Marvellous File Packer Unpacker

```java
// MarvellousMain.java

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.BorderLayout;
import java.awt.Dimension;
import javax.swing.JLabel;

class MarvellousLogin extends Template implements ActionListener,  Runnable
{
    JButton SUBMIT;
    JLabel label1,label2,label3,TopLabel;
    final JTextField  text1,text2;
    private static int attemp = 3;

    MarvellousLogin()
    {
        TopLabel = new JLabel();
        TopLabel.setHorizontalAlignment(SwingConstants.CENTER);
        TopLabel.setText("Marvellous Packer Unpacker : Login");
        TopLabel.setForeground(Color.BLUE);

        Dimension topsize = TopLabel.getPreferredSize();
        TopLabel.setBounds(50,40, topsize.width, topsize.height);
        _header.add(TopLabel);

        label1 = new JLabel();
        label1.setText("Username:");
        label1.setForeground(Color.white);

        Dimension size = label1.getPreferredSize();

        label1.setBounds(50,135,size.width,size.height);
        label1.setHorizontalAlignment(SwingConstants.CENTER);

        text1 = new JTextField(15);
        Dimension tsize = text1.getPreferredSize();
        text1.setBounds(200,135, tsize.width, tsize.height);

        text1.setToolTipText("ENTER USERNAME");

        label2 = new JLabel();
        label2.setText("Password:");
        label2.setBounds(50,175, size.width, size.height);
        label2.setForeground(Color.white);
        label2.setHorizontalAlignment(SwingConstants.CENTER);

        text2 = new JPasswordField(15);
        text2.setBounds(200,175, tsize.width, tsize.height);

        text2.setToolTipText("ENTER PASSWORD");

        text2.addFocusListener(new FocusListener()
        {
            public void focusGained(FocusEvent e)
            {

            }
            public void focusLost(FocusEvent e)
            {
                label3.setText("");
            }
        });

        SUBMIT=new JButton("SUBMIT");
        SUBMIT.setHorizontalAlignment(SwingConstants.CENTER);

        Dimension ssize = SUBMIT.getPreferredSize();

        SUBMIT.setBounds(50,220,ssize.width,ssize.height );

        label3 = new JLabel();
        label3.setText("");

        Dimension l3size = label3.getPreferredSize();

        label3.setForeground(Color.RED);
        label3.setBounds(50,250 , l3size.width , l3size.height);

        Thread t = new Thread(this);
        t.start();

        _content.add(label1);
        _content.add(text1);

        _content.add(label2);
        _content.add(text2);

        _content.add(label3);
        _content.add(SUBMIT);

        pack();
        validate();

        ClockHome();
        setVisible(true);
        this.setSize(500,500);
        this.setResizable(false);
        setLocationRelativeTo(null);
        SUBMIT.addActionListener(this);
    }
    public boolean Validate(String Username, String password)
    {
        if((Username.length()<8)||(password.length() < 8))
            return false;
        else
            return true;
    }

    public void actionPerformed(ActionEvent ae)
    {
        String value1=text1.getText();
        String value2=text2.getText();

        if ( ae.getSource() == exit )
        {
            this.setVisible(false);
            System.exit(0);
        }

        if ( ae.getSource() == minimize )
        {
            this.setState(this.ICONIFIED);
        }

        if(ae.getSource()==SUBMIT)
        {
            if(Validate(value1,value2) == false)
            {
                text1.setText("");
                text2.setText("");
                JOptionPane.showMessageDialog(this, "Short username","Marvellous Packer Unpacker", JOptionPane.ERROR_MESSAGE);
            }
            if (value1.equals("MarvellousAdmin") && value2.equals("MarvellousAdmin"))
            {
                NextPage page = new NextPage(value1);
                this.setVisible(false);
                page.pack();
                page.setVisible(true);
                page.setSize(500, 500);
            }
            else
            {
                attemp--;

                 if(attemp == 0)
                 {
                    JOptionPane.showMessageDialog(this, "Number of attempts finished”,”Marvellous Packer Unpacker", JOptionPane.ERROR_MESSAGE);
                       this.dispose();
                    System.exit(0);
                  }

                JOptionPane.showMessageDialog(this, "Incorrect login or password",
                        "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    public void run()
