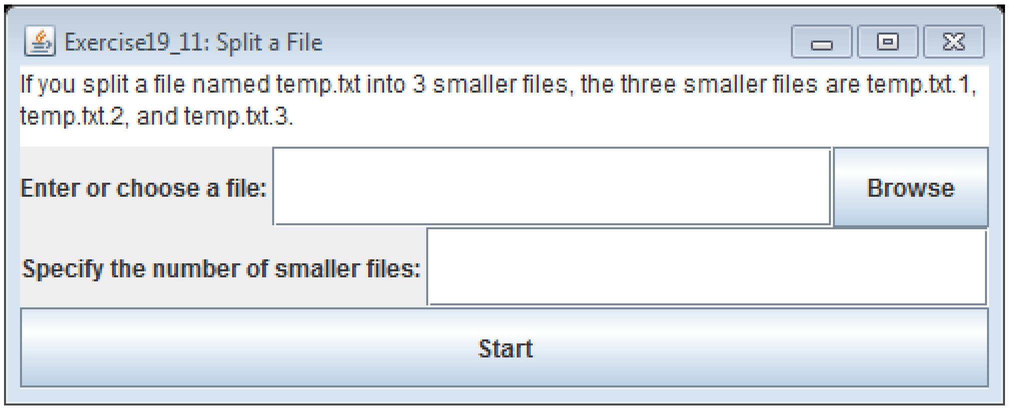
CIT 249: Java II
Chapter 19 Lab 3

In this lab we will complete # 11 on page 734. This program will take a file of your choosing and split it into multiple files. The number of files it splits into is based on the information you supply. When the file first runs it will display as:

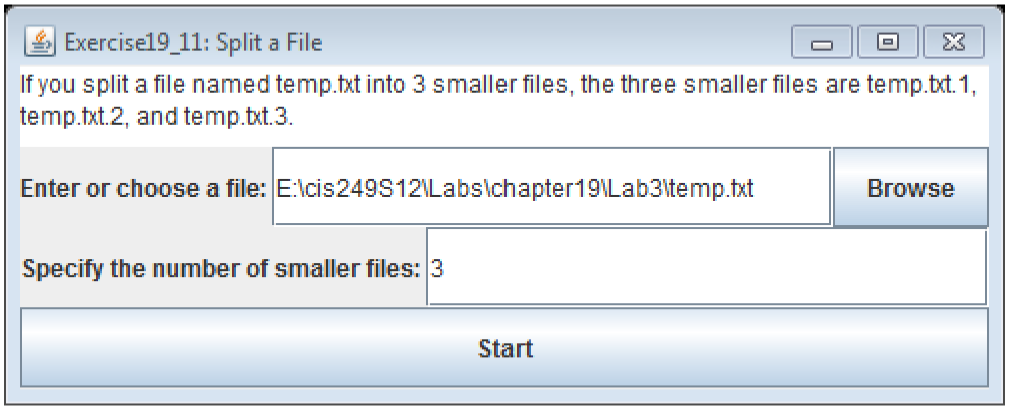


If you split a file named temp.txt into 3 smaller files, the three smaller files are temp.txt.1, temp.txt.2, and temp.txt.3.

Enter or choose a file:

Specify the number of smaller files:

You will browse to the specified file and enter the number of smaller files:

