

JAVA Case Study

A Simple Locker Utility that can be used to Passcode Protected Applications

Rollnos:

160114733111

160114733116

Introduction:

In today's security driven world there is no doubt that the privacy of a user is at stake. And thus protecting applications and sensitive data from prying eyes is truly important. The present programme demonstrates how a simple locker application may be created which authenticates user based on their login credentials entered, in this case, a passcode.

Although the application does not provide any advanced encryption formats to store the credentials or something, and they can be viewed by anyone a little familiar with computers, this is an attempt to demonstrate the use of JAVA AWT in handling such utilities together with the core IO functionality of JAVA™. Although any amount of encryption can be decrypted in the world of cyber security, though.

Functionality:

The Starting UI

The application has a starting dialog box that prompts the user to choose they want to do. Either they can enter a lock sequence, unlock an application, request to contact a system administrator, or simply exit.



The user may select any one.

Setting a new Sequence....

The Application provides a Lock with numbers from 0-9, and a submit and a quit button. The **Quit** button quits the application, the **Submit** button saves the credentials to a file named "crypt.dat".



The user is prompted saying that the credentials have been stored.

```

Command Prompt - java Locker
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

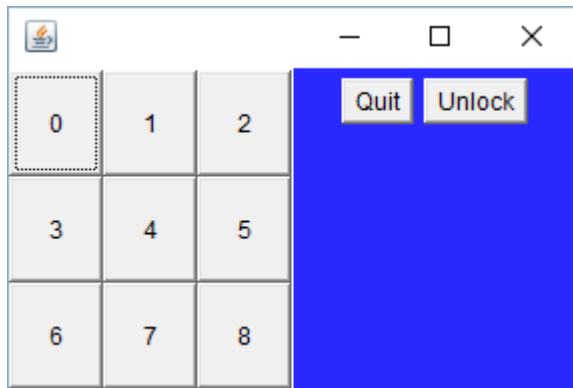
C:\Users\SVL>cd desktop
C:\Users\SVL\Desktop>java Locker
Your pattern is has been saved
  
```

The file “crypt.dat” stores the passcode entered

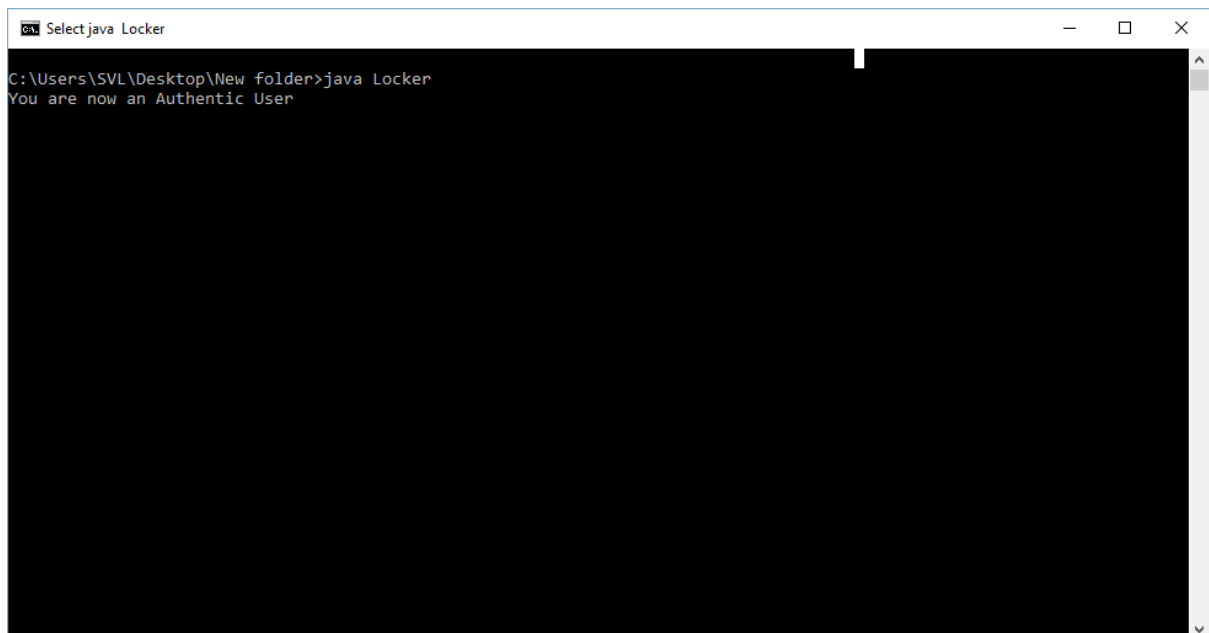
| <input type="checkbox"/> Name | Date modified | Type | Size |
|---|------------------|------------|------|
| Market | 09-05-2016 01:29 | JAVA File | 7 KB |
| <input checked="" type="checkbox"/> Crypt.dat | 09-05-2016 01:37 | DAT File | 1 KB |
| Start.class | 09-05-2016 01:29 | CLASS File | 2 KB |
| Unlock\$1.class | 09-05-2016 01:29 | CLASS File | 1 KB |
| Unlock\$2.class | 09-05-2016 01:29 | CLASS File | 2 KB |
| Locker.class | 09-05-2016 01:29 | CLASS File | 1 KB |
| Start\$1.class | 09-05-2016 01:29 | CLASS File | 1 KB |
| Start\$2.class | 09-05-2016 01:29 | CLASS File | 1 KB |
| Start\$3.class | 09-05-2016 01:29 | CLASS File | 1 KB |
| Start\$4.class | 09-05-2016 01:29 | CLASS File | 2 KB |
| Unlock.class | 09-05-2016 01:29 | CLASS File | 2 KB |
| Lock\$1.class | 09-05-2016 01:29 | CLASS File | 1 KB |
| Lock\$2.class | 09-05-2016 01:29 | CLASS File | 1 KB |
| Lock.class | 09-05-2016 01:29 | CLASS File | 2 KB |

