DAM. UNIT 1. ACCESS TO FILES. ASSESSABLE TASK 1

DAM. Acceso a Datos (ADA) (a distancia en inglés)

Unit 1. ACCESS TO FILES

Assessable Task 1

Abelardo Martínez

Year 2023-2024

Aspects to bear in mind

Important

If you look for the solutions surfing the Internet or asking the oracle of ChatGPT you will be fooling yourself. Keep in mind that **ChatGPT is not infallible or all-powerful**.

It is a great tool to speed up your work once you have mastered a subject, but using it as a shortcut when acquiring basic skills and knowledge seriously undermines your learning. If you use it to get solutions or advice on your own, check the proposed solutions carefully as well. Try to solve the activities using the resources we have seen and the extended documentation you will find in the "Virtual Classroom".

Tips for programming

We advice to follow the next coding standards:

- One instruction per line.
- Add comments to make your code clearer and more readable.
- Use the Hungarian notation to recognise the type of variables at first sight.
- Remember that there are several ways to implement a solution, so choose the one you like best. **We strongly recommend using buffer-based solutions**.

A. Instructions and guidelines

The project MUST be carried out in Java. **Other technologies -such as Spring Boot- will not be supported**. Any of the IDEs proposed in unit 1 can be used for its development, although **Eclipse is strongly recommended**.

1. OVERVIEW

You are required to create a Java application **on your own** that utilises concepts taught during UNIT 1 (Week1-Week5) to meet a provided specification.

2.TIMELINE AND EXPECTATIONS

- **Percentages within the TERM**: 50% of TERM total (AT2 would make the other 50%)
- **Percentages within the TASK**: 100% ADA skills (English skills must be PASSED).
- **Due/Deadline**: **11:59pm on Sunday, 5th November, 2023** (3 WEEKS)

3. GRADING

You must get 5 marks out of 10 in ADA and a PASSED in English to pass this ASSESSABLE TASK.

A detailed grading scale will be providing with this document (check LEARNING RUBRIC).

4.RESOURCES

You should make a comprehensive reading of all the materials provided by your teacher as well as the non-assessable tasks, but also dive the Internet to find examples which provide similar outcomes to the ones required by this task.

Feel free to copy & paste code from ANY resource as long as you understand every piece of it since you will be required to defend your work in an individual meeting.

5. PLAGIARISM

You must not allow other students to copy your work and must take care to safeguard against this happening.

In case of suspected plagiarism, an additional oral interview might be required.

6. HANDING AND FEEDBACK

- **The task will be delivered ONLY in a ZIP format file, compressing the project folder from your IDE** (i.e. Eclipse).
- Afterwards, **you WILL BE REQUIRED to attend an oral interview** with your teacher to discuss certain aspects of your task in English for a maximum of 15 minutes.

- You will receive your marks broken down by each criteria, and the total, together with any comments giving suggestions on how you could have done better.

B. Assessment details

> **ONLY ENGLISH IS ALLOWED** for the implementation of the assessable task, both comments and explanatory/clarifying texts.

1. **EVERY METHOD MUST BE PROPERLY DESCRIBED IN YOUR OWN WORDS**. At the beginning of each method you must add comments to explain in your own words how it works.
2. **ALSO, YOU MUST ADD A TEXT EXPLAINING IN YOUR OWN WORDS, YOUR EXPERIENCE IMPLEMENTING THIS SOLUTION**.

Create a text file and copy it into the project folder or create the text file within the project itself in the Eclipse IDE.

- **PARAGRAPH 1**. Describe briefly the solution provided.
- **PARAGRAPH 2**. Describe briefly the difficulties found.
- **PARAGRAPH 3**. Describe briefly several possible extensions you recommended.

B.1. Mandatory features

Activity (ASSESSABLE)

Create a program in Java to manage CHESS PLAYERS in a Chess Tournament by printing and using a specific menu. After each option, the user should see the same menu until option zero is pressed.

ATTENTION: Use the proper exceptions when accessing to files.

