****

**Netaji Subhash Institute Of Technology**

**University of Delhi,New Delhi**

**CEC07-Object Oriented Programming**

**Project Report On**

**Easy\_Note\_lite – Scratchpad**

**Submitted by-**

**Raghav Kumar, 2016UCO1640,COE3**

**Aryaman Mohapatra,2016UCO1643,COE3**

**(Teacher Incharge)**

**1. Problem Statement -**

Everyone face the need to note down some small piece of information that is important, private and should be available easily. The text editors that already exist are robust but they pose the following problems -

1. You need to create a file for every note, or you have to navigate to them everytime.

2. These files can be password protetcted but again that requires tweaking certain settings, if this feature is availabe at all.

The demanding criteria here is the **speed** of **accessing** one’s notes and **encrypting** them.

Thus we came up with the idea of **Easy\_Note\_lite – Sketchpad**.

**Sketchpad** is a **single screen** java Application that can be used to

1. Take down notes and save them with one click.

2. The notes are available as soon as the app fires up. There is no longer any need to remember where you saved your files.

3. If you do not want anyone to read what you have written you can encrypt them using 3 different encryption ciphers.

4. Only you can make any permanent changes to your notes because every major action can be performed only after entering your password.

Also since Easy\_Note\_lite – Scratchpad is written in java it can be accessed on multiple platforms and your notes can be made available acroos platforms too if the JAR file is placed in an appropriate directory.

**2.Object Orientation -**

This application is developed using **javax.swing** package and self developed **encrypt** package. So we will be understanding the involved object orentation from the perspective of a user defined package as well as an in-built package.

**A. encrypt Package:**

The encrypt package consists of one abstract class Encryption and its three derived classes viz. BCipher, CeaserShiftCipher and AtabashCipher.

The abstract class Encryption defines how any encryption should look like. All the encryption classes might be using some kind of “key” that may be a string or a character. Well it is imperative that this key is well protetcted(Data Hiding). To ensure this it is declared as a private member and cn be accessed using the String getKey() and void setKet(String str) methods.

Also all encryption classes must have functions that can codify and decode the messages passed to them. The rules of encoding and hence decoding will be different for different classes but they will be there. Also since the setKey and getKey have pretty straight forward funtions to set and extract the key for the encryption respectively, they are not required to be overridden by any derived class thus it is safe to declare them as **final**.

Our base class Encryption thus ensures that any fututre encryption class extending from it must fulfill the sturcture and behaviour specified by it. The derived classes may employ fairly complex algorithms to encode and decode text and numbers but they all will share the same structure defined by Encryption.

**B. javax.swing Package:**

We have used **javax.swing package** to make our graphics user interface. What this package has done is that it has defined the structure and behaviour of various different elements like Jframes,

Jbuttons, ActionListeners etc. The user of this package can use these definitions to create his/her own classes that inherit or implement structures defined in the package, override the methods that can be over ridden and emulate the desired behaviour.

We will take the example of the classes and interfaces that we have used in our software. An important thing that must be brought to notice here is that if we keep the input and the output i.e. the interface of any class or method unchanged then it just does not matter to other classes or methods how that method or class is achieving its results. That is to say that we have achieved **abstraction** and **encapsulation**.

**Abstraction** is achieved as now no two classes need to be aware of how the other class is performing its functions, they only need to know what functions the other class can perform.

**Encapsulation** is achieved as now the internal algorithms of the class or method are no longer affected by changes in the internal algorithms of any other classes or methods, as long as their functions do not change.

Also it should be noted that this package was written by people who none of us know and who themselves were unknown to how their code might be used. This level of **reusability** of code is simply great.

**(I) JFrame:** JFrame is a top level window with title and border, this means that any program must have at least one frame to have a GUI.In our case we have used three classes that extend Jframe viz. Main.java, PasswordPrompt.java and PasswordChange.java. These three classes differ widely in what they do but since all of them extend JFrame they define the functions needed to render a window on screen.

In Main.java we have **overridden** Main(String s) constructor in order to pass the password instead of passing title. In this class only we have defined many functions that run the entire app.

MAIN class also implements interface ActionListener, this means that is eligible to respond to any input that is happening on anyone of its component. Hence we define its actionPerformed function to achieve our desired result.

