CS 152 Computer Programming Fundamentals
Project 1: ASCII Art

Brooke Chenoweth

Spring 2024

1 ASCII Art

ASCII art is a graphic design technique that consists of pictures pieced together from the 95 printable (from a total of 128) characters defined by the ASCII Standard from 1963 and ASCII compliant character sets. ASCII art can be created with any text editor. Most examples of ASCII art require a fixed-width font (non-proportional fonts, as on a traditional typewriter) such as Courier for presentation.

Examples:

```
    _    _                .---.        .-----------
   (o)--(o)              /     \  __  /    ------
  /.______.\            / /     \(  )/    -----
  \________/           //////   ’ \/ ‘   ---               /\_/\
 ./        \.         //// / // :    : ---            ____/ o o \
( .        , )       // /   /  /‘    ’--            /~____  =ø= /
 \ \_\\//_/ /       //          //..\\             (______)__m_m)
  ~~  ~~  ~~       ============UU====UU====
                               ’//||\\‘
                                 ,,‘‘
```

2 Problem Specification

1. Using the IntelliJ IDE, create a Java project that contains one class: `AsciiArt.java`[1]

2. Your `AsciiArt.java` must use Java's `System.out.println()` function to display to the Java console an ASCII art image that fits within 60x24 columns and rows. It is ok to be smaller than that size. Your image must be a stylized version of your

[1]If you accidentally used a wrong name such as "Asciiart" or "AsciArt" you can rename the class with the option under the "Refactor" menu. The IDE will both rename the class and update the java file name for you. Ask your section leader for help if you have trouble.

name (or a nickname, if you prefer). Each "letter" must be at least 3x3 characters: `printf("Brooke\n")` would NOT get credit as an ASCII art image of my name.[2]

3. Your Java program must have the following format

```
/**
 * Brief description of the assignment here.
 * @author YourFirstName YourLastName
 */
public class AsciiArt {

    /**
     * Prints my name in ASCII Art to the console.
     * @param args Command-line arguments are ignored.
     */
    public static void main(String[] args) {
        // Your code goes here.
        // Use as many lines as you need.
        // No line may be more than 80 characters
        //     (length includes the leading spaces)
    }
}
```

Note: In Java, comments that begin with `/**` can be compiled using Javadoc, a documentation generator. The comment format used by Javadoc is the de facto industry standard for documenting Java classes. The `@author` and `@param` annotations[3] are Javadoc tags understood by the Javadoc generator. In this lab, you do not need to generate web pages with Javadoc, but you do need to follow the Javadoc format shown above.

3 Escaping Special Characters

The backslash character has a special meaning in Java strings: it is called an "escape character" and is used to modify the next character in order to generate special characters.

For example, `'\n'` is the newline character. `System.out.println("one fish\ntwo fish")` will display:

```
one fish
two fish
```

A backslash can be used to include a quotation mark in a string.
For example, `System.out.println("She said \"hello\"")` will display:

```
She said "hello"
```

[2]There are websites and tools out there that will generate ASCII art from text. (Try searching for "text to ASCII art") Feel free to make use of one of these if you aren't feeling particularly artistic. You will still need to write the Java code to print it.

[3]as well as several others, including `@version` and `@return`

If you want to actually print a backslash, you need to use a double backslash. `System.out.println("\\oo/")` will display:

```
\oo/
```

4 Turning in your assignment

Submit your **AsciiArt.java** file into the Project 1 assignment in Canvas. Do not attach `.class` files or any other files.

5 Grading Rubric (total of 20 points)

1 point : Submitted one file in Canvas with file name `AsciiArt.java`.

2 points : The source code contains neither errors nor warnings when viewed in IntelliJ.

5 points : The code adheres to the specified format. This includes:

1. The author's name commented in the format shown.
2. A description of the main method in the format shown.
3. All code in each block is indented exactly four spaces per block structure level. *Spaces* must be used for indenting, not *tabs*.[4]
4. No non-space characters may occur on a line after an open curly bracket ({).
5. No characters except spaces are on the same line as any closing curly brackets (}).
6. No line contains more than 80 characters.[5]

6 points : When compiled and run, the program outputs a beautifully stylized, and readable, ASCII art version of your name in the Java console. (I'm not actually judging your artistic ability here.)

3 points : Each character of your name is composed of at least 9 characters (3x3).

3 points : The full name is no larger than 60x24 characters.[6]

```
 ____                  _
|  _ \                | |
| |_) |_ __ ___   ___ | | _____
|  _ <| '__/ _ \ / _ \| |/ / _ \
| |_) | | | (_) | (_) |   <  __/
|____/|_|  \___/ \___/|_|\_\___|
```

[4]IntelliJ will automatically insert spaces for tabs and indent to the correct level, so you don't have to sit there counting how many times you pressed the space bar.

[5]You can configure IntelliJ to draw a line at 80 characters so you know when your lines are getting too long.

[6]If you have a really long name, you can use a nickname or initials to be shorter.