**Abstract:**

Developers in other platform like VB, ASP.NET do not have to worry about their layout. But for Java programmer, he/she must add layout code for each component, which will be very much cumbersome in big projects. Though there is software available called NetBeans through which developer can easily design forms using drag and drop method. But the code generated by NetBeans is not reusable as the code is generated in ad-hoc manner. Because of this IT industries don’t use NetBeans for GUI Design. This project provides Java Developers libraries for constructing layouts in Java. It will help user to design various forms for their application without worrying much about layouts. Also the code or the libraries of this project can be reused anywhere anytime just by installing single jar file of this application.

Objective of this project is to make Java layouts more simple and user friendly. User/developer should not get confused about code. He/she should be able focus on content without bothering much about presentation. This will reduce development time and as well as code complexity. So it is useful in developing industry compliant Java UI (User Interface) development.

**Introduction**

Java is purely object oriented language. This makes clear that it will also have object oriented features like previous languages like C++ i.e. Inheritance , Polymorphism , Data abstraction, Encapsulation etc. Also it has various new features listed below:

1. Compiled and Interpreted
2. Platform-Independent and Portable
3. Robust and Secure
4. Distributed
5. Familiar and Simple
6. Multithreaded and Interactive
7. High Performance
8. Dynamic and Extensible

Java found everywhere – on mobile phones, desktop computers, Blue ray disk players, set top boxes and even in your car. So scope java is more than all other languages.

Java API provides number of classes grouped into different packages according to their functionality. Each packages have their own benefit and purpose.

**Objective:**

Objective of this project is to make Java layouts more simple and user friendly. User/developer should not get confused about code. He/she should be able focus on content without bothering much about presentation. This will reduce development time and as well as code complexity. So it is useful in developing industry compliant Java UI (User Interface) development.

**Scope:**

Our java project is specially and only for “Developer”. Therefore it can be used by any Java Developer. As we all know Java is mostly used at Industry level. In Industries various IDE’s () for quick programming.eg.Net Beans , Eclipse , Jcreator. As NetBeans has various problems. So mostly Eclipse IDE is used. Our Project (JAR File) Can be used with Eclipse IDE to reduce much lines of code.

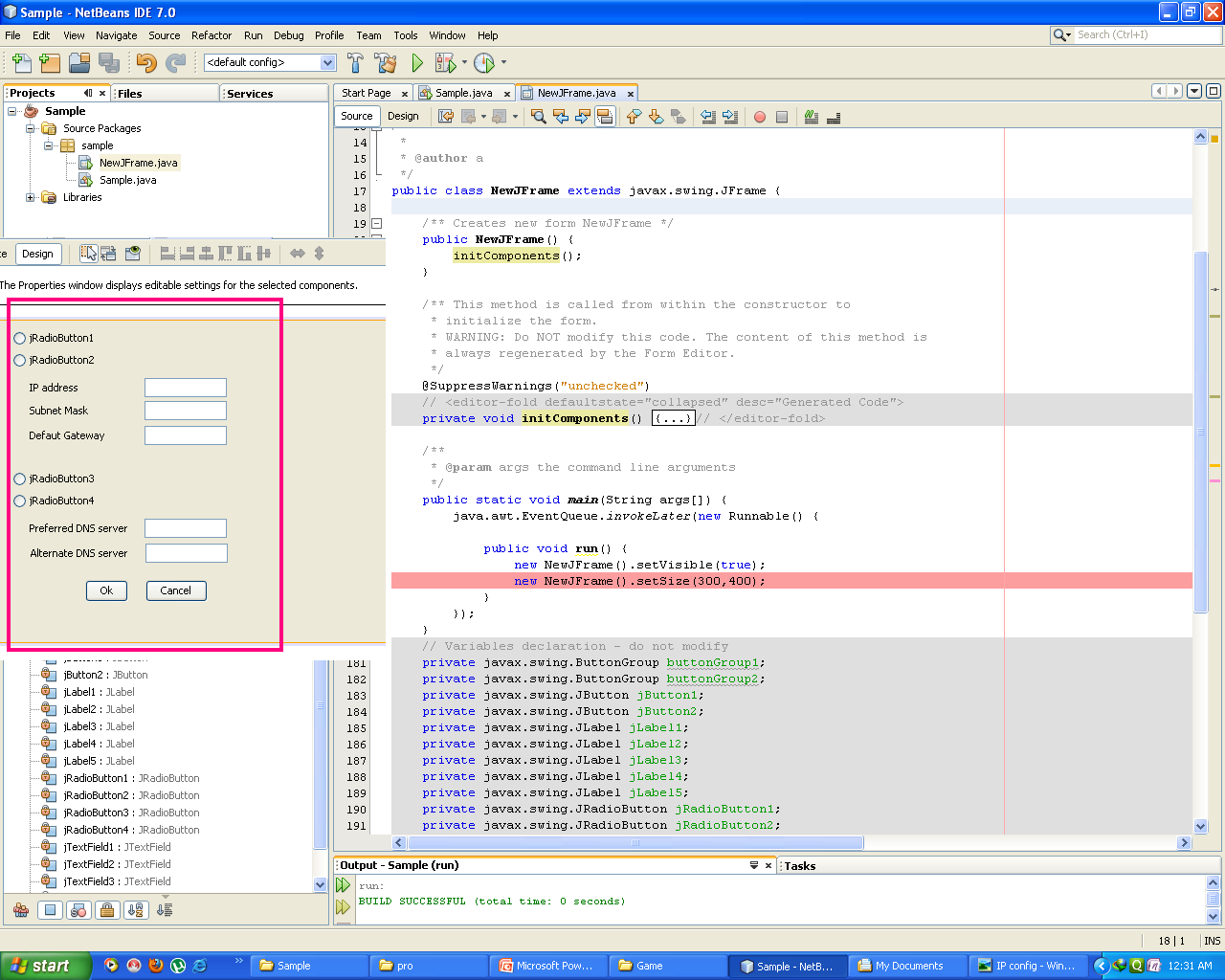
e.g. 98 Lines of code can be reduced up to only 46 line.

