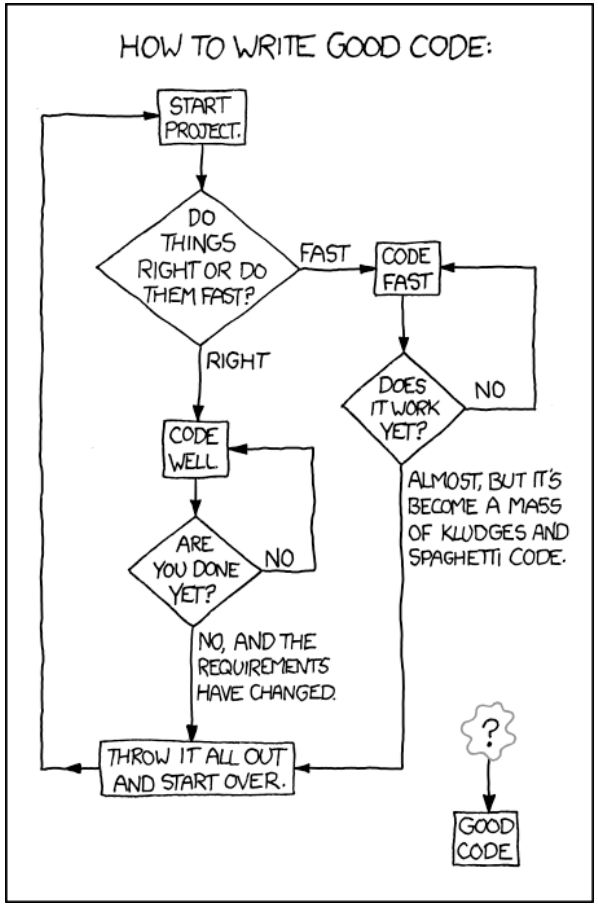


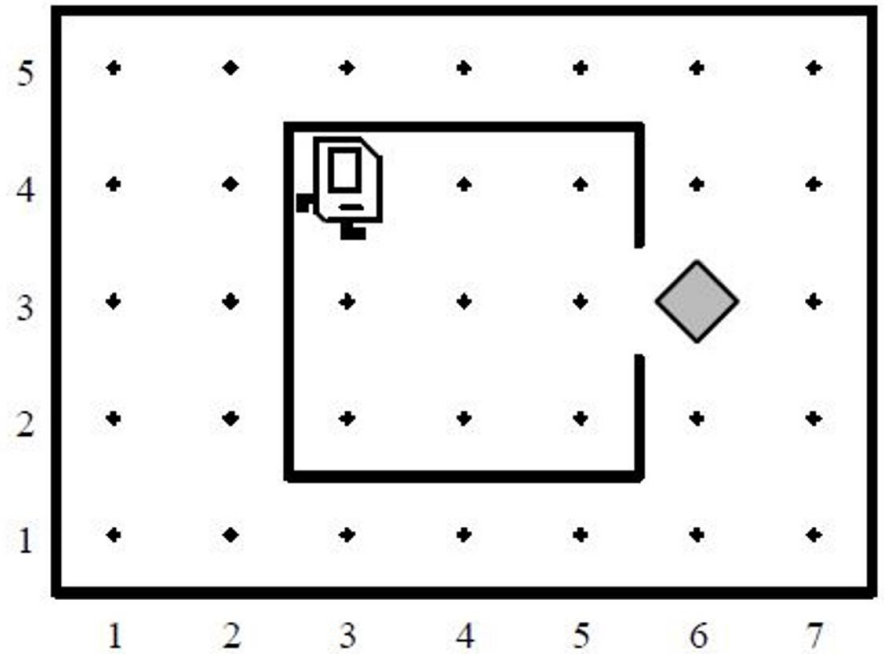
YEAH 2: Simple Java!

Avery Wang
Jared Bitz
7/6/2018

What are YEAH Hours?

- “**Y**our **E**arly **A**ssignment **H**elp”
- Only for some assignments
- Review + Tips for an assignment
- Lectures are recorded, slides are posted on website.

Bye Karel!



