NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

MESSAGE ORGANIZER

REPORT

A MINI PROJECT

REPORT

SUBMITTED BY

JITENDRA SAH

In partial fulfillment for the award of the degree of

Bachelor of Engineering

IN

COMPUTER SCIENCE AND ENGINEERING

ABSTRACT

In today's world of communication, it is seen that hundred and thousands of texts, codes and otp are being exchanged and generated. Due to this load of messages, it is very important to organize according to their types. It is very difficult to search a message of a particular person in a pile up messages containing otp, code and other type of the messages. This also results in the more consumption of the time and arises the problem of time on other activity.

The solution to the above problem statement is the Message Organizer project designed as per the user requirement of sorting the message. The message is sorted on the basis of their category like the text from a company are sorted in one category, otp in one category etc. The message is received from other sources and sent to a particular source where the organizer detects and categorizes the message.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be, but impossible without the mention of the people who made it possible, whose constant guidance and encouragement crowned my efforts with success.

I have great pleasure in expressing gratitude to **Dr. Mohan Manghnani,** Chairman of New Horizon Educational Institutions for providing necessary infrastructure and creating good environments

I take this opportunity to express my profound gratitude to **Dr. Manjunatha**,Principal NHCE, for his constant support and encouragement.

I am grateful to **Dr. Prashanth. C. S. R,** Dean Academics, for his unfailing encouragement and suggestions, given to me in the course of my project work.

I would also like to thank **Dr. B. Rajalakshmi**, Professor and Head of the Department of Computer Science and Engineering, for her constant support.

I express my gratitude to **Mr Santosh Kumar,** Senior Asst. Professor my project reviewer for constantly monitoring the development of the project and setting up precise deadlines. Her valuable suggestions were the motivating factors in completing the work.

Finally a note of thanks to all the teaching and non-teaching staff of Computer Science and Engineering Department for their cooperation extended to me and my friends, who helped me directly or indirectly in the course of the project work.

JITENDRA SAH

1NH17CS059

CONTENTS

List of Figures

CHAPTER 1

INTRODUCTION

1.1 PROBLEM DEFINITION

In today's world of communication, it is seen that hundred and thousands of texts, codes and otp are being exchanged and generated. Due to this load of messages, it is very important to organize according to their types. It is very difficult to search a message of a particular person in a pile up messages containing otp, code and other type of the messages. This also results in the more consumption of the time and arises the problem of time on other activity.

1.2 COURSE OBJECTIVES

The solution to the above problem statement is the Message Organizer project designed as per the user requirement of sorting the message. The message is sorted on the basis of their category like the text from a company are sorted in one category, otp in one category etc. The message is received from other sources and sent to a particular source where the organizer detects and categorizes the message.

1.3 METHODOLOGY TO BE FOLLOWED

In this project agile methodology is used. Every java file is developed in an incremental process. After judging the user requirements each screen or method is added step by step and only after completion of previous step.

1.4 OUTCOMES OF THE PROJECT

- This program allows to get rid of some circumstances where user has to check all of their transaction, OTP, Food, Banking, General messages.
- In this program user will also be able to lookup on their messages which are saved on database.
- It will also help know in which category user has received or send the messages.

1.5 REQUIREMENT SPECIFICATION

HARDWARE REQUIREMENT

Processor	:	i3 and higher which is X86 Compatible
RAM	:	512 MB or more
Hard disk	:	500 GB

SOFTWARE REQUIREMENT

Eclipse IDE for java.

CHAPTER 2

OBJECT ORIENTED CONCEPTS

Swing in java is part of Java foundation class which is lightweight and platform independent. It is used for creating window-based applications. It includes components like button, scroll bar, text field etc. Putting together all these components makes a graphical user interface. In this article, we will go through the concepts involved in the process of building applications using swing in **Java**. Following are the concepts discussed in this article:

- What is Java Swing?
- Container Class
- Difference Between AWT And Swing
- Java Swing Class Hierarchy
- Layout Manager
- Example-Chat Frame

What is Swing in Java?

Swing in Java is a lightweight GUI toolkit which has a wide variety of widgets for building optimized window-based applications. It is a part of the JFC (Java Foundation Classes). It is built on top of the AWT API and entirely written in java. It is platform independent unlike AWT and has lightweight components.

It becomes easier to build applications since we already have GUI components like button, checkbox etc. This is helpful because we do not have to start from the scratch.

2.1 Container Class

Any **class** which has other components in it is called as a container class. For building GUI applications at least one container class is necessary.

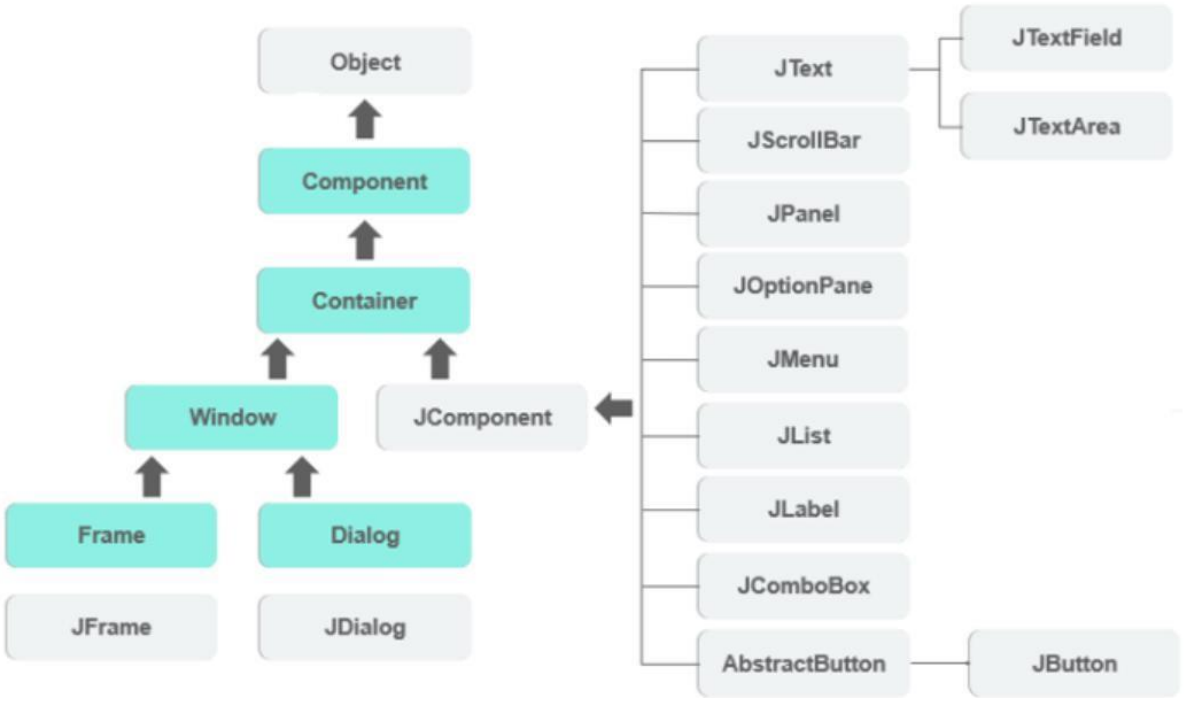
Following are the three types of container classes:

1. Panel – It is used to organize components on to a window
2. Frame – A fully functioning window with icons and titles
3. Dialog – It is like a pop-up window but not fully functional like the frame

Difference Between AWT and Swing

AWT	SWING
• Platform Dependent	• Platform Independent
• Does not follow MVC	• Follows MVC
• Lesser Components	• More powerful components
• Does not support pluggable look and feel	• Supports pluggable look and feel
• Heavyweight	• Lightweight

Java Swing Class Hierarchy:

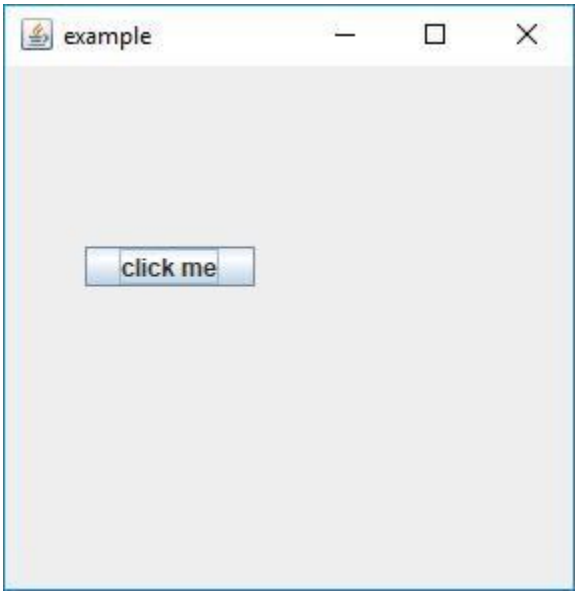


Explanation:

All the components in swing like JButton, JComboBox, JList, JLabel are inherited from the JComponent class which can be added to the container classes. Containers are the windows like frame and dialog boxes. Basic swing components are the building blocks of any gui application. Methods like setLayout override the default layout in each container. Containers like JFrame and JDialog can only add a component to itself. Following are a few components with examples to understand how we can use them.