Therefore it can be used in any software development Industry in Java. Any programmer having good knowledge about java can use our JAR File for smart programming. Using our project following ways:

1. For any GUI based application.
2. Web development using Java Applets.
3. For Designing any UI.

**Need:**

Developers in other platform like VB, ASP.NET do not have to worry about their layout. But for Java programmer, he/she must add layout code for each component, which will be very much complex in big projects. Though there is software available called NetBeans through which developer can easily design forms using drag and drop method. But the code generated by NetBeans is not reusable as the code is generated in ad-hoc manner. Because of this IT industries don’t use NetBeans for GUI Design.



This project provides Java Developers libraries for constructing layouts in Java. It will help user to design various forms for their application without worrying much about layouts. Also the code or the libraries of this project can be reused anywhere anytime just by installing single jar file of this application.

In big projects at designing phase have complex coding which make it necessary for the user/developer to concentrate on its design rather than logic. By using our project user/developer can focus on the logic as design phase has been simple.

**Hardware and software to be used:**

**Minimum requirements:**

**Software:**

**Operating System:**

**Windows:** Windows 97,98,Windows XP, Windows Vista, Windows 7 and other versions of Windows

**Linux:** Obuntu, RedHat, and other versions of Linux

JRE 1.6.0 or later.

Ellipse

**Hardware:**

**Processor:** IntelCore 2 Duo, I5, I3

**RAM:** 512 MB RAM

**History of Java:**

Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems, Inc. in 1991. It took 18 months to develop the first working version. This language was initially called "Oak" but was renamed "Java" in 1995. Between the initial implementation of Oak in the fall of 1992 and the public announcement of Java in the spring of 1995, many more people contributed to the design and evolution of the language. Bill Joy, Arthur van Hoff, Jonathan Payne, Frank Yellin, and Tim Lindholm were key contributors to the maturing of the original prototype.

**Problem Compared to Other Languages:**

Although Java was modeled after C and C++ languages, it differs from C and C++ in many ways. Java does not incorporate a number of features available in C and C++.

**Java and C**

Java is lot like C but major difference between Java and C is that Java is an object-oriented language and has mechanism to define class and objects. In an effort to build a simple and safe language, the Java team did not include some features of C in Java. Java does not include the C unique statements keywords like “sizeof” and “typedef”. Java does not contain the data types like “struct” and “union”. Java does not support an explicit pointer type.

**Java and C++**

Java is a true object-oriented language while C++ is basically C with object-oriented extension. That is what exactly the increment operator ++ indicates. Java appears to be similar to C++ when we consider only the “extension” part of C++. There are some major C++ features that where intentionally omitted from Java or significantly modified. Java does not support operator overloading. Java does have template classes as in C++. Java does not support multiple inheritance of classes. This is accomplished using a new feature called “interface”. Java does not support global variables. Every variable and method is declared within a class and forms part of that class.

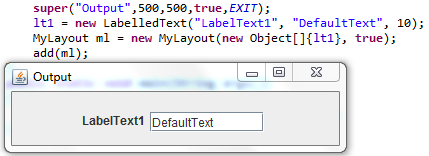
**Project Schedule**

|  |  |  |
| --- | --- | --- |
| Sr. no | DATE | Task |
| 1 | 12-12-2011  To  17-12-2011 | * Selection of Topic * Objectives and Scope * Project Planning |
| 2 | 18-12-2011  To  4-1-2012 | * Collection of snapshots * Study of swing * Study of components and layouts |
| 3 | 8-1-2012  To  15-1-2012 | * Selection of combinations |
| 4 | 17-1-2012  To  30-1-2012 | * Design of specification * Specification of Combinations * Specification of Layouts * Specification of Containers * Specification of Validations |
| 5 | 1-1-2012  To  17-2-2012 | Coding  Implementation of components  Implementation of layouts |
| 6 | 18-2-2012  To  6-3-2012 | Testing  Unit testing of combinations  Unit testing of layouts  Testing of API  Integration testing of whole project |
| 7 | 7-3-2012  To  16-3-2012 | Report Preparation |

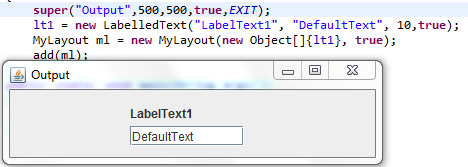
**Our new Features in Java:**

* **Components:**
  1. LabelledText:

Horizontal:

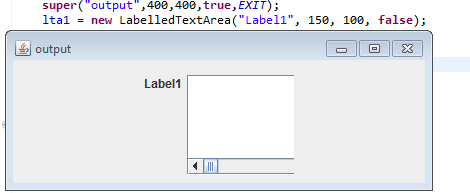


Vertical:

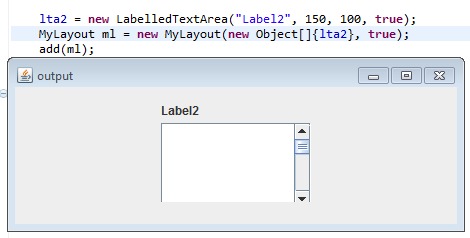


**2.LabelledTextArea:**

Horizontal:

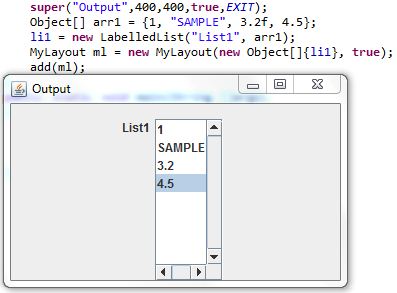


Vertical:

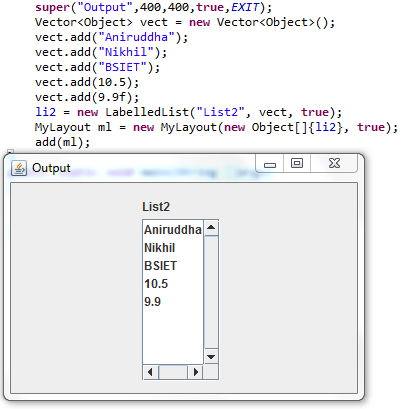


**3.LabelledList:**

Horizontal:

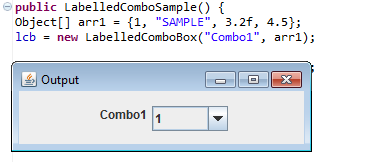
\

Vertical:

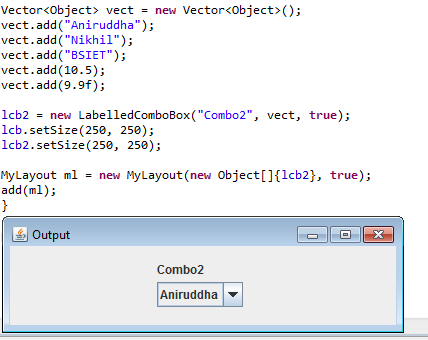


**4:LabelledComboBox:**

Horizontal:

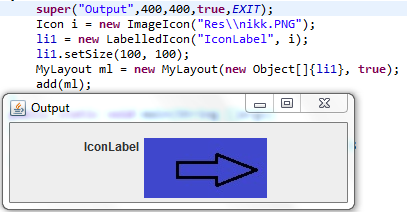


Vertical:

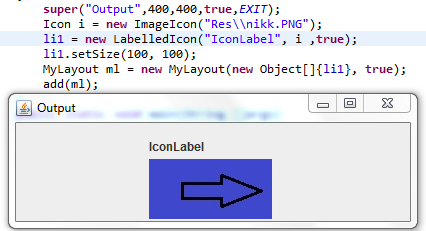


**5.LabelledIcon:**

Horizontal:

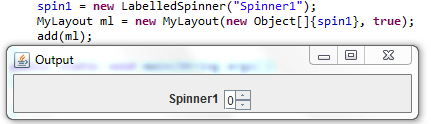


Vertical:

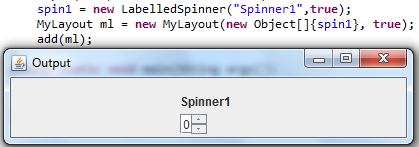


**6.LabelledSpinner:**

Horizontal:

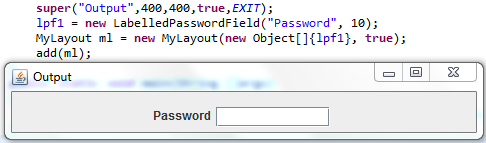


Vertical:

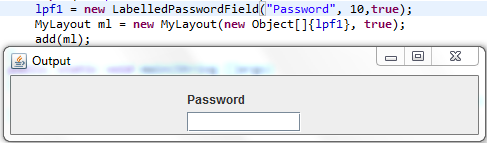


**7.LabelledPasswordField:**

Horizontal:



Vertical:



* **Containers:**

**1.InstallationWizzard:**

InstallationWizzard class is defined inside org.bsiet.containers package this class is used to create wizards with forward and backward jumping(e.g. Next, Previous or Back options).Here developers just need to pass JPanels.

**2.MyFrame:**

MyFrame class is defined inside org,bsiet.containers package this class can be is used instead of JFrame class of javax.swing .

e.g.

[**MyFrame**](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\containers\MyFrame.html#MyFrame%28java.lang.String,%20int,%20int,%20boolean,%20int%29)(java.lang.String frameName, int sizeX, int sizeY, boolean isVisible, int CLOSE\_OPERATION)

A frame that provides pre-handled events like visibility, default close operation and even more. Provides final int’s for default close operation. Exactly works as EXIT\_ON\_CLOSE and all other's. Just they are reconstructed by removing unnecessary words. Instantiates default MyFrame same as JFrame with specified title, width, height, visibility of MyFrame and specified defaultCloseOperation.

**3.OkCancelFrame:**

This is the readymade handled frame provides Pre-Aligned and handled "Ok" & "Cancel" buttons. The frame is already handled with all of its events and even you can also provide your custom choices of handling through parameters. You can decide title of frame, width and height, enabled property of OK-CANCEL buttons and their names and defaultCloseOperation of the frame.Instantiates default MyFrame same as JFrame with specified title, width, height, visibility of MyFrame and specified defaultCloseOperation.

e.g.

[**OkCancelFrame**](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\containers\OkCancelFrame.html#OkCancelFrame%28java.lang.String,%20java.lang.String,%20java.lang.String,%20boolean,%20boolean,%20int%29)(java.lang.String title, java.lang.String button1Name, java.lang.String button2Name, boolean isOkEnabled, boolean isCancelEnabled, int CLOSE\_OPERATION)

Instantiates OkCancelFrame with specified title, names of button1 and button2, visibility of both buttons and default close operation.

* **Layout:**

**1.MyLayout:**

This class allows you to manage your components in HORIZONTAL and VERTICAL group's of GroupLayout. It You just have to pass your Components in proper sequence in which you want in the Object's array. That Object's array then you should have to insert in MyLayout's Object as specified in sample. You'll get the output as all Components in array arranged in Single JPanel arranged by GroupLayout with their PREFERRED\_SIZE. There is limitation as you can pass maximum upto 15 Components for one Object but if you need more then you can manage it through multiple Objects(in multiple JPanel's)

* **Validations:**

**1.UIValidator:**

The UIValidator class helps delevloper's to make validations of various data types like Strict int, float, long, double, String etc.In advance, we've provided data types like e-mail, 10 Mobile Number etc. That are able to throw exception if important conditions related to data-type are not satisfied.

**2.ValidationException:**

ValidationException class is actual our own Exception created for checking runtime behavior .

**e.g.** [**ValidationException**](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\validations\ValidationException.html#ValidationException%28java.lang.String%29)(java.lang.String msg)

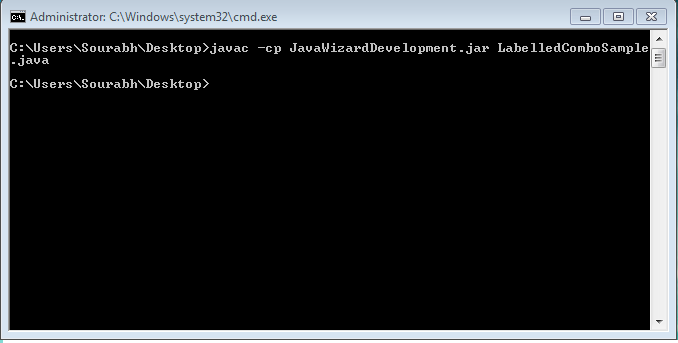
# 

**Installation:**

**a. Copy paste .jar file of our project.**

Our project installation is as simple as any software installation. First that machine should have java installed. If there is java installed in that machine, then you only need to copy paste the jar file of our project into that machine.