**(II) JButtons:** All the components that are rendered as buttons on the screen are of this type. These buttons can interact with any other visible(to the JButton object) object of any other class to produce the desired result of the action of pressing them. How pressing a button works is unknown to the programmer beyond a certain level of abstration but still he/she is able to work with it cause he/she knows what functions can be performed with any object of type JButton.

**(III)JTextArea:** A major part of out screen is covered by an object of this type.

**(IV)JPasswordField:** This class is used to get password from the user. Now how it renders the ‘\*’ in place of the actual keys pressed is hidden from us but still we are able to use the feature.

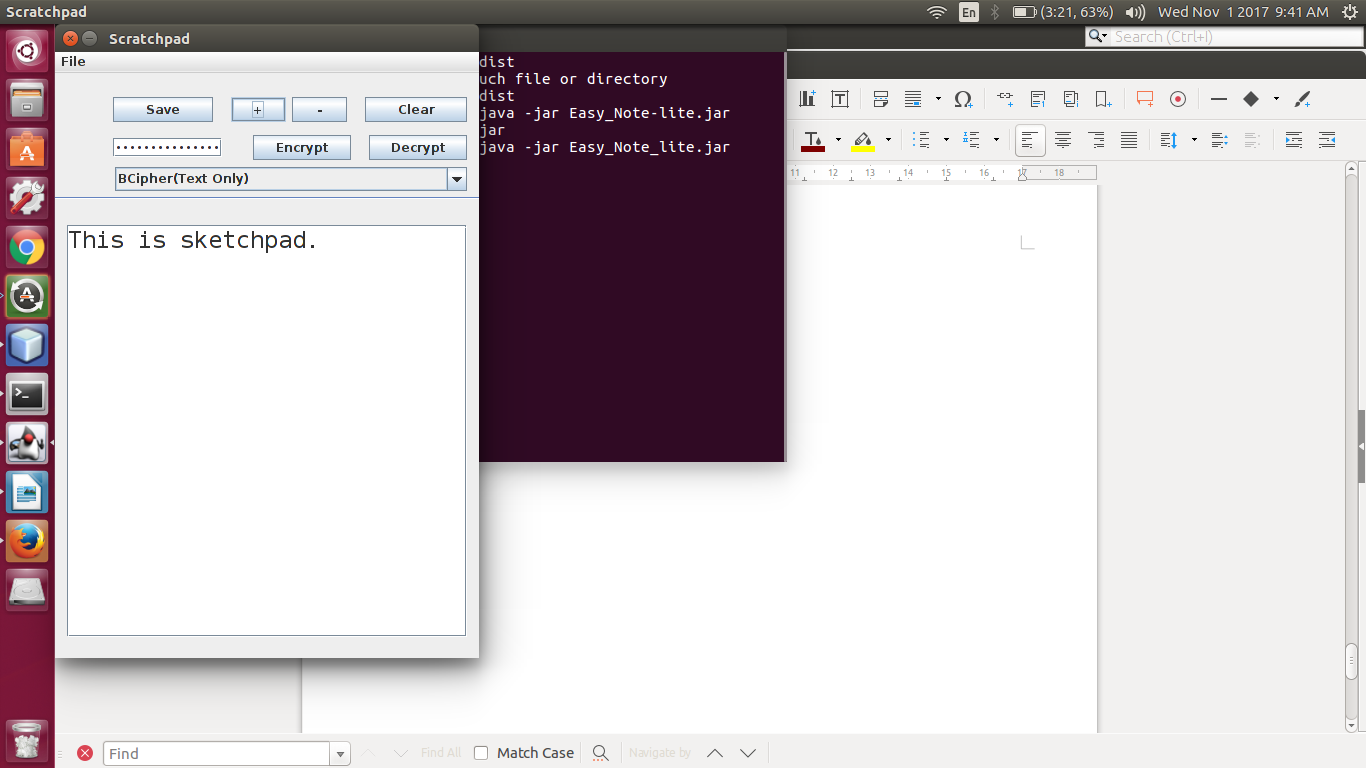
**(V)Other Components Used : JTextPane,** **Jmenubar, JMenuItem, Jseperator, JLabel, JComboBox.**

