Name: Choong Hui Min **Tutorial Group ID**: W14	

Code

```
package textbuddy;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;



/**
 * Assumptions: commands are correctly entered.
 * @author huimin
 *
 */
public class TextBuddy {

	private static ArrayList<String> textFile = new ArrayList<String>();
	private static String userCommand;
	private static String restOfText;
	private static String fileName;
	private static String toDisplay;
	private static Scanner sc = new Scanner(System.in);

	private static final String INVALID_COMMAND_MESSAGE = "Invalid command";
	private static final String WELCOME_MESSAGE = "Welcome to TextBuddy. %1$s is ready for use";
	private static final String FILE_ADDED_MESSAGE = "added to %1$s: \"%2$s\"";
	private static final String FILE_CLEARED_MESSAGE = "all content deleted from %1$s";
	private static final String FILE_EMPTY_DISPLAY_MESSAGE = "%1$s is empty";

	public static void main(String[] arg) {
		String userInput;
		fileName = arg[0];
		checkFileValid();
		welcomeMsg();
		while (true) {
			System.out.print("command: ");
			userInput = sc.nextLine();
			splitText(userInput);
			executeCommand();
			System.out.println(toDisplay);
			saveFile();
		}
	}

	public static void checkFileValid() {
		if (fileName.isEmpty()) {
			toDisplay = String.format(INVALID_COMMAND_MESSAGE, fileName);
			System.out.println(toDisplay);
		}
	}
```

```
public static void welcomeMsg() {
        toDisplay = String.format(WELCOME_MESSAGE, fileName);
        System.out.println(toDisplay);
}

/**
 * This operation splits the string into the user's command and the rest of the text.
 * @param userInput
 */
public static void splitText(String userInput) {
        String[] textArr = userInput.split(" ", 2);
        userCommand = textArr[0];
        if (textArr.length > 1 ) {
                restOfText = textArr[1];
        }
}

/**
 * This operation determines which of the supported command types the user wants to perform
 */
public static void executeCommand() {
        switch (userCommand) {
                case "add" :
                        add();
                        break;

                case "delete" :
                        delete();
                        break;

                case "clear" :
                        clear();
                        break;

                case "display" :
                        display();
                        break;

                case "exit" :
                        sc.close();
                        System.exit(0);
                        break;

                default :
                        //throw an error if the command is not recognized
                        throw new Error("Unrecognized command type");
        }
}

public static void display() {
        if (!textFile.isEmpty()) {
                int size = 1;
                for (int i = 0; i < textFile.size()-1; i++) {
                        System.out.println(size +". " + textFile.get(i));
                        size++;
                }
                // prints the last line in textFile as toDisplay will be printed last.
                toDisplay = size + ". " + textFile.get(textFile.size()-1);
        } else {
                toDisplay = String.format(FILE_EMPTY_DISPLAY_MESSAGE, fileName);
        }
}

public static void clear() {
        textFile.clear();
```

```
            toDisplay = String.format(FILE_CLEARED_MESSAGE, fileName);
        }

        public static void delete() {
            int indexToDelete = Integer.parseInt(restOfText.substring(0));

            if ((indexToDelete > textFile.size()) || (indexToDelete < 1)) {
                toDisplay = INVALID_COMMAND_MESSAGE;
            } else {
                String deletedLine = textFile.get(indexToDelete-1);
                textFile.remove(indexToDelete-1);
                toDisplay = "deleted from " + fileName + " \"" + deletedLine + "\"";
            }
        }

        public static void add() {
            textFile.add(restOfText);
            toDisplay = String.format(FILE_ADDED_MESSAGE, fileName, restOfText);
        }

        /**
         * This operation saves the file. If file already exists, it will be overwritten.
         */
        public static void saveFile() {
            try {
                BufferedWriter fileWrite = new BufferedWriter(new FileWriter(fileName));
                Iterator<String> textFileItr = textFile.iterator();
                while (textFileItr.hasNext()) {
                    fileWrite.write(textFileItr.next().toString());
                    fileWrite.newLine();
                }
                fileWrite.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }


}
```

TestInput.txt

```
add little brown fox
display
add jumped over the moon
display
delete 2
display
clear
display
add this line should be line one
exit
```

ExpectedOutput.txt

```
Welcome to TextBuddy. textbuddy.txt is ready for use
command: added to textbuddy.txt: "little brown fox"
command: 1. little brown fox
command: added to textbuddy.txt: "jumped over the moon"
command: 1. little brown fox
2. jumped over the moon
command: deleted from textbuddy.txt "jumped over the moon"
command: 1. little brown fox
command: all content deleted from textbuddy.txt
command: textbuddy.txt is empty
command: added to textbuddy.txt: "this line should be line one "
command:
```