Unlocking with a passcode.....

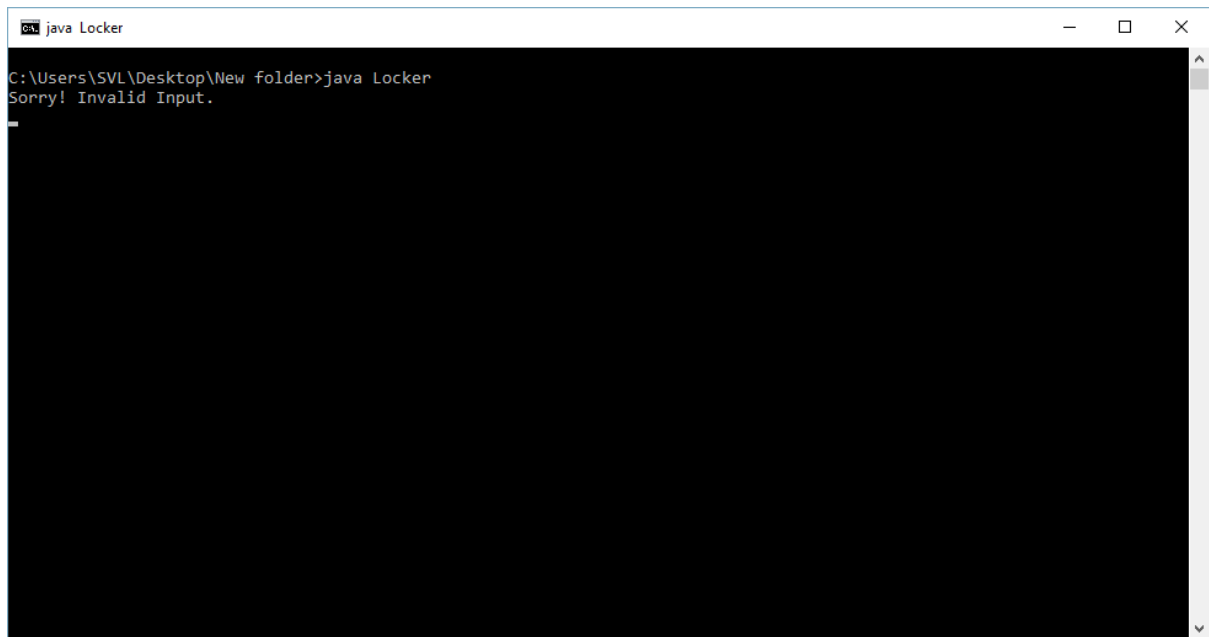
The UI for unlocking is much similar to that with a lock, except that it has an **Unlock** button instead of a **Submit** button.