    {
        for(;;)
        {
            if(text2.isFocusOwner())
            {
                if( Toolkit.getDefaultToolkit().getLockingKeyState (KeyEvent.VK_CAPS_LOCK ) )
                {
                    text2.setToolTipText("Warning : CAPS LOCK is on");
                }
                else
                    text2.setToolTipText("");

                if((text2.getText()).length() < 8)
                    label3.setText("Weak Password");
                else
                    label3.setText("");
            }
        }
    }
}

class MarvellousMain
{
    public static void main(String arg[])
    {
        try
        {
            MarvellousLogin frame=new MarvellousLogin();
            frame.setVisible(true);
        }
        catch(Exception e)
        {
            JOptionPane.showMessageDialog(null, e.getMessage());}
        }
}

// Template.java

import javax.swing.*;
import javax.swing.plaf.basic.BasicBorders;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.Serializable;
import java.text.SimpleDateFormat;
import java.util.Date;
```

```java
class ClockLabel extends JLabel implements ActionListener
{
    String type;
    SimpleDateFormat sdf;

    public ClockLabel(String type)
    {
        this.type = type;
        setForeground(Color.green);

        switch (type)
        {
            case "date" : sdf = new SimpleDateFormat("  MMMM dd yyyy");
                setFont(new Font("sans-serif", Font.PLAIN, 12));
                setHorizontalAlignment(SwingConstants.LEFT);
                break;
            case "time" : sdf = new SimpleDateFormat("hh:mm:ss a");
                setFont(new Font("sans-serif", Font.PLAIN, 40));
                setHorizontalAlignment(SwingConstants.CENTER);
                break;
            case "day"  : sdf = new SimpleDateFormat("EEEE  ");
                setFont(new Font("sans-serif", Font.PLAIN, 16));
                setHorizontalAlignment(SwingConstants.RIGHT);
                break;
            default     : sdf = new SimpleDateFormat();
                break;
        }

        Timer t = new Timer(1000, this);
        t.start();
    }

    public void actionPerformed(ActionEvent ae)
    {
        Date d = new Date();
        setText(sdf.format(d));
    }
}

class Template extends JFrame implements Serializable , ActionListener
{
    JPanel _header;
    JPanel _content;
    JPanel _top;

    ClockLabel dayLable;
    ClockLabel timeLable;
    ClockLabel dateLable;

    JButton minimize , exit;

    public Template()
    {
        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        GridBagLayout grid = new GridBagLayout();
        setLayout(grid);

        _top = new JPanel();
        _top.setBackground(Color.LIGHT_GRAY);

        _top.setLayout(null);

        getContentPane().add(_top,new  GridBagConstraints(0,0,1,1,1,5,GridBagConstraints.BASELINE,GridBagConstraints.BOTH,new Insets(0,0,0,0),0,0));

        _header = new JPanel();
        _header.setLayout(null);

        _header.setBackground(Color.white);

        getContentPane().add(_header,new  GridBagConstraints(0,1,1,1,1,20,GridBagConstraints.BASELINE,GridBagConstraints.BOTH,new Insets(0,0,0,0),0,0));

        _content = new JPanel();
        _content.setLayout(null);
        _content.setBackground(new Color(0,50,120));
        JScrollPane jsp = new JScrollPane(_content,JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        jsp.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);

        getContentPane().add(jsp,new GridBagConstraints(0,2,1,1,1,75,GridBagConstraints.BASELINE,GridBagConstraints.BOTH,new Insets(0,0,0,0),0,0));
        setTitle("Marvellous Packer-Unpacker");

        Clock();
        CloseAndMin();
    }

    void CloseAndMin()
    {
        minimize=new JButton("-");
        minimize.setBackground(Color.LIGHT_GRAY);
        minimize.setBounds(MAXIMIZED_HORIZ,0,45,20 );

        exit=new JButton("X");
        exit.setHorizontalAlignment(SwingConstants.CENTER);
        exit.setBackground(Color.LIGHT_GRAY);
        exit.setHorizontalTextPosition(0);
        exit.setBounds(MAXIMIZED_HORIZ+45,0,45,20 );

        _top.add(minimize);
        _top.add(exit);

        exit.addActionListener(this);
        minimize.addActionListener(this);
    }
    public void actionPerformed(ActionEvent ae)
    {
        if ( ae.getSource() == exit )
        {
            this.setVisible(false);
            System.exit(0);
        }

        if ( ae.getSource() == minimize )
        {