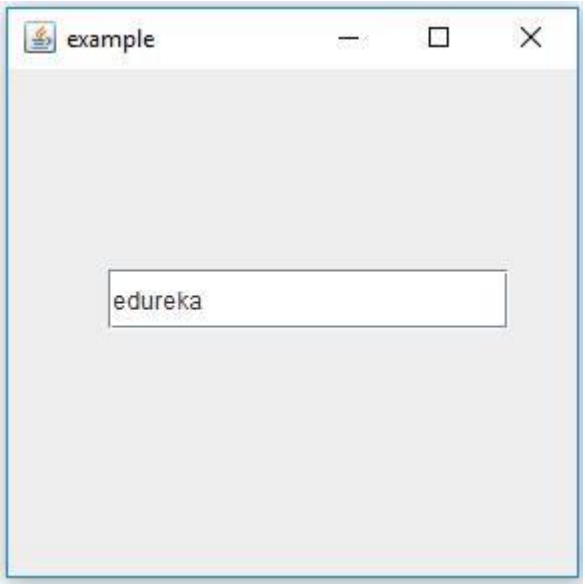
JButton Class

It is used to create a labelled button. Using the ActionListener it will result in some action when the button is pushed. It inherits the Abstract Button class and is platform independent.



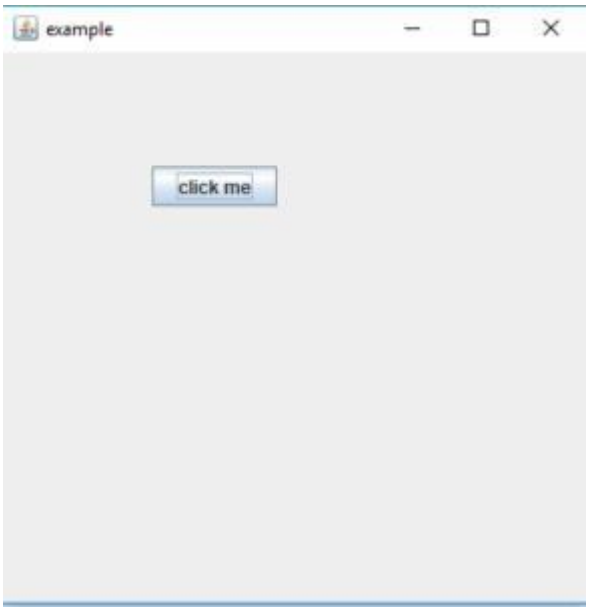
JTextField Class

It inherits the JTextComponent class and it is used to allow editing of single line text.



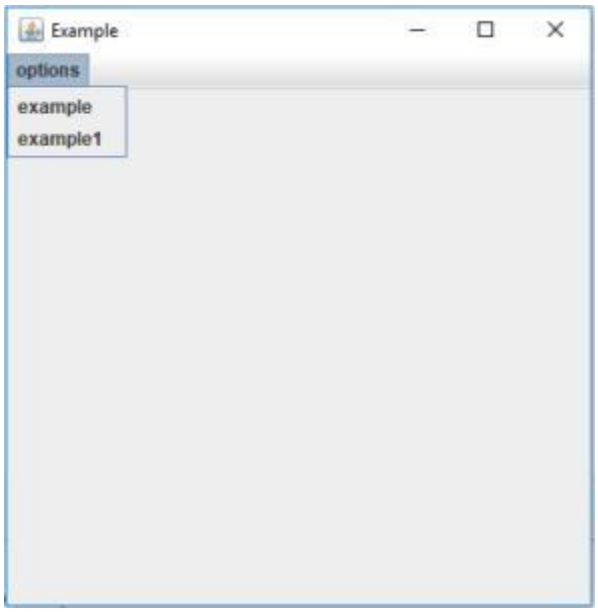
JScrollBar Class

It is used to add scroll bar, both horizontal and vertical.



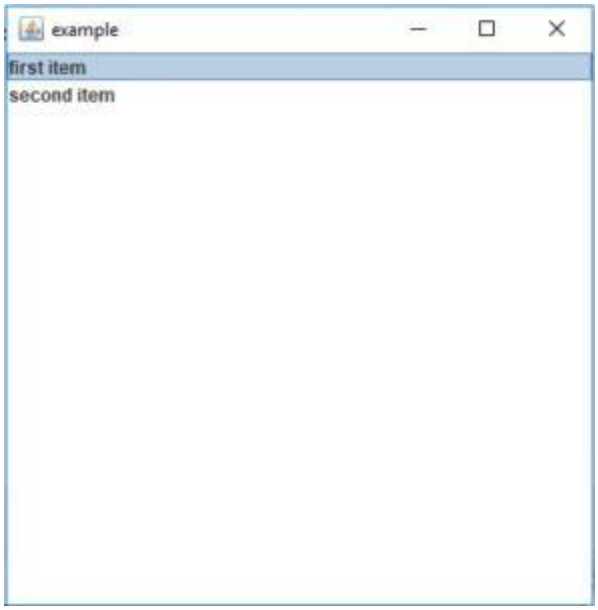
JMenu Class

It inherits the JMenuItem class, and is a pull-down menu component which is displayed from the menu bar.



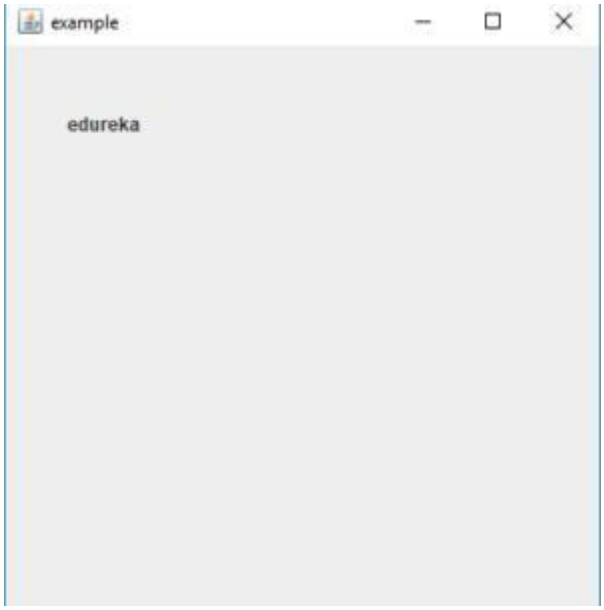
JList Class

It inherits JComponent class, the object of JList class represents a list of text items.



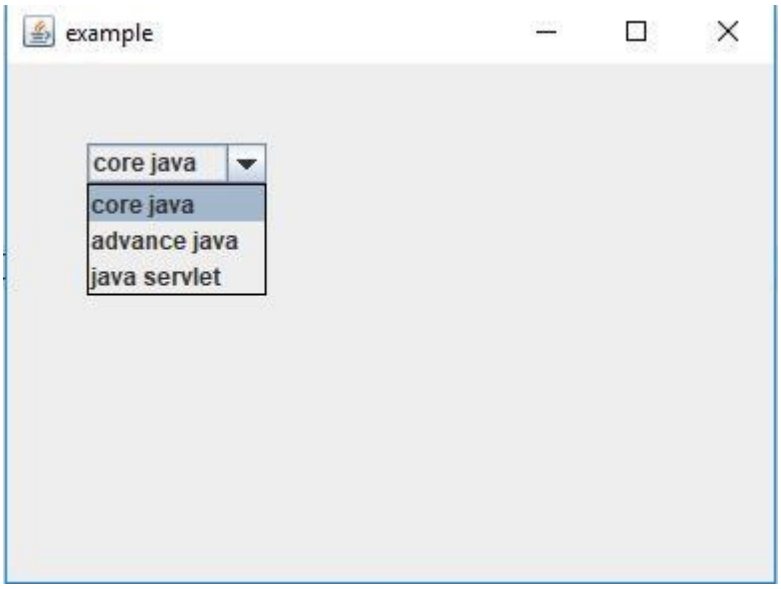
JLabel Class

It is used for placing text in a container. It also inherits JComponent class.



JComboBox Class

It inherits the JComponent class and is used to show pop up menu of choices.



2.2 OBJECT

Software objects also have a state and a behaviour. A software object's state is stored in fields and behaviour is shown via methods.

So in software development, methods operate on the internal state of an object and the object-to-object communication is done via methods.

An object has three characteristics:

- **State:** represents the data (value) of an object.
- **Behaviour:** represents the behaviour (functionality) of an object such as deposit, withdraw, etc.

- **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

So basically, an object is created from a class. In Java, the new keyword is used to create new objects.