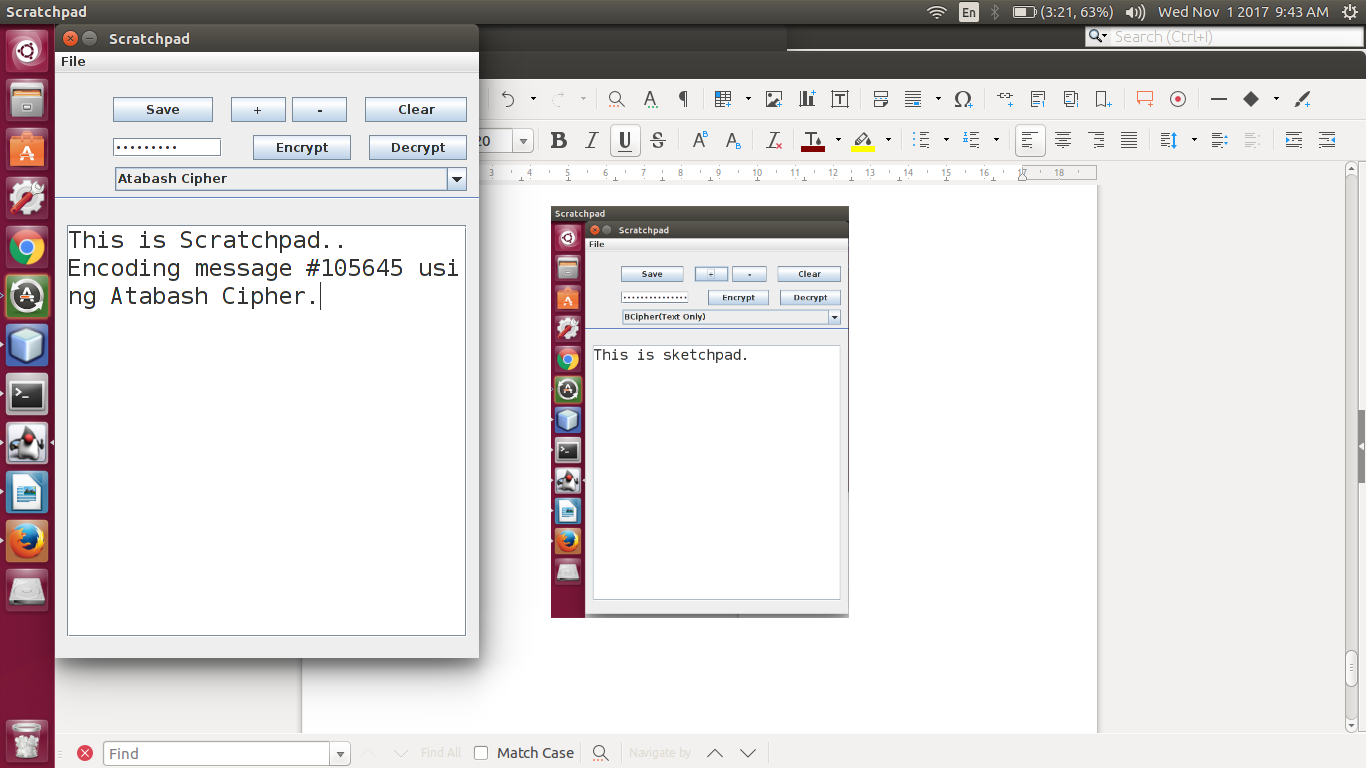
The MAIN class is **associated** with all the classes mentioned above. MAIN class **is a** JFrame and **has a** Jbutton,Jpane,JTextArea etc. That is to say that MAIN class will have **composite** objects.

