

Programming Assignment 3

Due: 11:59PM, Saturday, January 23

Overview

This assignment should get you familiar with

- If/else statements (part 1)
- Loops (part 2)

Setup

This assignment requires two starter files. Set up your directory via:

```
$ mkdir HW3
$ cd HW3
$ cp /home/linux/ieng6/cs11wb/public/HW3/* .
```

Part 1: The Guessing Game (15 pts)

For this part you will make a program that can play a simple number guessing game. You will think of a number and the computer will try to guess it, with help from you.

It should start by asking for a range in which to find your number. Then, when it starts guessing, the only hints you are allowed to give are “l” (too low), “h” (too high), and “c” for a correct guess. It will repeatedly guess numbers until it finds the correct one. You should make your program guess your number in as few steps as possible.

Some details:

- The range is inclusive, so if you say the range is “1200 to 1400” it could guess the min and max values 1200 and 1400.
- It should work with negative numbers
- If it is obvious what the answer is, it should print it and exit
- All of your code should go inside “GuessingGame.java” (it’s in the setup folder)

Here are some examples:

```
Your number is in this range...(enter 2 numbers)
1200 1400
Type 'h' for too high, 'l' for 'too low', and 'c' if I got it right
Is it 1300?
l
Is it 1350?
h
Is it 1325?
l
Is it 1337?
c
I always win
```

Another:

```
Your number is in this range...(enter 2 numbers)
-4 4
Type 'h' for too high, 'l' for 'too low', and 'c' if I got it right
Is it 0?
h
Is it -3?
h
I win, it's -4
```

Part 2: The Upside Down Triforce (25 pts)

In this part, you will use loops to print out ASCII art. Specifically, we're going to make this upside down triforce:

```
##### 
##### 
##### 
### 
# 
##### 
##### 
##### 
### 
#
```

The catch is, it needs to be whatever size the user enters. “size” refers to the height (in lines) of one of the individual 3 shaded-in triangles. For example, in the triforce shown previously, size=5.

The details:

- All code needs to go in “Triforce.java”
- It should just prompt the user for a size, then print a triforce, then exit.
- Use “#” as the shading character.

Here are some other examples:

```
$ java Triforce
Enter size: 1
# #
#
```

```
$ java Triforce
Enter size: 2
### ###
#   #
  ###
  #
```

```
$ java Triforce
Enter size: 3
##### #####
###   ###
#     #
  #####
  ###
  #
```

Programmer of the week (optional)

If you'd like to compete in the programmer of the week challenge, you must complete the above two parts of the assignment as described. Once you get that to work, create another class, POWeek.java, and get creative with your ASCII art. Either do something cool to decorate your triforce, or make any other kind of automatically generated ASCII art. The only requirement is that your program must use loops.

Your tutor will present your animation at our weekly meeting and we'll vote for the top 5 submissions from each class and show those to the students to vote for the programmer of the week. The winner will get an award, 1% extra credit added to their final grade, and be famous for a week.

Style Requirements (10 pts)

You will be graded for the style of programming on this assignment. A few key requirements for style are given below and at <https://google.github.io/styleguide/javaguide.html>. These guidelines for style will have to be followed for all the remaining assignments. I've underlined the comments that students have lost the most points on in the past. Read them carefully.

- Use reasonable comments to make your code clear and readable.
- Use comments for all classes and methods.
 - The comments should describe the purpose of your program and methods. See the starter file for examples on how to comment a method.
- Use reasonable variable names that are meaningful.
- Use static final constants to make your code as general as possible. No hardcoding constant values inline.
- No magic numbers.
- Judicious use of blank spaces around logical chunks of code makes your code much easier to read and debug.
- Keep all lines less than 80 characters. Make sure each level of indentation lines up evenly. The default settings in vim and emacs files that I provided in HW1 will

help on this. Anytime you see your code change to a gray color you've exceed 80 characters and should break your code into 2 lines.

- Every time you open a new block of code (use a '{'), indent farther by 2 spaces. Go back to the previous level of indenting when you close the block (use a '}'). Do not use tabs. The default setting files for vim and emacs that I gave you in HW1 will automatically change your tabs to spaces, so this should be taken care of for you, but double check that it's working.
- Always recompile and run your program right before turning it in, just in case you commented out some code by mistake.

Turnin Instructions

Remember the deadline to turn in your assignment is Saturday, January 23 by 11:59pm. When you are ready to turn in your program in, type in the following command and answer the prompted questions:

```
$ cd ~/
$ bundleP3
Good; all required files are present:

    HW3

Do you want to go ahead and turnin these files? [y/n]y
OK.  Proceeding.

Performing turnin of approx. 6144 bytes (+/- 10%)
Copying to /home/linux/ieng6/cs11wb/turnin.dest/cs11wb.P3
...
Done.
Total bytes written: 31744
Please check to be sure that's reasonable.
Turnin successful.
```

bundleP3 will look for the directory HW3 and will compress the files into a single turnin submission. You can turnin your program multiple times. The bundle program will ask you if you want to overwrite a previously-turned in homework. **ONLY THE LAST SUBMISSION IS USED!**