There are three steps when creating an object from a class –

Declaration – A variable declaration with a variable name with an object type.

Instantiation – The 'new' keyword is used to create the object.

Initialization – The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

Following is an example of creating an object –

Example

```
public class Puppy { public
   Puppy(String name) {
      // This constructor has one parameter, name.
      System.out.println("Passed Name is :" + name );
   }

   public static void main(String []args) {
      // Following statement would create an object
   myPuppy Puppy myPuppy = new Puppy( "tommy" ); }}
```

If we compile and run the above program, then it will produce the following result –

Output

```
Passed Name is :tommy
```

2.3 INHERITANCE

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviours of a parent object. It is an important part of OOPs (Object Oriented programming system).

The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

Inheritance represents the **IS-A relationship** which is also known as a parent child relationship.

The syntax of Java Inheritance **class** Subclass-name **extends** Superclass-name

```
{
  //methods and fields
}
```

The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

In the terminology of Java, a class which is inherited is called a parent or superclass, and the new class is called child or subclass.

On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.

Single Inheritance Example

When a class inherits another class, it is known as a single inheritance. In the example given below, Dog class inherits the Animal class, so there is the single inheritance.

Multilevel Inheritance Example

When there is a chain of inheritance, it is known as multilevel inheritance. As you can see in the example given below, Baby Dog class inherits the Dog class which again inherits the Animal class, so there is a multilevel inheritance.

Hierarchical Inheritance Example

When two or more classes inherits a single class, it is known as hierarchical inheritance. In the example given below, Dog and Cat classes inherits the Animal class, so there is hierarchical inheritance.

2.4 POLYMORPHISM

Polymorphism in java is one of core Object-oriented programming concepts with **Abstraction**, **encapsulation**, and **inheritance**.

Polymorphism means one name many forms. In Java, polymorphism can be achieved by **method overloading** and **method overriding**.

There are two types of polymorphism in java.

- Compile time polymorphism.
- Run time polymorphism.

Compile time Polymorphism

Compile time Polymorphism is nothing but **method overloading** in java. You can define various methods with same name but different method arguments. You can read more about **method overloading**.

Runtime Polymorphism

Runtime Polymorphism is nothing but **method overriding** in java.If subclass is having same

method as base class then it is known as method overriding Or in another word, If subclass provides specific implementation to any method which is present in its one of parents classes then it is known as. method overriding

2.5 ABSTRACT CLASS

A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).But first let us know more about Abstraction.

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

Abstraction lets you focus on what the object does instead of how it does it.

Ways to achieve Abstraction

There are two ways to achieve abstraction in java

1. Abstract class (0 to 100%)
2. Interface (100%)

A class which is declared as abstract is known as an **abstract class**. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.

- It can have constructors and static methods also.

- It can have final methods which will force the subclass not to change the body of the method.

A method which is declared as abstract and does not have implementation is known as an abstract method.

Example of abstract method abstract void

```
printStatus();//no method body and abstract
```

2.6 MULTITHREADING

Multithreading in Java is a process of executing multiple threads simultaneously.

A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

Advantages of Java Multithreading

1) It **doesn't block the user** because threads are independent and you can perform multiple operations at the same time.

2) You **can perform many operations together, so it saves time**.

3) Threads are **independent**, so it doesn't affect other threads if an exception occurs in a single thread.

Threads are light-weight processes within a process.

Threads can be created by using two mechanisms:

1. Extending the Thread class
2. Implementing the Runnable Interface

Thread creation by extending the Thread class

We create a class that extends the **java.lang.Thread** class. This class overrides the run() method available in the Thread class. A thread begins its life inside run() method. We create an object of our new class and call start() method to start the execution of a thread. Start() invokes the run() method on the Thread object.

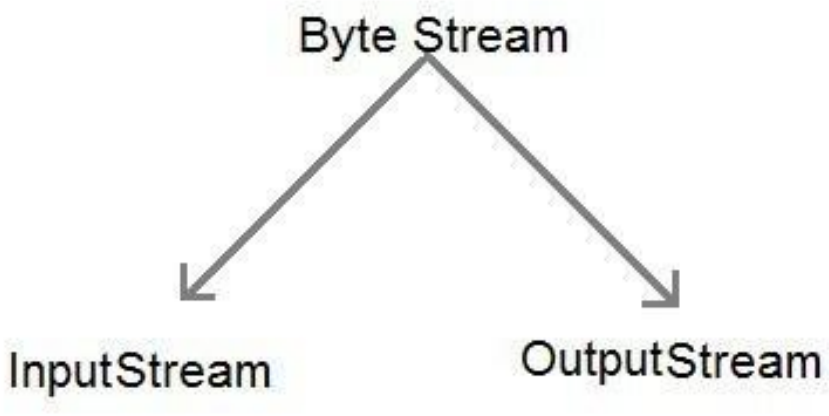
2.7 IO FUNCTIONS

Java performs I/O through **Streams**. A Stream is linked to a physical layer by java I/O system to make input and output operation in java. In general, a stream means continuous flow of data. Streams are clean way to deal with input/output without having every part of your code understand the physical.

Java encapsulates Stream under **java.io** package. Java defines two types of streams. They are,

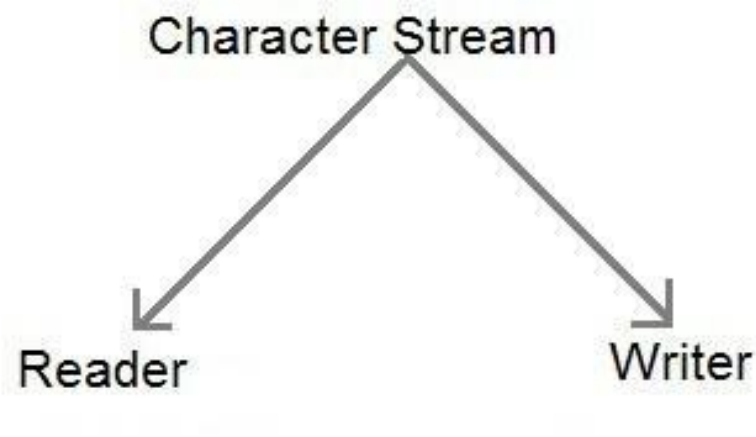
1. **Byte Stream:** It provides a convenient means for handling input and output of byte.
2. **Character Stream:** It provides a convenient means for handling input and output of characters. Character stream uses Unicode and therefore can be internationalized.

Byte stream is defined by using two abstract class at the top of hierarchy, they are Input Stream and Output Stream.



These two abstract classes have several concrete classes that handle various devices such as disk files, network connection etc.

Character stream is also defined by using two abstract class at the top of hierarchy, they are Reader and Writer.



These two abstract classes have several concrete classes that handle Unicode character.

2.8 JAVA PACKAGES

A package as the name suggests is a pack(group) of classes, interfaces and other packages. In java we use packages to organize our classes and interfaces. We have two **types of packages in Java**: built-in packages and the packages we can create (also known as user defined package). In this guide we will learn what are packages, what are user-defined packages in java and how to use them.

In java we have several built-in packages, for example when we need user input, we import a package like this:

```
import java.util.Scanner
```

Here:

→ **java** is a top-level package

→ **util** is a sub package

→ and **Scanner** is a class which is present in the sub package **util**.

Types of packages in Java

As mentioned in the beginning of this guide that we have two types of packages in java.

1) User defined package: The package we create is called user-defined package.
2) Built-in package: The already defined package like java.io.*, java.lang.* etc are known as built-in packages.

2.9 EXCEPTION HANDLING

Exception Handling in Java is a very interesting topic. Exception is an error event that can happen during the execution of a program and disrupts its normal flow. Java provides a robust and object-oriented way to handle exception scenarios, known as **Java Exception Handling**.

Java being an object oriented programming language, whenever an error occurs while executing a statement, creates an **exception object** and then the normal flow of the program halts and JRE tries to find someone that can handle the raised exception. The

exception object contains a lot of debugging information such as method hierarchy, line number where the exception occurred, type of exception etc. When the exception occurs in a method, the process of creating the exception object and handing it over to runtime environment is called **"throwing the exception"**.

Java provides specific keywords for exception handling purposes; we will look after them first and then we will write a simple program showing how to use them for exception handling.

1. **throw** – We know that if any exception occurs, an exception object is getting created and then Java runtime starts processing to handle them. Sometime we might want to generate exception explicitly in our code, for example in a user authentication program we should throw exception to client if the password is null. **throw** keyword is used to throw exception to the runtime to handle it.
2. **throws** – When we are throwing any exception in a method and not handling it, then we need to use **throws** keyword in method signature to let caller program know the exceptions that might be thrown by the method. The caller method might handle these exceptions or propagate it to its caller method using throws keyword. We can provide multiple exceptions in the throws clause and it can be used with main() method also.
3. **try-catch** – We use try-catch block for exception handling in our code. try is the start of the block and catch is at the end of try block to handle the exceptions. We can have multiple catch blocks with a try and try-catch block can be nested also. catch block requires a parameter that should be of type Exception.
4. **finally** – finally block is optional and can be used only with try-catch block. Since exception halts the process of execution, we might have some resources open that will not get closed, so we can use finally block. finally block gets executed always, whether exception occurred or not.

