Practice Project #2 Report
For Y2K a Text Editor

CSC 4320/6320 - Operating Systems
Spring 2016

Name: Yu Lin
Email: ylin21@student.gsu.edu

Introduction and Overview of App:

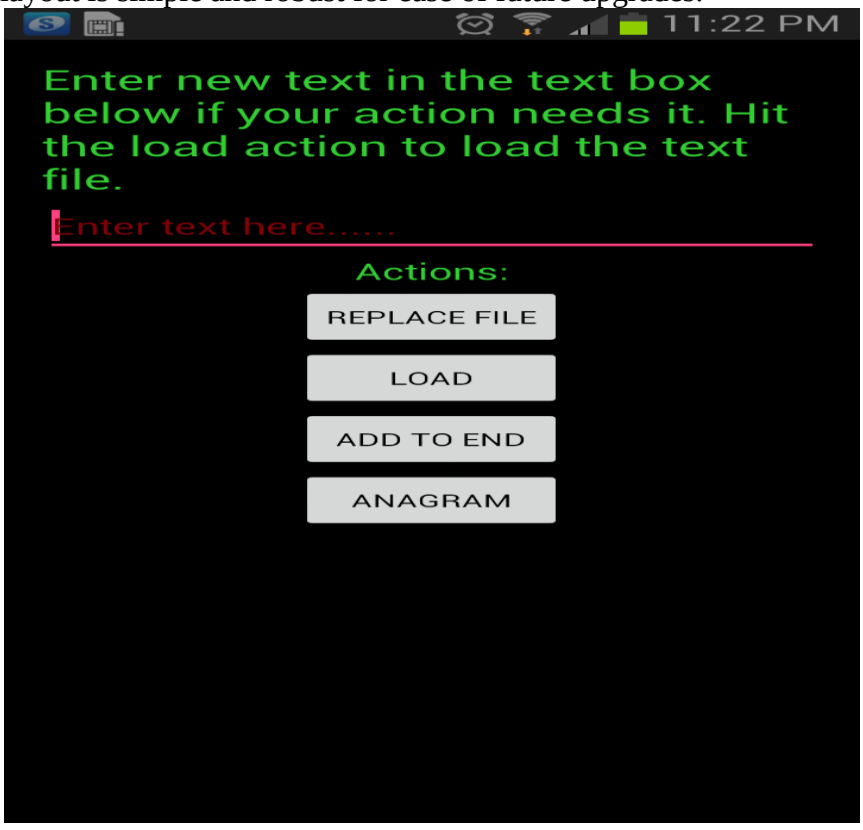
Y2K a Text Editor app takes a .txt file called t1.txt that is hardcoded into the system. A public File variable is declared in MyActivity.java via

```
public File file = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS) +
"/Samples/t1.txt");
```

where

```
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS) +
"/Samples/t1.txt"
```

is the absolute path to the t1.txt file and Samples directory (in DIRECTORY_DOWNLOADS) that is already created for the ease of this project. App is intended to be user friendly by having a clear outline supported by a visible color scheme through styles.xml. However, users have to use common sense to determine when to enter text into the text box whenever the user desired action requires it. At this moment, the app includes 4 different actions or buttons that users can choose from. The 1st button is "REPLACE FILE" which takes the text entered by the user (id: editText) and completely replaces t1.txt with the new text. The 2nd button is "LOAD" which prints the contents of t1.txt beneath the last button, "Anagram," from left to right. The 3rd button is "ADD TO END" which takes the text entered by the user (id: editText) and appends it to the end of t1.txt after skipping a line. The last and 4th button is "ANAGRAM" which takes words that are separated by the " " space character and lines, and then removes all non-letter characters from those lines. Then it replaces the old file with words that are anagrams written (.write) on the same line in alphabetical order and ignore words that are longer than 12 characters. To summarize, the 1st and 3rd buttons require user input. The only button that prints is the 2nd, and only the 4th button requires no input and prints no text. Lastly, app is designed in such a way for future developing to be easily done; MyActiviy.java has documented methods, and the entire app layout is simple and robust for ease of future upgrades.

