**IDE and JRE**

Predominantly, the development of the application MeerkatFX is done using Netbeans IDE 8.1 on Windows and Linux. The application is run on Java version 1.8.0 build 45 and above.

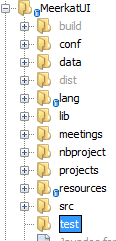
The project MeerkatFX from the developer’s perspective is divided into 3 projects:

* MeerkatUI
* MeerkatAPI
* MeerkatLogic

MeerkatUI

The MeerkatUI (or the Meerkat User Interface) project consists of all the components that are displayed on the screen and are available for the user to interact with the user. The UI components of this project are developed using the JavaFX components. Additional libraries can be added to it but as offline libraries. Offline libraries in the form of jars and other projects will not interrupt the application in case the online sources of the libraries are removed without notice.

Meerkat UI Folder Strucutres



The user defined the folders in the MeerkatUI Project are

* conf
* data
* lang
* lib
* src
* test

Folder ‘conf’:

The conf folder consists of the config files that are associate with the configuration of the Meerkat Application. The configuration files available are

conf.xml – Contains the mapping and related configuration for the application. This file is also explained in detail later

bizconf.xml – Contains the mapping between each class Ids and the classes to be invoked in the Business Layer of the application (MeerkatLogic Project). This file is explained in detail later.

settings.xml – The default values that are to be loaded everytime an application is started. The default values can be related to the UI features such as the vertex sizes, default colors, opacity, width and heights of various containers and Algorithm Parameters, the default values for various algorithms

conf.xml

The conf.xml file contains the configuration of the application that are related to the UI layer. The configurations are available inside the <app> tag as the root. Some of the configurations that are available in conf.xml are

<defaultlang> - The default Language file. The name should be saved as the same name of the file without file extension

<supportedfiles> - The imported format of the files that are supported by the user

Example of the format is

<filetype extension=".json" id=".json" type="0">value</filetype>

extension = the extension of the file

id = id that would be mapped with the readers of the file in the business layer

type = 0 for graphs and 1 for textual format

value = text to be shown in the user interface while opening a file

<exportformats> - The supported formats of the files in which a graph can be exported. That is, the writers should be available to export these formats

Example of the format is

<filetype extension=".meerkat" id=".meerkat" type="0">Meerkat Files</filetype>

extension = the extension of the file in which a graph can be exported

id = the mapping id of the extension to the writer

type = 0 for graphs and 1 for textual format

value = text to be shown in the user interface while saving it to a file

bizconf.xml

The bizconf.xml file contains the configuration of the application that are related to the Logic layer. The configurations are available inside the <app> tag as the root. Some of the configurations that are available in bizconf.xml are

<layouts> - The layouts that are displayed in the UI and their mapping to the respective class. The Ids are mapped with the Ids that are given in a <Language>.xml file (explained later).

<communityminings> - The mapping between the id and the respective classes that are to be invoked for executing community mining algorithms

<metrics> - The mapping between the id and the respective classes that are to be invoked for computing various metrics on the graph

<readers> - The mapping between the id and the respective classes that are to be invoked for reading from a file

<writers> - The mapping between the id and the respective classes that are to be invoked for writing the graphs/text to a file

Folder ‘lang’ :

This folder consists of the files that are specific to the Languages. Since MeerkatFX is an application where the language is extracted from different xml files, this is the folder that contains the all the languages that are available for the user to select from. Each language is an XML file with all the tags being mandatory for all languages. If any tag is missing, then the application would not start and close automatically. All the tags are to be in a single xml file and in the same format. Some of the examples of language files are “English.xml”, “Francais.xml” and “German.xml”. The selected language file will be referred to as <Language>.xml file.

<Language.xml> file

The <Language.xml> is a one file where all the text that would be shown on the user interface of MeerkatFX is available. It also serves to use the plug-in modules of the application (Plug-in Modules are explained later). Any additional information added to this file requires changes in other places. The most important thing after adding a feature through this xml is to add the specific tags in code in one of the config code files and then parse those files.

Folder ‘data’:

Any data files that form the intermediate results or needs to access the file are written or fetched from this folder.