CHAPTER 3

3.1 DESIGN GOALS

Platform Independent

The concept of Write-once-run-anywhere (known as the Platform independent) is one of the important key feature of java language that makes java as the most powerful language. Not even a single language is idle to this feature but java is closer to this feature. The programs written on one platform can run on any platform provided the platform must have the JVM.

Simple

There are various features that makes the java as a simple language. Programs are easy to write and debug because java does not use the pointers explicitly. It is much harder to write the java programs that can crash the system but we cannot say about the other programming languages. Java provides the bug free system due to the strong memory management. It also has the automatic memory allocation and deallocation system.

Object Oriented

To be an Object-Oriented language, any language must follow at least the four characteristics.

- Inheritance: It is the process of creating the new classes and using the behaviour of the existing classes by extending them just toreuse the existing code and adding the additional features as needed.
- Encapsulation: It is the mechanism of combining the information and providing

  the abstraction.

- Polymorphism: As the name suggest one name multiple form, Polymorphism is the way of providing the different functionality by the functions having the same name based on the signatures of the methods.
- Dynamic binding: Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

As the languages like Objective C, C++ fulfils the above four characteristics yet they are not fully object-oriented languages because they are structured as well as object-oriented languages. But in case of java, it is a fully Object-Oriented language because object is at the outer most level of data structure in java. No stand-alone methods, constants, and variables are there in java. Everything in java is object even the primitive data types can also be converted into object by using the wrapper class.

Robust

Java is the strong memory allocation and automatic garbage collection mechanism. It provides the powerful exception handling and type checking mechanism as compare to other programming languages. Compiler checks the program whether there any error and interpreter check any run time error and makes the system secure from crash. All of the above features make the java language robust.

Distributed

The widely is used protocols like HTTP and FTP are developed in java. Internet programmers can call functions on these protocols and can get access the files from any remote machine on the internet rather than writing codes on their local system.

Portable

The feature Write-once-run-anywhere makes the java language portable provided that the system must have interpreter for the JVM. Java also have the standard data size irrespective of operating system or the processor. These features make the java as a portable language.

Dynamic

While executing the java program the user can get the required files dynamically from a local drive or from a computer thousands of miles away from the user just by connecting with the Internet.

Secure

Java does not use memory pointers explicitly. All the programs in java are run under an area known as the sand box. Security manager determines the accessibility options of a class like reading and writing a file to the local disk. Java uses the public key encryption system to allow the java applications to transmit over the internet in the secure encrypted form. The bytecode Verifier checks the classes after loading.

Performance

Java uses native code usage, and lightweight process called threads. In the beginning interpretation of bytecode resulted the performance slow but the advance version of JVM uses the adaptive and just in time compilation technique that improves the performance.

Multithreaded

As we all know several features of Java like Secure, Robust, Portable, dynamic etc; you will be more delighted to know another feature of Java which is **Multithreaded.**

Java is also a Multithreaded programming language. Multithreading means a single program having different threads executing independently at the same time. Multiple threads execute instructions according to the program code in a process or a program. Multithreading works the similar way as multiple processes run on one computer.

Multithreading programming is a very interesting concept in Java. In multithreaded programs not even a single thread disturbs the execution of other thread. Threads are obtained from the pool of available ready to run threads and they run on the system CPUs. This is how Multithreading works in Java which you will soon come to know in details in

later chapters.

Interpreted

We all know that Java is an interpreted language as well. With an interpreted language such as Java, programs run directly from the source code.

The interpreter program reads the source code and translates it on the fly into computations.

Thus, Java as an interpreted language depends on an interpreter program.

The versatility of being **platform independent** makes Java to outshine from other languages.

The source code to be written and distributed is platform independent.

Another advantage of Java as an interpreted language is its error debugging quality. Due to this any error occurring in the program gets traced. This is how it is different to work with

Java.

Architecture Neutral

The term architectural neutral seems to be weird, but yes Java is an architectural neutral language as well. The growing popularity of networks makes developers think distributed. In the world of network, it is essential that the applications must be able to migrate easily to different computer systems. Not only to computer systems but to a wide variety of hardware architecture and Operating system architectures as well. The Java compiler does this by generating byte code instructions, to be easily interpreted on any machine and to be easily translated into native machine code on the fly. The compiler generates an architecture neutral object file format to enable a Java application to execute anywhere on the network and then the compiled code is executed on many processors, given the presence of the Java runtime system. Hence Java was designed to support applications on network. This feature of Java has thrived the programming language.

3.2 ALGORITHM

1. In login page -

    a) If Sign-in is pressed then => Sign-in class is called and it Proceeds to Sign-in window.
    b) If Sign-up is pressed then => Sing-up class is called and it Proceeds to Sign-up window.

2. In SIGN-UP –

    a) If First tab is selected-

        i. User need to fill their first name and last name and press next button to move on second tab .
        ii. If user want to move to login-in page then they need to press back button.

b) If Second tab is selected-

i. User need to fill their date of birth and gender and press next button to move on third tab.

ii. If user want to go to first tab then they need to press back button.

c) If Third tab is selected-

i. User need to create a new Gmail id then need to press next button to move on fourth tab.

ii. If user want to go to second tab then they need to press back button.

d) If tab is selected-

i. User need to create a strong password mixed of alphabets and numeric then need to press next button and you will get message as “Gmail has been successfully created”.

3. In SIGN-IN-

a) if username = username stored in database and password = password stored in database then => “Login successful” and Proceed to LOGGED.

b) Else “login unsuccessful” if either of username or password is incorrect.

4. In LOGGED-

a) If All mails tab is selected-

i. The message will be displayed according to categories such as inbox, food, otp, banking etc .

b) If Inbox tab is selected-

i. The message will be displayed which belongs to normal category.

c) If OTP tab is selected-

i. The message will be displayed which belongs to otp category.

d) If Food tab is selected-

i. The message will be displayed which belongs to food category.

e) If Banking tab is selected-

i. The message will be displayed which belongs to banking category.

f) If Spam tab is selected-

i. The message will be displayed which belongs to spam category.

g) If Compose tab is selected-

i. We can compose message and can send to whom we want to send and the message will be filtered as which categories they belongs to.

CHAPTER 4

IMPLEMENTATION

This mini project has been implemented using Java Programming Language.

The code for the following project is:

4.1 LOGIN PAGE:

This page allows user to log in the application using there username and password. User will be able to login only if there username and password are stored in the database otherwise they have sign up by providing their personal details such as username and password.

```
Login.java  signin.java  signup.java  signup1.java  signup2.java  signup3.java  compose.java  logged.java
1  package jitender;
2
3  import java.awt.EventQueue;
16
17 public class Login {
18
19     private JFrame frame;
20     private JTextField txtMesssageOrganizer;
21
22     /**
23      * Launch the application.
24      */
25     public static void main(String[] args) {
26         EventQueue.invokeLater(new Runnable() {
27             public void run() {
28                 try {
29                     Login window = new Login();
30                     window.frame.setVisible(true);
31                 } catch (Exception e) {
32                     e.printStackTrace();
33                 }
34             }
35         });
36     }
37
38     /**
39      * Create the application.
40      */
41     public Login() {
42         initialize();
43     }
44
45     /**
46      * Initialize the contents of the frame.
47      */
```

```
Login.java  signin.java  signup.java  signup1.java  signup2.java  signup3.java  compose.java  logged.java
47      */
48     private void initialize() {
49         frame = new JFrame();
50         frame.setBounds(100, 100, 450, 300);
51         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
52         frame.getContentPane().setLayout(null);
53
54         txtMesssageOrganizer = new JTextField();
55         txtMesssageOrganizer.setForeground(Color.MAGENTA);
56         txtMesssageOrganizer.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 14));
57         txtMesssageOrganizer.setText("MESSSAGE ORGANIZER");
58         txtMesssageOrganizer.setBackground(Color.BLACK);
59         txtMesssageOrganizer.setEditable(false);
60         txtMesssageOrganizer.setEnabled(false);
61         txtMesssageOrganizer.setBounds(131, 25, 184, 36);
62         frame.getContentPane().add(txtMesssageOrganizer);
63         txtMesssageOrganizer.setColumns(10);
64
65         JButton btnNewButton = new JButton("SIGN IN");
66         btnNewButton.addActionListener(new ActionListener() {
67             public void actionPerformed(ActionEvent e) {
68                 signin s1 = new signin();
69                 s1.func1();
70             }
71         });
72         btnNewButton.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 14));
73         btnNewButton.setBounds(39, 129, 109, 23);
74         frame.getContentPane().add(btnNewButton);
75
76         JButton signup = new JButton("SIGN UP");
77         signup.addActionListener(new ActionListener() {
78             public void actionPerformed(ActionEvent e) {
79
80                 signup up = new signup();
81                 up.sign1();
```

```java
        txtMesssageOrganizer.setEnabled(false);
        txtMesssageOrganizer.setBounds(131, 25, 184, 36);
        frame.getContentPane().add(txtMesssageOrganizer);
        txtMesssageOrganizer.setColumns(10);

        JButton btnNewButton = new JButton("SIGN IN");
        btnNewButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                signin s1 = new signin();
                s1.func1();
            }
        });
        btnNewButton.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 14));
        btnNewButton.setBounds(39, 129, 109, 23);
        frame.getContentPane().add(btnNewButton);

        JButton signup = new JButton("SIGN UP");
        signup.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

                signup up = new signup();
                up.sign1();
            }
        });
        signup.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 14));
        signup.setBounds(267, 129, 115, 23);
        frame.getContentPane().add(signup);

        JLabel lblNewLabel = new JLabel("");
        lblNewLabel.setIcon(new ImageIcon("C:\\Users\\JITENDRA SAH\\Downloads\\aab.jpg"));
        lblNewLabel.setBounds(0, 0, 436, 263);
        frame.getContentPane().add(lblNewLabel);
    }
}
```