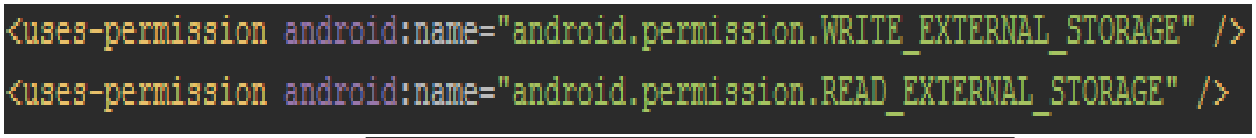


Screenshot1:
A screenshot of the Y2K a Text Editor on launch.

OS-Related Low-Level Functionalities:

As of Android API level 4 (API 4), the "WRITE_EXTERNAL_STORAGE" permission was implemented to restrict access to the shared storage space i.e. all AndroidManifest.xml files after API 4 needs to have permissions declared. Even though the use of "WRITE_EXTERNAL_STORAGE" permission implicitly gives any app the permission "READ_EXTERNAL_STORAGE," the **Y2K a Text Editor** declares both of the permissions in the manifest to ensure reliability as seen in Screenshot2. Since this app have (or potential) features in file management/manipulation/modification that create, modify, and print to/from files, they are considered to be low-level functions by definition as being part of the programs the Android Linux Kernel handles. Therefore, permissions can also be viewed as a low-level function.

At a later time, the app can be upgraded to run on API 23 (as of April, 2016 the newest API version is 23 for devices like Google Nexus 6P) where the app will then request the user for run time permissions where the app will then have to request user permission specifically to read and write to external storage files. To help fix this problem, the app checks if the API level is less than or equal to 22 as seen in Screenshot3.



Screenshot2:
"WRITE_EXTERNAL_STORAGE" AND "READ_EXTERNAL STORAGE" permissions located in AndroidManifest.xml.

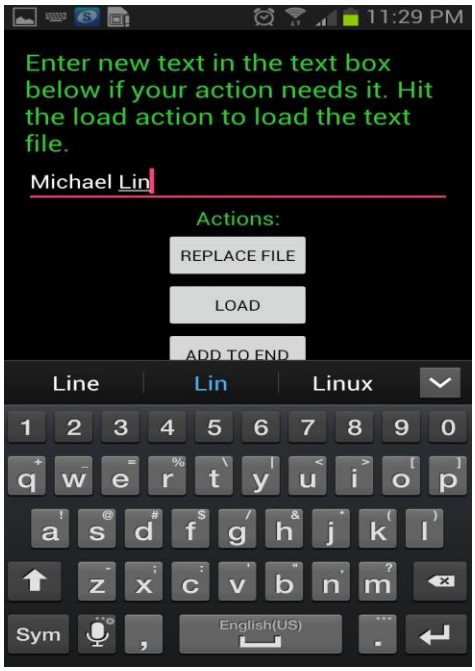
Screenshot3:
The max SDKVersion or API is set to 22. This line is located in AndroidManifest.xml.

Design of the App:

Y2K a Text Editor is designed to be user-friendly and easily upgradable later on for developers. As seen in Screenshot1, the color scheme for the only activity, MyActivity.java, is clearly visible and readable by the user. AppTheme style is based on textColor = lime green or #32cd32 in hex code and the background is black or #000000 as seen in Screenshot4. The user input text box has a hint text that says "Enter text here……" in red that disappears when the user clicks on the text box, at which time a keyboard appears as seen in Screenshot5 and Screenshot6. All user entered text will appear as white in (id = editText). And all buttons are of black textColor and white background. textView (id: textView) is designed to be located at the bottom of the last button, in this case the "ANAGRAM" button and appear as lime green as per app theme. In the event that the textView (id: textView) or buttons is too large for the user to view in one vertical screen, a vertical scroll view is implemented in content_my.xml. Furthermore by using the clear(View view) method and the line result.setText(""); it cleans up any and all unwanted texts from user UI when any actions are used. Furthermore, clear(View view) eliminates the user keyboard when it is not needed by the user.

```
<style name="AppTheme" parent="Base.Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
    <item name="android:background">#000000</item>
    <item name="android:textColor">#32cd32</item>
</style>
```

Screenshot4:
AppTheme used by **Y2K a Text Editor** located in String.xml.