Variables

(From Lecture 4)

`int`	integer	`int date = 7;`
`double`	real values	`double height = 5.8;`
`char`	letters	`char letter = 'A';`
`boolean`	true/false	`boolean lovesCS106A = true;`

Variables

(From Lecture 4)

Good vs. Bad names

`i`

`something`

`numDays`

`sum`

`double`

`CONSTANT`

Variables

(From Lecture 4)

Good vs. Bad names

`i`
(unless it is a loop counter)

`something`

`numDays`

`sum`

`double`

`CONSTANT`

Constants

(From Lecture 5)

Variables whose value doesn't change.

```
private static final double CIRCLE_RADIUS = 5.5;
```

Arithmetic Operators (From Lecture 5)

`+`
`-`
`*`

Evaluates as you'd expect.

`/` Careful when dividing ints – truncates decimals!

`%` "mod" operator

Arithmetic Operators (From Lecture 5)

`a % b` What's the remainder when you divide `a` by `b`?

`17 % 2` evaluates to `1`

`52 % 2` evaluates to `0`

`100 % 3` evaluates to `1`

Logical Operators

(From Lecture 5)

`!p`	NOT	evaluates to true if `p` is false.
`p && q`	AND	evaluates to true if both `p` and `q` are true
`p || q`	OR	evaluates to true if either `p` or `q` is true

Relational Operators (From Lecture 5)

`a == b`	evaluates to true if `a` is equal to `b`.
`a != b`	evaluates to true if `a` is not equal to `b`.

`a > b`
`a >= b`
`a < b`
`a <= b`
	evaluates to true or false as you'd expect.

Relational Operators (From Lecture 5)

`a == b`

checks if `a` is equal to `b`.

```
if(a == b){
  println("equal!");
}
```

`a = b`

assigns `a` to the value of `b`.

```
int b = 3;
int a = 2;
a = b; // now a is 3
```

Control Flow

(From Lecture 5)

```
for (init; test; step){
      statements
}
```

We know how many times to iterate.

```
init
while (test) {
      statements
}
```

We don't know how many times to iterate.

Control Flow

(From Lecture 5)

```
while (true) {
  // get input
  if (input == SENTINEL){
    break;
  }
  // rest of body
}
```

```
// get input - fencepost
while (input != SENTINEL){
  // rest of body
  // get input
}
```

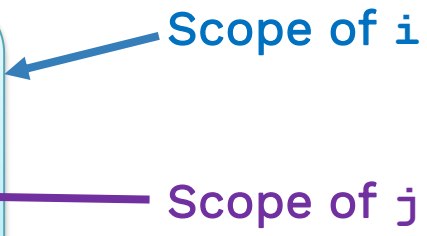
Scope

(From Lecture 6)

A variable's lifetime

- starts at initialization
- until end of code block

```
public void run(){
  for (int i = 0; i < 3; i++){
    if (i == 0){
      int j = 0;
      j++;
    }
    i--;
  }
}
```



Forbidden Java Features

(For Assignment 2)

- parameters
- return
- Strings
- instance variables (more on this later)
- concepts from Chapter 5 and beyond

Practice: FizzBuzz

- Write a program that prints all of the numbers in a range, separated by spaces
- For multiples of three print "Fizz" instead of the number
- For the multiples of five print "Buzz".
- For numbers which are multiples of both three and five print "FizzBuzz".
- Get the upper limit from the user
- For a limit of 100, the output would be:

1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz 16 17 Fizz 19 BuzzFizz 22 23 Fizz Buzz 26 Fizz 28 29 FizzBuzz 31 32 Fizz 34 Buzz Fizz 37 38 FizzBuzz 41 Fizz 43 44 FizzBuzz 46 47 Fizz 49 Buzz Fizz 52 53 Fizz Buzz 56 Fizz 5859 FizzBuzz 61 62 Fizz 64 Buzz Fizz 67 68 Fizz Buzz 71 Fizz 73 74 FizzBuzz 7677 Fizz 79 Buzz Fizz 82 83 Fizz Buzz 86 Fizz 88 89 FizzBuzz 91 92 Fizz 94 BuzzFizz 97 98 Fizz Buzz

```
public void run() {


}
```

```
public void run() {
      int limit = readInt("Limit? ");


}
```

```
public void run() {
      int limit = readInt("Limit? ");
      for (int i = 1; i <= limit; i++){

      }
}
```

```
public void run() {
      int limit = readInt("Limit? ");
      for (int i = 1; i <= limit; i++){
            if (i % 3 == 0 && i % 5 == 0){
                  print("FizzBuzz ");
            }
      }
}
```

```
public void run() {
      int limit = readInt("Limit? ");
      for (int i = 1; i <= limit; i++){
            if (i % 3 == 0 && i % 5 == 0){
                  print("FizzBuzz ");
            } else if (i % 3 == 0){
                  print("Fizz ");
            }
      }
}
```

```
public void run() {
      int limit = readInt("Limit? ");
      for (int i = 1; i <= limit; i++){
            if (i % 3 == 0 && i % 5 == 0){
                  print("FizzBuzz ");
            } else if (i % 3 == 0){
                  print("Fizz ");
            } else if (i % 5 == 0){
                  print("Buzz ");
            }
      }
}
```

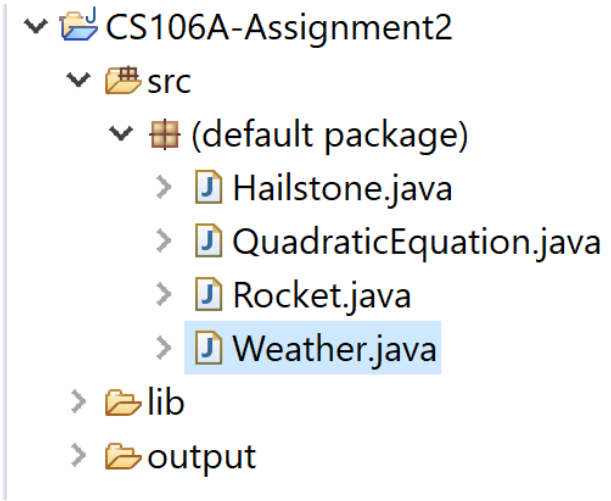
```
public void run() {
      int limit = readInt("Limit? ");
      for (int i = 1; i <= limit; i++){
            if (i % 3 == 0 && i % 5 == 0){
                  print("FizzBuzz ");
            } else if (i % 3 == 0){
                  print("Fizz ");
            } else if (i % 5 == 0){
                  print("Buzz ");
            } else {
                  print(i + " ");
            }
      }
}
```

Assignment 2: Intro to Java!

Due Date: **Wed**, Jul. 11, 2018 at 11 AM.

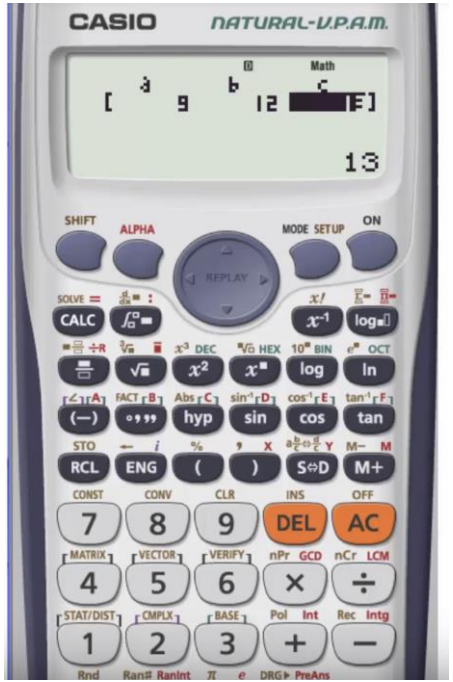
Assignment 2

- Consists of 4 console programs
- Applies concepts from lectures 4-6 (up to Tuesday's lecture) and section 2.
- Done individually.