4.2 SIGN UP PAGE:

This page allows user to sign up in the application by giving their username ,password and some personal details.

```java
package jitender;

import java.awt.EventQueue;

public class signup {

    private JFrame frame;
    public static JTextField fname;
    public static JTextField lname;
    private JButton btnNewButton_1;
    private JLabel lblNewLabel_2;

    /**
     * Launch the application.
     */
    public void sign1() {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    signup window = new signup();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the application.
     */
    public signup() {
        initialize();
    }

```

Login.java signin.java signup.java signup1.java signup2.java compose.java logged.java

```java
    /**
     * Initialize the contents of the frame.
     */
    private void initialize() {
        frame = new JFrame();
        frame.setBounds(100, 100, 450, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(null);

        JLabel lblNewLabel = new JLabel("CREATE A ACCOUNT");
        lblNewLabel.setBackground(new Color(240, 240, 240));
        lblNewLabel.setEnabled(false);
        lblNewLabel.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 14));
        lblNewLabel.setBounds(129, 11, 203, 14);
        frame.getContentPane().add(lblNewLabel);

        JLabel lblNewLabel_1 = new JLabel("Enter your name");
        lblNewLabel_1.setForeground(Color.RED);
        lblNewLabel_1.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 12));
        lblNewLabel_1.setBounds(147, 47, 116, 14);
        frame.getContentPane().add(lblNewLabel_1);

        fname = new JTextField();
        fname.setFont(new Font("Tahoma", Font.ITALIC, 12));
        fname.setToolTipText("First name");
        fname.setBounds(81, 86, 203, 20);
        frame.getContentPane().add(fname);
        fname.setColumns(10);

        lname = new JTextField();
        lname.setToolTipText("Last name");
        lname.setBounds(81, 133, 203, 20);
        frame.getContentPane().add(lname);
        lname.setColumns(10);
```

Login.java signin.java signup.java signup1.java signup2.java compose.java logged.java

```java
        frame.getContentPane().add(lname);
        lname.setColumns(10);

        JButton btnNewButton = new JButton("NEXT");
        btnNewButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String fname1 = fname.getText();
                String lname1 = lname.getText();
                if(fname1.equals("")|| lname1.equals("")) {
                    JOptionPane.showMessageDialog(null, "please enter all the fields");
                }
                else {
                new signup1().func4(fname.getText(),lname.getText());
                }
            }
        });
        btnNewButton.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 14));
        btnNewButton.setBounds(237, 207, 89, 23);
        frame.getContentPane().add(btnNewButton);

        btnNewButton_1 = new JButton("BACK");
        btnNewButton_1.addActionListener(new ActionListener() {
            @SuppressWarnings("static-access")
            public void actionPerformed(ActionEvent e) {
                new Login().main(null);
            }
        });
        btnNewButton_1.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 12));
        btnNewButton_1.setBounds(49, 209, 89, 23);
        frame.getContentPane().add(btnNewButton_1);

        lblNewLabel_2 = new JLabel("");
        lblNewLabel_2.setIcon(new ImageIcon("C:\\Users\\JITENDRA SAH\\Downloads\\aab1.png"));
        lblNewLabel_2.setBounds(0, -18, 436, 281);
        frame.getContentPane().add(lblNewLabel_2);
```

Login.java signin.java signup.java signup1.java signup2.java signup3.java compose.java logged.java

```java
package jitender;

import java.awt.EventQueue;

public class signup1 {

    private JFrame frame;
    private JTextField day;
    private JTextField month;
    private JTextField year;
    private JTextField gender;

    public static String fname,lname;
    private JLabel lblNewLabel_2;

    /**
     * Launch the application.
     */
    public void func4(String first,String last) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    fname = first;
                    lname = last;
                    //System.out.println(fname+lname);
                    signup1 window = new signup1();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
```

4.3 SIGN IN PAGE:

This page allows user to log in the application using there username and password. User will be able to login only if their username and password are stored in the database.

```
Login.java  signin.java  signup.java  signup1.java  signup2.java  signup3.java  compose.java  logged.java
1  package jitender;
2
3  import java.awt.EventQueue;
21
22 public class signin {
23
24     private JFrame frame;
25     private JTextField email1;
26     private JPasswordField passwordField;
27     int count =0,total=0;
28     //declaration for databases
29     Connection connection = null;
30     ResultSet resultSet = null;
31
32     //resultset1 declared to get count of the data
33     ResultSet resultSet1 = null;
34
35     Statement statement = null;
36
37     public static String email;
38
39     /**
40      * Launch the application.
41      */
42     public void func1() {
43         EventQueue.invokeLater(new Runnable() {
44             public void run() {
45                 try {
46                     signin window = new signin();
47                     window.frame.setVisible(true);
48                 } catch (Exception e) {
49                     e.printStackTrace();
50                 }
51             }
52         });
```

```
Login.java  signin.java  signup.java  signup1.java  signup2.java  signup3.java  compose.java  logged.java
55     /**
56      * Create the application.
57      */
58     public signin() {
59         initialize();
60     }
61
62     /**
63      * Initialize the contents of the frame.
64      */
65     private void initialize() {
66         frame = new JFrame();
67         frame.setBounds(100, 100, 434, 300);
68         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
69         frame.getContentPane().setLayout(null);
70
71         JButton btnNewButton = new JButton("LOGIN");
72         btnNewButton.addActionListener(new ActionListener() {
73             public void actionPerformed(ActionEvent e) {
74                 if(email1.getText().equals("")||passwordField.getText().equals("")) {
75                     JOptionPane.showMessageDialog(null, "please enter all the field");
76                 }
77                 else {
78                     try
79                         {
80                         Class.forName("org.sqlite.JDBC");
81                         connection = DriverManager.getConnection("jdbc:sqlite:D:\\customer.db");
82                         statement = connection.createStatement();
83                         resultSet1 = statement.executeQuery("SELECT COUNT(*) FROM customers");
84 //                      System.out.println(resultSet1.getString(1));
85                         total = Integer.parseInt(resultSet1.getString(1));
86                         System.out.println(total);
87
88                         resultSet = statement
```

```
Login.java  signin.java  signup.java  signup1.java  signup2.java  signup3.java  compose.java  logged.java
90
91                         while (resultSet.next())
92                         {
93                             count ++;
94                             if(email1.getText().equals(resultSet.getString(3))||passwordField.getText().equals(results
95                                 email = email1.getText();
96                                 logged lll = new logged();
97                                    lll.func2();
98                                    break;
99                             }
100
101                        }
102                        if(count==total) {
103                            JOptionPane.showMessageDialog(null, "please enter the correct email and password");
104                        }
105                    }
106                    catch (Exception e1)
107                    {
108                        e1.printStackTrace();
109                    }
110                    finally
111                    {
112                        try
113                        {
114                            resultSet.close();
115                            statement.close();
116                            connection.close();
117                        }
118                        catch (Exception e1)
119                        {
120                            e1.printStackTrace();
121                        }
122                    }
123                }
```

```java
    private JFrame frame;

    /**
     * Launch the application.
     */
    public void func2() {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    logged window = new logged();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the application.
     */
    public logged() {
        initialize();
    }

    /**
     * Initialize the contents of the frame.
     */
    private void initialize() {
        email.setBounds(20, 26, 70, 20);
        frame.getContentPane().add(email);

        JLabel password = new JLabel("PASSWORD");
        password.setForeground(Color.ORANGE);
        password.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 14));
        password.setBackground(new Color(240, 240, 240));
        password.setEnabled(false);
        password.setBounds(20, 76, 112, 23);
        frame.getContentPane().add(password);

        JButton btnNewButton_1 = new JButton("BACK");
        btnNewButton_1.setBackground(Color.GRAY);
        btnNewButton_1.addActionListener(new ActionListener() {
            @SuppressWarnings("static-access")
            public void actionPerformed(ActionEvent e) {
                Login l1 = new Login();
                l1.main(null);
            }
```