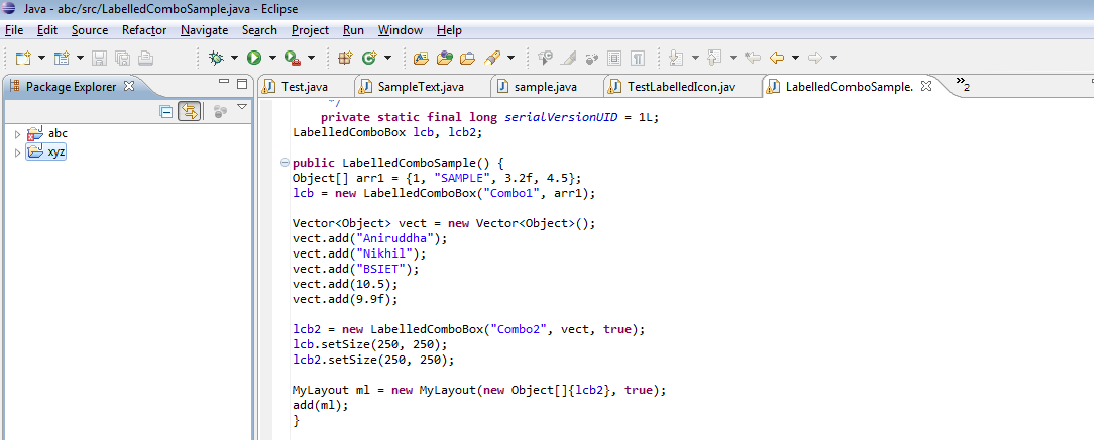
**b. For general text-editor’s.**

****

For general text-editor like cmd just type above line for compilation.

c**. For standard IDE’s like eclipse.**

**Step1:**



# Step2:

# D:\snapshoots\e2.png

# Step3:

# D:\snapshoots\e3.png

# Step4:

# D:\snapshoots\e4.png

# Step5:

# D:\snapshoots\e5.png

# On clicking on Finish Button our jar file is imported in ellipse.

# 

# Tree Diagram:

# Main Tree Diagram

# Hierarchy:

# Hierarchy For All Packages

# Package Hierarchies:

* [org.bsiet.combinations](file:///D:\Project%20Seminar\doc\org\bsiet\combinations\package-tree.html),
* [org.bsiet.containers](file:///D:\Project%20Seminar\doc\org\bsiet\containers\package-tree.html),
* [org.bsiet.layouts](file:///D:\Project%20Seminar\doc\org\bsiet\layouts\package-tree.html),
* [org.bsiet.validations](file:///D:\Project%20Seminar\doc\org\bsiet\validations\package-tree.html)

**Class Hierarchy**

java.lang.Object

* java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
  + java.awt.Container
    - javax.swing.JComponent (implements java.io.Serializable)
      * javax.swing.JPanel (implements javax.accessibility.Accessible)
        + org.bsiet.combinations.[LabelledComboBox](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\combinations\LabelledComboBox.html)
        + org.bsiet.combinations.[LabelledIcon](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\combinations\LabelledIcon.html)
        + org.bsiet.combinations.[LabelledList](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\combinations\LabelledList.html)
        + org.bsiet.combinations.[LabelledPasswordField](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\combinations\LabelledPasswordField.html)
        + org.bsiet.combinations.[LabelledSpinner](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\combinations\LabelledSpinner.html)
        + org.bsiet.combinations.[LabelledText](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\combinations\LabelledText.html)
        + org.bsiet.combinations.[LabelledTextArea](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\combinations\LabelledTextArea.html)
        + org.bsiet.layouts.[MyLayout](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\layouts\MyLayout.html)
    - java.awt.Window (implements javax.accessibility.Accessible)
      * java.awt.Frame (implements java.awt.MenuContainer)
        + javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)

org.bsiet.containers.[MyFrame](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\containers\MyFrame.html)

org.bsiet.containers.[InstallationWizard](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\containers\InstallationWizard.html) (implements java.awt.event.ActionListener)

org.bsiet.containers.[OkCancelFrame](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\containers\OkCancelFrame.html) (implements java.awt.event.ActionListener)

org.bsiet.validations.[ValidationDemo](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\validations\ValidationDemo.html) (implements java.awt.event.ActionListener)

* java.lang.Throwable (implements java.io.Serializable)
  + java.lang.Exception
    - org.bsiet.validations.[UIValidator](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\validations\UIValidator.html)
    - org.bsiet.validations.[ValidationException](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\validations\ValidationException.html)

**Hierarchy For Package org.bsiet.combinations**

Package Hierarchies:

* [All Packages](file:///D:\Project%20Seminar\doc\overview-tree.html)

**Class Hierarchy**

* java.lang.Object
  + java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
    - java.awt.Container
      * javax.swing.JComponent (implements java.io.Serializable)
        + javax.swing.JPanel (implements javax.accessibility.Accessible)

org.bsiet.combinations.[LabelledComboBox](file:///D:\Project%20Seminar\doc\org\bsiet\combinations\LabelledComboBox.html)

org.bsiet.combinations.[LabelledIcon](file:///D:\Project%20Seminar\doc\org\bsiet\combinations\LabelledIcon.html)

org.bsiet.combinations.[LabelledList](file:///D:\Project%20Seminar\doc\org\bsiet\combinations\LabelledList.html)

org.bsiet.combinations.[LabelledPasswordField](file:///D:\Project%20Seminar\doc\org\bsiet\combinations\LabelledPasswordField.html)

org.bsiet.combinations.[LabelledSpinner](file:///D:\Project%20Seminar\doc\org\bsiet\combinations\LabelledSpinner.html)

org.bsiet.combinations.[LabelledText](file:///D:\Project%20Seminar\doc\org\bsiet\combinations\LabelledText.html)

org.bsiet.combinations.[LabelledTextArea](file:///D:\Project%20Seminar\doc\org\bsiet\combinations\LabelledTextArea.html)