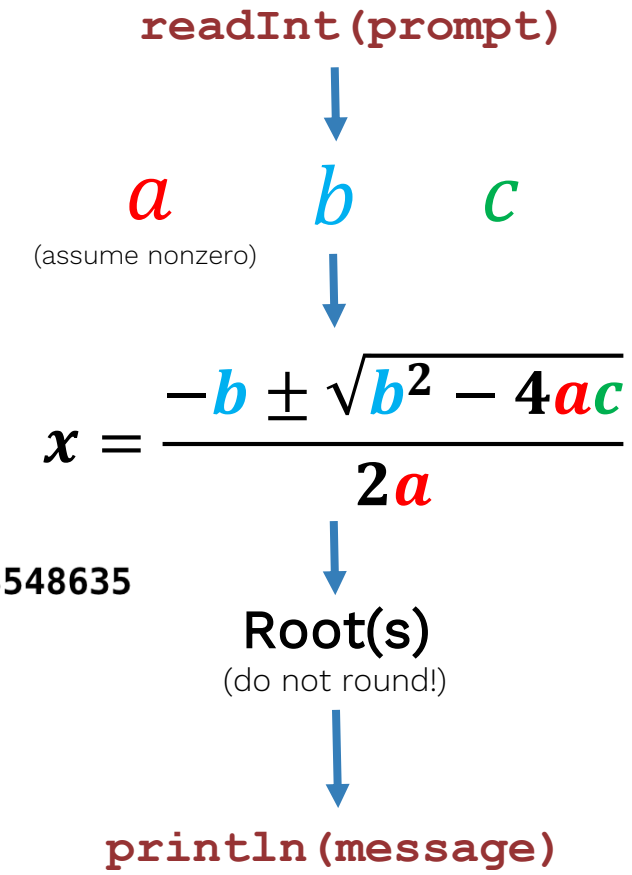
1. Quadratic Formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$



Quadratic Formula

```
CS 106A Quadratic Solver!
Enter a: 3
Enter b: 4
Enter c: -1
Two roots: 0.21525043702153024 and -1.5485837703548635
```



Discriminant

$$\Delta = b^2 - 4ac$$

$\Delta > 0$ $\Delta = 0$ $\Delta < 0$

Discriminant

$$\Delta = b^2 - 4ac$$

$\Delta > 0$	$\Delta = 0$	$\Delta < 0$
Two real roots	One root	No real roots
CS 106A Quadratic Solver! Enter a: 1 Enter b: -3 Enter c: -4 Two roots: 4.0 and -1.0	CS 106A Quadratic Solver! Enter a: 1 Enter b: 6 Enter c: 9 One root: -3.0	CS 106A Quadratic Solver! Enter a: 2 Enter b: 4 Enter c: 6 No real roots

Quadratic Formula

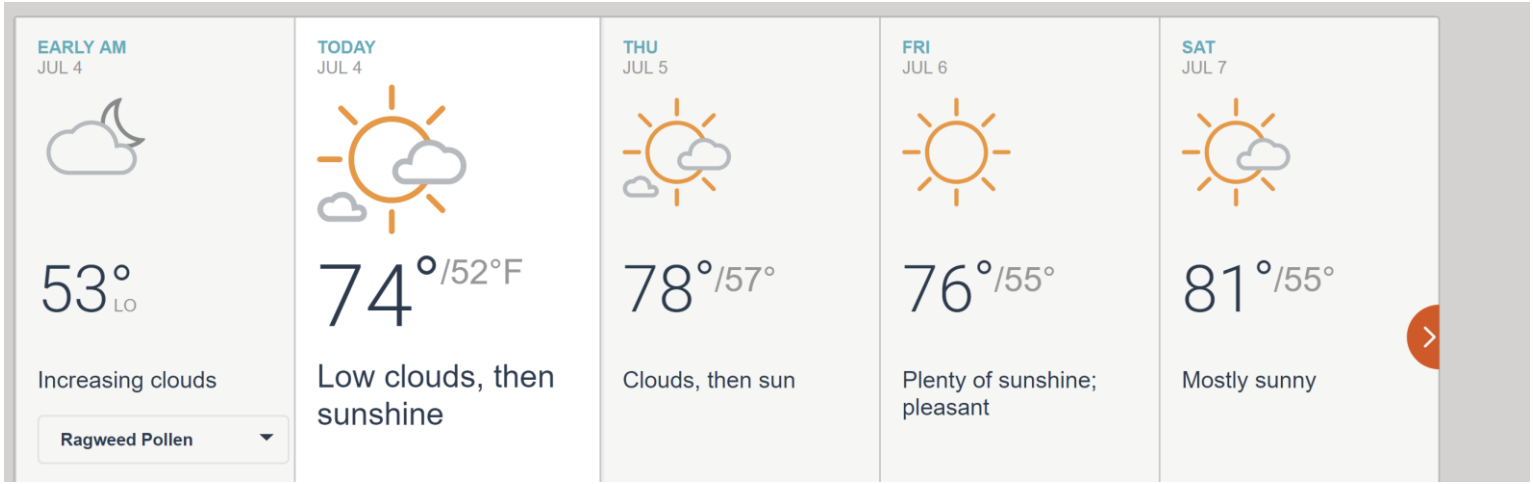
- Assume $a \neq 0$.
- Assume $a, b,$ and c are integers.
- Do not round your answer(s).

```
double y = Math.sqrt(x);
```