Screenshot5:
Screenshot after user has put input text in text box (id: editText). Notice the absence of the hint text in Screenshot1.

```
android:hint="Enter text here....."
android:textColorHint="#800007"
```

Screenshot6:
Hint text and text color code located in content_my.xml under id: editText.

Sample Documentation/Comments of Relevant Methods in MyActivity.java:

All necessary variables(identified by id in the Manifest) requiring different actions upon them are initialized by the onCreate method when the app is ran. Method clears editText(ed) when called. Also, keyboard is hidden upon call.

When user hits the "REPLACE FILE" button, the buttonReplace method is called at which time the user input at editText(ed) is put into a String array called addText. The addToText method is then called with a single parameter addText. Then clear(view) and result is both cleared of any text.

Method addToText takes one String[] and overwrites the global variable File file by replacing it with the contents stored in the given array. By using FileWriter writer = new FileWriter(file, false), anything writen(.write) to file will now overwrite the old file.

When user hits the "Load" button, the buttonLoad method is called at which time the contentsof File file global variable will be placed in a String called "input" Lastly, clear(view) is calledand user input is printed into printView by using result.setText(input).

When user hits the "ADD TO END" button, the buttonAddToEnd method is called at which time the whole user input in editText(ed) is appended to the end of File file. This is possible by FileWriter writer = new FileWriter(file, true);where putting boolean true in the second parameter makes it possible to .write by appending to file. In this method, line is skipped before anything is appended. Lastly, clear(view) and textView(result) is called, where result is cleared of any text viewable to user.

When user hits the "ANAGRAM" button, the buttonAnagram method is called. Purpose of anagram is to take words from an input file(in this case, File file a global variable), and compares it to each other to see if they are anagrams of each other. And anagrams are permutations of the letters in that word. Also, anagram disregards any punctuation and/or upper cases in the words; any punctuations in this case, will be eliminated when stringSort is cast, which eliminates all the punctuations and converts all the words into Strings containing of alphabetical letters. When the program is comparing words, it will use testAnagram algorithm to see if the words are anagrams of each other or not. In the method, any output will be written temporarily in a temporary file File temp(global variable), where the original words, with their punctuations, will be printed on the same lines; however, all the lines will be in alphabetical order determined by the first letter of the first word in each line.

Furthermore, buttonAnagram will implement 2 try-catch statements to find errors where the input file is empty or does not exist. Also, it can also find the errors where the temp file cannot be written. Under all these errors, the program will call printStackTrace(). If an word is 12 chars or more, then the app will ignore it. The first try-catch runs through the input file to find if the words exceed 12 chars or more, and if the input file exists; if any of these is true, the first try-catch will call printStackTrace(), and at the end of the first try-catch fileInput will be reset for the next try-catch. The 2nd try-catch is for output, where it will print anagrams on the same line, or words by themselves. Temp and input is there to compare the words in the loop to

see if they are anagrams. The catch phrases at the end will attempt to find where output.txt cannot be created, and again where input file does not exist.

In the 2nd try-catch statement, the method uses a nested for loop where the first for loop runs from index 0 and 2nd for loop runs from the first loops index + 1.And testAnagram is called in the 2nd loop to find anagrams for the word in the first index. Upon finding an anagram, method will .write anagrams on the same line with a space between every word. Lastly, clear(view) is called and File file is replaced with the contents in temp.

testAnagram method takes two inputs as Strings x and y. First, it sets x and y to lower case and sorts it according to the stringSort method created. Second, the method will check if the lengths of the 2 Strings are not equivalent; if they are, then the method returns false. If the lengths are equivalent, then the method tests if x=y, if they are equal, then the method returns false. If other conditions appear, this method will return false. Since this method is boolean, it will always have one output that is either true or false.