**Hierarchy For Package org.bsiet.containers**

Package Hierarchies:

* [All Packages](file:///D:\Project%20Seminar\doc\overview-tree.html)

**Class Hierarchy**

* java.lang.Object
  + java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
    - java.awt.Container
      * java.awt.Window (implements javax.accessibility.Accessible)
        + java.awt.Frame (implements java.awt.MenuContainer)

javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)

org.bsiet.containers.[MyFrame](file:///D:\Project%20Seminar\doc\org\bsiet\containers\MyFrame.html)

org.bsiet.containers.[InstallationWizard](file:///D:\Project%20Seminar\doc\org\bsiet\containers\InstallationWizard.html) (implements java.awt.event.ActionListener)

org.bsiet.containers.[OkCancelFrame](file:///D:\Project%20Seminar\doc\org\bsiet\containers\OkCancelFrame.html) (implements java.awt.event.ActionListener)

**Hierarchy For Package org.bsiet.layouts**

Package Hierarchies:

* [All Packages](file:///D:\Project%20Seminar\doc\overview-tree.html)

**Class Hierarchy**

* java.lang.Object
  + java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
    - java.awt.Container
      * javax.swing.JComponent (implements java.io.Serializable)
        + javax.swing.JPanel (implements javax.accessibility.Accessible)

org.bsiet.layouts.[MyLayout](file:///D:\Project%20Seminar\doc\org\bsiet\layouts\MyLayout.html)

**Hierarchy For Package org.bsiet.validations**

Package Hierarchies:

* [All Packages](file:///D:\Project%20Seminar\doc\overview-tree.html)

**Class Hierarchy**

java.lang.Object

* java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
  + java.awt.Container
    - java.awt.Window (implements javax.accessibility.Accessible)
      * java.awt.Frame (implements java.awt.MenuContainer)
        + javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)

org.bsiet.validations.[ValidationDemo](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\validations\ValidationDemo.html) (implements java.awt.event.ActionListener)

* java.lang.Throwable (implements java.io.Serializable)
  + java.lang.Exception
    - org.bsiet.validations.[UIValidator](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\validations\UIValidator.html)
    - org.bsiet.validations.[ValidationException](file:///C:\Users\Sourabh\Desktop\doc\org\bsiet\validations\ValidationException.html)

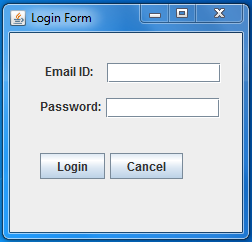
**Methodology:**

As the aim of Java Wizzard Developement is to reduce lines of code. Now How to use this feature ?

Following are the guidelines for developer/user that How to use

“Java Wizzard Developement”.

Now Suppose Developer wants to create Following Form using our Project then Developer/user can refer following guidelines:

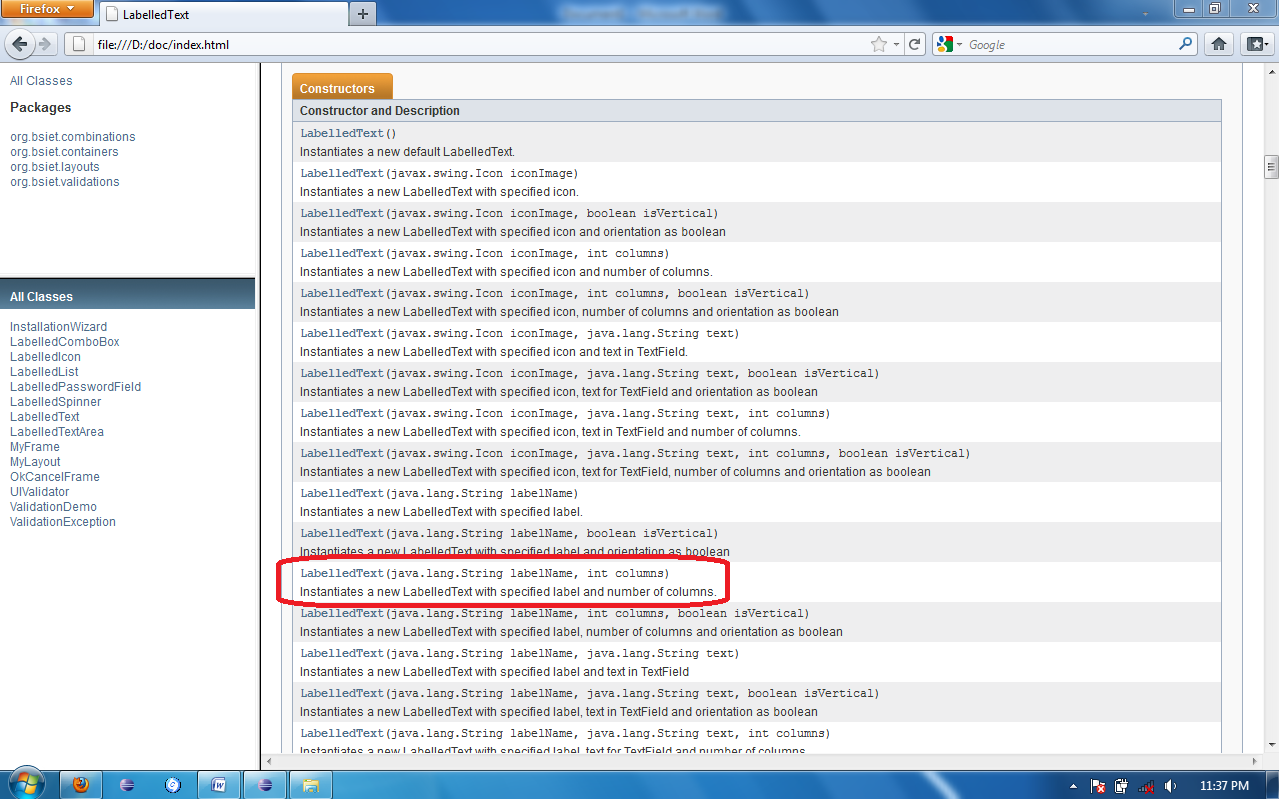


From the Design Here Developer need Following Jcomponents:

* 1. Two JLabel’s
  2. One JTextField
  3. One JPasswordField
  4. Two JButton’s

But using our Project here Developer can use Following Components:

* 1. One LabelledText
  2. One LabelledPasswordField
  3. Two JButton’s
* Now After gathering Requirements refer API and decide suitable Constructor for component as per required Design(Login Form).



