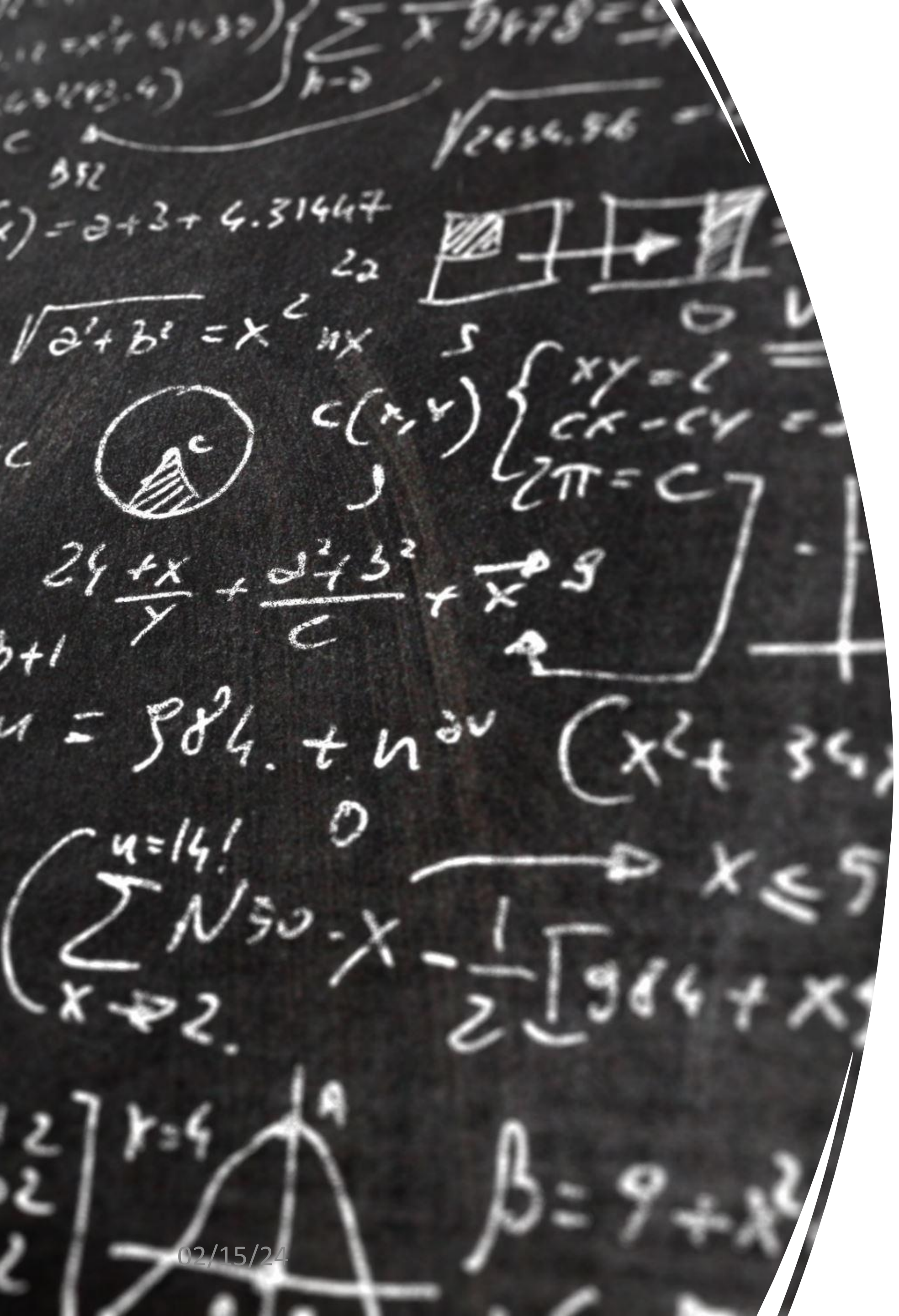


CS 113 – Computer Science I

Lecture 08 – String Methods & Recursion

Thursday 02/15/2024

Announcements

- HW02 deadline extended to Sunday

Answer the Piazza OH poll

Agenda

String Comparison review

Recursion

Comparing strings

- In Java, you cannot directly compare strings using ==

- Instead, use **compareTo**
  - Javadocs: https://docs.oracle.com/javase/7/docs/api/java/lang/String.html

Recursion

Recursion

a function that calls itself

Base case that handles the smallest problem

Rule that *does something* then *calls itself* on a smaller version of the problem

Recursion example – print “hello” 5 times

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

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

Recursion

a function that calls itself

Each recursive call should move towards a base case where a direct solution can be found.

Base case that tells us when to stop

Rule that *does something* then *calls itself* on a smaller version of the problem

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

- What is a factorial? n!
- product of all integers less than or equal to n
    - n! = n * n-1 * n-2 1
    - 5! = 5 * 4 * 3 * 2 * 1
    - 4! = 4 * 3 * 2 * 1
    - 3! = 3 * 2 * 1

- Factorial.java
- What is the base case?

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
```

Exercise: Blast Off

Write a recursive method: `void BlastOff(int n)`

Which prints a count down from n to 1 and then prints “Blast off!”

Example:

`BlastOff(3)` prints

```
3
2
1
Blast off!
```

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 character to look for
2. The string to be looking in

Question: What is the return type?

Recursion Visualization – Contains letter

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

Recursion containsLetter

Recursion Example – printVowels

Your turn!

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

Recursion Example – IndexOf letter

Your turn again! Write a method called IndexOf.

Arguments: String (haystack), Character (needle)

Return: the index of the character in the String. You can assume needle is in haystack.

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 gg=G

- 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

Filing Cabinet

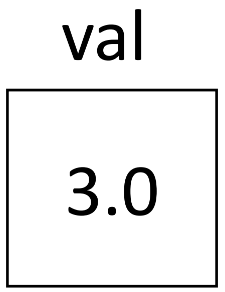
Idea: Store multiple values into a single variable

Values are sequential

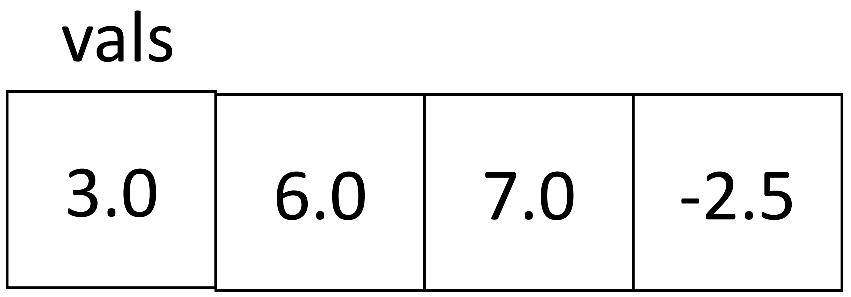
Analogous to a list

Arrays

```
double val = 3.0;
```



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



Array Indexing

Access individual elements of an array with indexing

    array[index]

We use *zero*-based indexing

    first element is **0**

    last element is **length-1**

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

Arrays

Three ways to initialize an array

1. With an initial value

```java
int[] numbers = {1, 2, 5};
```

2. With allocated space, but uninitialized

```java
int[] numbers = new int[3];
```

3. With an empty array reference

```java
int[] numbers = null;
```

Arrays

```
int[] x = {2, 7, 5};
System.out.println(x.length); //what will this print?
```

.length field tells us how many elements are in the array

Once an array is full, you cannot add more elements to it!