Useful Concepts

- Conditionals
- Math with **`int`** and **`double`**.
- Reading input.

2. Weather



Accuweather forecast for CA 94305

Weather

Prompt until **SENTINEL**.

Print the following:

- Highest temperature
- Lowest temperature
- Average temperature
- Cold days (50 degrees or less)

```
CS 106A "Weather Master 4000"!
Next temperature (or -1 to quit)? 68
Next temperature (or -1 to quit)? 94
Next temperature (or -1 to quit)? 76
Next temperature (or -1 to quit)? 45
Next temperature (or -1 to quit)? 89
Next temperature (or -1 to quit)? 36
Next temperature (or -1 to quit)? 73
Next temperature (or -1 to quit)? -1
Highest temperature = 94
Lowest temperature = 36
Average = 68.71428571428571
2 cold day(s).
```

Weather

```
CS 106A "Weather Master 4000"!
Next temperature (or -1 to quit)? 68
Next temperature (or -1 to quit)? 94
Next temperature (or -1 to quit)? 76
Next temperature (or -1 to quit)? 45
Next temperature (or -1 to quit)? 89
Next temperature (or -1 to quit)? 36
Next temperature (or -1 to quit)? 73
Next temperature (or -1 to quit)? -1
Highest temperature = 94
Lowest temperature = 36
Average = 68.71428571428571
2 cold day(s).
```

SENTINEL has value −1

(value you should set as default).

Weather

```
CS 106A "Weather Master 4000"!
Next temperature (or -42 to quit)? 76
Next temperature (or -42 to quit)? 89
Next temperature (or -42 to quit)? 83
Next temperature (or -42 to quit)? -42
Highest temperature = 89
Lowest temperature = 76
Average = 82.66666666666667
0 cold day(s).
```

SENTINEL has value −42

(one of many values you should test).

Weather

```
CS 106A "Weather Master 4000"!
Next temperature (or -1 to quit)? -10
Next temperature (or -1 to quit)? -1
Highest temperature = -10
Lowest temperature = -10
Average = -10.0
1 cold day(s).
```

SENTINEL has value –1

If only one temperature:

Highest, lowest, and average temperature are equal.

Weather

```
CS 106A "Weather Master 4000"!
Next temperature (or -1 to quit)? -1
No temperatures were entered.
```

SENTINEL has value −1

If no temperatures:

Print error message.

Weather

- `SENTINEL` must be a constant.
- Assume inputs are integers.
- Do not round your answer(s).

- Output should match **exactly**.

Useful Concepts

- Fencepost.
- Scope.
- Sentinel loops.

3. Hailstone Sequence

Hailstone Sequence

Pick some positive integer and call it n.
Do the following until n is equal to 1:

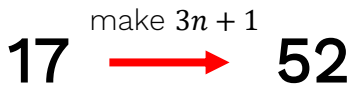
- If n is odd, multiply it by three and add one.
- If n is even, divide it by two.

17

Hailstone Sequence

Pick some positive integer and call it n.
Do the following until n is equal to 1:

- If n is odd, multiply it by three and add one.
- If n is even, divide it by two.



