YEAH Hours: Assignment 3

CS 106A
Summer 2018

Jared Bitz
Previously: Methods are a way of organizing and efficiently repeating identical blocks of code.

```java
private static final int SIZE = 5;

private void drawTriangle() {
    for(int i = 1; i <= SIZE; i++) {
        for(int j = 1; j <= i; j++) {
            print("*");
        }
        println();
    }
}
```
Previously: Methods are a way of organizing and efficiently repeating identical blocks of code

```java
private static final int SIZE = 5;

private void drawTriangle() {
    for(int i = 1; i <= SIZE; i++) {
        for(int j = 1; j <= i; j++) {
            print("*");
        }
        println();
    }
}
```

Now: Methods are a way of organizing and efficiently repeating similar blocks of code

```java
private void drawTriangle(int size) {
    for(int i = 1; i <= size; i++) {
        for(int j = 1; j <= i; j++) {
            print("*");
        }
        println();
    }
}
```
Key Question: When I do this task multiple times, what changes from one run to another?
Don’t Overthink It!

https://imgs.xkcd.com/comics/the_general_problem.png
Previously: Calling a method is like giving a command to the computer

```java
private void foo(int x){
    println(2 * x);
}

public void run(){
    foo(2); //prints 4
}
```
Previously: Calling a method is like giving a command to the computer

```java
private void foo(int x)
{
    println(2 * x);
}

public void run()
{
    foo(2); //prints 4
}
```

Now: Calling a method can also be like asking a question to the computer

```java
private int bar(int x)
{
    return 2 * x;
}

public void run()
{
    int result = bar(2);
    println(result); //prints 4
}
```
Return Values – Things to Remember

- Returning is **different from printing**!
  - Sometimes, we want to do things with the result of an operation rather than immediately print them out (e.g. store them in a variable for later use).
Return Values – Things to Remember

- Returning is different from printing!
  - Sometimes, we want to do things with the result of an operation rather than immediately print them out (e.g. store them in a variable for later use).

- Methods can be both commands and questions!
  - Just because a method returns something doesn’t mean that it doesn’t also perform an action.
Return Values – Things to Remember

● Returning is **different from printing**!
  ○ Sometimes, we want to do things with the result of an operation rather than immediately print them out (e.g. store them in a variable for later use).

● Methods can be **both commands and questions**!
  ○ Just because a method returns something doesn’t mean that it doesn’t also perform an action.

● You are allowed to have **multiple return statements** in the same function!
  ○ Let’s see an example!
Example: Hailstone Revisited

```java
public int nextHailstoneStep(int n) {
    if(n % 2 == 0) {
        return n / 2;
    } else {
        return 3 * n + 1;
    }
}
```

**Note:** With code that branches like this, make sure that every possible path eventually reaches a return statement. Otherwise, your code won’t compile!
# Strings

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int length()</td>
<td>Returns the number of characters in the string</td>
</tr>
<tr>
<td>char charAt(int index)</td>
<td>Returns the character at index</td>
</tr>
<tr>
<td>string substring(int begin)</td>
<td>Returns the part of the string after index begin</td>
</tr>
<tr>
<td>string substring(int begin, int end)</td>
<td>Returns the part of the string between indices begin and end</td>
</tr>
<tr>
<td>int indexOf(string str)</td>
<td>Returns the first index where str appears in this string (or -1 if not found)</td>
</tr>
<tr>
<td>string toLowerCase()</td>
<td>Returns an uppercase/lowercase version of the string</td>
</tr>
<tr>
<td>string toUpperCase()</td>
<td></td>
</tr>
</tbody>
</table>

**Note**: All of these methods are called on a specific string!

```java
String s = "Hello World!";
println(s.substring(1, 3));
```
String Indexing

Hello, World
s.substring(3, 12);

**How I Remember It:** Think of the indices as being at the bottom left of each character. The result of `substring(begin, end)` is the letters that are physically between `begin` and `end`. 
Strings - Things to Remember

- Strings are **immutable** - they can’t be modified directly
  - If we want to modify a string, we need to build up a new version from scratch
- Make sure not to use indices **beyond the end of the string** - your program will crash if you do
- Useful paradigm: looping over the characters of a string:

```java
for(int i = 0; i < s.length(); i++) {
    char c = s.charAt(i);
    //Do something with c
}
```
try {
    Scanner input = new Scanner(new File("myfile.txt"));
    while(input.hasNext()) {
        String s = input.nextLine();
        println(s.toUpperCase());
    }
} catch (Exception e) {
    println(e.getMessage());
}
try {
    Scanner input = new Scanner(new File("myfile.txt"));
    while(input.hasNext()) {
        String s = input.nextLine();
        println(s.toUpperCase());
    }
} catch (Exception e) {
    println(e.getMessage());
}
try {
    Scanner input = new Scanner(new File("myfile.txt"));
    while(input.hasNext()) {
        String s = input.nextLine();
        println(s.toUpperCase());
    }
} catch (Exception e) {
    println(e.getMessage());
}
try {
    Scanner input = new Scanner(new File("myfile.txt"));
    while(input.hasNext()) {
        String s = input.nextLine();
        println(s.toUpperCase());
    }
} catch (Exception e) {
    println(e.getMessage());
}