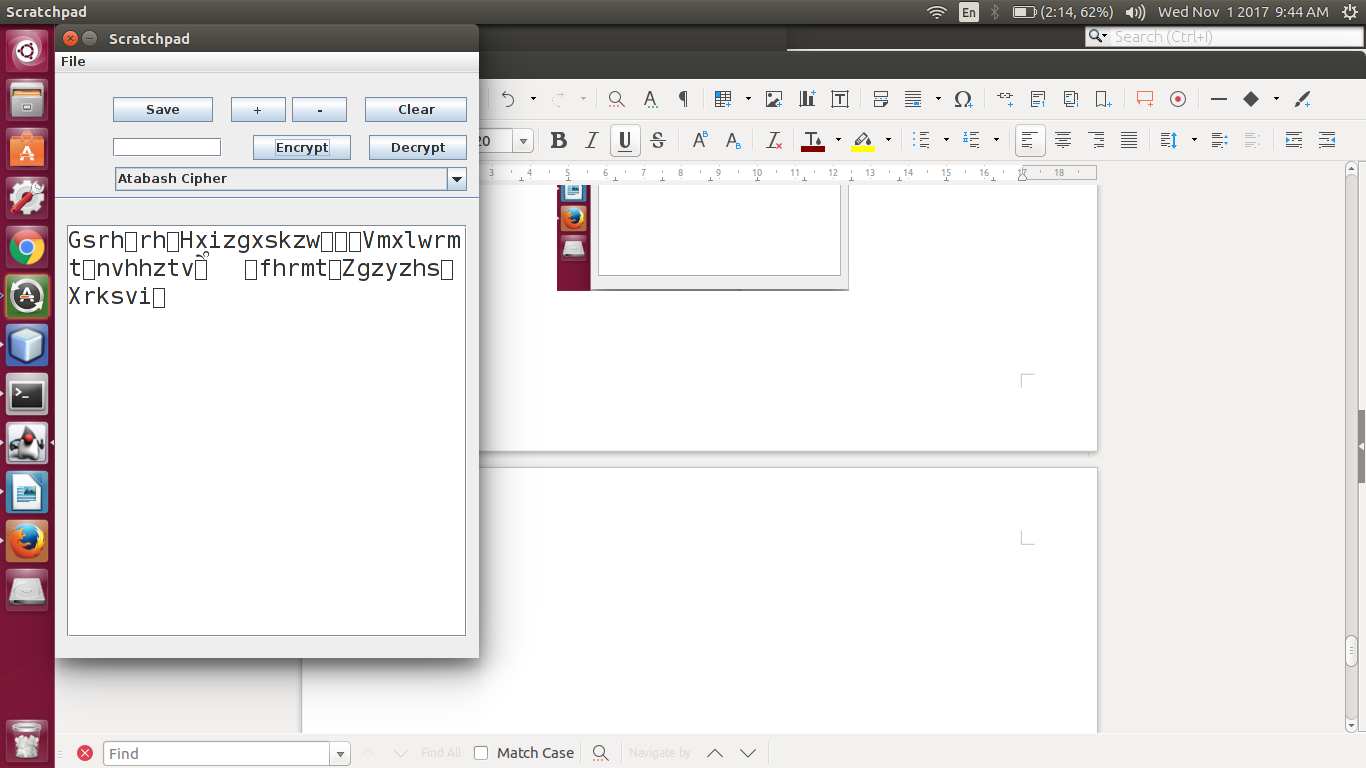
Another interesting concept of OOPs that we have used is that we have used the reference to a base class to hold an object of derived class. This feture can come in very handy. For example when one is unsure of what type of exception may be thrown by a function, or when a function can throw from an overwhelming variety of exceptions the programmer can simply declare a reference of type Exception to catch any of those exceptions. This is possible cause all the exceptions derived from the inbuilt class Exception.