Menu options:

- **Press 0 to "Exit"**
- **Press 1 to "Get chess players and scores (to CSV & XML)"**
  - For every CHESS PLAYER we need player ID (String), full name (String with spaces), country (String with spaces), score/rating of game 1 (score1, Float), score/rating of game 2 (score2, Float) and score/rating of game 3 (score3, Float), added to an ArrayList of CHESS PLAYERS. A player who wins his/her game, or who wins by default, receives one point (1); a player who loses his/her game, or who loses by default, receives zero points (0); and a player who ends his/her game in a draw receives half a point (½). Therefore, there are only 3 possible score values: 0, 0.5 and 1 point.
  - **Check if the chess player ID already exists in the array list**. If yes, you must display a message on the screen. You must ask for each value (in loop) until the user enters a valid ID.
  - Once zero is entered as ID, all CHESS PLAYERS will be saved both in a CSV and XML file, with a proper format, overwriting all the whole files if exist. Before saving, you should check if the ArrayList of CHESS PLAYERS is empty to avoid executing unnecessary code.
  - While you read the ArrayList to create the XML, you have to add a node called **playerscoretotal** and save the sum of the scores for every CHESS PLAYER.
  - **ATTENTION**: score1, score2 and score3 must be FLOAT! For every CHESS PLAYER, you must ask for each value (in loop) until the user enters a valid float.

  - Call the **CSV file**: chessplayers.csv
  - Call the **XML file**: chessplayers.xml

- **Press 2 to "List all chess players stored (using DOM)"**
  - Just read the XML file and print every CHESS PLAYER information. **You may use DOM**.

- **Press 3 to "Generate HTML with all chess players via XSL"**
  - Editing and adapting the XSL template you can find at the end of this document or any other XSL you can find on the Internet, you have to read the XML file and generate an HTML using the XSL. The header table must have a background color.
  - **Call the HTML file**: chessplayers.html
  - **Call the XSL file**: chessplayers.xsl

Menu example:

```
*****
MENU
*****
========================================================================
  0. Exit
  1. Get chess players and scores (to CSV & XML)
  2. List all chess players
  3. Generate HTML with all chess players via XSL
  4. [optional] Modify the rating of a chess player's game
  5. [optional] Generate HTML with chess master candidates via XSL
========================================================================
    Select an option:
```

Here you have a SUGGESTION of how your CSV could look like:

```
playerID, Name, Country, Score1, Score2, Score3
ELO2780, ANATOLY KARPOV, Russia, 1, 1, 1
```

```
ELO2851, GARRY KASPAROV, Azerbaijan, 1, 1, 1
ELO2882, MAGNUS CARLSEN, Norway, 1, 1, 1
ELO2530, ARTURO POMAR, Spain, 1, 0.5, 0
ELO2785, BOBBY FISCHER, USA, 1, 1, 1
ELO2725, JOSÉ RAÚL CAPABLANCA, Cuba, 1, 1, 1
ELO2100, JOHN LOST, United Kingdom, 0.5, 0, 0
```

Here you have a SUGGESTION of how your XML could look like:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Players>
     <Player>
          <playerid>ELO2780</skaterid>
          <playername>ANATOLY KARPOV</playername>
          <playercountry>Russia</playercountry>
          <playerscore1>1</playerscore1>
          <playerscore2>1</playerscore2>
          <playerscore3>1</playerscore3>
          <playerscoretotal>3</playerscoretotal>
     </Player>
     <Player>
          <playerid>ELO2851</skaterid>
          <playername>GARRY KASPAROV</playername>
          <playercountry>Azerbaijan</playercountry>
          <playerscore1>1</playerscore1>
          <playerscore2>1</playerscore2>
          <playerscore3>1</playerscore3>
          <playerscoretotal>3</playerscoretotal>
     </Player>
     <Player>
          <playerid>ELO2530</skaterid>
          <playername>ARTURO POMAR</playername>
          <playercountry>Spain</playercountry>
          <playerscore1>1</playerscore1>
```