Login.java | signin.java | signup.java | signup1.java | signup2.java | signup3.java | compose.java | logged.java

```java
        frame.getContentPane().add(email);

        JLabel password = new JLabel("PASSWORD");
        password.setForeground(Color.ORANGE);
        password.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 14));
        password.setBackground(new Color(240, 240, 240));
        password.setEnabled(false);
        password.setBounds(20, 76, 112, 23);
        frame.getContentPane().add(password);

        JButton btnNewButton_1 = new JButton("BACK");
        btnNewButton_1.setBackground(Color.GRAY);
        btnNewButton_1.addActionListener(new ActionListener() {
            @SuppressWarnings("static-access")
            public void actionPerformed(ActionEvent e) {
                Login l1 = new Login();
                l1.main(null);
            }
        });
        btnNewButton_1.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 14));
        btnNewButton_1.setBounds(71, 167, 89, 23);
        frame.getContentPane().add(btnNewButton_1);

        passwordField = new JPasswordField();
        passwordField.setEchoChar('*');
        passwordField.setBounds(155, 75, 195, 20);
        frame.getContentPane().add(passwordField);

        JLabel lblNewLabel = new JLabel("");
        lblNewLabel.setIcon(new ImageIcon("C:\\Users\\JITENDRA SAH\\Downloads\\aab10.jpg"));
        lblNewLabel.setBounds(-11, -17, 1951, 1108);
        frame.getContentPane().add(lblNewLabel);
    }

}
```

4.4 LOGGED PAGE:

This page shows all the messages according to the categories such as OTP category, BANK category, FOOD category, INBOX category, ALLMAILS category & we can compose message in this page.

Login.java | signin.java | signup.java | signup1.java | signup2.java | signup3.java | compose.java | logged.java

```java
    private void initialize() {
        frame = new JFrame();
        frame.setBounds(100, 100, 631, 503);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(null);

        JButton btnNewButton = new JButton("ALL MAILS");
        btnNewButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                allmail m1 = new allmail();
                m1.main(null);
            }
        });
        btnNewButton.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 18));
        btnNewButton.setBounds(10, 57, 137, 38);
        frame.getContentPane().add(btnNewButton);

        JButton btnNewButton_1 = new JButton("SPAM");
        btnNewButton_1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                spam s1 = new spam();
                s1.spam1();
            }
        });
        btnNewButton_1.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 18));
        btnNewButton_1.setBounds(413, 228, 165, 38);
        frame.getContentPane().add(btnNewButton_1);

        JButton btnNewButton_2 = new JButton("OTP");
        btnNewButton_2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

                otp o1 = new otp();
                o1.otp1();
            }
```

```java
        btnNewButton_4.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 18));
        btnNewButton_4.setBounds(10, 362, 137, 38);
        frame.getContentPane().add(btnNewButton_4);

        JButton btnNewButton_5 = new JButton("BANKING");
        btnNewButton_5.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                bank b1 = new bank();
                b1.bank1();
            }
        });
        btnNewButton_5.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 18));
        btnNewButton_5.setBounds(413, 73, 165, 38);
        frame.getContentPane().add(btnNewButton_5);

        JButton btnNewButton_6 = new JButton("INBOX");
        btnNewButton_6.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                inbox i1 = new inbox();
                i1.inbox1();
            }
        });
        btnNewButton_6.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 16));
        btnNewButton_6.setForeground(UIManager.getColor("InternalFrame.inactiveTitleForeground"));
        btnNewButton_6.setBackground(UIManager.getColor("FormattedTextField.selectionBackground"));
        btnNewButton_6.setBounds(224, 183, 124, 46);
        frame.getContentPane().add(btnNewButton_6);

        JLabel lblNewLabel_1 = new JLabel("");
        lblNewLabel_1.setIcon(new ImageIcon("C:\\Users\\JITENDRA SAH\\Downloads\\jkjkj.jpg"));
        lblNewLabel_1.setBounds(-12, 0, 629, 466);
        frame.getContentPane().add(lblNewLabel_1);
    }
}
```

```java
        });
        btnNewButton_2.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 18));
        btnNewButton_2.setBounds(10, 215, 137, 38);
        frame.getContentPane().add(btnNewButton_2);

        JButton btnNewButton_3 = new JButton("COMPOSE");
        btnNewButton_3.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                compose c1 = new compose();
                c1.func3();
            }
        });
        btnNewButton_3.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 18));
        btnNewButton_3.setBounds(427, 362, 165, 38);
        frame.getContentPane().add(btnNewButton_3);

        JLabel lblNewLabel = new JLabel("CHOOSE");
        lblNewLabel.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 14));
        lblNewLabel.setBounds(217, 11, 111, 38);
        frame.getContentPane().add(lblNewLabel);

        JButton btnNewButton_4 = new JButton("FOOD");
        btnNewButton_4.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {

                food fo = new food();
                fo.food1();
            }
        });
        btnNewButton_4.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 18));
        btnNewButton_4.setBounds(10, 362, 137, 38);
        frame.getContentPane().add(btnNewButton_4);

        JButton btnNewButton_5 = new JButton("BANKING");
        btnNewButton_5.addActionListener(new ActionListener() {
```

4.5 COMPOSE MESSAGE PAGE:

This page allows to compose message and the message will be organized according to the categories and the message get store in the file and the database.

```java
package jitender;

import java.awt.EventQueue;

public class compose extends signin{

    private JFrame frame;
    private JTextField from;
    private JTextField to;
    private JTextField subject;
    private JButton btnNewButton_1;
    private JButton btnNewButton_2;

    public static String email1;

    /**
     * Launch the application.
     */
    public void func3() {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    email1 = signin.email;
                    compose window = new compose();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the application.
     */
```

Login.java signin.java signup.java signup1.java signup2.java signup3.java compose.java logged.java

```java
 */
public compose() {
    initialize();
}

/**
 * Initialize the contents of the frame.
 */
private void initialize() {
    frame = new JFrame();
    frame.setBounds(0, 0, 550, 650);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().setLayout(null);

    JLabel lblNewLabel = new JLabel("FROM");
    lblNewLabel.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 13));
    lblNewLabel.setBounds(10, 11, 49, 14);
    frame.getContentPane().add(lblNewLabel);

    JLabel lblNewLabel_1 = new JLabel("TO");
    lblNewLabel_1.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 13));
    lblNewLabel_1.setBounds(10, 62, 49, 14);
    frame.getContentPane().add(lblNewLabel_1);

    from = new JTextField();
    from.setEnabled(false);
    from.setEditable(false);
    from.setText(email1);
    from.setBackground(new Color(255, 255, 255));
    from.setFont(new Font("Tahoma", Font.ITALIC, 12));
    from.setBounds(90, 9, 305, 20);
    frame.getContentPane().add(from);
    from.setColumns(10);

    to = new JTextField();
```

Login.java signin.java signup.java signup1.java signup2.java signup3.java compose.java logged.java

```java
    to.setBounds(90, 60, 305, 20);
    frame.getContentPane().add(to);
    to.setColumns(10);

    JLabel lblNewLabel_2 = new JLabel("SUBJECT");
    lblNewLabel_2.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 13));
    lblNewLabel_2.setBounds(10, 124, 71, 14);
    frame.getContentPane().add(lblNewLabel_2);

    subject = new JTextField();
    subject.setBounds(90, 122, 305, 20);
    frame.getContentPane().add(subject);
    subject.setColumns(10);

    JTextArea emailtext = new JTextArea();
    emailtext.setToolTipText("Compose email");
    emailtext.setFont(new Font("Monospaced", Font.ITALIC, 13));
    emailtext.setTabSize(13);
    emailtext.setBounds(10, 162, 481, 351);
    frame.getContentPane().add(emailtext);

    btnNewButton_1 = new JButton("SEND");
    btnNewButton_1.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            //to check all the field is entered
            if(from.getText().equals("") || to.getText().equals("") || emailtext.getText().equals("") || subject.ge
                JOptionPane.showMessageDialog(null, "please enter all the field ");
            }

            else {
                String message = emailtext.getText();
                if(message.contains("otp")) {
                    FileWriter fw;
                    try {
                        fw = new FileWriter("D://file//otp//new.txt");
```