            setState(JFrame.ICONIFIED);
        }
    }


    void Clock ()
    {
        dateLable = new ClockLabel("date");
        timeLable = new ClockLabel("time");
        dayLable = new ClockLabel("day");

        dateLable.setForeground (Color.blue);
        timeLable.setForeground (Color.blue);
        dayLable.setForeground (Color.blue);

        dayLable.setFont(new Font("Century",Font.BOLD,15));

        dayLable.setBounds(700,10,200, 100);

        dateLable.setFont(new Font("Century",Font.BOLD,15));

        dateLable.setBounds(800,-40,200, 100);

        timeLable.setFont(new Font("Century",Font.BOLD,15));

        timeLable.setBounds(760,-15,200, 100);

        _header.add(dateLable);
        _header.add(timeLable);
        _header.add(dayLable);
    }

    void ClockHome()
    {
        dateLable = new ClockLabel("date");
        timeLable = new ClockLabel("time");
        dayLable = new ClockLabel("day");

        dateLable.setForeground (Color.blue);
        timeLable.setForeground (Color.blue);
        dayLable.setForeground (Color.blue);
        dayLable.setFont(new Font("Century",Font.BOLD,15));
        dayLable.setBounds(200,20,200, 100);
        dateLable.setFont(new Font("Century",Font.BOLD,15));
        dateLable.setBounds(300,-40,200, 100);

        timeLable.setFont(new Font("Century",Font.BOLD,15));
        timeLable.setBounds(260,-10,200, 100);

        _header.add(dateLable);
        _header.add(timeLable);
        _header.add(dayLable);
    }
}

//NextPage.java

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

class NextPage extends Template implements ActionListener
{
    JLabel label;
    JButton pack , unpack;

    NextPage(String value)
    {
        setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

        label = new JLabel("Welcome: "+value);
        Dimension size = label.getPreferredSize();
        label.setBounds(40,50, size.width + 60, size.height);
        label.setFont(new Font("Century",Font.BOLD,17));
        label.setForeground (Color.blue);

        pack=new JButton("Pack Files");
        Dimension bsize = pack.getPreferredSize();
        pack.setBounds(100,100, bsize.width, bsize.height);
        pack.addActionListener(this);

        unpack=new JButton("Unpack Files");
        Dimension b2size = unpack.getPreferredSize();
        unpack.setBounds(300,100, b2size.width, b2size.height);
        unpack.addActionListener(this);

        _header.add(label);
        _content.add(pack);
        _content.add(unpack);

        ClockHome();
        this.setSize(600,600);
        this.setResizable(false);
        this.setVisible(true);
    }

    public void actionPerformed(ActionEvent ae)
    {
        if ( ae.getSource() == exit )
        {
            this.setVisible(false);
            System.exit(0);
        }
        if ( ae.getSource() == minimize )
        {
            this.setState(this.ICONIFIED);
        }
        if ( ae.getSource() == pack )
        {
            this.setVisible(false);
            try
            {
                MarvellousPackFront obj = new MarvellousPackFront();
            }
            catch(Exception e){}
        }
        if ( ae.getSource() == unpack )
        {
            this.setVisible(false);
            MarvellousUnpackFront obj = new MarvellousUnpackFront();        }
    }
}

//MarvellousPackFront.java

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class MarvellousPackFront extends Template implements ActionListener
{
    JButton SUBMIT ,PREVIOUS;
    JLabel label1,label2, title ;
    final JTextField  text1,text2 ;

   public MarvellousPackFront()
    {
        setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

        title = new JLabel("Marvellous Packing Portal");
        Dimension size = title.getPreferredSize();
        title.setBounds(40,50, size.width + 60, size.height);
        title.setFont(new Font("Century",Font.BOLD,17));
        title.setForeground (Color.blue);

        label1 = new JLabel();
        label1.setText("Directory name");
        label1.setForeground(Color.white);
        label1.setBounds(350,50, size.width, size.height);

        text1 = new JTextField(15);
        Dimension tsize = text1.getPreferredSize();
        text1.setBounds(500,50, tsize.width, tsize.height);
        text1.setToolTipText("Enter name of directory ");

        label2 = new JLabel();
        label2.setText("Destination file name");
        label2.setForeground(Color.white);
        label2.setBounds(350,100, size.width + 60, size.height);

