Software Development Frameworks

Dimitrios Simopoulos

Department of Mathematics
University of the Aegean

Dissertation

Structure

- Software Development Frameworks, Platforms and Applications
- Application Requirements
- Introduction to Social Network and Graph Theory
- Development of application, based on the Swing library
- Development of application , based on the NetBeans Platform
- Conclusions

Introduction to Software Development Frameworks

Some of the highest priorities of an application are usually its extensibility, its maintenance and its internal reusability.
The last one is needed to avoid re-designing and re-implementing the same components.

Software Development Frameworks

A *Software Development Framework* plays a significant role in the development of an application. Because of that, the programmer is able to customize the general functions of a software, so that it can fulfill all the desired requirements
The purpose of such a framework is the optimization of the efficiency for creating new softwares.

Features of a Software Development Framework

The Software Development Frameworks contain the following discrete features:

Features of a Software Development Framework

The Software Development Frameworks contain the following discrete features:

- Extensibility

Features of a Software Development Framework

The Software Development Frameworks contain the following discrete features:

- Extensibility
- Default Behavior

Features of a Software Development Framework

The Software Development Frameworks contain the following discrete features:

- Extensibility
- Default Behavior
- Inversion of Control

Features of a Software Development Framework

The Software Development Frameworks contain the following discrete features:

- Extensibility
- Default Behavior
- Inversion of Control
- Non-editable Code

What is a Platform

It is well known that almost every application is based on a third application, like the operating systems and the database management softwares.
Anything being executed from the software, contributes to an application platform. By default, an application platform provides application services.

Platforms and Applications

The platform provides a structure for one or more applications (which provides a view to the final user). That is what facilitates the programmer to develop the application. For example, the definition of the "*customer*" differs radically between a platform and an application:

- For the application, a customer is defined as the end-user or the buyer.
- For the platform, it is the programmer who has to fulfill all the desired requirements of the developed application.

Social Network Application Requirements

The way that both of the applications have been developed, is different. And that, for specific reasons for examining different abilities and mechanisms that each one provides.It is required for both of them the depiction of the users, of their personal information and of their friendships.

Basic Concepts of Social Networks

Social Networking is the gathering or participation of people in specific groups. The social networks are defined as a group of actors (people, organizations or other social groups) and a group of relationships (friendships, bonds, money transactions) between them.

Basic Concepts of Social Networks

The actors are connected with social “ties”. The range and the type of of those ties can be really extensive. A tie is defined as a creation of a connection between a couple of actors.
A *social network* consists of one or more (finite) actor groups and of one or more ties, which connect those actors between them.

Social Network Representation

One of the most useful ways of representing a network, is through the *Graph Theory*. In Mathematics a graph consists of nodes, some of whom (possibly all of them) are connected between them. The connections (links) between nodes can be either bows or lines. That depends on the direction of the connections between those graph nodes.

The two implementations

Both of the implementations were based on the structure and the principles of a social network. There are users, who act as ‘nodes’ of a social network. They have to be connected with *friendship connections*.Such a connection is obviously bidirectional. But except of the logic of an application like that, there is also the graphical user interface. In both applications the depiction of the users, of their personal information and of their friends is mandatory.

Implementation with the basic classes of Java

The *Java* language is an object-oriented language and supports all the features of the object-oriented philosophy. It is also an interpreted language, which means that the Java compiler does not produce executable code but a bytecode, which cannot be executed alone. Through an appropriate Java interpreter, the code can be executed on any device.

The Java Swing graphical library

The Java Swing makes up a group of classes, which implement components that can be found on windowed applications, such as:

- windows
- menus
- dialog boxes
- buttons
- text fields

The Java Swing graphical library

The Swing provides a wide variety of components. For each one, there is the corresponding class. Some usual components are the following:

- Buttons (JButton class)
- Text labels (JLabel class)
- Text fields (JTextField class)
- Selecting-command menus (JMenu classes and JMenuItems)
- Text Areas, where also rich text can be inserted (JTextArea class)

Event Handling

The role of a user interface of an application is to show data and to execute commands as response to the application user, like:

- Press a button from the user
- Select an item from a menu
- Choose/delete some elements from a list or a matrix.

Event Handling

The sequence of the events/messages of the program after the button press, is the following:

Event Handling

The sequence of the events/messages of the program after the button press, is the following:

- The user presses the button.

