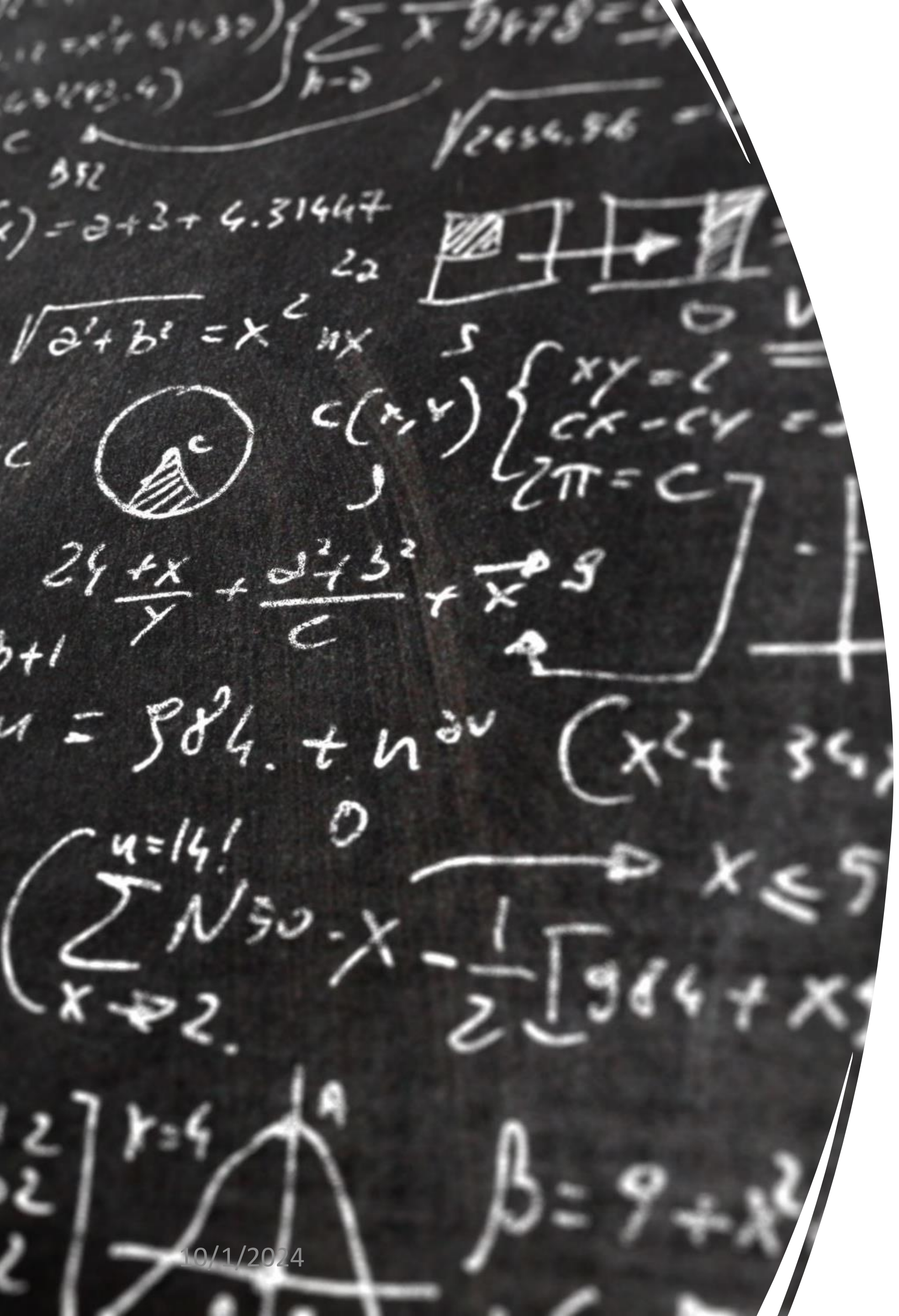


CS 113 – Computer Science I

Lecture 08 – Recursion & Arrays

Tuesday 10/01/2024

Announcements

- HW04 – released
  - Due Monday 10/07 11:59pm
- Thursday 10/03
  - No class
- Office hours:
  - Adam’s Thursday 2:40-4:00pm are cancelled this week

Agenda

Recursion

Arrays

Exercise: Blackjack

Write a program Blackjack.java which generates a random value between 2 and 21

- If the value is 21, print the value and “Blackjack” to the console
- If the value is between 17 and 20, print the value and “Stand” to the console
- If the value is less than 17, print the value and “Hit me!” to the console

Top down design

1. Identify features of the program
    1. List them out!
2. Identify verbs and nouns in feature list
    1. Verbs: functions
    2. Nouns: objects/variables
3. Sketch major steps – how features should fit together
    1. Algorithm!
4. Write program skeleton
    1. Include function **stubs** (placeholders for our functions)
    2. Function **stub:** empty function with parameters and return type
5. Implement and test function stubs one at a time

Recursion

a function that calls itself

“Simple” way to solve “similar” problems

Creating a recursive algorithms

Rule that “does work” then ”calls itself” on a smaller version of the problem

Base case that handles the smallest problem

Prevents “infinite recursion”

Recursion example – print “hello” 5 times

Rule: Print “hello” once and then print “hello” 4 times

Base case: When the number of times to print is 0, stop printing

Recursive functions – base case

Conditional statement that prevents infinite repetitions

Usually handles cases where:

input is empty

problem is at its smallest size

Recursion Example - Factorial

$$n! = n \ * (n-1) * (n-2) * \ \ldots * 1$$

3! = 3 * 2 * 1 = 6

4! = 4 * 3 * 2 * 1 = 24

Visualizing recursion – Factorial example

factorial(5) =

= 5 * factorial(4)

= 5 * 4 * factorial(3)

= 5 * 4 * 3 * factorial(2)

= 5 * 4 * 3 * 2 * factorial(1)

= 5 * 4 * 3 * 2 * 1

Recursion Example – Contains letter

Write a method called “containsLetter” that determines if a String contains a given character

Question: What are the parameters?