We can loop through a file by continually checking input.hasNext(), which returns a boolean indicating whether we’ve hit the end or not.
try {
    Scanner input = new Scanner(new File("myfile.txt"));
    while(input.hasNext()) {
        String s = input.nextLine();
        println(s.toUpperCase());
    }
} catch (Exception e) {
    println(e.getMessage());
}
The Assignment: Snowman!
Assignment Logistics

- Due 11AM on Thursday, July 19th
- **Partners** are allowed, but you must work with someone from your section!
- Uses lecture material up through Wednesday, July 11th
- We give you the decomposition (though you’re encouraged to decompose further if you see fit)
CS 106A Snowman!
I will think of a random word. You'll try to guess its letters. Every time you guess a letter that isn't in my word, a new piece of the snowman appears. Guess correctly to avoid bringing him to life in the sun!

Task #0 – Introduction Message

- Requires writing the `intro` method
- Just prints an intro message to the console. You’re all experts at this by now :)
Task #1 – Single Game

1. Program presents user with a hint and tells them what they’ve guessed so far and how many guesses they have left

Secret word: ---------
Your guesses:
Guesses left: 8
Task #1 – Single Game

1. Program presents user with a hint and tells them what they’ve guessed so far and how many guesses they have left
2. User guesses a letter

Secret word : ----------
Your guesses: 
Guesses left: 8
Your guess? r
Task #1 – Single Game

1. Program presents user with a hint and tells them what they’ve guessed so far and how many guesses they have left
2. User guesses a letter
3. Program tells user whether or not they were correct

Secret word: ---------
Your guesses:
Guesses left: 8
Your guess? r
Correct!
Task #1 – Single Game

1. Program presents user with a hint and tells them what they’ve guessed so far and how many guesses they have left
2. User guesses a letter
3. Program tells user whether or not they were correct
4. Repeat the process with a (possibly) new hint string

Secret word : *******
Your guesses:
Guesses left: 8
Your guess? r
Correct!
Secret word : -R--R----R
Your guesses: R
Guesses left: 8
Your guess? s
Incorrect.
Task #1 – Single Game

1. Program presents user with a hint and tells them what they’ve guessed so far and how many guesses they have left
2. User guesses a letter
3. Program tells user whether or not they were correct
4. Repeat the process with a (possibly) new hint string
5. Keep going until the user runs out of guesses or loses

Secret word : ---------
Your guesses:
Guesses left: 8
Your guess? r
Correct!
Secret word : -R--R----R
Your guesses: R
Guesses left: 8
Your guess? s
Incorrect.
...
Secret word : PROGR-MMER
Your guesses: RSTPXONGYMDE
Guesses left: 2
Your guess? a
Correct!
You win! My word was "PROGRAMMER".
Task #1 – Single Game

- Requires writing the `playOneGame` method
  - What other methods will you need to write in order to get `playOneGame` to work?
  - Once you’ve decided, consider writing (and testing) the smallest parts first, then building up to a complete solution
- How will you keep track of the guessed letters?
- For testing, you can print out the secret word before the game begins, so you know what it is you’re trying to guess
- User input should be checked for errors (e.g. not a single character, already guessed) - reprompt if it’s bad
- Creating the hint requires manipulating strings - good examples in Lecture 9
Task #2 - Display Snowman (ASCII Art)

- Requires writing the `displaySnowman` method
- ASCII art is located in files named `display0.txt` through `display8.txt`
- Use a `Scanner` to read the files:
  ```java
  Scanner input = new Scanner(new File("display0.txt"));
  ```
- Consult Lecture 10 for good examples of reading and displaying text files
Task #3 - Choosing Random Words

73
ABSTRACT
AMBASSADOR
... (70 lines omitted)
ZIRCON
Task #3 - Choosing Random Words

- Requires writing the `getRandomWord` method
- Algorithm to find a random word:
  - Find the number of words from the first line of the file
  - Pick a random line in the file
  - Advance the scanner to that line
  - Return whatever word you find
- Requires more file reading (see previous slide) and `RandomGenerators` (see Lecture 8)
  - `RandomGenerator.getInstance().nextInt(min, max);`
- `Scanner.nextLine()` behaves strangely when used with `nextInt()`. For this assignment, just use `next()` instead.
- You can use the `promptUserForFile()` to ask the user for a filename.
Task #4 – Multiple Games and Statistics

- When the game is over, ask the user if they want to play again
  - We already have a `playOneGame` method - how can we take advantage of that for this task?
  - Can use the `while-readBoolean` idiom:
    ```java
    while (readBoolean("prompt text", "Y", "N")) { ... }
    ```
- Track **statistics** such as win rate and best game and display them at the end
  - Requires writing the `stats` method
  - Where should the variables to track these quantities go? What scope should they live in?
  - How do we know the results of a single game and communicate them across methods?
Common Pitfalls

- **String comparison** - remember to use `str1.equals(str2)`
- Remember **not to change the parameters or return types** of the given methods
- **Off-by-one errors:**
  - `RandomGenerator.nextInt(low, high)` is inclusive
  - `String.substring(begin, end)` is inclusive at the beginning and exclusive at the end
- **No instance variables are allowed** - it is 100% possible to do all the required tasks without them