StringSort method takes an input of String z to lower the cases of all the chars in String, and uses trim() to eliminate spaces in z. Next, each of the chars in z is put in the char array characters and the array is sorted using Arrays.sort into alphabetical order. The for loop then checks for any individual char that doesn't equal lower case alphabets from a-z and replaces them with a empty space. Lastly, characters is converted back into a String which is returned(output).

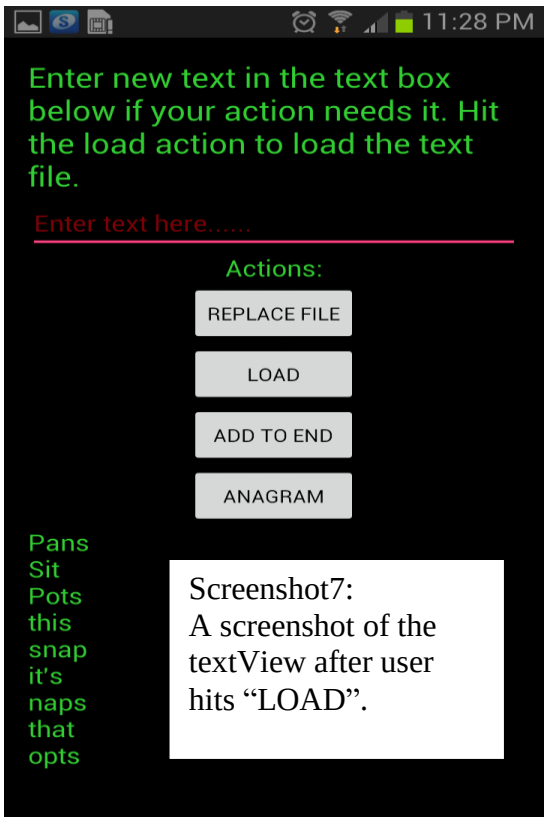
removeChar method takes one parameter of type String and converts it to lower case and removes non letter characters in the string. The method then returns another String after trim() and replaceAll is called from the java String library. replaceAll(" ", "") replaces any space characters found with "".

alphabeticalOrder method takes a parameter of ArrayList<String> and returns nothing. Then the method sorts the ArrayList<String> in alphabetical order regardless of capitalization and characters that aren't letters. Method removes non letter characters by calling the removeChar(String) method.

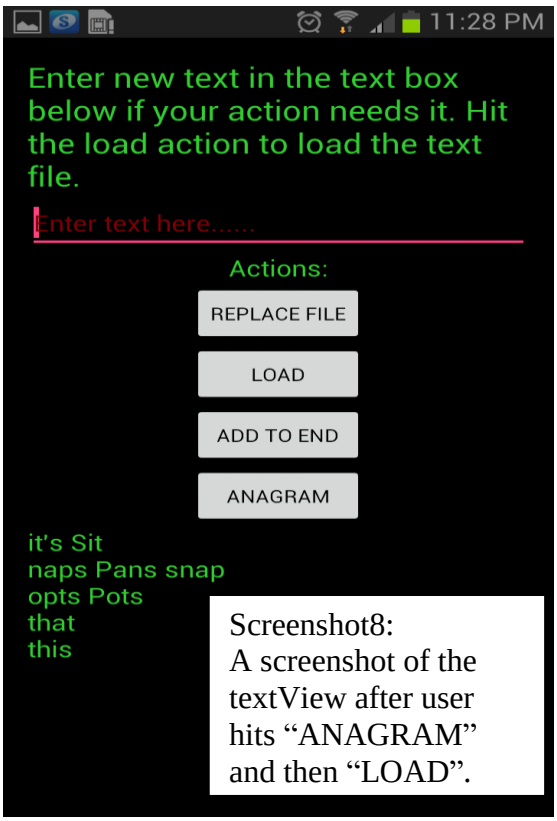
Final Notes:

Users do not need to create t1.txt or temp.txt beforehand. However, user need to create a directory called Samples in “DIRECTORY_DOWNLOADS” in the SD_CARD before using the app.