Login.java signin.java signup.java signup1.java signup2.java signup3.java compose.java logged.java

```java
                        fw = new FileWriter("D://file//otp//new.txt");
                        for (int i = 0; i < message.length(); i++)
                            fw.write(message.charAt(i));

                        System.out.println("Writing successful");
                        //close the file
                        fw.close();
                        JOptionPane.showMessageDialog(null, "MESSAGE SUCCESSFULLY SEND AND FILTERED AS OTP");
                        frame.dispose();

                    } catch (IOException e1) {
                        // TODO Auto-generated catch block
                        System.out.println(e);
                    }

                }

                else if(message.contains("loan") || message.contains("insurance")) {
                    FileWriter fw;
                    try {
                        fw = new FileWriter("D://file//spam//new.txt");
                        for (int i = 0; i < message.length(); i++)
                            fw.write(message.charAt(i));

                        System.out.println("Writing successful");
                        //close the file
                        fw.close();
                        JOptionPane.showMessageDialog(null, "MESSAGE SUCCESSFULLY SEND AND FILTERED AS SPAM");

                        frame.dispose();
                    } catch (IOException e1) {
                        // TODO Auto-generated catch block
                        System.out.println(e);
                    }
```

4.6 MESSAGE ORGANIZER PAGE:

This page will show the messages of the mail according to the category.

```java
package jitender;

import java.awt.EventQueue;

public class allmail {

    private JFrame frame;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    allmail window = new allmail();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the application.
     */
    public allmail() {
        initialize();
    }

    /**
     * Initialize the contents of the frame.
     */
    private void initialize() {
```

```java
        frame = new JFrame();
        frame.setBounds(100, 100, 450, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(null);

        JButton btnNewButton = new JButton("OTP");
        btnNewButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                otp o1 = new otp();
                o1.otp1();
            }
        });
        btnNewButton.setBounds(10, 148, 132, 33);
        frame.getContentPane().add(btnNewButton);

        JButton btnNewButton_1 = new JButton("FOOD");
        btnNewButton_1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                food f1 = new food();
                f1.food1();
            }
        });
        btnNewButton_1.setBounds(276, 45, 139, 33);
        frame.getContentPane().add(btnNewButton_1);

        JButton btnNewButton_2 = new JButton("BANKING");
        btnNewButton_2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                bank b1 = new bank();
                b1.bank1();
            }
        });
        btnNewButton_2.setBounds(276, 148, 139, 33);
        frame.getContentPane().add(btnNewButton_2);
```

```java
            public void actionPerformed(ActionEvent e) {
                bank b1 = new bank();
                b1.bank1();
            }
        });
        btnNewButton_2.setBounds(276, 148, 139, 33);
        frame.getContentPane().add(btnNewButton_2);

        JButton btnNewButton_3 = new JButton("BACK");
        btnNewButton_3.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                logged l1 = new logged();
                l1.func2();
            }
        });
        btnNewButton_3.setBounds(152, 208, 118, 33);
        frame.getContentPane().add(btnNewButton_3);

        JButton btnNewButton_4 = new JButton("INBOX");
        btnNewButton_4.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                inbox i1 = new inbox();
                i1.inbox1();
            }
        });
        btnNewButton_4.setBounds(10, 45, 132, 33);
        frame.getContentPane().add(btnNewButton_4);

        JLabel lblNewLabel = new JLabel("");
        lblNewLabel.setIcon(new ImageIcon("C:\\Users\\JITENDRA SAH\\Downloads\\aab6.jpg"));
        lblNewLabel.setBounds(-25, -23, 1046, 1534);
        frame.getContentPane().add(lblNewLabel);
    }
}
```

4.7 OTP CATEGORY PAGE:

This page will show all the message of otp category.

```java
package jitender;

import java.awt.EventQueue;

public class otp {

    private JFrame frame;
    //public static String path;

    /**
     * Launch the application.
     */
    public void otp1() {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    otp window = new otp();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the application.
     */
    public otp() {
        initialize();
    }

    /**
     * Initialize the contents of the frame.
     */
```

```java
    private void initialize() {
        String[] name = new String[10];
        int count =0;
        //path ="D://file//otp";
        try {
            File file = new File("D://file//otp");
            File[] files = file.listFiles();
            for(File f: files){
                System.out.println(f.getName());
                name[count] = f.getName();
                count = count+1;
            }
        }
        catch(Exception e) {
            System.out.println(e);
        }

        frame = new JFrame();
        frame.setBounds(100, 100, 450, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(null);

        JLabel lblNewLabel = new JLabel("THE OTP YOU HAVE SEND ARE");
        lblNewLabel.setForeground(Color.YELLOW);
        lblNewLabel.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 16));
        lblNewLabel.setBounds(68, 11, 302, 36);
        frame.getContentPane().add(lblNewLabel);

        JButton button1 = new JButton(name[0]);
        button1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                //path = path+"//"+button1.getText();
                //System.out.println(path);
                view_message v1 = new view_message();
                v1.show(button1.getText());
```

```java
        button1.setBounds(10, 73, 89, 23);
        //button1.setText(name[0]);
        frame.getContentPane().add(button1);

        JButton button2 = new JButton(name[1]);
        button2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
//                path = path+"//"+button2.getText();
//                System.out.println(path);
                view_message v1 = new view_message();
                v1.show(button2.getText());
            }
        });

        button2.setBounds(243, 73, 89, 23);
        //button2.setText(name[1]);
        frame.getContentPane().add(button2);

        JButton button3 = new JButton(name[3]);
        button3.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
//                path = path+"//"+button3.getText();
//                System.out.println(path);
                view_message v1 = new view_message();
                v1.show(button3.getText());            }
        });

        button3.setBounds(10, 137, 89, 23);
        button3.setText(name[2]);
        frame.getContentPane().add(button3);

        JButton button4 = new JButton(name[4]);
        button4.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
```

```java
        button3.setBounds(10, 137, 89, 23);
        button3.setText(name[2]);
        frame.getContentPane().add(button3);

        JButton button4 = new JButton(name[4]);
        button4.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
//                path = path+"//"+button1.getText();
//                System.out.println(path);
                view_message v1 = new view_message();
                v1.show(button4.getText());
            }
        });

        button4.setBounds(243, 137, 89, 23);
        button4.setText(name[3]);
        frame.getContentPane().add(button4);

        JButton btnNewButton_4 = new JButton("BACK");
        btnNewButton_4.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                logged l1 = new logged();
                l1.func2();
            }
        });
        btnNewButton_4.setBounds(154, 229, 89, 23);
        frame.getContentPane().add(btnNewButton_4);

        JLabel lblNewLabel_1 = new JLabel("");
        lblNewLabel_1.setIcon(new ImageIcon("C:\\Users\\JITENDRA SAH\\Downloads\\aab7.jpg"));
        lblNewLabel_1.setBounds(10, 11, 728, 410);
        frame.getContentPane().add(lblNewLabel_1);
    }
}
```