Event Handling

The sequence of the events/messages of the program after the button press, is the following:

- The user presses the button.
- The instance of the button activates an event, which is represented by an ActionEvent.

Event Handling

The sequence of the events/messages of the program after the button press, is the following:

- The user presses the button.
- The instance of the button activates an event, which is represented by an ActionEvent.
- The button instance sends the above message to all the (listeners) of the specific message (ActionListeners), which is connected with that instance. This is done through the method's actionPerformed(ActionEvent e) call on every ActionListener.

Event Handling

The sequence of the events/messages of the program after the button press, is the following:

- The user presses the button.
- The instance of the button activates an event, which is represented by an ActionEvent.
- The button instance sends the above message to all the (listeners) of the specific message (ActionListeners), which is connected with that instance. This is done through the method's actionPerformed(ActionEvent e) call on every ActionListener.
- The message receiver (listener) implements the interface ActionListener and executed the appropriate code, through appropriate implementation of the actionPerformed method.

Depiction of the graphical application

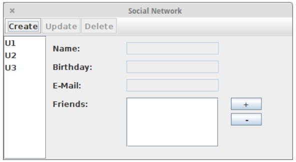
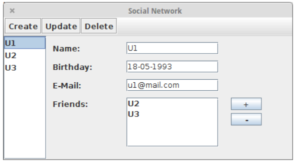


Figure: Main display of the application.



Name: U1
Birthday: 18-05-1993
E-Mail: u1@mail.com
Friends: U2
U3

Figure: Windowed application view after choosing a user.

Description of the first application

The graphical depiction of the application is based on the Swing toolbar. The main display contains 3 areas:

- The toolbar with the choices, like creating a new user, changing the personal data of the current user and deleting a chosen user.
- The list, which contains all the available users (on the left of the display).
- The view area of the personal data of the selected user, providing the ability of adding/removing a friend.

Adding a friend

The user can add or remove friends with the "+"/"-" buttons, after a relevant check from the available users. After that, the appropriate window pops up.

Adding a friend

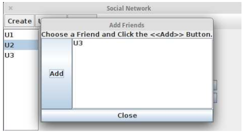


Figure: View of a pop up window, showing all the available choices for selecting a user as a friend.

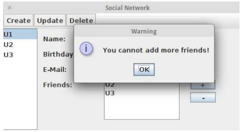
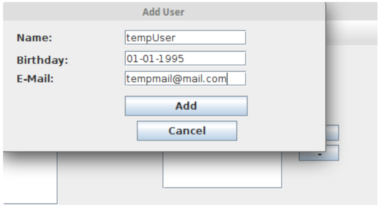


Figure: View of a pop up window, if there are not available users.

Adding a new user

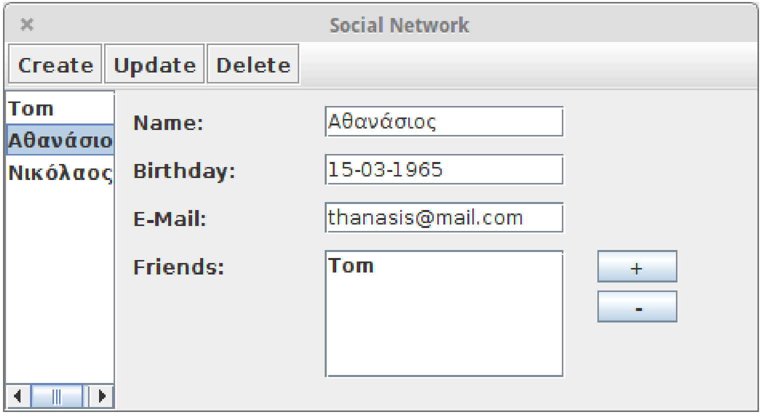
On the upper-side area, there is a toolbar. Through this, three choices are given to the application user. To create or delete a user, but also to edit the personal information of a selected user. The new window that pops up when someone presses the "Create" button, is collocated.



Name:	tempUser
Birthday:	01-01-1995
E-Mail:	tempmail@mail.com

Figure: Adding a new user.

Adding a new user



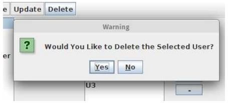
Tom
Αθανάσιο
Νικόλαος

Name: Αθανάσιος
Birthday: 15-03-1965
E-Mail: thanasis@mail.com
Friends: Tom

Figure: Main display after adding a user.