        text2 = new JTextField(15);
        text2.setBounds(500,100, tsize.width, tsize.height);
        text2.setToolTipText("Enter Destination file name");

        SUBMIT=new JButton("SUBMIT");
        Dimension bsize = SUBMIT.getPreferredSize();
        SUBMIT.setBounds(350,200, bsize.width, bsize.height);
        SUBMIT.addActionListener(this);

        PREVIOUS = new JButton("PREVIOUS");
        Dimension b2size = PREVIOUS.getPreferredSize();
        PREVIOUS.setBounds(500, 200, b2size.width, b2size.height);
        PREVIOUS.addActionListener(this);

        _header.add(title);
        _content.add(label1);
        _content.add(label2);
        _content.add(text1);
        _content.add(text2);
        _content.add(SUBMIT);
        _content.add(PREVIOUS);

        this.setSize(1000,400);
        this.setResizable(false);
        this.setVisible(true);
        text1.requestFocusInWindow();
    }

    public void actionPerformed(ActionEvent ae)
    {
        if ( ae.getSource() == exit )
        {
            this.setVisible(false);
            System.exit(0);
        }

        if ( ae.getSource() == minimize )
        {
            this.setState(this.ICONIFIED);
        }
        if ( ae.getSource() == SUBMIT )
        {
            try
            {
                MarvellousPacker obj = new MarvellousPacker(text1.getText(),text2.getText());
                this.dispose();
                NextPage t = new NextPage("MarvellousAdmin");
            }
            catch(Exception e){}
        }
        if ( ae.getSource() == PREVIOUS )
        {
            this.setVisible(false);
            this.dispose();
            NextPage t = new NextPage("MarvellousAdmin");
        }
    }
}

// MarvellousPacker.java

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;
import java.util.stream.Stream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Arrays;

public class MarvellousPacker
{
    FileOutputStream outstream = null;

    String ValidExt[] = {".txt",".c",".java",".cpp"};

    public MarvellousPacker(String src, String Dest) throws Exception
    {
        String Magic = "Marvellous11";
        byte arr[] = Magic.getBytes();
        File outfile =new File(Dest);

        File infile = null;
        outstream = new FileOutputStream(Dest);
        outstream.write(arr, 0, arr.length);

        File folder = new File(src);

        System.setProperty("user.dir",src);

        listAllFiles(src);
    }

    public void listAllFiles(String path)
    {
        try
        (Stream<Path> paths = Files.walk(Paths.get(path)))
        {
            paths.forEach(filePath ->
                       {
                           if (Files.isRegularFile(filePath))
                           {
                               try
                               {
                                   String name = filePath.getFileName().toString();
                                   String ext = name.substring(name.lastIndexOf("."));

                                   List<String> list = Arrays.asList(ValidExt);

                                   if(list.contains(ext))
                                   {
                                       File file=new File(filePath.getFileName().toString());

                                       Pack(file.getAbsolutePath());
                                   }
                               }
                               catch (Exception e)
                               {
                                       System.out.println(e);
                               }
                           }
                       });
        }
        catch (IOException e)
        {
            System.out.println(e);
        }
    }

    public void Pack(String filePath)
    {
        FileInputStream instream = null;

        try
        {
            byte[] buffer = new byte[1024];

            int length;

            byte temp[] = new byte[100];

            File fobj = new File(filePath);

            String Header = filePath+" "+fobj.length();

            for (int i = Header.length(); i < 100; i++)
                Header += " ";

            temp = Header.getBytes();

            instream = new FileInputStream(filePath);

            outstream.write(temp, 0, temp.length);

            while ((length = instream.read(buffer)) > 0)
            {
                outstream.write(buffer, 0, length);
            }

            instream.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}

// MarvellousUnpackFront.java