Hailstone Sequence

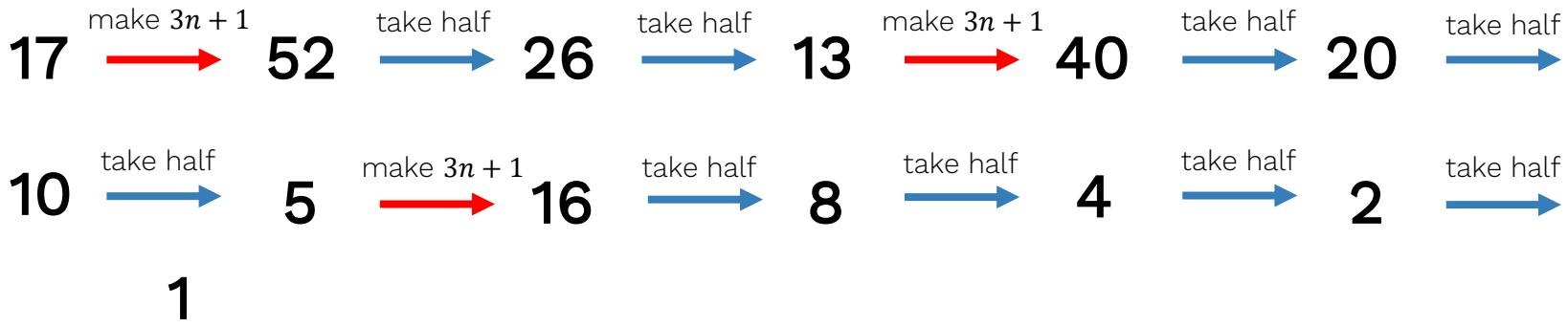
Pick some positive integer and call it n.
Do the following until n is equal to 1:

- If n is odd, multiply it by three and add one.
- If n is even, divide it by two.

Hailstone Sequence

Pick some positive integer and call it n.
Do the following until n is equal to 1:

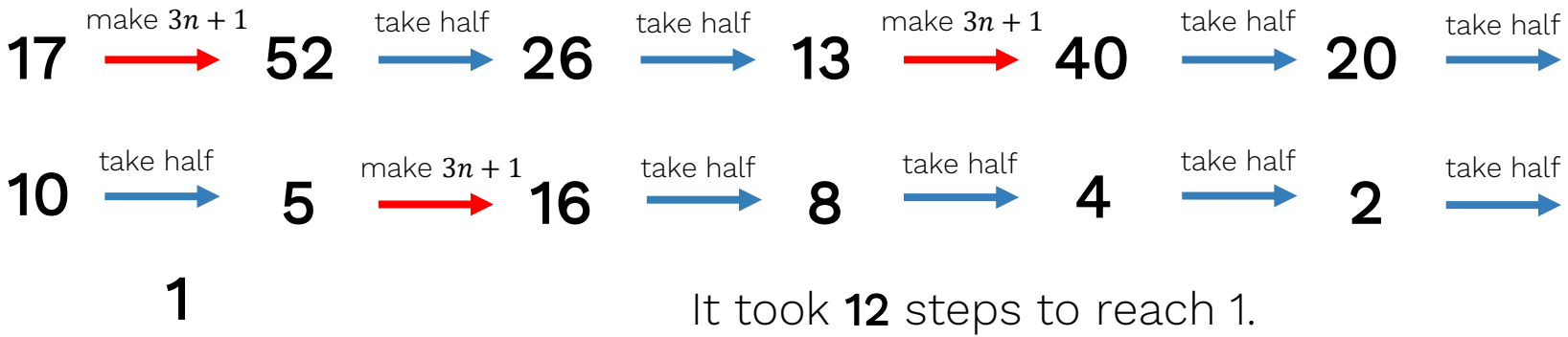
- If n is odd, multiply it by three and add one.
- If n is even, divide it by two.



Hailstone Sequence

Pick some positive integer and call it n.
Do the following until n is equal to 1:

- If n is odd, multiply it by three and add one.
- If n is even, divide it by two.



It took **12** steps to reach 1.

```
This program computes Hailstone sequences.

Enter a number: 17
17 is odd, so I make 3n + 1: 52
52 is even, so I take half: 26
26 is even, so I take half: 13
13 is odd, so I make 3n + 1: 40
40 is even, so I take half: 20
20 is even, so I take half: 10
10 is even, so I take half: 5
5 is odd, so I make 3n + 1: 16
16 is even, so I take half: 8
8 is even, so I take half: 4
4 is even, so I take half: 2
2 is even, so I take half: 1
It took 12 steps to reach 1.
Run again? y

Enter a number: 4
4 is even, so I take half: 2
2 is even, so I take half: 1
It took 2 steps to reach 1.
Run again? y

Enter a number: 1
It took 0 steps to reach 1.
Run again? n
Thanks for using Hailstone.
```

Must have a method to output a single Hailstone sequence.

