Exam Projects

Advanced Programming
Summer session June 2016

Details about how to submit the project

Deliver your project as a zip file named summer-exam-2016-USER-STUDENTID.zip (substitute 'USER' with your name and 'STUDENTID' with your student id number). In this zip file put the **full project** and a **short project description** where you list the **implemented 7 AP techniques**, i.e.

Technique	Short description (Java files & source code)
a) **JUnit Test Cases and Test Suite**	**I created tests to check if a user was created properly …** **TestUser.java Lines: 20-50** **…**
b) **…**	
c) **…**	
d) **…**	
e) **…**	
f) **…**	
g) **…**	

I must be able to compile and execute the program without any additional library that is not already included in the standard Java API. **If you really need special instructions** for compiling and running the project in **Eclipse**, then include a readme.txt file to your submission.

- Double check that one of your colleagues using a Mac with **Eclipse** can successfully load and execute your project without any problem.
- Do not put in your code absolute references to files that cannot be found when running your project in another computer (my computer).

Note that if I cannot run your project or the application does not find the required files (see the requirements) you will **fail the project**.

Upload your zip file to **ole.unibz.it** before the deadline: **June 10th, 2016 at noon.**

Student Code Ethics
Students are expected to maintain the highest standards of academic integrity. Work that is not of the student's own creation will receive no credit. Remember that you cannot give or receive unauthorized aid on any assignment, quiz, or exam. A student cannot use the ideas of another and declare it as his or her own. Students are required

to properly cite the original source of the ideas and information used in his or her work.

Application requirements

Implement a swing-awt-based "**AP Manager 2016"** application where a user can perform the tasks described in the following.

The implemented application must be displayed in the main window (frame). **Do not create an Applet.**

There must be a main window menu containing the following menu items:

- *Exit:* kills the application
- *About:* pops up a dialogue that tells something about the developer of the application (you) and some information about the application: number of classes, number of methods, total number of lines of code.

"**AP Manager 2016" Specification**

The goal of this application is to provide an "**AP Manager 2016" user interface** with the following features.

1. **Login:**
   1.1. When the program starts it shows a **login window** where the user has to enter a username and a password.
   1.2. All the user information has to be managed in a structured **users.xml** file. That file has to be loaded at the start of the application and updated on each user change.
   1.3. The application provides the possibility to **add a new user** with
      1.3.1. a **username**
         - Usernames can contain letters (a-z), numbers (0-9), dashes (-), underscores (_), apostrophes ('}, and periods (.).
         - Usernames can't contain an equal sign (=), brackets (<,>), plus sign (+), a comma (,), or more than one period (.) in a row.
      1.3.2. a **password**
         - Passwords can contain any combination of ASCII characters and must contain a minimum of 8 characters.
         - A password should be stored as encrypted password to the users.xml file. (Hint: http://howtodoinjava.com/security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/)
      1.3.3. a **user image**, i.e. as jpeg file
      1.3.4. **and user roles**
         - **read** data
         - **search** or **query** data
         - **add** data
         - **delete** data
         - **import** new data from a file

2. **Data Manager:**

2.1. The application has to provide a user friendly GUI. The graphical components used for this application can be chosen freely.

2.2. After the login the data (see **possible data sources**) stored in the file **data.xml is loaded**.

2.3. Depending on the user roles it is possible to

- **read** data
- **search** or **query data**

  The application has to provide a user-friendly way to search and query data.
- **add** data
- **delete** a data record
- **import** new data from a file

2.4. The system should provide **data information** like

2.4.1.1. File path to the data file

2.4.1.2. Total number of data records

2.4.1.3. Total number of changed records

2.4.1.4. Total number of deleted records

2.5. All changes to the data have to be updated in the **data.xml** file when closing the application.

3. **Possible data sources:**

3.1. **Invoice list** (company)

3.2. **Apartment list** (real estate agents)

3.3. **Restaurant list** (restaurants)

3.4. **Car list** (car seller)

3.5. **Flight list** (airport)

3.6. **News list** (i.e. RSS web feed - http://www.computerweekly.com/rss/Latest-IT-news.xml)

3.7. **Movies list** (personal)

3.8. **Music list** (artists, genres, activity type, title, album)

3.9. **Address book** (people, their addresses, phone numbers, relationships…)

3.10. **Personal budget** (personal incomes/outcomes)

Each data source has to contain **at least 50 records**.

4. **7 Advanced Programming techniques**

   The system has to implement **all 7 Advanced Programming techniques**:

   4.1. **JUnit Test Cases and Test Suite**

   Provide a **meaningful set of test classes** to cover the greatest part of the application functionalities.

   4.2. **Exception handling**

   Provide proper exception handling such that the application does not necessary terminate when an exception is called.

   4.3. **XML**

   4.4. **Generics & Collections**

   Use Generics to manage the data read from data files, i.e. Arraylist of User objects

   4.5. **Reflection & RTTI**

   4.6. **I/O & Streams**

   Hint: Reading data and writing to files

   4.7. **Regular expression**

   Hint: Data format check for username and password

5. **Javadoc Documentation**

   Provide a Javadoc documentation to describe the source code of your application.

6. **Optional requirements**

   Implement the following additional requirements:

   6.1. Export the visualized data as .html file

   6.2. Export the visualized data as .rtf file

Project evaluation

This project counts 50% of the total Advanced Programming mark.