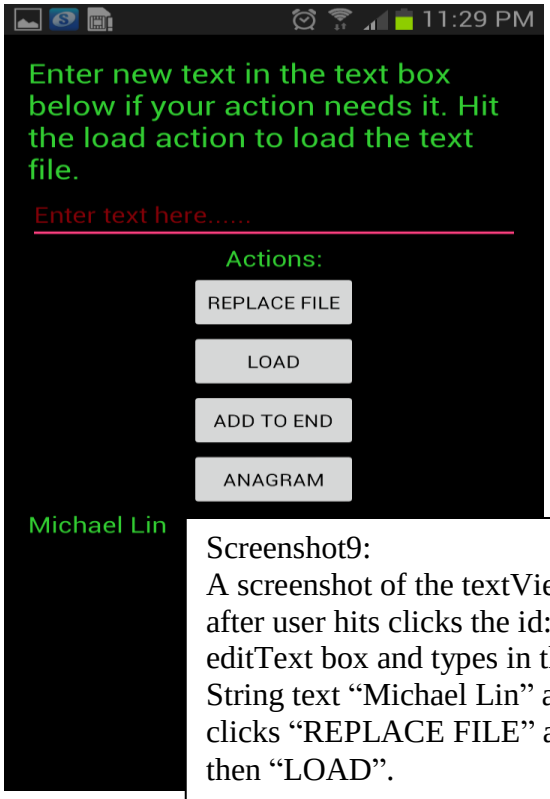
More Screenshots of the UI:



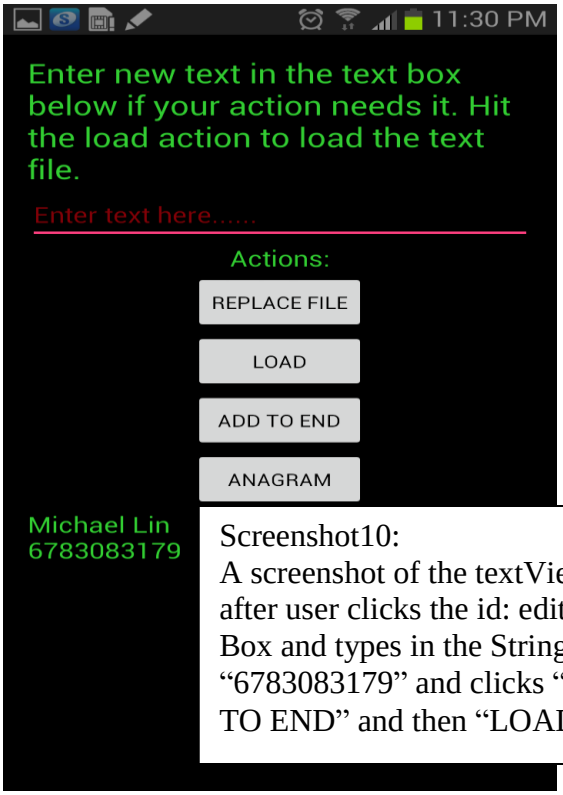
Screenshot7:
A screenshot of the textView after user hits "LOAD".



Screenshot8:
A screenshot of the textView after user hits "ANAGRAM" and then "LOAD".



Screenshot9:
A screenshot of the textView after user hits clicks the id: editText box and types in the String text "Michael Lin" and clicks "REPLACE FILE" and then "LOAD".



Screenshot10:
A screenshot of the textView after user clicks the id: editText Box and types in the String text "6783083179" and clicks "ADD TO END" and then "LOAD".