Updating and Deleting Users

Similarly, with the appropriate buttons the application user can update the personal data of a user. After pressing the “Update” button, the newly replaced data are stored and the application gets automatically updated.
In case of deleting a user, after a warning message, not only the selected user but also all of his friendships with other users are deleted.



Would You Like to Delete the Selected User?

Figure: Warning message of a user deletion.

Rich Client

In a server-client architecture, the definition *rich client* is used for clients, where the data edit takes place mainly in the client's side.
The client provides also to graphical user interface (GUI). Usually the rich clients are applications that are extendable through plug-ins and modules. Because of that, rich clients are able to solve one or more problems.

Rich Client

Below there is a brief list of the rich client features:

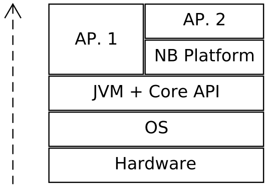
- Flexible and modular application architecture
- Platform independence
- Adaptability to the end user
- Simplified updating of the client

Rich Client Platform

A rich client platform is an application lifecycle environment, a basis for desktop applications. Most desktop applications have similar features, such as menus, toolbars, status bars, progress visualizations, data displays, customization settings, the saving and loading of user-specific data and configurations, splash screens, about boxes, internationalization, help systems, and so on. For these and other typical client application features, a rich client platform provides a framework with which the features can quickly and simply be put together.

NetBeans Platform

The NetBeans Platform is a general framework offering the ability of developing any application, which is based on Swing. It's about a powerful tool that helps the programmer to develop demanding apps with high requirements.
The main *benefit* is that, because of its operational mode, there is no need for each component and for their communication to be developed from scratch.

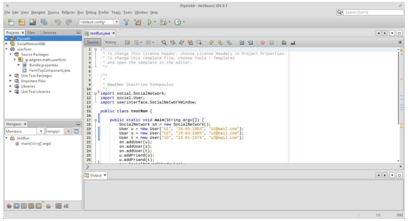


NetBeans Modules

A *module* is a collection of functionally-related classes together with a description of the interfaces that the module exposes, as well as a description of the other modules that it needs in order to function.

NetBeans IDE

The NetBeans IDE is a very good example of a modular rich client application. The functionality and the characteristics of an IDE, such as its Java language support or the code editor, are created in the form of modules on top of the NetBeans Platform.



```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Dimitrios Simopoulos
 */
import social.SocialNetwork;
import social.User;
import userinterface.SocialNetworkWindow;

public class testRun {

    public static void main(String argv[]) {
        SocialNetwork sn = new SocialNetwork();
        User u = new User("U1", "18-05-1993", "u1@mail.com");
        User s = new User("U2", "15-03-1965", "u2@mail.com");
        User t = new User("U3", "22-01-1974", "u3@mail.com");
        sn.addUser(u);
        sn.addUser(s);
        sn.addUser(t);
        u.addFriend(s);
        u.addFriend(t);
```

The Lookup concept

The Lookup is a Map, with Class objects as keys and instances of the Class objects as values.
The main idea behind the Lookup is decoupling components. That means, letting modules communicate with each other, which plays an important role in component-based systems, such as applications based on the NetBeans Platform.

Implementation with NetBeans Platform

The platform divides the display to default areas.

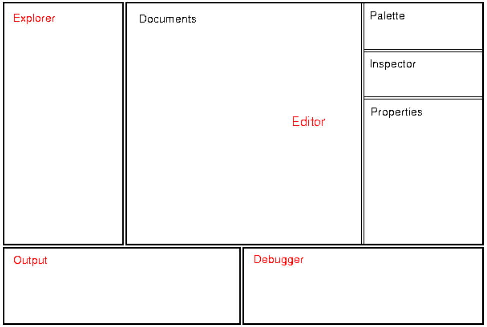


Figure: Depiction of the display areas with NetBeans Platform.