In diagram of API the red marked Constructor is suitable for 1st component LabelledText as per Design.

Eg:

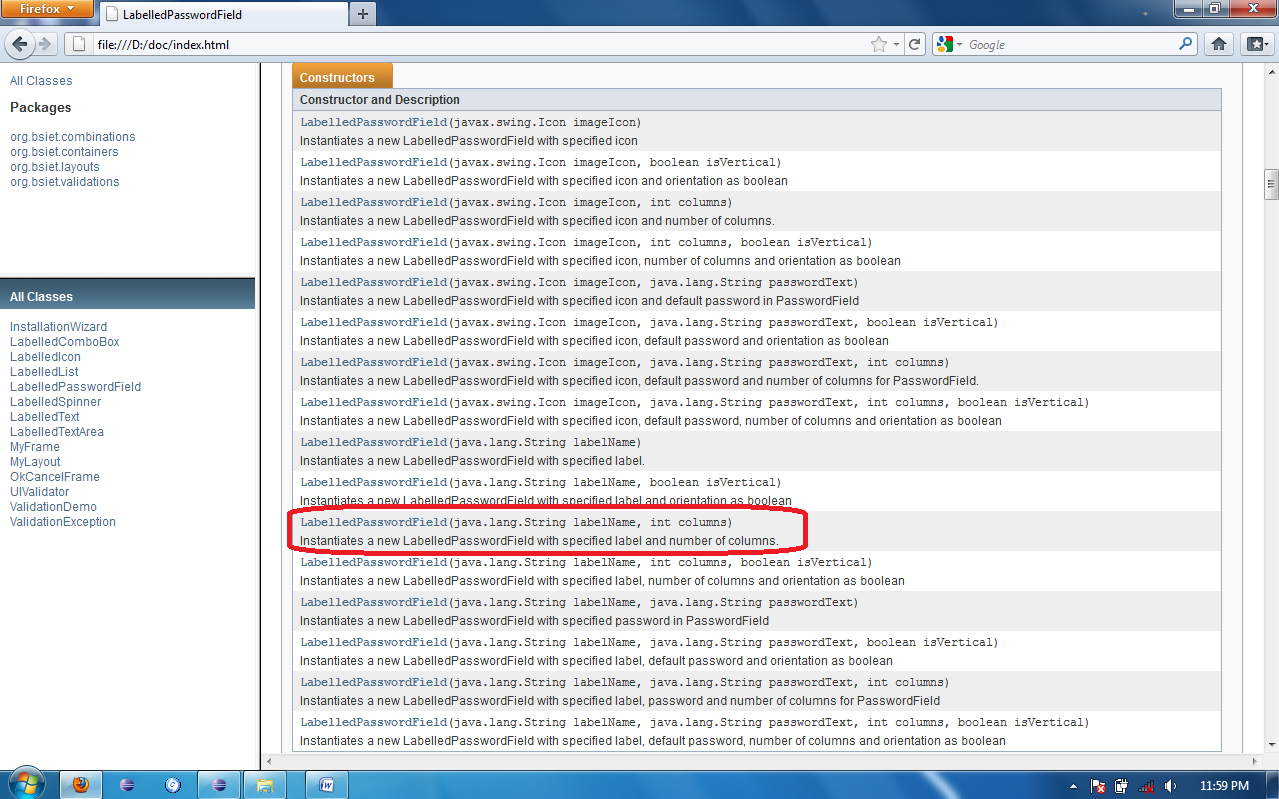
LabelledText emailID;

emailID=new LabelledText(“Email ID”,10);

/\* Here 1st parameter “Email ID” as a String is Label Name for JLabel And 2nd parameter 10 is Integer value for indicating number of coloumns for JTextField in LabelledText \*/

* Now the next Component is LabelledPasswordField

Here the red marked ellipse indicates suitable Constructor for designing LabelledPasswordField as per required Design.



E.g.:

LabelledPasswordField pass;

Pass=new LabelledPasswordField(“PasswordField”,10);

/\* Here 1st parameter String “Password” is basically for JLabel And 2nd parameter Integer 10 is for indicating number of coloumns of JPasswordField \*/

* Now the next Components are 2 JButton’s i.e.Login & Cancel

E.g.:

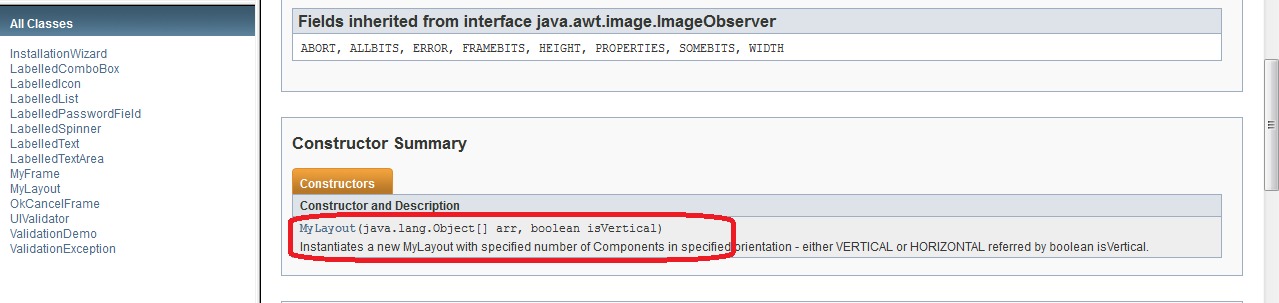
JButton loginbtn,cancelbtn;

loginbtn=new JButton(“Login”);

cancelbtn=new JButton(“Cancel”);

* Now here the initialization phase is completed the next step is to arrange these Components as per Design using “MyLayout” from org.bsiet.layouts.MyLayout;

Refer API



Here red marked single Constructor can be used for arranging Components as per required Design.