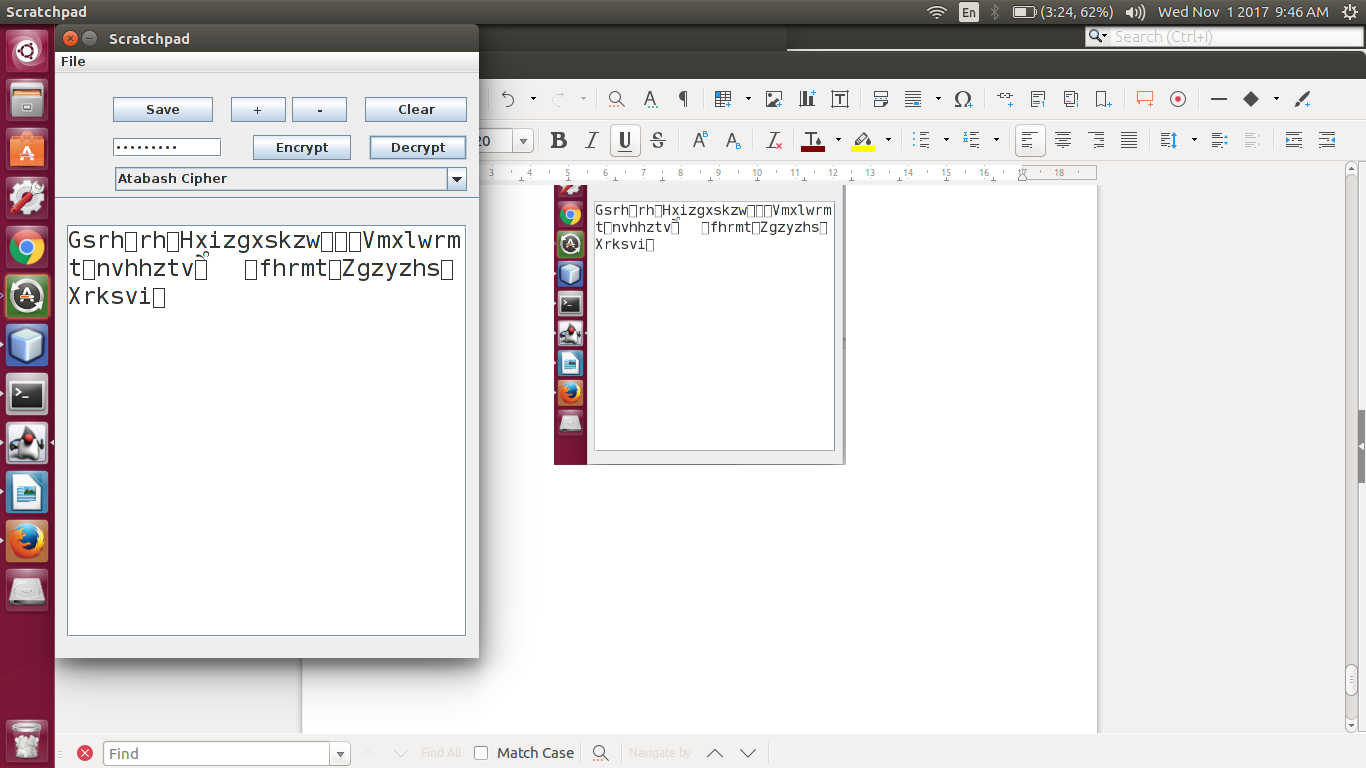
We have seen how we used OOPs concepts to our benefit that even though we are ignorant of many of the functionalities and intricacies of GUI components, still we were able to develop a stable graphical app. The concepts of Inheritance, Abstraction, Encapsulation, Association, Data Hiding etc. can be used to get a lot of leverage while developing software. The very reason that the mostly used programming paradigm today is Object Oriented is because it let’s us focus on the information we want to process and removes from our sight the deviating concerns of how to go about it. The fact that the code I have written can be used by a complete stranger also gives rise to a framework where people can take off from places that were reached by someone before them. This framework allows exponential growth in the field of development.