Relevant Source Code:

content_my.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.c1tech.textfilemanager.MyActivity"
    tools:showIn="@layout/activity_my"
    android:weightSum="1">

    <TextView
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Enter new text in the text box below if your action needs it.
Hit the load action to load the text file. "
        android:id="@+id/irrelevant1" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Actions:"
        android:id="@+id/irrelevant2"
        android:layout_below="@+id/editText"
        android:layout_centerHorizontal="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#000000"
        android:text="Load"
        android:id="@+id/load"
        android:onClick="buttonLoad"
        android:layout_below="@+id/replace"
        android:layout_alignLeft="@+id/replace"
        android:layout_alignStart="@+id/replace"
        android:layout_alignRight="@+id/addToEnd"
        android:layout_alignEnd="@+id/addToEnd" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#000000"
        android:text="Replace File"
        android:id="@+id/replace"
        android:onClick="buttonReplace"
        android:layout_below="@+id/irrelevant2"
        android:layout_centerHorizontal="true" />
```

```
    <EditText
        android:textColor="#FFFFFF"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/editText"
        android:layout_below="@+id/irrelevant1"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:hint="Enter text here......"
        android:textColorHint="#800007"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#000000"
        android:text="Add to End"
        android:id="@+id/addToEnd"
        android:layout_below="@+id/load"
        android:layout_alignRight="@+id/replace"
        android:layout_alignEnd="@+id/replace"
        android:onClick="buttonAddToEnd"
        android:layout_alignLeft="@+id/replace"
        android:layout_alignStart="@+id/replace" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#000000"
        android:text="Anagram"
        android:id="@+id/anagram"
        android:layout_below="@+id/addToEnd"
        android:layout_alignRight="@+id/addToEnd"
        android:layout_alignEnd="@+id/addToEnd"
        android:onClick="buttonAnagram"
        android:layout_alignLeft="@+id/addToEnd"
        android:layout_alignStart="@+id/addToEnd" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:id="@+id/textView"
        android:layout_below="@+id/anagram"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignRight="@+id/editText"
        android:layout_alignEnd="@+id/editText" />

    </RelativeLayout>
</ScrollView>
```

MyActivity.java

```
package com.c1tech.textfilemanager;

import android.content.Context;
import android.os.Bundle;
import android.os.Environment;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.*;

public class MyActivity extends AppCompatActivity
{
    public EditText ed;
    public TextView result;
    public Button save, load;
    public File file = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS) +
"/Samples/t1.txt");
    public File temp = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS) +
"/Samples/tempFile.txt");

    /*All necessary variables(identified by id in the Manifest) requiring different
actions upon them are initialized
    * by the onCreate method when the app is ran. */
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        ed = (EditText) findViewById(R.id.editText);
        result = (TextView) findViewById(R.id.textView);
        save = (Button) findViewById(R.id.replace);
        load = (Button) findViewById(R.id.load);
    }

    /*Method clears editText(ed) when called. Also, keyboard is hidden upon call.*/
    public void clear(View view)
    {
        ed.setText("");
        InputMethodManager input = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
        input.hideSoftInputFromWindow(view.getApplicationWindowToken(), 0);
    }

    /*When user hits the "REPLACE FILE" button, the buttonReplace method is called at
which time the user
    * input at editText(ed) is put into a String array called addText. The addToText
method is then called
```

```
    * with a single parameter addText. Then clear(view) and result is both cleared of
any text.*/
    public void buttonReplace (View view){
        String [] addText =
String.valueOf(ed.getText()).split(System.getProperty("line.separator"));
        addToText(addText);
        clear(view);
        result.setText("");
    }

    /*Method takes one String[] and overwrites the global variable File file by
replacing
    * it with the contents stored in the given array. By using FileWriter writer =
    * new FileWriter(file, false), anything writen(.write) to file will now overwrite
the
    * old file.*/
    public void addToText(String[] text)
    {
        FileWriter writer;

        try
        {
            writer = new FileWriter(file, false);
            for (int i = 0; i < text.length; i++)
            {
                writer.write(text[i].toString());

                if (i < text.length - 1)
                {
                    writer.write("\n");
                }
            }
            writer.flush();
            writer.close();
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }

    /*When user hits the "Load" button, the buttonLoad method is called at which time
the contents
    of File file global variable will be placed in a String called "input" Lastly,
clear(view) is called
    and user input is printed into printView by using result.setText(input).*/
    public void buttonLoad(View view)
    {
        Scanner fileinput = null;
        File inFile = file;
        String input = "";

        try
        {
            fileinput = new Scanner(inFile);
            while(fileinput.hasNext())
            {
                input += fileinput.nextLine() + "\n";
            }
        }
        catch(FileNotFoundException e)
        {
            e.printStackTrace();
```