Hailstone Sequence

Hailstone Sequence

- Assume input is an integer.
- Output should match **exactly** (including all spaces on the console).
- Ask the user whether to play directly inside the while loop:

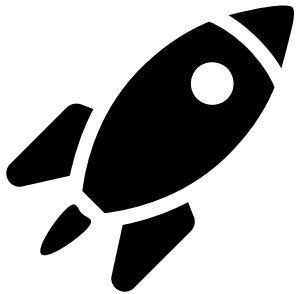
```
while (readBoolean("Run again?", "y", "n")) {
```

Useful Concepts

- Fencepost.
- Scope & loops.
- Binary Operators.

4. Rocket

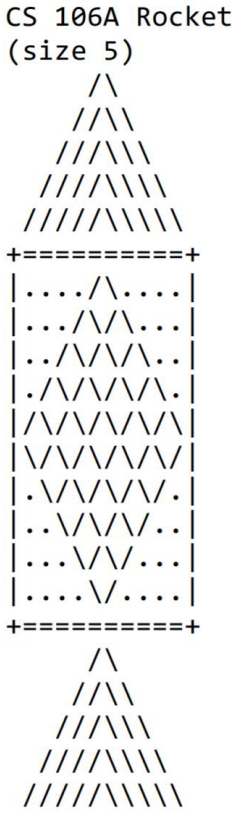
```
CS 106A Rocket
(size 5)
     /\
    //\\
   ///\\\
  ////\\\\
 /////\\\\\
+==========+
|..../\....|
|.../\/\...|
|../\/\/\..|
|./\/\/\/\.|
|/\/\/\/\/\|
|\/\/\/\/\/|
|.\/\/\/\/.|
|..\/\/\/..|
|...\/\/...|
|....\/....|
+==========+
     /\
    //\\
   ///\\\
  ////\\\\
 /////\\\\\
```



```
CS 106A Rocket
(size 3)
   /\
  //\\
 ///\\\
+=======+
|../\..|
|./\/\.|
|/\/\/\|
|\/\/\/|
|.\/\/.|
|..\/..|
+=======+
   /\
  //\\
 ///\\\
```

Rocket

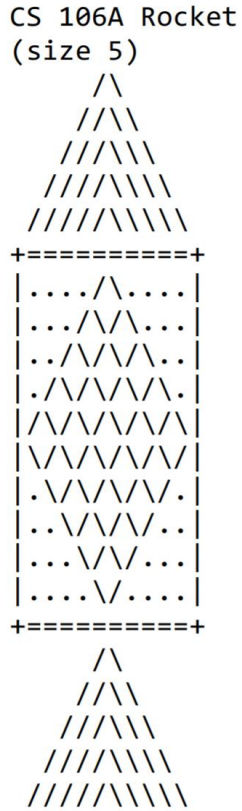
- Program is **non-interactive**.
- **`SIZE`** must be a constant.
- Assume **`SIZE`** is 2 or greater.
- Must use a **nested `for` loop**.



Rocket

`SIZE` has value 5

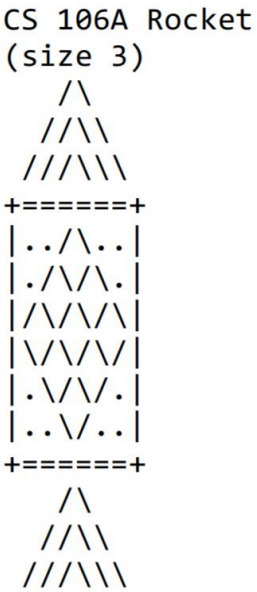
(value you should set as default).



Rocket

`SIZE` has value 3

(one of many values you should test).



Rocket

- Decompose each part of the rocket.
  - No `println()` inside `run()`
- Output should match **exactly**
- Helpful Tips:
  - Make a table.
  - Solve the default size (5) before using constant.

