# User manual for ClonSAGe tool

ClonSAGe tool provides information about simple clones and structural clones of the system, extracts genealogies for both categories of clones and visualizes them. It also extracts historical information of clone refactorings in previous versions and visualizes them. ClonSAGe is implemented in .Net framework.

**Download and Run**

ClonSAGe tool is available at <https://github.com/j-kanwal/ClonSAGe>. You can download it and just click on the cloneGen.exe file and IDE of ClonSAGe will be open. It takes input files from the current directory where tool exe is placed. A sample of input files are also provided in the link.

**Input of ClonSAGe**

Input files for *CloneGenealogyView* consist of clone information (simple clones and structural clones) for each version as input. Format of input files are given in the link.

Input files for *CloneRefactoringView* consist of clone information (simple clones) and refactoring information. Sample of input files are also provided in the link.

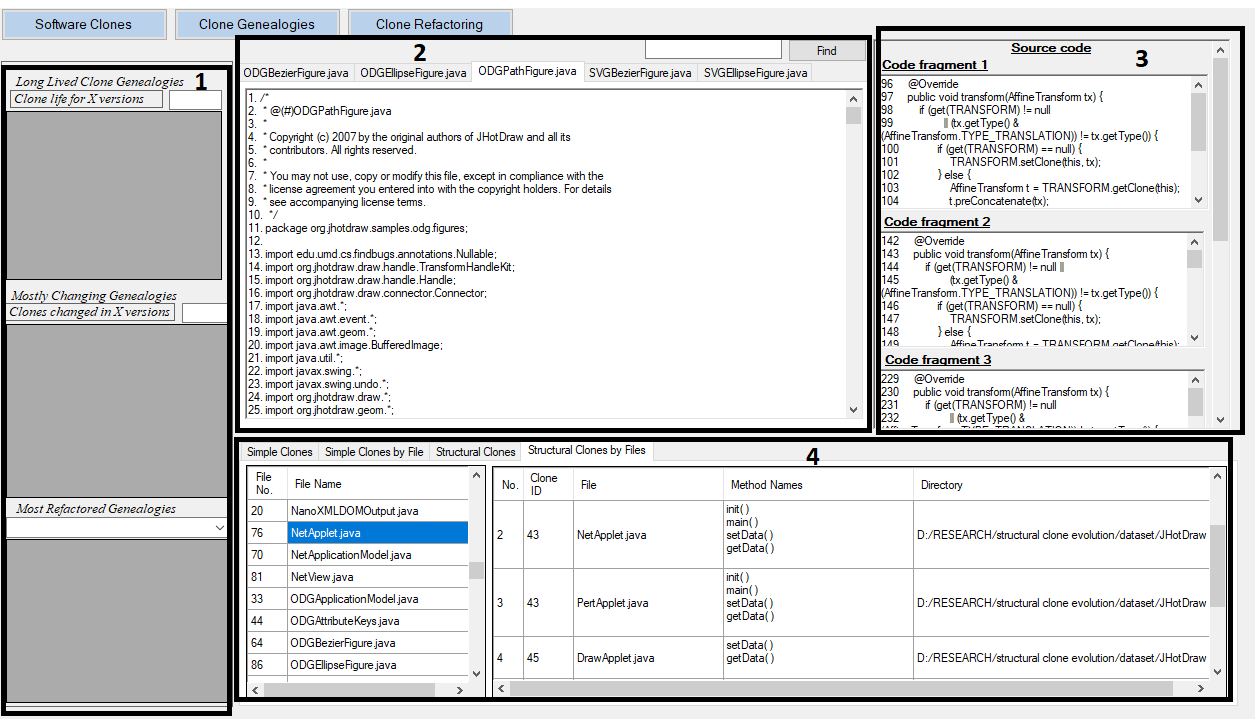


Figure : Clone class view

**Tool IDE**

Tool IDE consists of three views clonesView, GenealogyView and RefactoingView. Detail of each view is given in the following.