```
        }
        finally
        {
            fileinput.close();
        }

        clear(view);
        result.setText(input);
    }

    /*When user hits the "ADD TO END" button, the buttonAddToEnd method is called at
which time the whole user
    * input in editText(ed) is appended to the end of File file. This is possible by
FileWriter writer = new FileWriter(file, true);
    * where putting boolean true in the second parameter makes it possible to .write
by appending to file. In this method, line is
    * skipped before anything is appended. Lastly, clear(view) and textView(result) is
called, where result is cleared of any
    * text viewable to user.*/
    public void buttonAddToEnd (View view) throws IOException
    {
        FileWriter writer = new FileWriter(file, true);
        writer.write("\n" + ed.getText().toString());
        writer.flush();
        writer.close();
        clear(view);
        result.setText("");
    }

    /* When user hits the "ANAGRAM" button, the buttonAnagram method is called.
     * Purpose of anagram is to take words from an input file(in this case, File file
a global variable), and compares
     * it to each other to see if they are anagrams of each other. And anagrams are
     * permutations of the letters in that word. Also, anagram disregards any
     * punctuation and/or upper cases in the words; any punctuations in this case,
     * will be eliminated when stringSort is cast, which eliminates all the
     * punctuations and converts all the words into Strings containing of
     * alphabetical letters. When the program is comparing words, it will use
     * testAnagram algorithm to see if the words are anagrams of each other
     * or not. In the method, any output will be written temporarily in a temporary
file File temp(global variable),
     * where the original words, with their punctuations, will be printed on the
     * same lines;however, all the lines will be in alphabetical order determined
     * by the first letter of the first word in each line.
     *
     * Furthermore, buttonAnagram will implement 2 try-catch statements to find errors
     * where the input file is empty or does not exist. Also, it can also find
     * the errors where the temp file cannot be written. Under all these errors,the
     * program will call printStackTrace(). If an word is 12 chars or more, then the
program will
     * ignore it. The first try-catch runs through the input file to find if the words
exceed 12 chars or more, and if the
     * input file exists; if any of these is true, the first try-catch will call
printStackTrace(),
     * and at the end of the first try-catch fileInput will be reseted for the
     * next try-catch. The 2nd try-catch is for output, where it will print anagrams
on the
     * same line, or words by themselves. Temp and input is there to compare the words
in the
     * loop to see if they are anagrams. The catch phrases at the end will attempt to
find
     * where output.txt cannot be created, and again where input file does not exist.
     *
```

```
     * In the 2nd try-catch statement, the method uses a nested for loop where the
first
     * for loop runs from index 0 and 2nd for loop runs from the first loops index +
1.
     * And testAnagram is called in the 2nd loop to find anagrams for the word in the
first
     * index. Upon finding an anagram, method will .write anagrams on the same line
     * with a space between every word.
     *
     * Lastly, clear(view) is called and File file is replaced with the contents in
     * temp.*/
    public void buttonAnagram(View view) throws IOException {
        String input;
        Scanner fileInput = null;
        File inFile = file;
        ArrayList<String> list = new ArrayList<>();
        result.setText("");

        try
        {
            fileInput = new Scanner(inFile);

            while (fileInput.hasNext())
            {
                input = fileInput.next();

                if (input.length() > 12)
                    continue;

                list.add(input);
            }

            alphabeticalOrder(list);
            FileWriter writer = new FileWriter(temp);

            try
            {
                for (int i = 0; i < list.size(); i++)
                {
                    writer.write(list.get(i) + " ");
                    for (int j = i + 1; j < list.size(); j++)
                    {
                        if (testAnagram(list.get(i), list.get(j)))
                        {
                            writer.write(list.get(j) + " ");
                            list.remove(j);
                        }
                    }
                    writer.write("\n");
                }
            }
            catch (IOException e)
            {
                e.printStackTrace();
            }
            finally
            {
                writer.flush();
                writer.close();
            }
        }
        catch (FileNotFoundException e)
        {
```