The user can enter the passcode. If it matches, the user is authenticated successfully



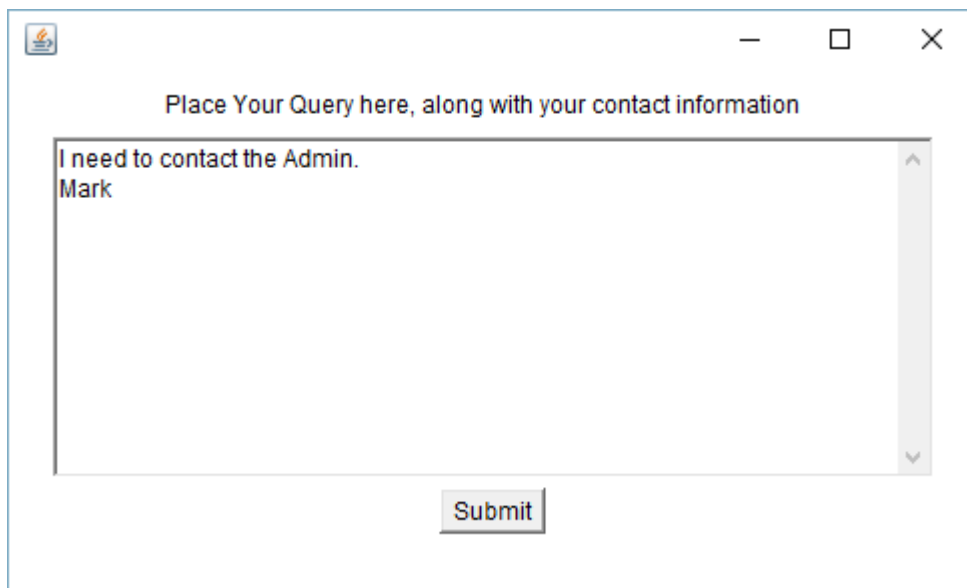
A Failure is reported as well.



```
java Locker
C:\Users\SVL\Desktop\New folder>java Locker
Sorry! Invalid Input.
```

Contacting Admin, just in case.....

The Administrator would be requested to check into an issue. The user would be prompted to enter his credentials and his query, which is stored inside an "ReportLogs.dat" File. The existing queries are not overridden.





















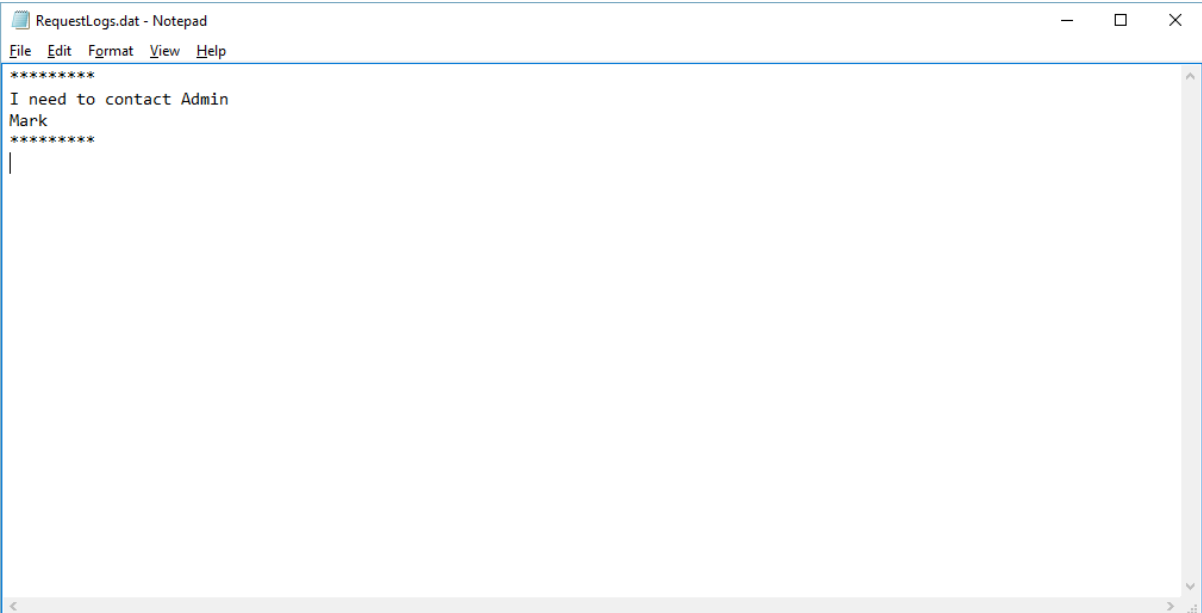
Place Your Query here, along with your contact information

