**CS 410 Software Engineering - Spring 2018**

**Assignment 3**

**Due Date: Tuesday, June 11, 2019**

**Refactoring**

You are given the following Java implementation of a simple notepad GUI application. The implementation for the following menu items is already given: **new file, save file, print file, copy and paste. You are asked to do the followings:**

1. Identify at least 4 “code smells” you found in the source code by indicating the line number/ code block and the type of code smell. Remember to start with the worst smell.
2. Action performed method needs to be decoupled.
3. Main method needs to be moved to a separate class.
4. Variables need to be more specific and refactored.
5. Code needs to be commented.
6. Refactor the code. Make changes to the code to be more readable, maintainable and scalable enough to add new operations to the application. Please provide a log for your updates with your reasoning. You don’t need to write every detail, but make sure you provide a list of refactoring practices you applied.

Ex:

Line #: changed the variable from \_\_\_\_\_\_ to \_\_\_\_, because \_\_\_\_\_\_\_\_\_\_\_.

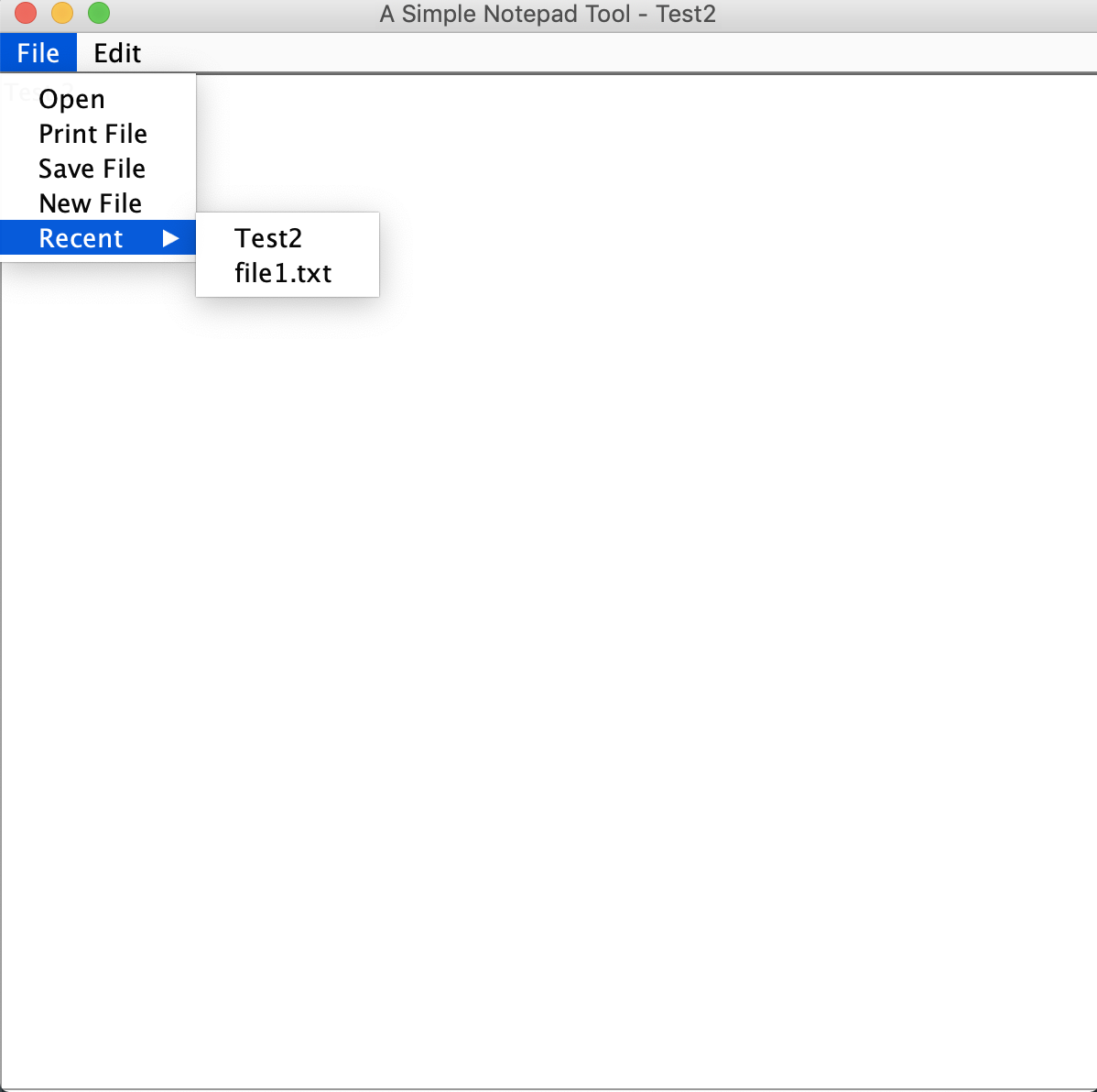
I created a new class \_\_\_\_\_\_\_, because \_\_\_\_\_\_\_\_\_\_\_.