Folder ‘lib’ :

The library folder that contains all the external offline libraries that must be included in the UI package for it compile and run successfully without breaking.

Folder ‘src’:

All the source code is from this folder. Although this folder is created by the Netbeans IDE, the src folders contains the custom code written for the application. All the java packages go within this folder.

Folder ‘test’:

If there are any test cases written to run the application under specific scenarios or to run the application to test the application partially on a specific module, the test cases have to be put into this folder called test.

**Overview of the Meerkat Application**

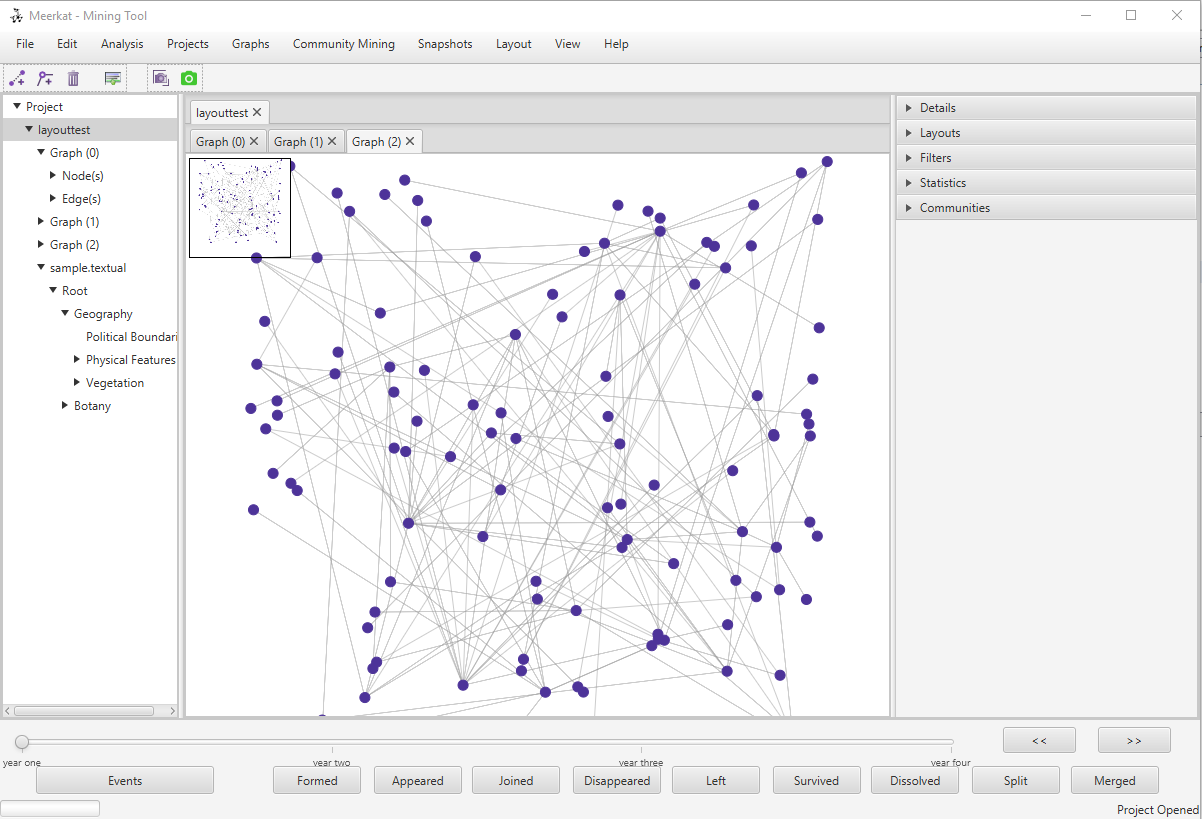
Project Tabs Palette

Tools Ribbon

Title Bar

Graph Tabs Palette

Menu Bar



Thread Tree Panel

Minimap

Accordion Pane

Status Bar

Event Analyzer

Graph Canvas

The various panels of Meerkat FX are explained below

Title Bar: The title of the Main Window of the application. The title to be displayed is language dependent and the title to be displayed is retrieved from the file <Language>.xml.