Useful Concepts

- Nested `for` loop.
- Constants.
- Decomposition.

Example from Tuesday

```
for (int i = 0; i < 5; i++) {
    for (int j = 0; j < i + 1; j++) {
        print("*");
    }
    println();
}
```

A.
```
*****
*****
*****
*****
*****
```

B.
```
*****
****
***
**
*
```

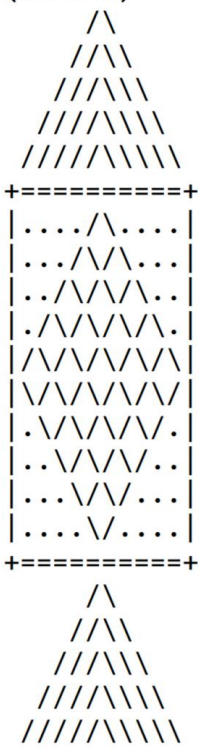
C.
```
*
**
***
****
*****
```

D.
```
1
22
333
4444
55555
```

E.
```
12345
```

(How would you modify the code to produce each output above?)

CS 106A Rocket (size 5)

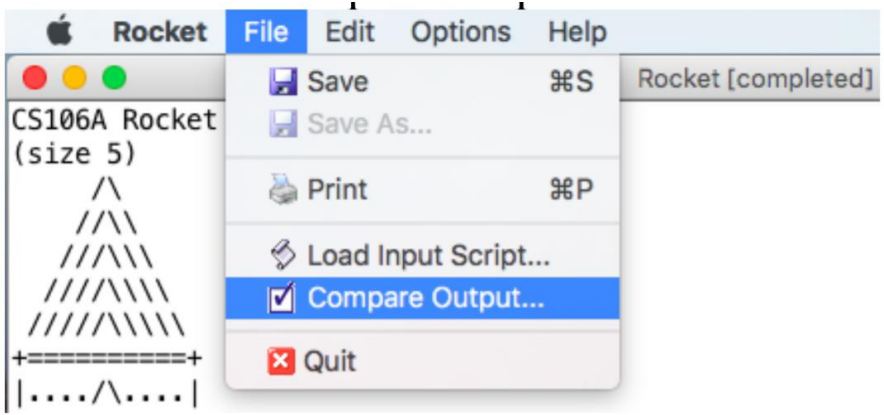


Note about Instance Variables

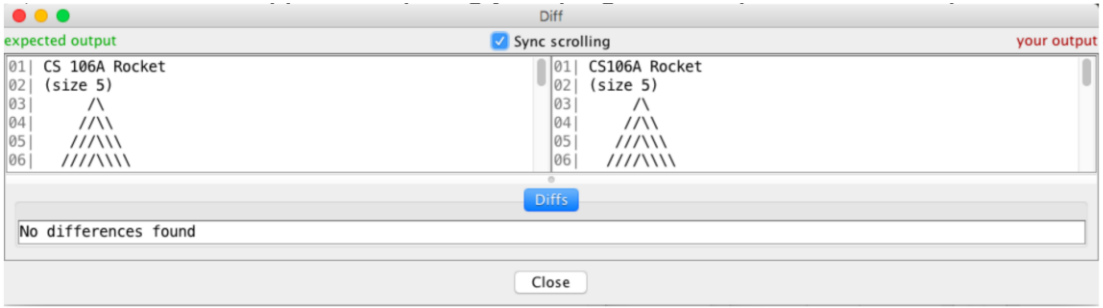
```
public class Example {
	private static final int SIZE = 5; // constant
	private int num = 0; // instance variable – bad!

	public void run() {
		int sum = 0; // local variable
	}
}
```

For this assignment, don't use non-constant variables declared outside of methods to get avoid having to deal with scope issues!

Output Comparison Tool



Output should match **exactly.**



Other Advice

- Read spec very carefully about requirements.
- Use constant, but no instance variables.
- Read the Assignment 2 style guide.
- Fix a bug, before moving on.
- Make sure output matches exactly (Output Comparison Tool).
- Test your programs extensively.
- Visit the LaIR if you get stuck.
- Incorporate feedback from Assignment 1!

Questions?

Have fun!