Depiction of the second implementation

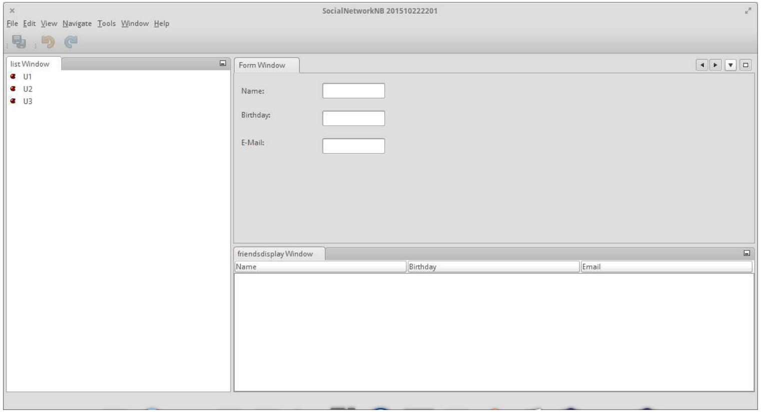


Figure: The main display of the second implementation.

The Structure of the Application Modules

Under these, three independent modules were implemented and they are named as:

1. explorerlist
2. friendsdisplay
3. userform

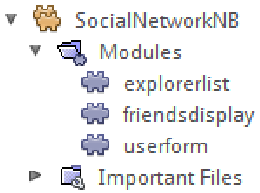


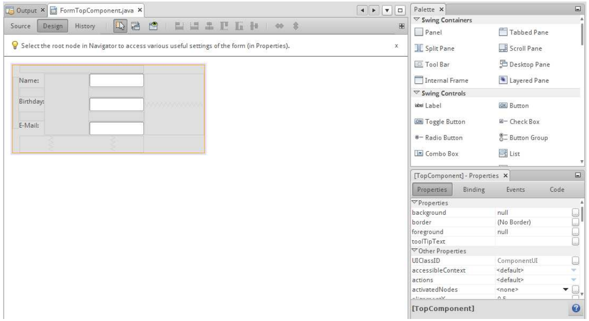
Figure: The structure of the application modules in NetBeans Platform

The usage of the Lookup concept in the application

In the explorerlist module, a *Lookup Provider* is created, which is responsible to send data (like the user choice through the mouse click) to the other modules.
On the other hand, the *friendsdisplay* and the *userform* modules embody a *Lookup Listener* mechanism. They receive, in that way, the necessary data that each one of them needs. After that, through proper code the application display gets automatically refreshed.

Design of the modules and the Lookup concept

The userform module is created with automatic visual programming tools. The personal data of the selected user are stated there. So, a Lookup concept is implemented there for the data refresh. It's about a Lookup Listener.



Select the root node in Navigator to access various useful settings of the form (in Properties).

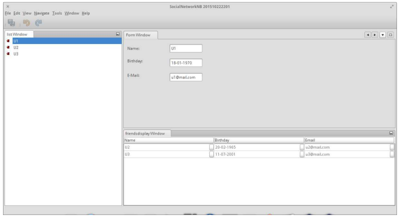
Properties	
background	null
border	(No Border)
foreground	null
toolTipText	
Other Properties	
UIClassID	ComponentUI
accessibleContext	<default>
actions	<default>
activatedNodes	<none>

Design of the modules and the Lookup concept

The userform and the friendsdisplay modules are implemented with the help of the built-in wizard. That wizard determines also the display area of each module. The basic is the successful communication between all the modules.
A Lookup Provider is inserted into the explorerlist module to provide data, related to the user choice. Similarly, a Lookup Listener is added to the friendsdisplay module, to show the friendships of the selected user.

Depiction of the second implementation

The construction of that window encloses the following. There are three text labels, named *Name*, *Birthday* and *E-Mail*. On the right side of each label, there is an appropriate field where the personal data are displayed. Of course, the function of that module is similar to that one of the friendsdisplay module.


SocialNetworkNB 201510222201
File Edit View Navigate Tools Window Help
list Window
U1
U2
U3
Form Window
friendsdisplay Window

Name: U1

Birthday: 18-01-1970

E-Mail: u1@mail.com

Name	Birthday	Email
U2	20-02-1965	u2@mail.com
U3	11-07-2001	u3@mail.com

Figure: Choosing a user from the list on the left.

Conclusions

According to the first implementation, it is easy to learn how to develop an application, but its difficulty lies in how the different application components can communicate between them. And that is needed, for its proper function.
On the other hand, concerning the NetBeans Platform, it is a really time consuming procedure to learn from scratch how to use it. But the offered benefit is the quick implementation, not only for simple but also for complex applications, providing easier maintenance and future modification.