Assignment 1: Getting Familiar with jEdit and PDFsam

Due date: 02/20/2020

1 Introduction

In this assignment, you will get familiar with two open-source software projects: *jEdit*[1], a free and open-source mature text editor; and *PDFsam*[2], a free and open-source application to manage PDF files. You will use both systems and check their source code to gain knowledge on the features they provide, their internal and external design, their coding style, etc.

When preparing this assignment, consider that the goals of this activity are:

i. To introduce students to active, medium-sized, open-source software systems;
ii. To provide students with the opportunity to become familiar with software systems that will be used in upcoming assignments; and
iii. To introduce students to change request writing.

Remember that the assignments in this class are to be developed in pairs, so your first task is to find a teammate.

Read carefully the assignment before proceeding!

2 The surface

There are several ways to get familiar with the functionality provided by jEdit and PDFsam. You could, for example, check their websites and search for documents related to features and usage. A more hands-on approach is to download, install and set up the applications in your machine to give them a try as a user. The good news is that both applications provide **installers** for Windows, Linux and Mac OS, so this process should be quick and painless.

In the case of PDFsam, make sure you download the basic edition.

For each system, you are to execute as many features as possible. You should be at least an intermediate user of each system by the end of this phase of the assignment.

3 Beyond the surface

Now that you are familiar with the features provided by jEdit and PDFsam, you are to gain some knowledge on their internals. Specifically, you are to download their sources, deploy them locally, and navigate the code to get an insight on their design, main modules, dependencies, test suites and technologies, code style, etc.

The following instructions are a starting point to build and deploy jEdit and PDFsam in **macOS Catalina**. These instructions are based on the **Eclipse IDE**, but you could use any other IDE you are

[1] http://www.jedit.org/
[2] https://pdfsam.org/

familiar with, such as, *IntelliJ* or *NetBeans*. You will also need *git* to get the sources of the projects, as well as *maven* and *ant* to build them.

3.1 Downloading and deploying jEdit

1. Create a new repository in GitHub (see naming conventions in Section 5) and **import** the sources of jEdit version **5.5.0** (i.e., the latest stable version to date) by providing the URL:

   https://svn.code.sf.net/p/jedit/svn/jEdit/tags/jedit-5-5-0

2. Clone the repository in your machine.
3. Checkout the head of the repository in your Eclipse Workspace.
4. Start Eclipse and open the Java Perspective (if not already open).
5. Select **File » New Java Project**.
6. Enter jEdit-5-5-0 (or the name of the folder containing the code) in the Project name input field.
7. Click on the **Finish** button.
8. For building and running jEdit, execute the task: **ant retrieve build run**

By now, the jEdit Java project should present some compilation errors. Some of them can be solved by following the "Tips for Eclipse/NetBeans/IDE users" step in the README file.

Most open source projects provide a **README file** with instructions and tips to build and run their sources. jEdit is not the exception. Make sure you always read such a file.

3.2 Downloading and deploying PDFsam

PDFsam provides a detailed guide to build and run the project[3]. Read it before following the next steps.

1. Fork or duplicate the PDFsam repository[4] in your GitHub account (see naming conventions in Section 5).
2. Clone the repository in your machine.
3. Checkout the tag **v4.0.5** (i.e., the latest stable version to date) in your Eclipse Workspace.
4. Start Eclipse and open the Java Perspective (if not already open).
5. Select **File » Import » Existing Maven Projects** and click **Next**.
6. Provide the path of the directory containing the sources of PDFsam as Root Directory.
7. Select all the pom files.
8. Click on the **Finish** button.
9. Instructions to build and run PDFsam are provided in its wiki.

[3] https://github.com/torakiki/pdfsam/wiki/Build-and-run
[4] https://github.com/torakiki/pdfsam

4 Deliverables

The deliverable of this assignment is a report that **for each software system** describes:

1. Its overall internal design. You could, for example, provide an overview of the purpose of projects and packages composing the system.
2. At least one bug (i.e., unexpected behavior) that you experienced while using the application. Provide as many details as possible. At the minimum, you should include: observed behavior, expected behavior and steps to reproduce. Stack traces (if they exist) and screenshots are recommended.
3. At least two new features that you would like to see implemented in the application. Once again, provide as many details as possible. Include adapted screenshots if they make your feature request clearer.

The reports must be submitted through Canvas before class (i.e., before 12:30pm).

Wondering what makes a good bug report? You're not the first one! Check Zimmermann et al.'s work [Zimmermann'10] to find out. By the way, there is a lot of research on bug reports and their management!

5 Notes

- Name the repository after your team name as follows: **cs515-001-s20-teamname-projectname**, where **teamname** is the name that you have chosen for your team, and **projectname** is either jedit or pdfsam.
- Name your report as follows: **cs515-001-teamname**.
- Proofread carefully your report.
- You will not receive credit for this assignment if:
  - You do not submit the report.
  - The file you submit cannot be opened.
- You will get points off if:
  - Your report contains typos or grammar errors.
  - Your report does not fit the brief.
  - Your file does not follow the specified naming conventions.

6 References

[Zimmermann'10] Zimmermann, Thomas, Rahul Premraj, Nicolas Bettenburg, Sascha Just, Adrian Schroter, and Cathrin Weiss. "What makes a good bug report?" *IEEE Transactions on Software Engineering 36*, no. 5 (2010): 618-643.