I need to contact the Admin.
Mark

Submit

The user is expected to enter his query and credentials, and then by hitting the **Submit** button, the message gets stored in the file.

| <input type="checkbox"/> Name | Date modified | Type | Size |
|---|------------------|------------|------|
|  Crypt.dat | 09-05-2016 01:37 | DAT File | 1 KB |
|  javac - Shortcut | 09-05-2016 01:29 | Shortcut | 2 KB |
|  Lock\$1.class | 09-05-2016 02:18 | CLASS File | 1 KB |
|  Lock\$2.class | 09-05-2016 02:18 | CLASS File | 1 KB |
|  Lock.class | 09-05-2016 02:18 | CLASS File | 2 KB |
|  Locker.class | 09-05-2016 02:18 | CLASS File | 1 KB |
|  Market | 09-05-2016 02:17 | JAVA File | 7 KB |
|  QueryBox\$1.class | 09-05-2016 02:18 | CLASS File | 2 KB |
|  QueryBox.class | 09-05-2016 02:18 | CLASS File | 1 KB |
| <input checked="" type="checkbox"/>  RequestLogs.dat | 09-05-2016 02:18 | DAT File | 1 KB |
|  Start\$1.class | 09-05-2016 02:18 | CLASS File | 1 KB |
|  Start\$2.class | 09-05-2016 02:18 | CLASS File | 1 KB |
|  Start\$3.class | 09-05-2016 02:18 | CLASS File | 1 KB |
|  Start\$4.class | 09-05-2016 02:18 | CLASS File | 1 KB |
|  Start.class | 09-05-2016 02:18 | CLASS File | 2 KB |
|  Unlock\$1.class | 09-05-2016 02:18 | CLASS File | 1 KB |
|  Unlock\$2.class | 09-05-2016 02:18 | CLASS File | 2 KB |
|  Unlock.class | 09-05-2016 02:18 | CLASS File | 2 KB |



```

RequestLogs.dat - Notepad
File Edit Format View Help
*****
I need to contact Admin
Mark
*****
|

```

Finally, the **Quit** button on the main window simply quits the application.

Thank You.

The Source code is attached here for reference.

/*

Locker Utility

Single Person Usage

```
*/
```

```
import java.io.*;
```

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

```
import java.awt.*;
```

```
import java.awt.event.*;
```

```
import java.awt.color.*;
```

```
/*
```

```
    The Query Box
```

```
*/
```

```
class QueryBox extends Frame
```

```
{
```

```
    Label l1;
```

```
    static TextArea ta;
```

```
    Button b;
```

```
    QueryBox()
```

```
{
```

```
    setSize(500,300);
```

```
    setVisible(true);
```

```
    setLayout(new FlowLayout());
```

```
    b=new Button("Submit");
```

```
    b.addActionListener(new ActionListener(){
```

```
        public void actionPerformed(ActionEvent e){
```

```
            try(FileWriter fw = new FileWriter("RequestLogs.dat", true);
```

```
                BufferedWriter bw = new BufferedWriter(fw);
```

```
                PrintWriter out = new PrintWriter(bw))
```

```
            {
```

```
                out.println("*****");
```

```
                out.println(ta.getText());
```

```
                out.println("*****");
```

```
            } catch (IOException k) {
```

```

        System.out.println(k);
    }
    System.exit(0);    }

});

l1=new Label("Place Your Query here, along with your contact information");
ta=new TextArea();
add(l1);
add(ta);
add(b);
}
}

```

/* The starting UI Class

Provides the options for the user to choose

*/

class Start extends Frame

```

{
    Button b1,b2,b3,b4;
    static String Sequence;
    static boolean Authentic;
    Start()
    {
        Authentic=false;
        b1=new Button("Set new Lock Sequence");
        b1.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                setSequence();
            }
        });
        b1.setBackground(Color.ORANGE);
    }
}