Eg:

MyLayout layout1=**new** MyLayout(**new** Object[]{emailID,pass}, **true**);

MyLayout layout2=**new** MyLayout(**new** Object[]{loginbtn,cancelbtn}, **false**);

MyLayout layout=**new** MyLayout(**new** Object[]{layout1,layout2}, **true**);

add(layout);

/\* Here 1st of all MyLayout class returns JPanel and it has paramrters array of Objects(Components) and orientation Boolean true or false

If true then all Components on array will arranged in Vertically

And If false then all Components in array will arranged Horizontally parallel to each other.

layout1 indicates 1st JPanel having Components emailID & pass. orientation true indicates both Components will Vertically parallel to each other.

layout2 indicates 2nd JPanel having Components loginbtn &cancelbtn.orientation false indicates both Components will be Horizontally parallel to each other.

Now layout JPanel contains 2 JPanels layout1 & layout2. Orientation true indicates both JPanel’s are Vertically parallel to each other.

Now after adding JPanel layout the Design is ready.

E.g.: add(layout);

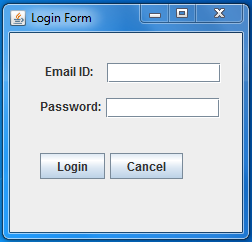
\*/

* We can also use methods of Components while using our project.All methods are overridden.

Eg:

str1=emailID.getLabel();

System.*out*.println(str1);



Following code generates the above “Login Form”.

***import*** *javax.swing.\*;*

***import*** *org.bsiet.combinations.LabelledPasswordField;*

***import*** *org.bsiet.combinations.LabelledText;*

***import*** *org.bsiet.containers.MyFrame;*

***import*** *org.bsiet.layouts.MyLayout;*

***public******class*** *LoginForm* ***extends*** *MyFrame*

*{*

*LabelledText emailID;*

*LabelledPasswordField pass;*

*JButton loginbtn,cancelbtn;*

*String str1;*

***public*** *LoginForm()*

*{*

***super****("Login Form",500,500,****true****,EXIT);*

*emailID=****new*** *LabelledText("Email ID: ", 10);*

*pass=****new*** *LabelledPasswordField("Password:", 10);*

*loginbtn=****new*** *JButton("Login");*

*cancelbtn=****new*** *JButton("Cancel");*

*MyLayout layout1=****new*** *MyLayout(****new*** *Object[]{emailID,pass},* ***true****);*

*MyLayout layout2=****new*** *MyLayout(****new*** *Object[]{loginbtn,cancelbtn},* ***false****);*

*MyLayout layout=****new*** *MyLayout(****new*** *Object[]{layout1,layout2},* ***true****);*

*add(layout);*

*str1=emailID.getLabel();*

*System.out.println(str1);*

*}*

***public******static******void*** *main(String args[])*

*{*

*LoginForm l=****new*** *LoginForm();*

*}*

*}*

**Testing:**

Testing is the important phase in the development of any application or project. There are many definition of Testing:

* Testing is the process which is performed to prove that there are no errors in a program
* Testing is the process to prove that software works correctly.
* Testing is the process to detect the defects and minimize the risk associated with the residual defects.

The objective of testing is to uncover as many bugs as possible. The testing should be done perfect and without any attachments to the software. The software testing is destructive than constructive.

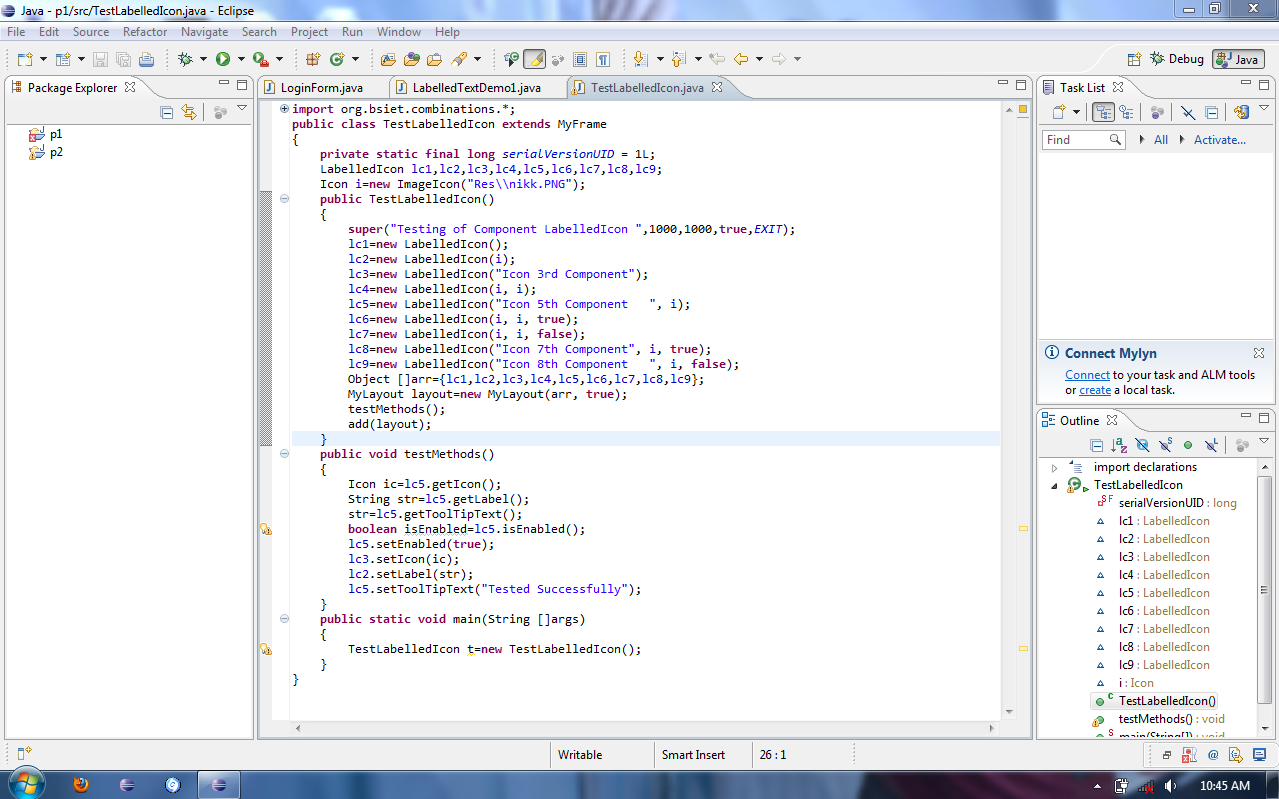
Our project required many testing. Testing was done according to the need of the project. We divided our testing into number of phases.

