
   }
   class Sphere extends ThreeDimensionalShape {
      public void vol(){System.out.println("Sphere volume");}
      public void draw() {System.out.println("Sphere draw");}
    }

   public Polymorphism() {
      Shape shape;
      Object object;
      Area twoDShape;
      Volume  threeDShape = new ThreeDimensionalShape();
      TwoDimensionalShape rectangle = new Rectangle();

      threeDShape.vol();
      ((TwoDimensionalShape) rectangle).area();
      shape = (ThreeDimensionalShape) threeDShape;
      shape.draw();
      twoDShape = rectangle;
      ((Rectangle) twoDShape).draw();
      object = twoDShape;
      ((TwoDimensionalShape) object).area();
      threeDShape = new Sphere();
      ((Area) threeDShape).area();
   }
   public static void main(String[] args) {
      new Polymorphism();
   }
}
```

What output will be produced by the above-presented code fragment? Display the output in the box given below.

3D Shape volume
Rectangle area
3D Shape draw
2D shape draw
Rectangle area
3D shape area

2. Suppose a class called `Sub` implements an interface called `Sandwich`. Assume also that you are given the following Java statements:

```
Sandwich x = new Sub();
Sub y = new Sub();
```

Which of the following assignments are legal? Circle all of the correct answers.

a. y = x;
b. x = y;
c. y = (Sub) x;
d. x = (Sandwich) y;

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

1. Design and implement a Java program that reads a string from the end user and then converts the input string into an integer. If the user enters an invalid string, your program should generate an exception message alerting him to the fact that he entered a string that does not contain a numeric value. Once the exception message gets printed, your program should ask the user to enter another string. This process is repeated until the end user provides a properly formatted string, in which case the program would eventually terminate its execution.

```
import java.util.Scanner;
public class NumberFormat {
    public static void main(String[] args) {
        String input;
Reading line in  int input_int; Scanner scan = new Scanner(System.in);
   (3)           boolean continueLoop = true;
using try+catch  while (continueLoop) {
+right exception     System.out.println("Enter a String containing " +
   class                    "a numeric value");
   (3)               input = scan.nextLine();
                     try {
ParseInt +               input_int = Integer.parseInt(input);
While loop               continueLoop = false;
   (4)               } catch (NumberFormatException e) {
                        System.out.println("Input is not numeric");
                     }
                }
                System.out.println("Input as int: " + input_int);
            }
        }
```

2. Design and implement a Java program that creates an exception class called `InvalidIndexException`. Supplement this class with a driver class that reads from the end user an `int` value followed by a `String` of characters. The obtained `int` value serves as an index to one of the characters in the input string. Therefore, your program should throw an `InvalidIndexException` if the int index value is found to be out of bounds. Under such a condition, your program should handle the exception and get another `int` value from the user. The repetition should continue until the user provides a valid index and at that point your program should print the character stored at the supplied index.

```
Public class InvalidIndexException extends Exception {
    Public InvalidIndexException(String msg) {
        super(msg);
    }
}
```
(3)

```
import java.util.Scanner;
Public class Driver {
    Public static void main(String[] args) {
        String str; int index;
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter an int value: ");
        index = scan.nextInt();
        scan.nextLine();
        S.o.p("Enter a line of text: ");
        str = scan.nextLine();
        InvalidIndexException exception =
                new InvalidIndexException("Index out of bounds");
        boolean continueLoop = true;
        While (continueLoop) {
        try { if( index >=0 && index < str.length())
                throw exception;
            continueLoop = false;
        } catch (InvalidIndexException e) {
            S.o.p(" Exception: " + e.getMessage());
            S.o.p("Please enter a valid index ");
            index = scan.nextInt();
        } }
        S.o.p("Character at supplied index: " + str.charAt(index));
```

(+3) (7) (+4)

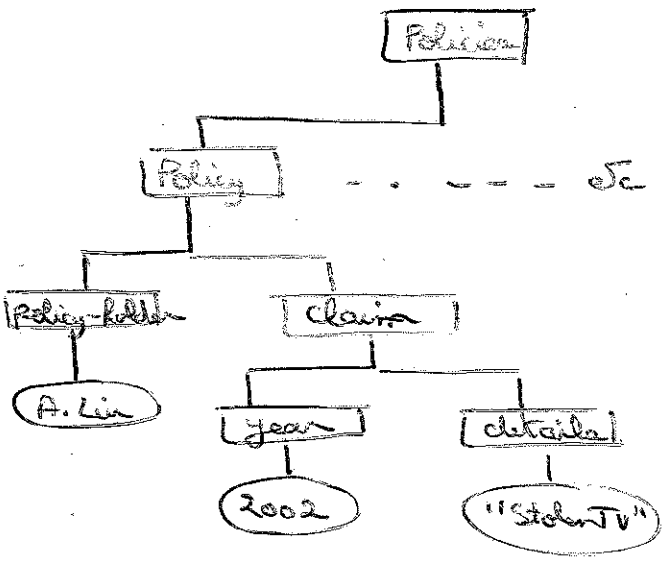
Problem 3: DOM Parser (**20 minutes**) [20 points]

Consider the following XML code snippet:

```
<?xml version="1.0"?>
<policies>
     <policy>
          <policy-holder>A. Liu</policy-holder>
          <claims>
               <claim>
                    <year>2002</year>
                    <details>Stolen TV</details>
               </claim>
          </claims>
     </policy>
     <policy>
          <policy-holder>B. Singh</policy-holder>
          <claims>
               <claim>
                    <year>2004</year>
                    <details>Damage to Kitchen</details>
               </claim>
          </claims>
     </policy>
     <policy>
          <policy-holder>D. Umaga</policy-holder>
          <claims>
               <claim>
                    <year>1998</year>
                    <details>Stolen bicycle</details>
               </claim>
               <claim>
                    <year>2004</year>
                    <details>Dropped Vase</details>
               </claim>
          </claims>
     </policy>
</policies>
```

The XML code above is stored in an **XML file** called "policies.xml" that resides in a **folder** named "XML" on the "http://www.wissamfawaz.com" **webserver**. So, the said file can be identified by the following URL: "http://www.wissamfawaz.com/XML/policies.xml". In what follows, "policies.xml" is used as the source XML for parsing purposes.

1. Draw a **tree** that captures the structure of the **Document Object Model** (DOM) corresponding to the XML code given above.



6

2. Using the **DOM parser**, write a Java program that outputs the number of the "claim" **elements** found in the "policies.xml" file as well as the "year" and "details" information pertaining to each "claim" element.

```
import java.io.InputStream;
import java.net.URL;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

public class ParsingClaimInformation {
     public static void main(String[] args) {
     try {
          URL url = new URL("file", "localhost", "policies.xml");
          InputStream is = url.openStream();
          Element claim;
          Node year, details;
          NodeList yearList, detailsList;
          String yearText, detailsText;
          StringBuilder sb = new StringBuilder();
```

14

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
DocumentBuilder db = dbf.newDocumentBuilder();
Document document = db.parse(is);
NodeList claimList = document.getElementsByTagName("claim");
S.o.p("Nb of claims" + claimList.getLength());
for(int i=0; i < claimList.getLength(); i++) {
    claim = (Element).claimList.item(i);
    yearList = claim.getElementsByTagName("year");
    year = yearList.item(0);
    yearText = year.getFirstChild().getNodeValue();
    sb.append(yearText + "\n");
    detailsList = claim.getElementsByTagName("details");
    details = detailsList.item(0);
    detailsText = details.getFirstChild().getNodeValue();
    sb.append(detailsText + "\n");
}
S.o.p(sb);
} catch(Exception e) {
        S.o.p("Exception: "+e);
}
```

```
    } // end main method
} // end ParsingClaimInformation class
```

Problem 4: JSON Parser (**20 minutes**) [20 points]