```

```

b2=new Button("Unlock Application");
b2.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        unlockApplication();
    }
});
b2.setBackground(Color.GREEN);
b3=new Button("Quit");
b3.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        System.exit(0);
    }
});
b3.setBackground(Color.red);
b4=new Button("Contact Admin");
b4.setBackground(Color.WHITE);
b4.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent ae){
        QueryBox QB=new QueryBox();
    }
});
add(b1);
add(b2);
add(b4);
add(b3);
setSize(500,100);
setVisible(true);
setLayout(new FlowLayout());
setBackground(new Color(10,10,100));
}
void setSequence(){

```



```

        Lock l=new Lock();
    }
    void unlockApplication(){
        Unlock u=new Unlock();
    }
}

/*
    The interface for locking is provided here.
    Uses TextBased data files to store login credentials.
    Overwrites any existing credentials.
*/
class Lock extends Frame implements ActionListener
{
    Button[] b;
    Button quit,submit;
    String temp;
    Panel p1,p2;
    boolean first_press;
    Lock()
    {
        first_press=true;
        quit=new Button("Quit");
        quit.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                System.exit(0);
            }
        });
        submit=new Button("Submit");
        submit.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){

```

```

        System.out.println("Your pattern is has been saved");
        try {
            File file = new File("Crypt.dat");
            FileWriter fileWriter = new FileWriter(file);
            fileWriter.write(temp);
            fileWriter.flush();
            fileWriter.close();
        }

        catch(IOException f){
            System.out.println(f);
        }
    }
});

p1=new Panel(new GridLayout(3,3,0,0));
add(p1);
p2=new Panel(new FlowLayout());
p2.setBackground(new Color(40,40,255));
add(p2);
p2.add(quit);
p2.add(submit);
b=new Button[9];
for(int i=0;i<b.length;i++)
{
    b[i]=new Button(String.valueOf(i));
    b[i].addActionListener(this);
    p1.add(b[i]);
}

setVisible(true);
setSize(300,200);
setLayout(new GridLayout(1,1,0,0));

```

```

        setBackground(new Color(40,40,255));
    }
    public void actionPerformed(ActionEvent e)
    {
        if(first_press){
            temp=e.getActionCommand();
            first_press=false;
        }
        else{
            temp+=e.getActionCommand();
        }
    }

}

/*
    The UI for unlock is provided here
    Reads credentials from a textBased Data file
*/

class Unlock extends Frame implements ActionListener
{
    Button[] b;
    Button quit,unlock;
    String temp;
    Panel p1,p2;
    boolean first_press;
    Unlock()
    {
        first_press=true;
        quit=new Button("Quit");
    }

```

```

quit.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        System.exit(0);
    }
});
unlock=new Button("Unlock");
unlock.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        try{
            Start.Sequence=new Scanner(new File("crypt.dat")).useDelimiter("\\A").next();
            if(Start.Sequence.equals(temp))
            {
                Start.Authentic=true;
                System.out.println("You are now an Authentic User");
            }
            else
            {
                System.out.println("Sorry! Invalid Input "+Start.Sequence);
            }
        }
        catch(FileNotFoundException FOF){
            System.out.println(FOF);
        }
    }
});
p1=new Panel(new GridLayout(3,3,0,0));
add(p1);
p2=new Panel(new FlowLayout());
p2.setBackground(new Color(40,40,255));
add(p2);
p2.add(quit);

```

```

        p2.add(unlock);

        b=new Button[9];
        for(int i=0;i<b.length;i++)
        {
            b[i]=new Button(String.valueOf(i));
            b[i].addActionListener(this);
            p1.add(b[i]);
        }

        setVisible(true);
        setSize(300,200);
        setLayout(new GridLayout(1,1,0,0));
        setBackground(new Color(40,40,255));
    }

    public void actionPerformed(ActionEvent e)
    {
        if(first_press){
            temp=e.getActionCommand();
            first_press=false;
        }
        else{
            temp+=e.getActionCommand();
        }
    }
}

```

```

class Locker
{
    public static void main(String[] args)
    {
        Start s=new Start();
    }
}

```

}

The Application and its corresponding binaries are available at:

<https://drive.google.com/open?id=0B2nRXxnrGsY0OGZ3Z3BZOGF5c0E>