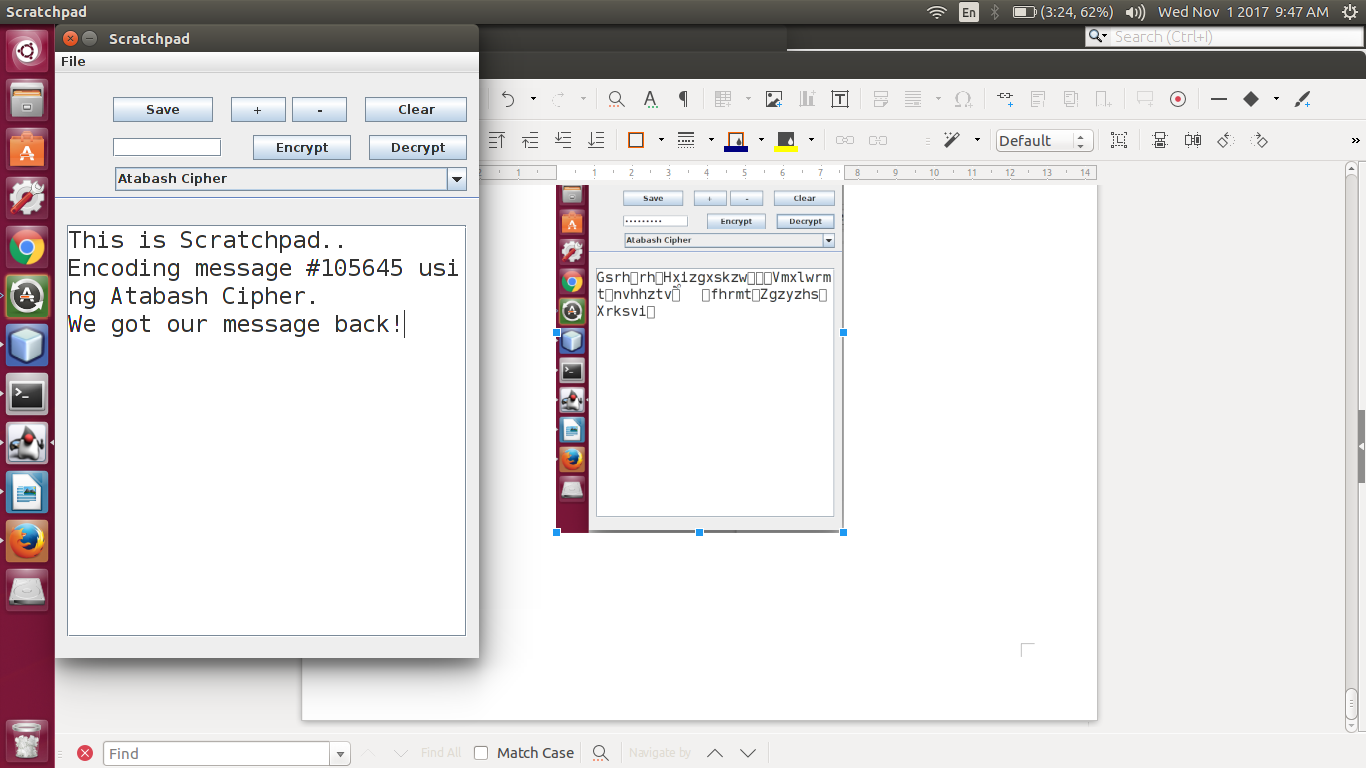
**3.Easy\_Notes\_lite – Scratchpad:**

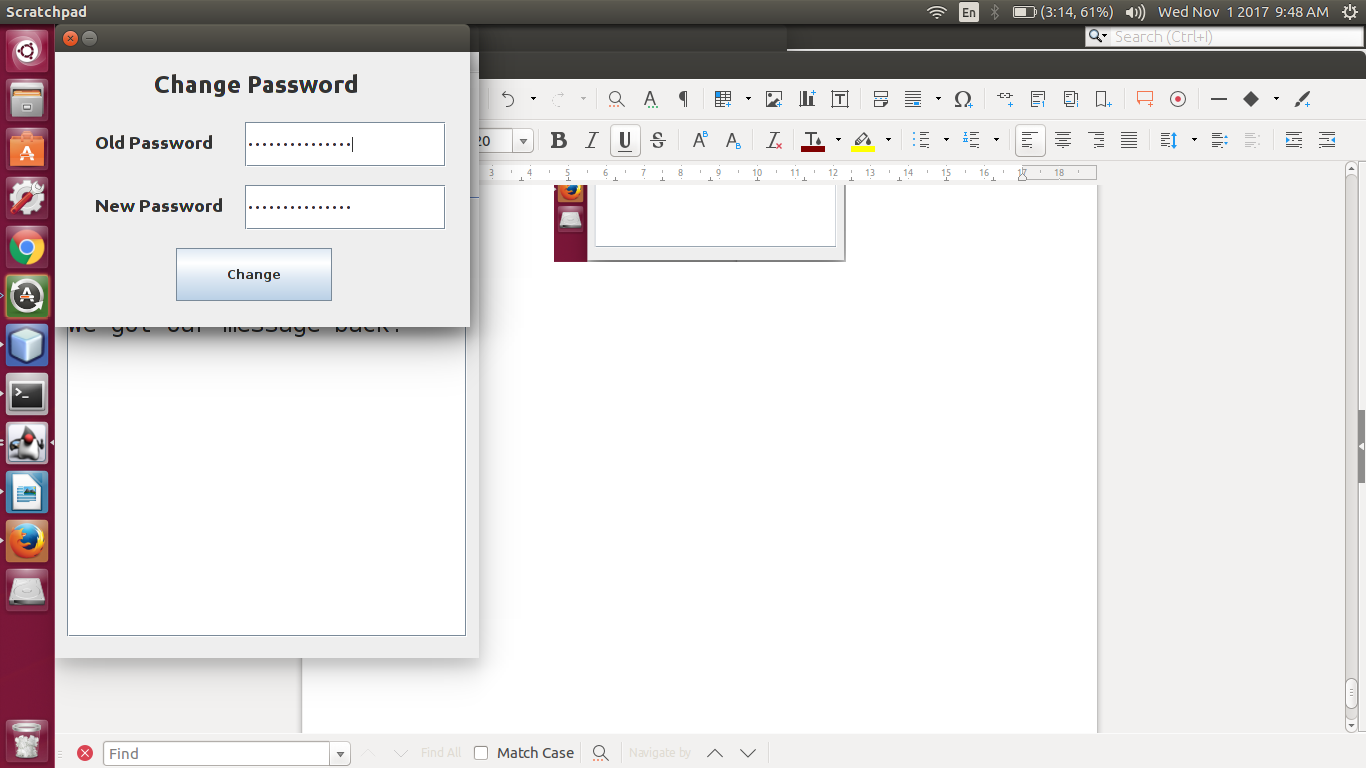


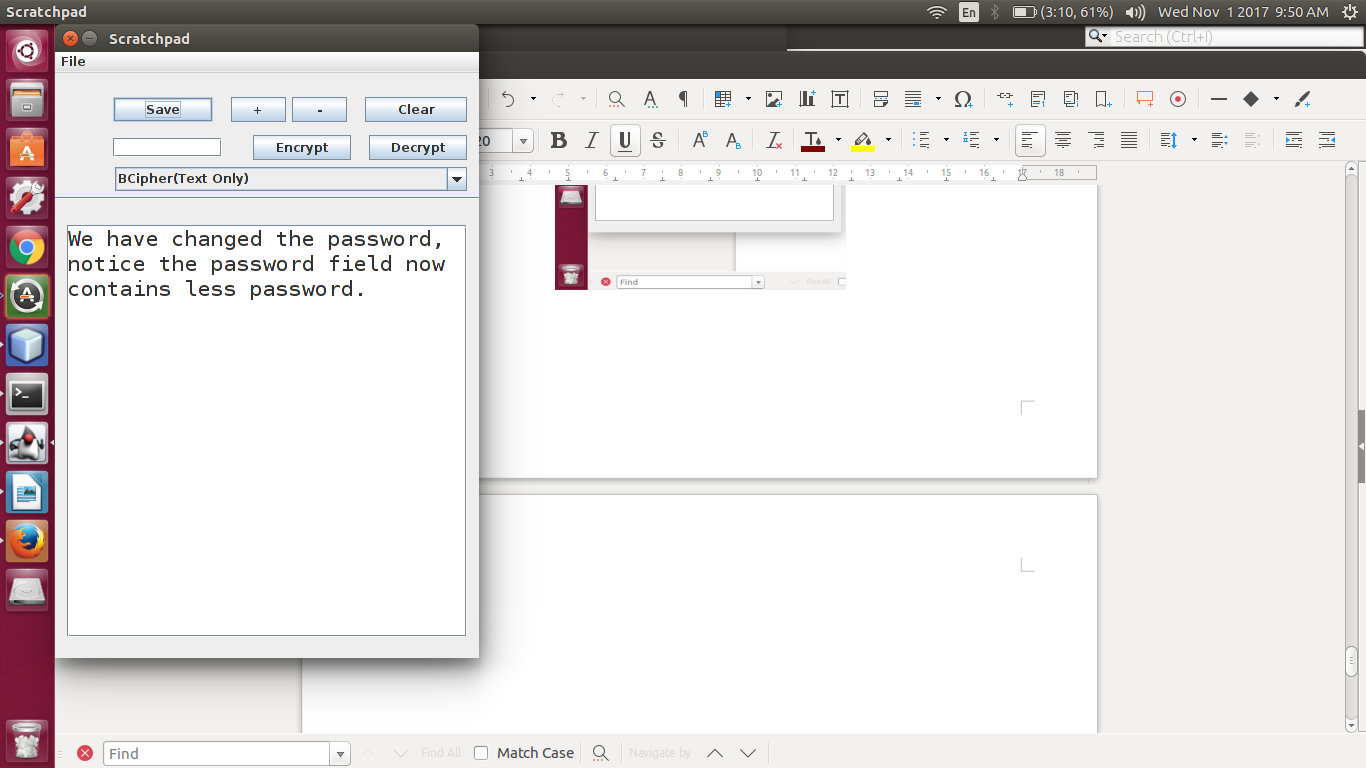












This was the Easy\_Note\_lite – Sketchpad that we have developed. We focused on speed, ease and security. This pad cannot be tampered with wihout the password.