```
            e.printStackTrace();
        } finally
        {
            fileInput.close();
        }

        clear(view);
        file = temp;
    }

    /*testAnagram method takes two inputs as Strings x and y. First, it sets
     * x and y to lower case and sorts it according to the stringSort method
     * created. Second, the method will check if the lengths of the 2 Strings
     * are not equivalent; if they are, then the method returns false. If the
     * lengths are equivalent, then the method tests if x=y, if they are equal,
     * then the method returns false. If other conditions appear, this method
     * will return false. Since this method is boolean, it will always have one
     * output that is either true or false.*/
    public static boolean testAnagram (String x, String y)
    {
        x = x.toLowerCase();
        x = stringSort(x);
        y = y.toLowerCase();
        y = stringSort(y);

        if (x.length() != y.length())
            return false;
        else if (x.equals(y))
            return true;
        else
            return false;
    }

    /* StringSort method takes an input of String z to lower the cases of all the
     * chars in String, and uses trim() to eliminate spaces in z. Next,
     * each of the chars in z is put in the char array characters and the array
     * is sorted using Arrays.sort into alphabetical order. The for loop then
     * checks for any individual char that doesn't equal lower case alphabets
     * from a-z and replaces them with a empty space. Lastly, characters is converted
     * back into a String which is returned(output).*/
    public static String stringSort (String z)
    {
        z = z.toLowerCase();
        z = z.trim();

        char[] characters = z.toCharArray();

        Arrays.sort(characters);
        for ( int i = 0; i < characters.length; i++)
        {
            if (characters[i] <= 96 || characters[i] >= 123)
                characters[i] = ' ';
        }

        return String.valueOf(characters).trim();
    }

    /*removeChar method takes one parameter of type String and
    converts it to lower case and removes non letter characters in the string.
    The method then returns another String after trim() and replaceAll
    is called from the java String library. replaceAll(" ", "") replaces any space
    characters found with "".*/
    public static String removeChar (String s)
```

```
    {
        s = s.toLowerCase();
        char[] characters = s.toCharArray();

        for ( int i = 0; i < characters.length; i++)
        {
            if (characters[i] <= 96 || characters[i] >= 123)
                characters[i] = ' ';
        }

        return String.valueOf(characters).trim().replaceAll(" " , "");
    }

    /*alphabeticalOrder method takes a parameter of ArrayList<String> and returns
nothing.
    Then the method sorts the ArrayList<String> in alphabetical order regardless of
    capitalization and characters that aren't letters. Method removes non letter
characters
    by calling the removeChar(String) method.*/
    public static void alphabeticalOrder (ArrayList<String> list)
    {
        int minIndex;
        String x, y;
        for (int i = 0; i < list.size(); i++)
        {
            minIndex = i;
            x = list.get(i);
            y = x;

            for (int j = i + 1; j < list.size(); j++)
            {
                if
(removeChar(list.get(minIndex)).compareToIgnoreCase(removeChar(list.get(j))) > 0)
                {
                    x = list.get(j);
                    minIndex = j;
                }
            }
            list.add(i, x);
            list.remove(i + 1);
            list.add(minIndex, y);
            list.remove(minIndex + 1);
        }
        list.trimToSize();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_my, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
```

```
        }

        return super.onOptionsItemSelected(item);
    }
}
```

styles.xml

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Base.Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
        <item name="android:background">#000000</item>
        <item name="android:textColor">#32cd32</item>
    </style>

    <style name="AppTheme.NoActionBar">
        <item name="windowActionBar">false</item>
        <item name="windowNoTitle">true</item>
    </style>

    <style name = "whiteText">
        <item name="android:textColor">#FFFFFF</item>
    </style>

    <style name="AppTheme.AppBarOverlay"
parent="ThemeOverlay.AppCompat.Dark.ActionBar" />

    <style name="AppTheme.PopupOverlay" parent="ThemeOverlay.AppCompat.Light" />

</resources>
```