* Line 70 actionPerformed() has been decupled into separate methods new, save, print, copy, paste and undo for more readability.
* Line 71 moved logic to method newFile() and call newFile().
* Line 74 moved logic to method saveFile() and call saveFile.
* Line 87 moved logic to method printFile() and call printFile().
* Line 110 moved logic to method copy() and call copy().
* Line 112 moved logic to method paste() and call paste().
* Line 117 moved logic to method undo() and call undo().
* Line 66 moved main method to class Main.java and call NotePad class.
* Line 24 – 33 refactored short abbreviations to full button names.
* Line 25 – 60 updated variable names that were changed in lines 24-33.
* Updated variables in all logic methods that were decupled from actionPerformed().
* Updated variables in object creation.

1. Line 24, mb to menuBar to be more descriptive.
2. Line 25, fm to fileMenu to be more descriptive.
3. Line 26, em to editMenu to be more descriptive.
4. Line 27, d to textPane to be more descriptive.
5. Line 28, nf to newFileButton to be more descriptive.
6. Line 29, sf to saveFileButton to be more descriptive.
7. Line 30, pf to printFileButton to be more descriptive.
8. Line 31, u to undoButton to be more descriptive.
9. Line 32, c to copyButton to be more descriptive.
10. Line 33, p to pasteButton to be more descriptive.
11. Add these new features:

* Add **open file** feature to the **file menu** to be able to open an existing file.
* Add **open recent** feature to the **file menu.** Open recent can show a list for the last 5 files opened in the right order, most recent in the top of the list.
* Add **simple-replace** to the **edit menu**. This will replace a highlighted text with the clipboard content/a user input.

Here are sample runs for the notepad after adding the new features



Open and Open Recent

|  |  |
| --- | --- |
|  |  |
| Replace Functionality | |

**SimpleNotePad.java**

import java.awt.Dimension;

import java.awt.Graphics;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.print.PageFormat;

import java.awt.print.Printable;

import java.awt.print.PrinterException;

import java.awt.print.PrinterJob;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.JScrollPane;

import javax.swing.JTextPane;

import javax.swing.text.Position;

import javax.swing.text.StyledDocument;

public class SimpleNotePad extends JFrame implements ActionListener{

JMenuBar mb = new JMenuBar();

JMenu fm = new JMenu("File");

JMenu em = new JMenu("Edit");

JTextPane d = new JTextPane();

JMenuItem nf = new JMenuItem("New File");

JMenuItem sf = new JMenuItem("Save File");

JMenuItem pf = new JMenuItem("Print File");

JMenuItem u = new JMenuItem("Undo");

JMenuItem c = new JMenuItem("Copy");

JMenuItem p = new JMenuItem("Paste");

public SimpleNotePad() {

setTitle("A Simple Notepad Tool");

fm.add(nf);

fm.addSeparator();

fm.add(sf);

fm.addSeparator();

fm.add(pf);

em.add(u);

em.add(c);

em.add(p);

nf.addActionListener(this);

nf.setActionCommand("new");

sf.addActionListener(this);

sf.setActionCommand("save");

pf.addActionListener(this);

pf.setActionCommand("print");

c.addActionListener(this);

c.setActionCommand("copy");

p.addActionListener(this);

p.setActionCommand("paste");

u.addActionListener(this);

u.setActionCommand("undo");

mb.add(fm);

mb.add(em);

setJMenuBar(mb);

add(new JScrollPane(d));

setPreferredSize(new Dimension(600,600));

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

pack();

}

public static void main(String[] args) {

SimpleNotePad app = new SimpleNotePad();

}

@Override

public void actionPerformed(ActionEvent e) {

if(e.getActionCommand().equals("new")) {

d.setText("");

}else if(e.getActionCommand().equals("save")) {

File fileToWrite = null;

JFileChooser fc = new JFileChooser();

int returnVal = fc.showSaveDialog(null);

        if (returnVal == JFileChooser.APPROVE\_OPTION)

        fileToWrite = fc.getSelectedFile();

        try {

PrintWriter out = new PrintWriter(new FileWriter(fileToWrite));

out.println(d.getText());

JOptionPane.showMessageDialog(null, "File is saved successfully...");

out.close();

        } catch (IOException ex) {

}

}else if(e.getActionCommand().equals("print")) {

try{

PrinterJob pjob = PrinterJob.getPrinterJob();

pjob.setJobName("Sample Command Pattern");

pjob.setCopies(1);

pjob.setPrintable(new Printable() {

public int print(Graphics pg, PageFormat pf, int pageNum) {

if (pageNum>0)

return Printable.NO\_SUCH\_PAGE;

pg.drawString(d.getText(), 500, 500);

paint(pg);

return Printable.PAGE\_EXISTS;

}

});

if (pjob.printDialog() == false)

return;

pjob.print();

} catch (PrinterException pe) {

JOptionPane.showMessageDialog(null,

"Printer error" + pe, "Printing error",

JOptionPane.ERROR\_MESSAGE);

}

}else if(e.getActionCommand().equals("copy")) {

d.copy();

}else if(e.getActionCommand().equals("paste")) {

StyledDocument doc = d.getStyledDocument();

        Position position = doc.getEndPosition();

        System.out.println("offset"+position.getOffset());

d.paste();

}else if(e.getActionCommand().equals("undo")) {

//TODO: implement undo operation

}

}

}

Here are a couple of resources you can use to understand and implement a Java GUI application:

* <http://www.java2s.com/Tutorials/Java/Java_Swing/0400__Java_Swing_JFrame.htm>
* <https://docs.oracle.com/javase/tutorial/uiswing/components/menu.html>