MenuBar : The Menu Bar is a set of Menu and Menu Items. Menu Items are extracted from <Language>.xml.

Example :

<MenuItem class="MenuItemNewProject" icon="file:resources/icons/new.png" mnemonicParsing="true" text="\_New Project" check="false" shortcut="n,ctrl,shift" disabled="0" SeparatorCount="0" />

* class – The class which contains the implementation of the ‘Click’ method that would be called on selecting the specific MenuItem
* icon – The file path of the icon that is displayed in the MenuItem. The location of the icon are relative to the project
* mnemonicParsing – Mnemonic Parsing is the mnemonics that can be used to invoke the specific MenuItem
* text – The text to be displayed in the MenuItem
* check – If the MenuItem is supposed to be checked MenuItem or just a MenuItem
* shortcut – The keyboard shortcut that can be used to invoke the MenuItem
* disabled – the level at which the MenuItem can be disabled. Currently there are 4 levels
  + level 0 – This option will never be disabled
  + level 1 – This option will be disabled when there are no projects open
  + level 2 – This option will be disabled when there are no graphs open in a graph
  + level 3 – This option will be disabled when there are no textual graphs open in a graph
* SeparatorCount – the number of lines (separators) that should be drawn in the menu after this Menu Item

Tools Ribbon:

The tools ribbon is a panel of shortcuts with different tool boxes. Currently there are two toolboxes, Snapshot Toolbox and Graph Editing Toolbox

Thread Tree:

The thread is a tree representation of the projects that are open in the application. It is also a useful navigation tool between projects. Each project can expanded into all the graphs and textual networks that have been added to the project. Each Graph can again be expanded into two sub sections, Nodes and Edges where all the Vertices of the Graph and Edges of the graph are listed respectively. The Textual Network is in a hierarchical structure and the hierarchy is shown by expanding each level of the textual network in the thread tree.

Canvas:

Canvas is a area where the graph is drawn. The Canvas is implemented in multiple layers and is drawn as a JavaFX Group. Each graph can have a set of Vertices and Edges

Project Tab Panel

The Project Tab Panel is a tabulated panel with a single row of tabs, with each tab representing a Project. Selecting a project tab shows the tab panel of all the graphs that are a part of the selected project and the graphs of other projects are hidden from the current view. The title of the Project’s Tab is the name of the selected project.

Graph Tab Panel

Graph Tab Panel is a tabulated panel with a single row of tabs, with each tab representing a Graph. The Graph shown is a set of Nodes and Edges that can be visualized. The title of the Graph’s Tab is the title of the Graph. When selecting the Textual network from the Thread Tree View in the current project, the last selected visible tab would be selected. This Graph is defined using the Graph Tab ID.

Accordion Tabs

The Accordion Tabs consists of 5 different panels as follows

1. Details Tab
2. Layouts Tab
3. Filters Tab
4. Statistics Tab
5. Communities Tab

Details Tab:

The Details Tab consists of the basic details about the graph that is selected. The selected graph is the graph that is currently displayed on the Graph Tab Panel. The Details panel consists of information in two forms. The first set is computed as soon as the graph is loaded and changed whenever there is a change in the graph. The second set is computed on demand. There is corresponding “Run” button that for each measure which is computed on pressing the run button and on demand. Any changes made to the graph are not reflected immediately and have to requested once again to get the measure of the modified graph.

Layout Tab:

The layout Tabs are a set of Radio Buttons that would run the corresponding Layout algorithms on the current graph. The Layout algorithms are retrieved based on the language.xml file and more alogirhtms can be added/removed by editing the language.xml file. The layouts are basically provided into 3 different groups – standard layouts, metric layouts and community layouts. The metric layouts should invoke the bulls eye layout and each metric layout can be included in

Icons to be done

1. Enable Visualization – Just the opposite of disable visualization without the cross line C:\Users\Talat-AICML\Documents\NetBeansProjects\MeerkatUI\resources\icons\dissablevisualization.png
2. Tool top for Vertices
3. Exit the application
4. Hide/Show Minimap