**ClonesView:** shows all the detected simple clones and structural clones in a software. To make the clone detection results understandable, this view presents them in tables. For both types of clones, this view represents clone classes in a table as shown in Figure 6.1 so that developers can investigate each clone class in detail. It also shows clones according to their file names so that developers can locate all the clones present in a file. Each row represents a clone class i.e. its clone ID, number of instances, location of clones (i.e. start and end line numbers of a code fragment, method and file name where a clone fragment resides). Clone classes can be sorted by columns. Clicking on the clone IDs of a clone class, it shows the source code of all the clone instances of a clone class (marked ‘3’ in Figure 1). By clicking on the file name, it also opens the file in text editor (marked ‘2’ in Figure 1) where clone instance resides so that user can understand clones with respect to the context. In case of structural clones, it shows source code of all the files in text editors in multiple tabs because structural clones consist of multiple simple clone classes and showing source code of all clone instances becomes complicated.

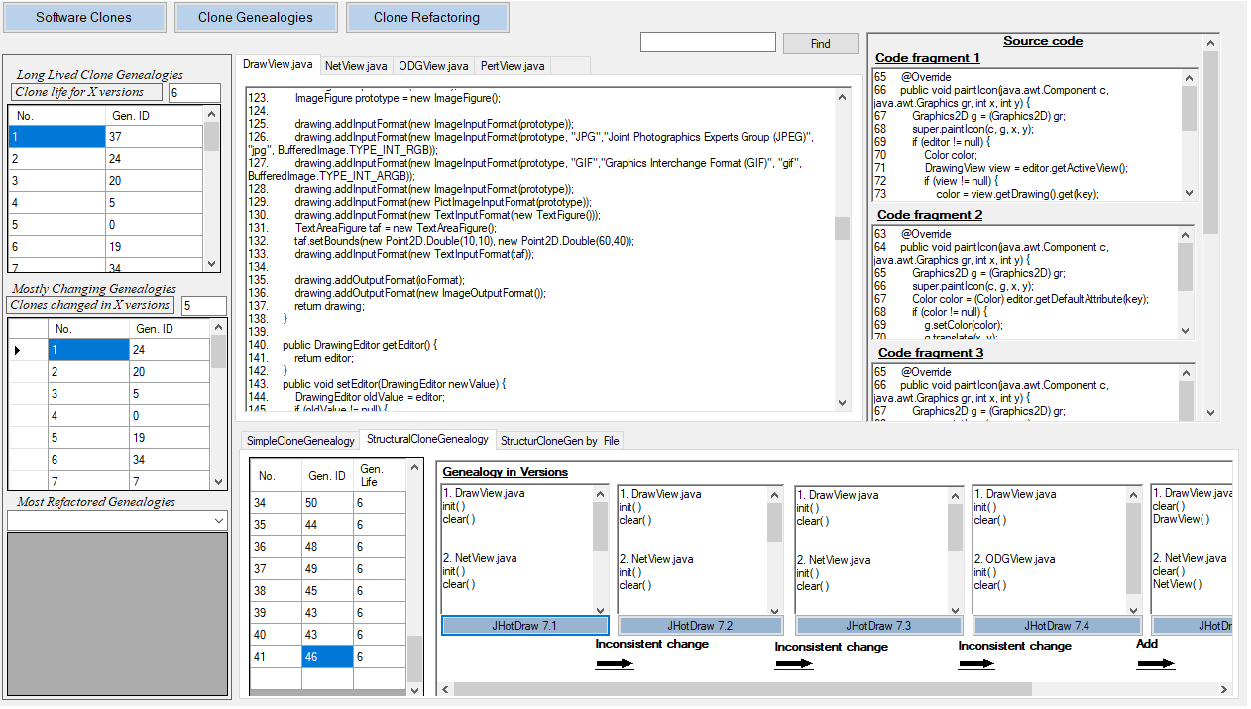


Figure : Clone genealogy View

**GenealogyView**: It shows all clone genealogies for simple clones and structural clones. It shows genealogy ID and lifetime for each clone genealogy in a table as shown in Figure 2. By clicking on a row, it shows the life line of the genealogy in versions.