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

class InvalidFileException extends Exception
{
    public InvalidFileException(String str)
    {
        super(str);
    }
}

public class MarvellousUnpackFront extends Template implements ActionListener
{
    JButton SUBMIT ,PREVIOUS;
    JLabel label1,label2, title ;
    final JTextField  text1 ;

   public MarvellousUnpackFront()
    {

        setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

        title = new JLabel("UnPacking Portal");
        Dimension size = title.getPreferredSize();
        title.setBounds(40,50, size.width + 60, size.height);
        title.setFont(new Font("Century",Font.BOLD,17));
        title.setForeground (Color.blue);

        label1 = new JLabel();
        label1.setText("File Name");
        label1.setForeground(Color.white);
        label1.setBounds(350,50, size.width, size.height);

        text1 = new JTextField(15);
        Dimension tsize = text1.getPreferredSize();
        text1.setBounds(500,50, tsize.width, tsize.height);
        text1.setToolTipText("Enter name of directory ");


        SUBMIT=new JButton("Extract Here");
        Dimension bsize = SUBMIT.getPreferredSize();
        SUBMIT.setBounds(350,200, bsize.width, bsize.height);
        SUBMIT.addActionListener(this);

        PREVIOUS = new JButton("PREVIOUS");
        Dimension b2size = PREVIOUS.getPreferredSize();
        PREVIOUS.setBounds(500, 200, b2size.width, b2size.height);
        PREVIOUS.addActionListener(this);

        _header.add(title);
        _content.add(label1);
        _content.add(text1);
        _content.add(SUBMIT);
        _content.add(PREVIOUS);

        this.setSize(1000,400);
        this.setResizable(false);
        this.setVisible(true);
        text1.requestFocusInWindow();
    }

    public void actionPerformed(ActionEvent ae)
    {
        if ( ae.getSource() == exit )
        {
            this.setVisible(false);
            System.exit(0);
        }
        if ( ae.getSource() == minimize )
        {
            this.setState(this.ICONIFIED);
        }
        if ( ae.getSource() == SUBMIT )
        {
            try{
            MarvellousUnpack obj = new MarvellousUnpack(text1.getText());
                this.dispose();
                NextPage t = new NextPage("admin");
            }
            catch(InvalidFileException obj)
            {
                this.setVisible(false);
                this.dispose();

                JOptionPane.showMessageDialog(this, "Invalid Packed File",
                                "Error", JOptionPane.ERROR_MESSAGE);

                NextPage t = new NextPage("MarvellousAdmin");
            }
            catch(Exception e)
            {}
        }
        if ( ae.getSource() == PREVIOUS )
        {
            this.setVisible(false);
            this.dispose();
            NextPage t = new NextPage("admin");
        }
    }
}

//MarvellousUnpack.java

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.FileInputStream;
import java.io.FileOutputStream;

public class MarvellousUnpack
{
    FileOutputStream outstream = null;

    public MarvellousUnpack(String src) throws Exception
    {
        unpack(src);
    }

    public void unpack(String filePath) throws Exception
    {
        try
        {
            FileInputStream instream = new FileInputStream(filePath);

            byte header[]= new byte[100];
            int length = 0;

            byte Magic[] = new byte[12];
            instream.read(Magic,0,Magic.length);

            String Magicstr = new String(Magic);

            if(!Magicstr.equals("Marvellous11"))
            {
                throw new InvalidFileException("Invalid packed file format");
            }

            while((length = instream.read(header,0,100)) > 0)
            {
                String str = new String(header);

                String ext = str.substring(str.lastIndexOf("/"));
                ext = ext.substring(1);

                String[] words=ext.split("\\s");

                String filename = words[0];

                int size = Integer.parseInt(words[1]);

                byte arr[] = new byte[size];

                instream.read(arr,0,size);

                FileOutputStream fout=new FileOutputStream(filename);
                fout.write(arr,0,size);
            }
        }
        catch(InvalidFileException obj)
        {
            throw new InvalidFileException("Invalid packed file format");
        }
        catch(Exception e)
        {}
    }
```