CHAPTER 5

RESULTS

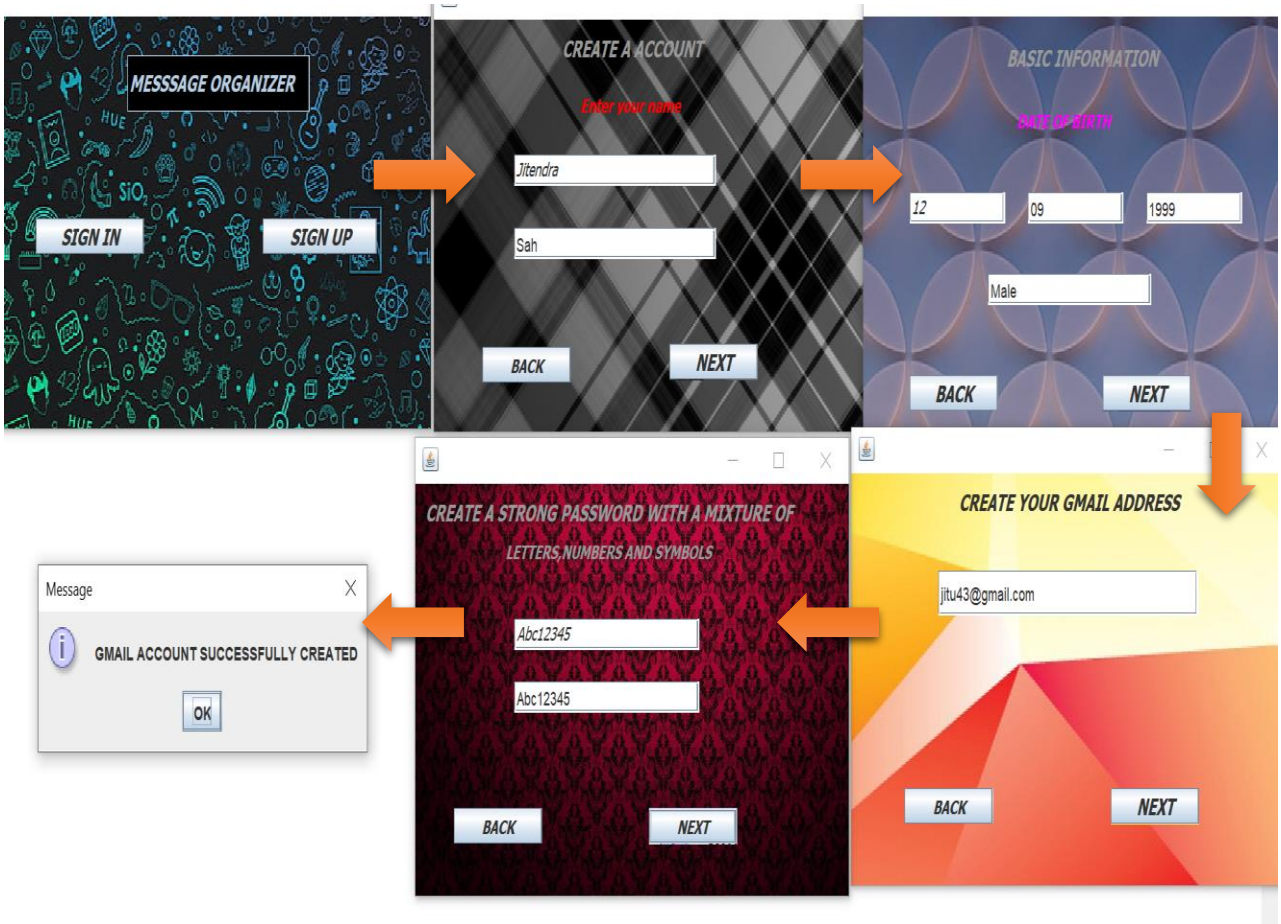


Fig 5.1: - Sign-up Steps

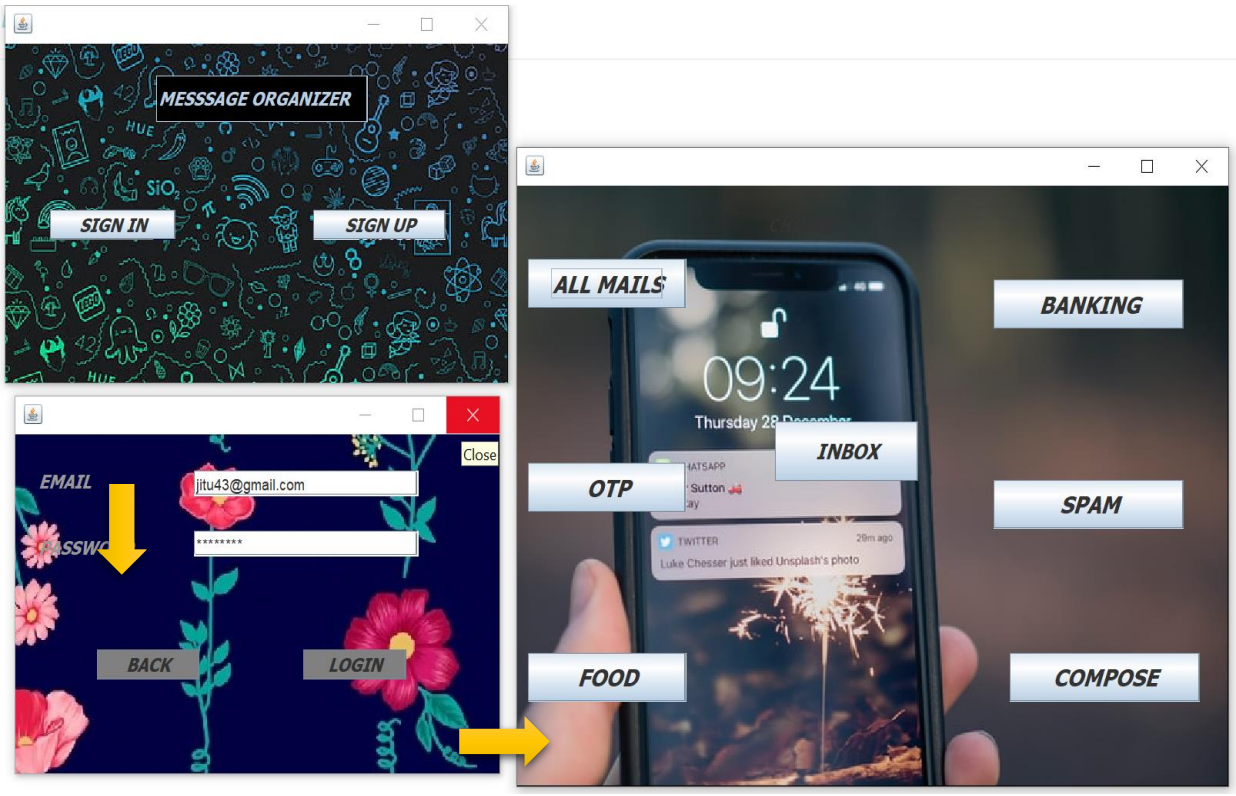


fig 5.2: - Sign-in Steps

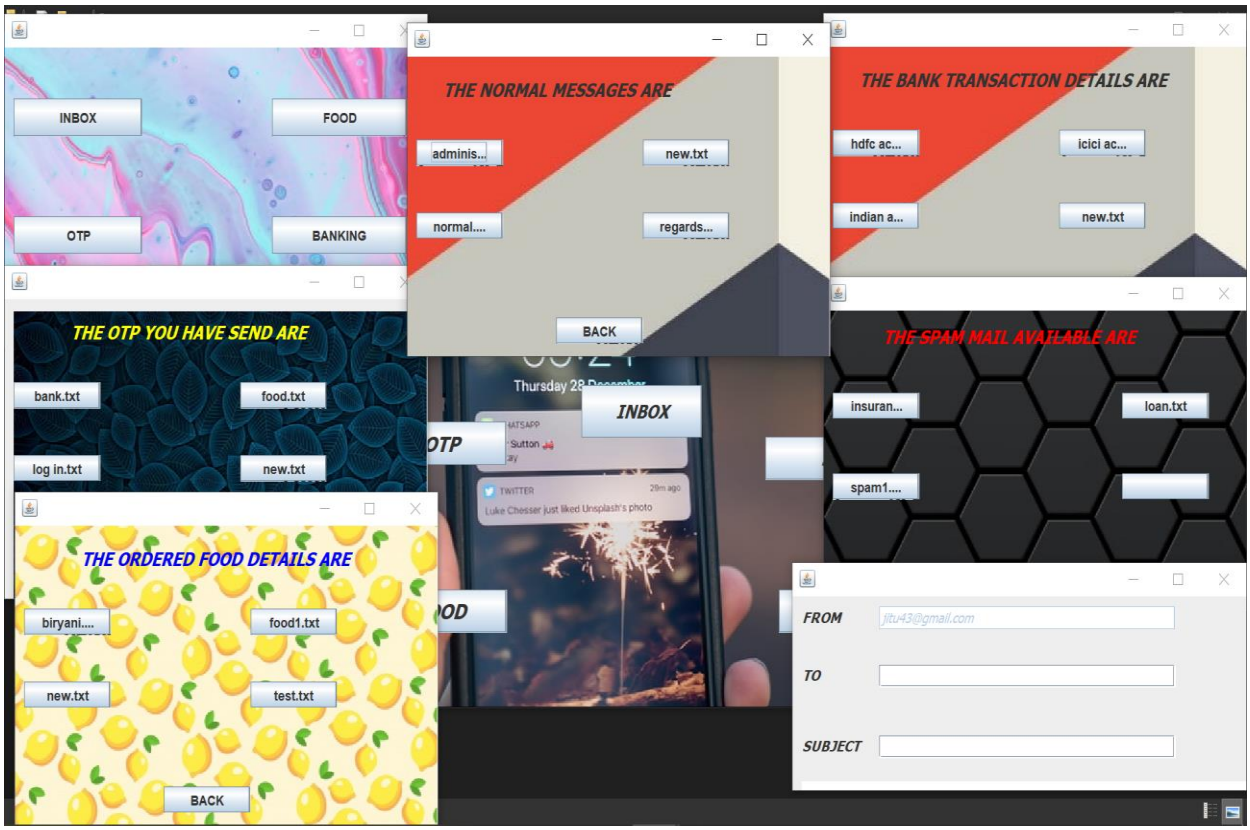


Fig 5.3: - Categorized messages

CHAPTER 6

CONCLUSION

The Text field won't take any invalid username and password. Every time it will match the id against the values entered in login table in the database providing a secure way of working. After login whatever the data is shown is with respect to the user.

This software makes user friendly environment for the user to get the job done exactly what they hope this software to offer. The user can simply select the buttons provided to complete the desired functionality. if the user wants to see All mails. he/she can simply click on "All mails" button and the output will be generated based on the user's entered/provided data. Simply by pressing "Compose" button can help user to compose the messages to send anyone and the software helps us to identify that the sent message belongs to which categories.

CHAPTER 7

REFERENCES

1. https://www.ntu.edu.sg/home/ehchua/programming/java/J4a_GUI.html
2. https://www.javatpoint.com/java-jdbc
3. https://www.javatpoint.com/java-swing