For each genealogy, all instances of a clone class (e.g. method names and file name of each clone instance) are shown for every version. Clone evolution pattern of a clone class between every two versions is also presented. In this view developers can view the life line of each clone class and also what changes happened to a clone class in its lifetime.

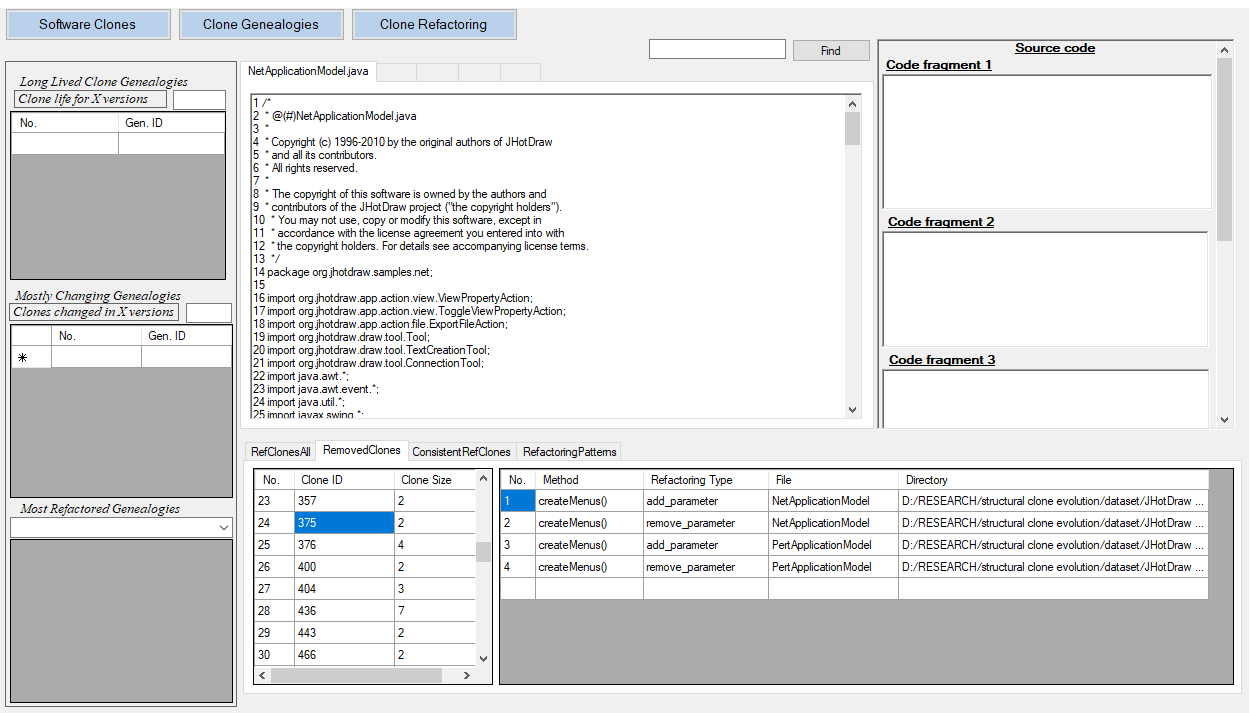


Figure : Clone refactoring view

**CloneRefactoringView**: It displays all the refactorings performed on the clones between any two versions as shown in Figure 3. It displays clone ID and clone class size for each refactored clone class in a table. By clicking on the clone ID, it displays method and file name of clone instances of a clone class, refactoring patterns applied on each clone instance and directory (file path) of that file. By clicking on the directory it opens the file in a text editor. User can search the relevant method in the file. Information of all clone classes that are removed as a result of refactoring and information of clone classes that are consistently refactored are also shown in tables. In another tab, frequently applied refactoring patterns are shown and frequency of each pattern is also shown.