1. The String to be looking in
2. The character to look for

Question: What is the return type?

Recursion Example – Contains letter

How can we break this problem down into smaller problems?

contains(“l”, “apple”) =

    contains(“l”, “a”) OR

    contains(“l”, “p”) OR

    contains(“l”, “p”) OR

    contains(“l”, “l”) OR

    contains(“l”, “e”) OR

Recursion Visualization – Contains letter

```
contains(“l”, “apple”) =
        contains(“l”, “apple”)
                contains(“l”, “pple”)
                        contains(“l”, “ple”)
                                contains(“l”, “le”)
                                        return true
```

Recursion Example – IndexOf letter

Write a method called IndexOf.

Arguments: String (haystack), Character (needle)

Return: the index of the character in the String, if the chatacter isnt there, return:

-1.

Recursion Example – printVowels

Write a recursive function that prints just the vowels in a String

Recursion limitations

- Limited number of times we can recurse
  - Stackoverflow – too many frames
- Potentially memory inefficient
  - If we copy data in subproblems – we'll worry about this in a few weeks
- Performance: might duplicate unnecessary work
  - We'll define performance later in the semester

Style

- How we format our programs is **very** important
    - Like rules of etiquette around eating and keep a clean appearance
    - Like punctuation rules, it helps make text more readable
- Variable names should be descriptive
- Indentation is **very** important
    - Every statement inside a pair of braces must be indented
- Braces should be placed consistently

Arrays

Arrays

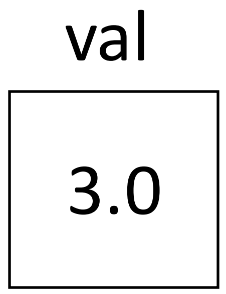
Idea: Store multiple values into a single variable

Values are sequential

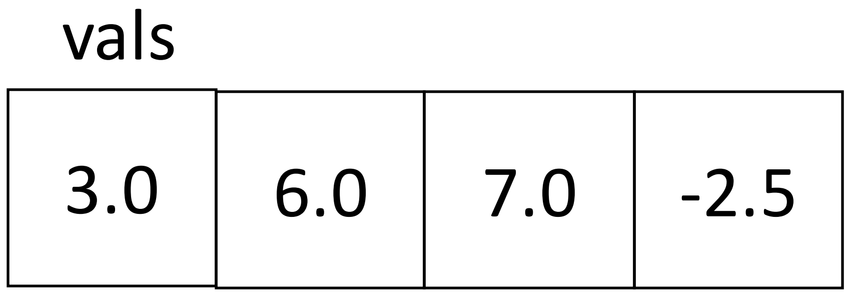
Analogous to a list

Arrays

```
double val = 3.0;
```



```
double[] vals = {3.0, 6.0, 7.0, -2.5};
```



Arrays

Three ways to initialize an array

1. With an initial value
```
int[] numbers = {1, 2, 5};
```
2. With allocated space, but uninitialized
```
int[] numbers = new int[3];
```
3. With an empty array reference
```
int[] numbers = null;
```

Array Indexing

Access individual elements of an array with indexing

```
array[index]
```

Variable name

Integer

We use *zero*-based indexing

first element is **0**

last element is **length-1**

Accessing indices out of range results in a **runtime error**!

Exercise: print backwards

Write a program, Backwards.java, that asks the user for 3 integers and then prints the list of numbers in reverse order

Strings

Strings are implemented as *arrays of characters*

Get the length of a string with length()

```
        String greeting = “hola”;
        int len = greeting.length(); // what is the length?
        char c = greeting[2]; // what character is in index 2?
```

char: built-in Java type, denoted with single quote, e.g. ‘a’ or ‘{‘

Strings as an array of characters

String str = “hello world”

- How many characters in this String?

  10
- How do we access the first character?

  str.charAt(0)

- How do access the 5th character?

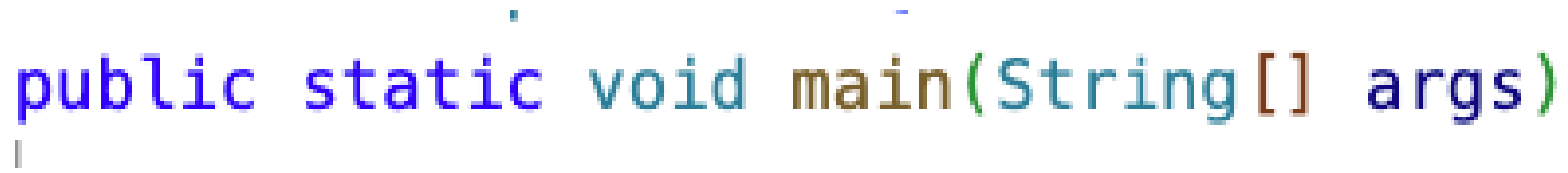
  str.charAt(4)

Exercise: GetCharacters.java

Write a program, GetCharacters.java, that asks the user for a word and then prints the first, last and middle character.

```
Enter a word: hola!
FirstIndex: 0 FirstCharacter: h
MiddleIndex: 2 MiddleCharacter: l
LastIndex: 5 LastCharacter: !
```

Command line arguments



```java
public static void main(String[] args)
```

Command line arguments are an *array of String*

Exercise: Write a program called commandLineArgs.java that

1) prints out 3 command line arguments that are passed in.
2) Compute the sum of three command line arguments (assuming they are integers)

Recursion Example – printList

Write a recursive function that prints the contents of an array