```
        <playerscore2>0.5</playerscore2>
        <playerscore3>0</playerscore3>
        <playerscoretotal>1.5</playerscoretotal>
    </Player>
</Players>
```

B.2. Optional features

Activity (ASSESSABLE)

Optionally, you can implement these following entries within the menu to reach more than 8 marks out of 10 at this ASSESSABLE TASK.

ATTENTION: Use the proper exceptions when accessing to files.

Menu options:

- **Press 4 to "[optional] Modify the rating of a chess player's game"**
  - The Referee Committee may review a game on its own initiative or at the request of a player in the tournament. If the result of the game is affected, the score will be modified according to the decision taken.
  - After asking for the player ID, game number (1, 2 or 3) and the new score, we walk through the DOM and change the score of the corresponding game for the chess player whose ID matches the former introduced ID. In addition, we will update the total score.
  - Then we (over)write the XML again.
- **Press 5 to "[optional] Generate HTML with chess master candidates via XSL"**
  - A chess player shall be considered a candidate for master if he has reached 3 points in the 3 games of the tournament.
  - Read the CSV and **create another XML** with the chess master candidates. Call the XML file: **candidates.xml**
  - Create another XSL from the existing one, changing the font foreground colour to red when listing the chess players.
  - Then, apply this new XSL to get another HTML:
    - **Call the HTML file**: candidates.html
    - **Call the XSL file**: candidates.xsl

Suggested XSL pattern

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:template match="/">
        <html>
            <head>
                <style type="text/css">
                    table.tfmt {
                    border: 1px ;
                    }

                    td.colfmt {
                    border: 1px ;
                    background-color: white;
                    color: black;
                    text-align:center;
                    }

                    th {
                    background-color: #2E9AFE;
                    color: white;
                    }
                </style>
            </head>
            <body>
                <table class="tfmt">
                    <tr>
                        <th style="width:100px">Player ID:</th>
                        <th style="width:350px">Full Name:</th>
                        <th style="width:200px">Country:</th>
                        <th style="width:100px">Score game 1:</th>
                        <th style="width:100px">Score game 2:</th>
                        <th style="width:100px">Score game 3:</th>
                        <th style="width:100px">Score Total:</th>
                    </tr>
                    <xsl:for-each select="Skaters/Skater">
                    <tr>
                            <td class="colfmt">
                                <xsl:value-of select="playerid" />
                            </td>
                            <td class="colfmt">
                                <xsl:value-of select="playername" />
                            </td>
                            <td class="colfmt">
                                <xsl:value-of select="playercountry" />
                            </td>
                            <td class="colfmt">
                                <xsl:value-of select="playerscore1" />
                            </td>
                            <td class="colfmt">
                                <xsl:value-of select="playerscore2" />
                            </td>
                            <td class="colfmt">
                                <xsl:value-of select="playerscore3" />
                            </td>
                            <td class="colfmt">
                                <xsl:value-of select="playerscoretotal" />
                            </td>
                        </tr>
                    </xsl:for-each>
                </table>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

C. Learning Rubric

C.1. ADA skills

Minimum of 5 out of 10 required for this part.

These marks will be invalidated (mark 4) if you fail to defend your work in an oral interview.

ASSESSMENT ITEMS	ASSESSMENT ITEM DETAILS	SCORE (POINTS)
Classes and methods	Classes and methods are structured properly	0.75
Menu	The menu complies with the specifications	0.75
Get chess players and marks (to CSV & XML)	The ArrayList is populated properly The CSV & XML are generated properly	2.5
List chess players	Reads the XML properly (using DOM) Prints the data in a proper way	1.5
Generate HTML with all chess players via XSL	Reads the information from the XML Generates the HTML properly from a XSL	2.5
[optional] Modify the rating of a chess player's game		1
[optional] Generate HTML with chess master candidates via XSL		1

C.2. English skills

Mandatory to be COMPETENT to pass this part.

ASSESSMENT ITEMS	ASSESSMENT ITEM DETAILS	SCORE
Writing skills	Every method is described properly	COMPETENT/NOT COMPETENT
	A proper text is provided (within the code or in a text file) to describe the AT using THREE PARAGRAPHS	COMPETENT/NOT COMPETENT
Oral skills	Uses a vocabulary appropriate for the purpose	COMPETENT/NOT COMPETENT
	Shows fluency and confidence	COMPETENT/NOT COMPETENT
Comprehension skills		Accomplished since all materials are in English
Reading skills		Accomplished since all materials are in English