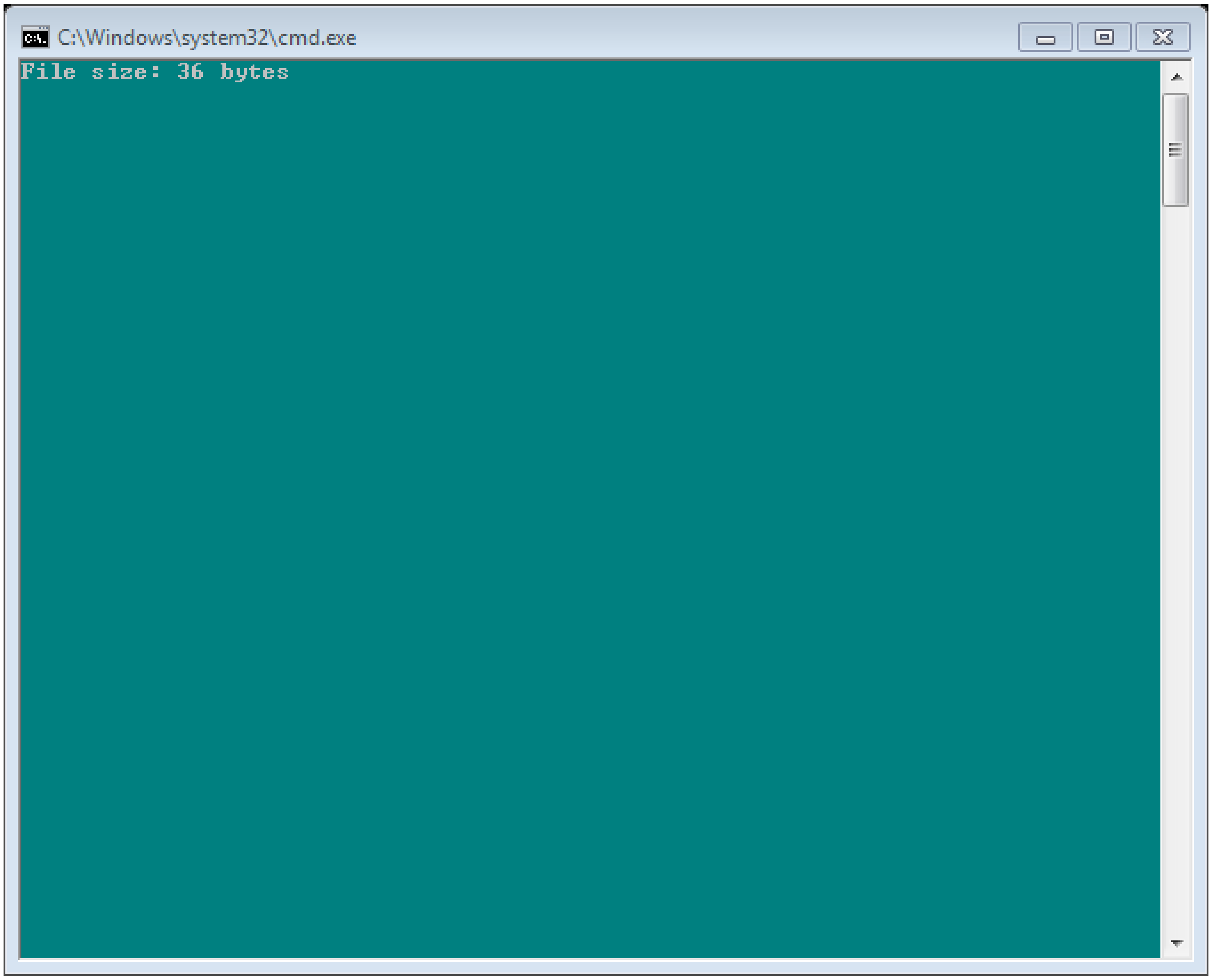


If you split a file named temp.txt into 3 smaller files, the three smaller files are temp.txt.1, temp.txt.2, and temp.txt.3.

Enter or choose a file: E:\cis249S12\Labs\chapter19\Lab3\temp.txt

Specify the number of smaller files: 3

When you click the "Start" button you will receive a message concerning the file size at the command prompt and the files will be split into "3" files, in this case, that has the same name as the base file but will contain a period and number at the end of the file name. The contents of the base file will be split among the split files.


C:\Windows\system32\cmd.exe

```
File size: 36 bytes
```

In my base file of temp.txt I had the words:
“Hello out there. Hope you are fine.”

My three files contain:
Hello out th
ere. Hope y
ou are fine.

So let’s get started!

1. First create a file in the same folder where you will save your program. A simple text file will be fine. Type some words in the file. It doesn’t matter what. Save the file.
2. Open a new document window and save the program as Ch19Lab3.java. Remember you may use either TextPad or another IDE of your choosing such as Eclipse.
3. We need to add some documentation as to the purpose of the program. Type:

   ```
   /* Purpose:  Split files into multiple files
   */
   ```

4. First we have to import several classes to use. Type:

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
```

- Since we are inputting and exporting information we need the classes in the java.io package.
- This is a GUI application so we need classes in both the java.awt and javax.swing packages.
- Since we will need event handlers we need classes in the java.awt.event package. Remember that Java imports classes but not an entire folder. This is why a separate import statement is required even though the event package is located in the java.awt package.

5. Type your class header and opening brace having the class extend the JFrame class.
6. We will need a couple of text fields and buttons in our program so we must create them. Type:

```
//Create components including text fields, and buttons
private JTextField jtfInputFile = new JTextField(20);
private JTextField jtfNumberOfFiles = new JTextField(2);
private JButton jbtBrowse = new JButton("Browse");
private JButton jbtStart = new JButton("Start");
```

7. In GUI applications the constructing is done within the constructor method. This method has the same name as the class. Type:

```
public Ch19Lab3()
{
```

- Whatever you do, do not include the keyword void in this method header or your frame will not display.

8. We will create a new panel and add a text field and button to the panel. Type:

```
//Construct panels and add components to the panels
JPanel panel1 = new JPanel(new BorderLayout());
panel1.add(new JLabel("Enter or choose a file: "), BorderLayout.WEST);
panel1.add(jtfInputFile, BorderLayout.CENTER);
panel1.add(jbtBrowse, BorderLayout.EAST);
```

- Here we set the layout manager for the panel to BorderLayout which is a coordinate system with locations of NORTH, SOUTH, EAST, WEST, CENTER
- A new label is added to the WEST
- A text field is added to the CENTER
- A button is added to the EAST

9. We create another panel and add a new label and the second text field to it. Type:

```
JPanel panel2 = new JPanel(new BorderLayout());
panel2.add(new JLabel("Specify the number of smaller files: "),BorderLayout.WEST);
panel2.add(jtfNumberOfFiles, BorderLayout.CENTER);
```

- The new panel has also been given a layout of BorderLayout
- The new label has been added to the WEST
- The second text field has been added to the CENTER

10. A third panel has been created, as well as a TextArea object. TextArea objects are like text fields except they have multiple rows. Type:

```
//construct main panel, a text area and set text area's wrapping
JPanel panel = new JPanel(new GridLayout(4, 1));
JTextArea jta = new JTextArea("If you split a file named temp.txt into 3 smaller files, the three
smaller files are temp.txt.1, temp.txt.2, and temp.txt.3.");
jta.setWrapStyleWord(true);
jta.setLineWrap(true);
```

- The panel has been given a layout manager of GridLayout with 4 rows and 1 column specified.
- A JTextArea is set to wrap the words typed in if they fill up a row, and to allow each line to wrap.

11. We add several components to our panel and add the panel to the actual frame. Type:

```
//add textarea, panels, and jbtStart button to the last panel
panel.add(jta);
panel.add(panel1);
panel.add(panel2);
panel.add(jbtStart);

//add panel to the frame
add(panel);
```

- First we add the JTextArea to the panel
- Then we add the two previous panels to this last panel.
- We also add the last button to the panel.
- We finally add this last panel to the frame.

12. We have to designate some events to be handled. First for the jbtStart button. Type:

```
//add listeners to the two buttons and invoke the appropriate methods for the action
jbtStart.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        splitFile(jtfInputFile.getText(), Integer.parseInt(jtfNumberOfFiles.getText()));
    }
});
```

- Here we add a new listener to the jbtStart button. If this button is clicked the splitFile() method is invoked and passed the values of the text that has been typed in the jtfInputFile field, and also the value typed in the jtfNumberOfFiles field after converting that to an integer.

13. We also have to add a listener to the other button. Type:

```
jbtBrowse.addActionListener(new ActionListener()
{
   public void actionPerformed(ActionEvent e)
   {
     //Create a JFileChooser in order to allow browsing and selecting of a file
     JFileChooser fileChooser = new JFileChooser();
     if (fileChooser.showOpenDialog(null) == JFileChooser.APPROVE_OPTION)
         {
       // Get the selected file
       // Get the selected file
       java.io.File file = fileChooser.getSelectedFile();
       jtfInputFile.setText(file.toString());
     }
   }
 });
```

- The JFileChooser class allows you to browse to a file and choose it.
- The APPROVE_OPTION returns true if you Open and the file was found.
- The File object received the value of the file that was selected.
- The text field received the value of the file as a String.

14. Close the constructor method.
15. The splitFile() method does the actual splitting. Type:

```
public void splitFile(String filename, int numberOfPieces)
{
```

16. Since a possible error might occur our code is included with a try/catch statement. First the try statement is created by typing:

```
try
  {
     //construct a BufferedInputStream to check for file to be split
    BufferedInputStream input = new BufferedInputStream(new FileInputStream(new
File(filename)));

    System.out.println("File size: " + input.available() + " bytes");
    long fileSize = input.available();
    int splitFileSize = (int) Math.ceil(1.0 * fileSize / numberOfPieces);

    for (int i = 1; i <= numberOfPieces; i++)
```

```
      {
        //construct new BufferedOutputStream which is used to split the file
        BufferedOutputStream output = new BufferedOutputStream(new FileOutputStream(new
File(filename + "." + i)));
        int value;
        int count = 0;
        // What is wrong if these two conditions are placed in a different
        // order?
        while (count++ < splitFileSize && (value = input.read()) != -1)
        {
          output.write(value);
        }
        output.close();
      }

      input.close();
    }
```

- First we create a BufferedInputStream which has an argument of a new FileInputStream which supplied the file name, which is entered by the user.
- The available() method of the BufferedInputStream supplies the size of the file and displays it.
- An integer named splitFileSize receives its value from the file size divided by the number of files you entered to split into.
- A while loop runs as long as the value of the variable *count* is less than the value of *splitFileSize* AND the result of the variable *value* does not equal -1 after it receives the value from the BufferedInputStream's read() method.
    - If both conditions are true then the stream writes the value.
- The stream is closed. You must always close your stream once you are finished with it.

16. If you have a try statement it must be followed by a catch statement. Type:

```
catch (IOException ex)
{
    ex.printStackTrace();
}
```

- The printStackTrace() throws the standard message for Input/Output errors.

17. Close the method.
18. Our main method is pretty basic. Type:

```
public static void main(String[] args)
{
    Ch19Lab3 frame = new Ch19Lab3();
    frame.setSize(500,200);
    frame.setTitle("Exercise19_11: Split a File");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setLocationRelativeTo(null); // Center the frame
```

```
        frame.setVisible(true);
    }
```

- We create an instance of our class, which is the frame.
- Set the size of the frame.
- Set the title that appears on the title bar of the frame.
- Sets it so that if the closing button in the upper right-hand corner is pressed the program ends. If this is not included it appears that the program has ended but only the frame closes and the program is still running.
- We center the frame on the screen.
- We set its visibility. You must set the visibility of the frame to true or it will not display. Why would you have to do this? You might have multiple frames.

19. Close the class.
20. Compile the program and compress all files into a single zip or rar file and submit it.