Exam Project

Advanced Programming
Summer session June 2017

Details about how to submit the project

Each student submits her/his own project. Group work is not allowed.

Upload your project data as **a zip file** named AP_201617_PR_studentID.ZIP (substitute “studentID” with your student id number, i.e. AP_201617_PR_156243652431.ZIP) to the AP course page on **ole.unibz.it**.
In this zip file put:

1. the **fully functional Eclipse project (Eclipse Workspace)** , even if you use another IDE

   Important:
   - **Do not use extra libraries (except for JDBC drivers and JUnit)**
   - **Use Eclipse Mars**
   - **Use Java 1.8**
   - **Use Linux (Ubuntu 16.04 LTS https://www.ubuntu.com/download/desktop) to test your application**

2. the **project report as .pdf** (template will be provided in OLE) in which you list the **implemented 7 AP techniques**, e.g.

Technique	**Short description (Java files & source code)**
1. **JUnit Test Cases and Test Suite**	**I created tests to check if a user was created properly …** **TestUser.java Lines: 20-50** **…**
2. **…**	
3. **…**	
4. **…**	
5. **…**	
6. **…**	
7. **…**	

3. **JavaDoc folder** with your exported **Javadoc documentation as html files**

We must be able to compile and execute the program without any additional libraries that are not already included in the standard Java API. **If your project requires special instructions** for compiling and running the project in **Eclipse**, then include a readme.txt file to your submission.

We recommend to:

- Double check with one of your colleagues using the Linux OS with **Eclipse** that she/he can successfully load and execute your project.
- Do not put in your code absolute references to files that cannot be found when running your project in another computer (e.g. "c:\Users\Mickey\...").

Note that you will **fail the project**:

- if your application does **not run and function on Ubuntu 16.04 LTS** under **Eclipse Mars** and **Java 1.8** even if the code shows that you did all the 7 programming techniques

For date reference, as of now, the last lecture is scheduled for the 18. May 2017, the last lab for the 18/19. May 2017 and the exams are scheduled for the 14. July and 5. September 2017.

Upload your zip file to **ole.unibz.it** before the deadline.

If you will attend the exam on 14. July 2017, the deadline for the submission of the project is 9. June 2017.

If you will attend the exam on 5. September 2017, the deadline for the submission of the project is 15. August 2017.

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-based "**AP Collection Manager 2017"** 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).

"AP Collection Manager 2017" Specification

The goal of this application is to provide an "**AP Collection Manager 2017" 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.

Important: Ensure that there exists a standard user (user: **admin** – password: **admin**) in your application. This user will be used to test your application.

1.2. All the user information has to be managed in a structured **users** 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 **any combination** of letters (a-z) and numbers (0-9)
- Usernames can contain **only 1** dash (-), 1 underscore (_) or 1 period (.).
- **Minimum length** of a username is **5**

1.3.2. a **password**

- Passwords can contain any combination of ASCII characters and must contain a **minimum of 5 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

1.4. The application provides the possibility to **delete a user.**

2. **AP Collection Manager:**

2.1. The application has to provide a **user friendly GUI** (i.e. SWING). 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 **data** file **is loaded**.

2.3. The application allows to

- **read** data from a data source
- **write/update** a data source
- **search** or **query data**

  The application has to provide a user-friendly way to search and query the content of each attribute of your data.
- **add** data records
- **delete** a data record
- **import/add data** from another data source

  It should be possible to import data from other files. This data has to be added to your application data. Afterwards all the visualized data should be stored in the **data** file**.**

2.4. The system should provide **data information (Summary of data)** like

- File path to the data file
- Total number of data records

- Total number of added records
- Total number of deleted records

Notice: You do not have to store this data information in a file. It is enough to calculate it each time when you start your application or when you add/delete new records.

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

3. **Possible data sources:**

3.1. **Book list**

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

3.3. **Movie list** (personal)

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

3.5. **Invoice list** (company)

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

3.7. **Restaurant list** (restaurants)

3.8. **Flight list** (airport)

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

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

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

3.12. **Anything similar or your own data source**

Each data source has to contain **at least 50 records** and **minimum 5 attributes** (i.e. title, author, length, etc.)

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. **Logging using java.util.logging**

4.3. **Exception throwing**

Provide proper exception handling such that the application does not necessary terminate when an exception is called. In addition your application should throw your own exceptions.

4.4. **data I/O (files or JDBC or web)**

4.5. **Generics & Collections**

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

4.6. Javadoc documentation

Provide a Javadoc documentation to describe the source code of your application. (**minimum one sentence** for function/class description and **minimum one sentence** for parameter description)

4.7. Design patterns

You should specify **at least 3 design patterns** that are used within your application. You must implement the design patterns, not just use the ones provided by Java. For example, if you use the Iterator of a collection, this does not count as your implementation of a design pattern.