Consider the following JSON document that was generated by Google's geo-coding API and that is available at:
"http://maps.googleapis.com/json?address=1600+Amphitheatre+Parkway"

```
{
   "results" : [
      {
         "address_components" : [
            {
               "long_name" : "1600",
               "short_name" : "1600",
               "types" : [ "street_number" ]
            },
            {
               "long_name" : "Amphitheatre Pkwy",
               "short_name" : "Amphitheatre Pkwy",
               "types" : [ "route" ]
            }
         ],
         "formatted_address" : "1600 Amphitheatre Pkwy",
         "geometry" : {
            "location" : {
               "lat" : 37.42208060,
               "lng" : -122.0845760
            }
         }
      }
   ]
}
```

1. How many JSON objects are contained in the above-presented JSON document?

six JSON objects (6)

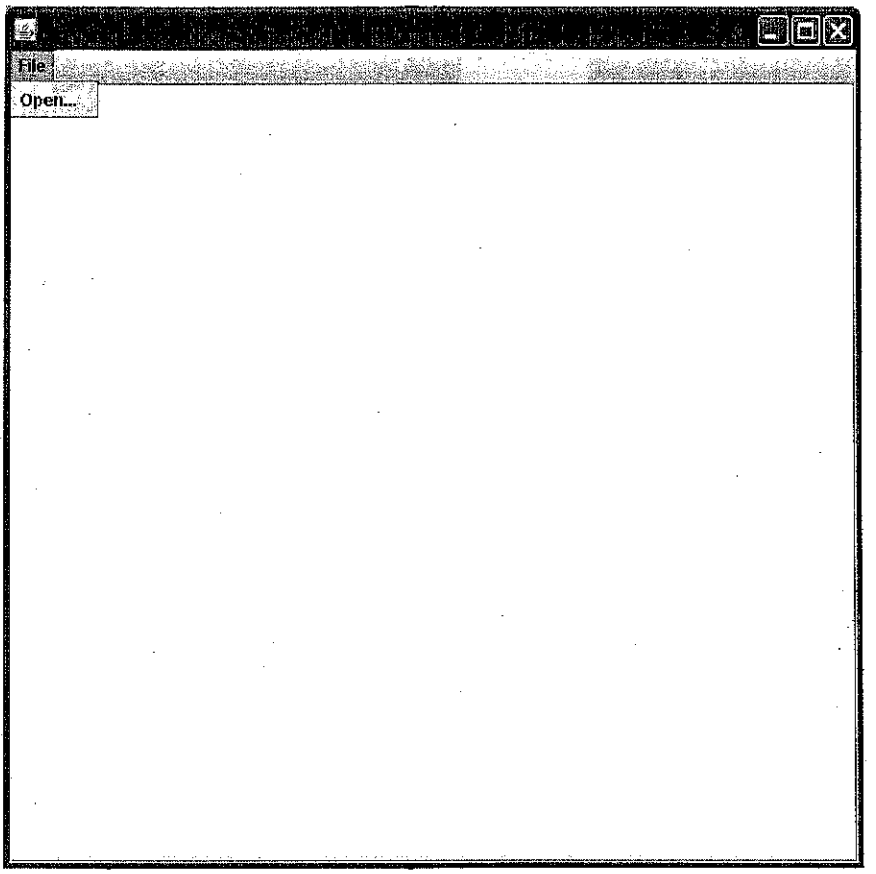
2. How many JSON arrays does it have?

four JSON arrays (4)

3. Using the JSON parser, write a Java program that first extracts the latitude and longitude GPS coordinates wrapped inside the JSON document presented earlier and then prints them to the screen.

```
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 GPSCoordinates {
    public static void main(String[] args) {
    try {
        URL url = new URL("file","localhost", "Address.json");
        InputStream is = url.openStream();
        JSONObject arrElt;
        double lat, lng;
        JSONObject jsonObject2, jsonObject3;
        StringBuilder sb = new StringBuilder();
      JSONTokener jst = new JSONTokener(is);
      JSONObject jsonObject1 = new JSONObject(jst);
      JSONArray jsonArray1 = jsonObject1.getJSONArray("results");
      for(int i=0; i< jsonArray1.length();i++) {
          arrElt=jsonArray1.getJSONObject(i);
          jsonObject2 = arrElt.getJSONObject("geometry");
          jsonObject3 =jsonObject2.getJSONObject("location");
          lat = jsonObject3.getDouble("lat");
          lng = jsonObject3.getDouble("lng");
          sb.append(lat + ", "+ lng);
    }
        sop(sb);
    } // end main method
} // end GPSCoordinates class
```

