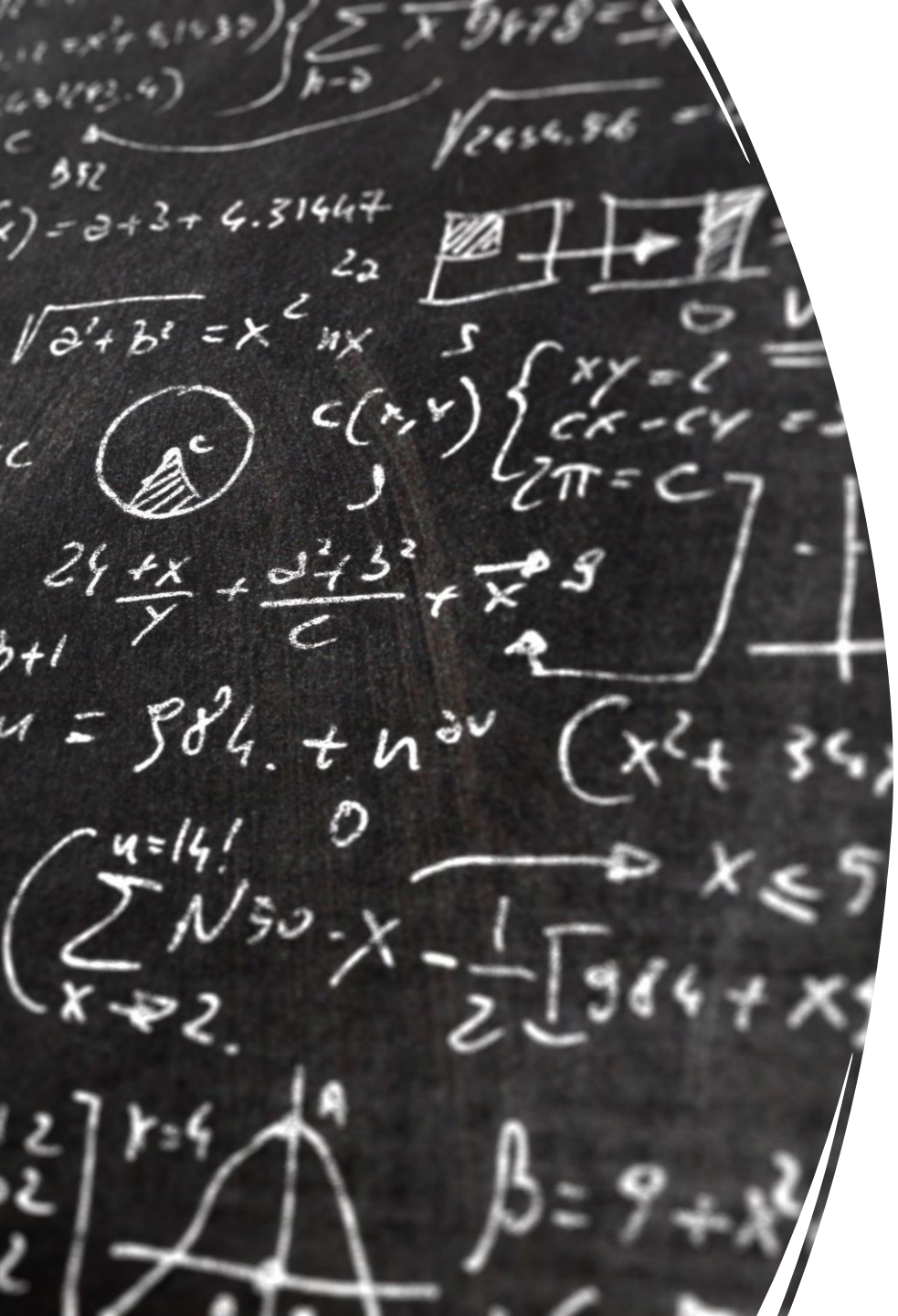

CS 113 – Computer Science I

Lecture 10 – Functions

Adam Poliak

10/06/2022

Announcements

- Assignment 04
  - Due Thursday 10/06 - tonight
- Office hours:
  - Today: 2:45-3:45pm

Unit testing

Verify that function is implemented correctly

Call the function with different inputs and check the results

In a library, we can use the main method to test functions

Exercise: guess number

Write a program that asks the user to guess a random number between 1 and 100 and check if it's the same as the computer's number:

- If the user's guess is too low, the computer should say "<num> is too low!"
- If the user's guess is too high, the computer should say "<num> is too high!"
- If the user guesses the right number, the computer should say "You win!"

Guess my number

- Let’s use IsInteger to check the user’s input

Scope

What variables are in scope in area()? in main()?

Scope

```
public class Area {

    public static double area(double width, double height) {
        float result = width * height;
        return result;
    }

    public static void main(String[] args) {

        double size = area(10.0, 5);
        System.out.println("Area is "+ size);
    }
}
```

Scope

What variables are in scope in shuffle()?

```
/**
 * Rearranges the elements of the specified array in uniformly random order.
 *
 * @param  a the array to shuffle
 * @throws IllegalArgumentException if {@code a} is {@code null}
 */

public static void shuffle(char[] a) {
    validateNotNull(a);
    int n = a.length;
    for (int i = 0; i < n; i++) {
        int r = i + uniformInt(n-i);     // between i and n-1
        char temp = a[i];
        a[i] = a[r];
        a[r] = temp;
    }
}
```

Draw a stack diagram for this program

```
class Add1 {

  public static int Add(int a, int b) {
    int result = a + b;
    return result;
  }

  public static void main(String[] args) {
    int a = 4;
    int b = 8;
    int c = Add(b, a);
    System.out.printf("%d + %d = %d\n", a, b, c);
  }
}
```

Draw a stack diagram for this program

```
class Add2 {

  public static int Add(int a, int b) {
    a = 2;
    int result = a + b;
    return result;
  }

  public static void main(String[] args) {
    int a = 4;
    int b = 8;
    int c = Add(a, b);
    System.out.printf("%d + %d = %d\n", a, b, c);
  }
}
```

Draw a stack diagram for this program

```
class Add3 {

  public static int Add(int[] a) {
    if (a.length != 2) return -1;
    int result = a[0] + a[1];
    return result;
  }

  public static void main(String[] args) {
    int[] a = {4, 8};
    int c = Add(a);
    System.out.printf("%d + %d = %d\n", a[0], a[1], c);
  }
}
```

Draw a stack diagram for this program

```
class Add4 {

  public static int Add(int[] a) {
    if (a.length != 2) return -1;
    a[0] = 2;
    int result = a[0] + a[1];
    return result;
  }

  public static void main(String[] args) {
    int[] a = {4, 8};
    int c = Add(a);
    System.out.printf("%d + %d = %d\n", a[0], a[1], c);
  }
}
```

Immutability vs. mutability

What happens when we change an arguments value in a function?

Immutable: values do not change, they are copied

Mutable: values can be changed without copying

Importance: subtlety of passing in arguments

Immutability vs. mutability

Immutable types

- String
- Boolean
- ints
- double
- char

Mutable types

- Arrays
- Objects
  - Will cover this after fall break

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

Exercise: math quiz

Welcome to math quiz!

4 + 9 = rtr

Invalid input!

4 + 9 = 12

Sorry the answer is 13

8 + 6 = 14

Correct!!!!

1 + 3 = 4

Correct!!!!

8 + 0 = 8

Correct!!!!

2 + 9 = 11

Correct!!!!

Your score is 0.80 (4/5)