Learning Aims:

- Expand knowledge and experiences with TDD and pair programming
- Get to know and test Open Source development and adjustment processes

Outlook: After these tasks we will deal with Open Source development processes more theoretically. You will get to know other projects, analyze them and discuss scientific literature concerning Open Source.

Task 4-1: (Selection and implementation of a JabRef feature)

[If you want to participate in the Saros experiment, this task has to be conducted under the experiment's conditions.]

Select and implement one of the following JabRef features. For further information go to the JabRef Feature Request Tracker (http://sourceforge.net/tracker/?group_id=92314&atid=600309) and filter for the appropriate Feature ID (referred to as Artifact ID)

1. Feature: Automatically rename files - ID: 2899374
2. Feature: Add Citeseer-Search and Google Scholar to JabRef - ID: 2934181
3. Feature: Highlight new entries that were "forgotten" - ID: 3061822
4. Feature: Highlight searched word in "review" or "abstract" - ID: 3029358

Remember that most of the time there are various ways to implement. The challenge is to find and implement a reasonable solution. If possible, use the so-called ping-pong programming method while programming.

Be prepared to present your preliminary results to your fellow students in the tutorial on 2010-11-15 and your final results in the tutorial on 2010-11-22.

Remember the following:

- Presentation of the requirement, possibly via a presentation of a running JabRef.
- Brief presentation of additional information (also from third sources) necessary for the completion of the task.
- Which parts/classes/packets of JabRef are important for handling the request?
- Which challenges concerning TDD may possibly arise from the task? How can you meet those? How do you have to expand your TDD knowledge (e.g. testing of GUIs, use of mock objects)?
- How far have you progressed with the implementation?
- In which way did pair programming help you? Which experiences have you gained with the ping-pong method? E.g. when does refactoring happen while using the ping-pong method?
- Which project-specific difficulties concerning your role as JabRef developer did you meet (e.g. while gathering information)?

1 http://www.sticky minds.com/sitewide.asp?Function=edetail&ObjectType=COL&ObjectId=9101
http://fest.easytesting.org/swing/wiki/pmwiki.php
3 http://www.michaelminella.com/testing/the-concept-of-mocking.html
http://connextra.com/aboutUs/mockobjects.pdf
Advice:

- Deal *pragmatically* with new test practices and technologies and keep in mind the benefit-cost ratio. This also means being able to justify your decisions and considerations!
- In case of doubts, contact the lecture's mailing list or directly the JabRef project (JabRef users' mailing list) when feeling confident enough.

Task 4-2: (Patch compiling and presentation)

1. Compile a patch containing your extensions until Wednesday, **2010-11-18, 12pm**.

   *You compile a patch in Eclipse with a right-click on your project > Team > Create Patch... (possibly use the Eclipse help function)*

   **Pay attention that your test cases are included!**

2. Present (»serve«) your extension in such a way that other developers (your fellow students) feel like examining your patch and may understand it easily.

   a. Create a blog entry in the Blackboard blog “Blog zur VL Softwareprozesse”
      
      - describing your feature and
      - **presenting briefly and coherently what the patch does and which changes and extensions to the code you carried out.**
      - **Please do not forget to add the information whether you are in Monday’s first or second tutorial.**

   b. Append (upload) the patch to your blog-post.
Practice sheet 4 – due on 2010-11-15 and 2010-11-22

Task 4-3: (Review process)

Now work alone and not in pairs. Review and evaluate one of the other developers’ patches until Monday, 2010-11-22, 8am so that the best patches may later be improved and presented to the right JabRef developers team.

- Evaluate just patches from your fellow students in your tutorial.
- Leave your review results as a comment on the blog-post.
- You may refer to each other.

Evaluate the patch reviewed in the following way:

- Approval (+1 point) and reason for approval.
- Restricted approval (0 points) and constructive criticism (concrete suggestions for improvement)
- Rejection (-1 point) and reason for rejection.

Criteria for your decision should be amongst others:

- Does the patch realize the aspired feature?
- Is the solution sensible, consistent and easily traceable?
- Is the source code clean and comprehensible?

Advice:

- Continue working on the SVN trunk of JabRef.
- Even if you know the solution: Do not yet send your solution to the project.