**Unit Testing:**

Unit testing focuses verification effort on the smallest unit of software design i.e the software components or module. This is the general definition of Unit Testing.

Our project consists of various components which are have different structure, different methods. These components(LabelledComboBox, LabelledIcon, LabelledList, LabelledPasswordField, abelledSpinner, LabelledText, LabelledTextArea) where tested by their constructors, methods etc. We made test cases for these components.

**Sample Code:**

****

****

Here is the sample code of test case for a single component LabelledIcon. We tested its constructors and methods of LabelledIcon. We did this for all the components i.e. unit. We also checked Boundary conditions for all the components.

**Integration Testing:**

Integration Testing is the systematic technique for constructing the software architecture while at the same time conducting tests to uncover errors associated with interface. In integration testing the program is constructed in small increments, where errors are easier t o isolate and correct, interfaces are more likely to be tested completely and systematic test approach can be applied.

We applied integration testing in our project. We use bottom-up approach for our project. In bottom-up approach we first tested different components we have made like LabelledText, LabelledTextArea, LabelledPassword etc. Then we tested container package. It included InstalationWizard, OkCancelFrame, MyFrame. After that we tested Layout package. We have created MyLayout class which takes care of complexity of GroupLayout.

**Performance Testing:**

Performance Testing is designed to test the runtime performance of software in the context of a system. To make true performance test all the system elements must be fully integrated.

We first arranged all the our components, containers, layout, validations in single jar file. Then tested jar file as whole for its performance.

**Advantages :**

1. Rapid Application Development.

In java the developer needs to write lots of code to develop a single frame. The java wizard development provides the package for developing various components and layouts. By using the Java wizard development we can develop the code faster than normal coding. As the Java wizard development provides the ready-made packages for various components and layouts the programmer can use these packages in his/her code and can override the methods.

1. No need to bother about complexity of GroupLayout

There are multiple types of forms which use multiple layouts which are complex and change their layout alignments according to requirements. The java wizard development provides the package for managing the layouts this reduces the burden of developer. And the developer can concentrate more on the other parts of the development. The complex Layouts can be handled properly and easily.

1. Reliability of using project

Our project is more reliable in consideration of Developers point of view. Various features in our project help to Developer to save time in coding. This features include components, containers, layouts which help our project being more reliable. Also it can be used in various ways in many applications.

1. Open Source

Open Source as the name suggest it can be used by anyone. Sorce code is available to the developers free of cost. If developer wish any changes he can make changes in his code he has to submit the changes to us then we will write test cases for it. We will test it with methods as required for validating the component. If that feature passes our validation process then we will add it into this project and will release it through next version.

1. User friendly API documentation with proper description like as of Java

The java wizard development provides its own API for the user. The java wizard development API gives the information to user about How to use? What it does? What constructors should be used? And What are the methods?. The java wizard development API is easy to understand and easy to use. API of java wizard development contents hierarchy of packages which will make user to find the class easily.

1. Generalized class, method and fields names.

The java wizard development has generalized names for class, method and fields so that the user can easily understand what it does? And he/she can find the class or method easily. The generalized names will also help the user to remember the names.

1. No need to setup, available in single JAR file

The requirements of The java wizard development are low. It requires the single JAR file and the JDK software to be installed.

**Conclusion:**

**What Developers will get?**

Our project is basically made in Developers point of view. It will just save lot of time of Developer. First of all it will reduce the time for coding. With our Combination package which have various components will help the Developer design easily. Our Container Package which has InstalationWizard, MyFrame, OkCancelFrame will help in provides pre-handled events like visibility, default close operation and even more. Layout Package which contain MyLayout class helps the Developer to deal with complexity of GroupLayout. Also Developer will get help from our API which is arranged in user friendly manner.

**What performance will be achieved?**

Greater performance can be achieved by using our project. It will boost the morale of Developer due to various features our project have provided. Complexity of GroupLayout which is handle in our project will reduce time which will in directly save the money.

**Where it is useful?**

Our project can be use full in many ways. First case would be in GUI Applications. We know that GUI applications are user friendly. To make user friendly application means to create the application like it will feel user good while development and also for end user to use. Making user friendly application requires lots of coding as well it takes more time and money. This GUI based applications are difficult to design for programmer, as programmer has to design his application according to user friendly approach. By using our code, total time required for development, money and efforts of programmer can be saved. And its all is without losing reliability of application.

Second would be the case of complex Layout. Some layouts are complex to design. This requires lots of coding. Also it should be designed in such a way that it must be user friendly manner. Arranging the components in complex layout is also a difficult job. This job can be reduced by using our API library and Layout, Container packages. Forms may contain multiple layouts in single one. Managing these multiple layouts need lots of efforts. These efforts can be reduced by using our API library and Layout, Container packages.

Third will be the case of installation of software wizard. To install any software, we need wizard to install it. This wizard is combination of many phases. This includes configuring hardware, initialization of software etc. To make this, each phase is taken in framework. This complexity while installing the software can be reduced. The same components are used again and again making it easy for user. But for each framework, there is necessity to write new code again and again.

Forth case will for strict validation of datatypes. Validations are important part of any system. It is very important to verification everything. Validation acts like key in many applications. Validations may a combination of alphabets, numeric or combination of both. In our project we have took necessary attention to this point.

**Future scope:**

1. Some new components and containers will be implemented.
2. Try to extend this project with more flexibility and reliability.
3. Provide Exception summary.
4. To provide our own Wizard Development IDE.
5. Try to implement all important necessities studied from FEEDBACK.

**Reference Books:**

1. The Complete Reference-Java2 by Patrick Naughton and Herbert Schildt

2. Core Swing: Advance Programming by Kim Topley

3. Java Swing : by O.Reilly