Problem 5: GUI-based Applications (**20 minutes**) [22 points]



Design and implement a Java application that uses the GUI shown above. This application enables the user to select a text file from the file system, retrieve the selected file and then display its content in a text area. As illustrated through the figure above, the GUI is made up of the following components: 1) **File menu** containing an **Open...** menu item that creates a **JFileChooser** when it gets selected, 2) **JTextArea** used for the purpose of showing the content of the file obtained through the file chooser, and 3) **JScrollPane** providing scrolling for the text area.

It is important to note that the content of the selected file is to be read through the Scanner class by means of the `hasNext()` and nextLine() methods.

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 java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;

public class OpenFile extends JFrame {
     // Include instance variables below
       JFileChooser fileChooser; JMenuBar menuBar;
       JMenu fileMenu; JMenuItem openItem;
       JTextArea fileContentArea;
       JScrollPane scrollPane; File selectedFile;
       Scanner fileScan;

     // Complete implementation of the constructor
     public OpenFile() { selectedFile = null;
          menuBar = new JMenuBar();
          fileMenu = new JMenu("File");
          openItem = new JMenuItem("Open...");
          openItem.addActionListener(new OpenListener());
          fileContentArea = new JTextArea();
          scrollPane = new JScrollPane(fileContentArea);
          fileMenu.add(openItem);
          menuBar.add(fileMenu);
          setJMenuBar(menuBar);
          add(scrollPane);
     } // end OpenFile constructor method

     private class OpenListener implements ActionListener {
          public void actionPerformed(ActionEvent e) {
               selectedFile = null;
               // create filechooser and show its dialog
               fileChooser = new JFileChooser();
               int result = fileChooser.showOpenDialog(this);
               // retrieve selected file
               if (result == JFileChooser.APPROVE_OPTION)
                    selectedFile = fileChooser.getSelectedFile();
```

(8)

(3)

(4)

```
                if(selectedFile != null)
                        showFileContent(selectedFile);

            }
        } // end OpenListener private class

        // method used to display content of file in text area
        private void showFileContent(File file){
            String output = "";
            try {
                fileScan = new Scanner(selectedFile);
            } catch(IOException e) {
                System.out.println("Could not open file"); return;
            }
            while(fileScan.hasNext())
                output += fileScan.nextLine();
            fileContentArea.setText(output);


        } // end showFileContent method

        public static void main(String[] args) {
            OpenFile frame = new OpenFile();
            frame.setDefaultCloseOperation(
                        JFrame.EXIT_ON_CLOSE);
            frame.setSize(600, 600);
            frame.setLocationRelativeTo(null);
            frame.setVisible(true);
        } // end main method
    } // end OpenFile class
```

(7)

Appendix: Classes and Methods

1. DOM parser related classes along with their associated methods.

Document	**NodeList**
Element getDocumentElement() Element getElementById(String) NodeList getElementsByTagName(String)	int getLength() Node item(int index)

Node	**Element**
NodeList getChildNodes() Node getFirstChild() Node getLastChild() String getNodeValue() Node getParentNode() boolean hasChildNodes()	NodeList getElementsByTagName(String) String getTagName() String getAttribute(String) void removeAttribute(String) boolean hasAttribute(String)

2. JSON parser classes and some of their related methods.

JSONTokener	**JSONObject**
JSONTokener(InputStream)	JSONObject(JSONTokener) JSONArray getJSONArray(String key) JSONObject getJSONObject(String key) String getString(String key)

JSONArray
JSONObject getJSONObject(int index) String getString(int index) JSONArray getJSONArray(int index) double getDouble(int index) long getLong(int index) int getInt(int index) int length()

3. Classes related to GUI-based application

Scanner	**JFileChooser**	**JTextArea**
Scanner(File) boolean hasNext() String nextLine()	int APPROVE_OPTION int CANCEL_OPTION int showInputDialog(Component) File getSelectedFile()	setText(String)