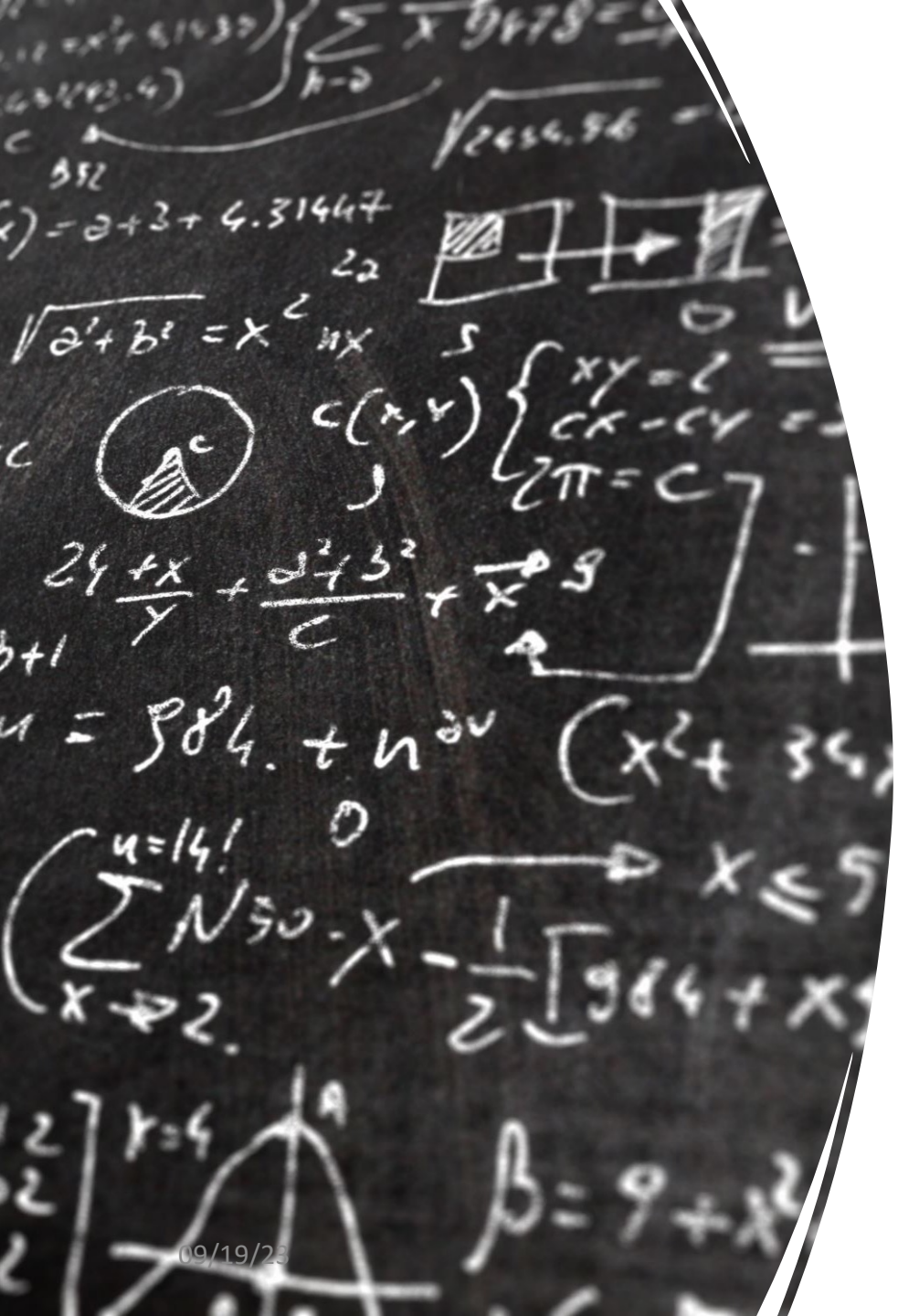

CS 113 – Computer Science I

Lecture 05 – Methods II

Tuesday 09/19/2023

Announcements

- HW01 – due last night
- HW02 – releasing later today
  - Due Monday 09/25

- **Read & Follow Instructions**
  - Don’t just skim the labs & homework

- Office hours:
  - Monday: 2:45-4:00pm
  - Tuesday: 9:30-10:45am (I’ll try to get there by 9:15)
  - This week: Thursday 9:30-10:45am

Creating Methods

Idea: Define re-useable portions of code

Analogy: machines with inputs and outputs

Two steps for programming with functions:

1. Define the function (name, inputs, outputs, implementation)
2. Call the function with inputs and wait for its output

All methods should be contained inside a class

Anatomy of a method

- All methods have the following things:
  - Name
  - Parameter
  - Body
  - Return Type

```
public static int method1 (int param1,
                           String param2) {
  /**
    body of the method
  */
  return 0;
}
```

Method documentation

```
/**
Description of the method
* @param param1 description
* @param param2 description
* @return what the method returns
*/
public static int method1 (int param1,
                           String param2) {
   /**
```

Defining methods in Java: syntax

```
public static void main(String[] args) {
    // function statements
}
```

```
public static float foo(int a, float b, String c) {
    // function statements
    System.out.println(c);
    return a*b;
}
```

Calling methods in Java: syntax

```java
public static float foo(int a, float b, String c) {
    // function statements
    System.out.println(c);
    return a*b;
}
```

parameters

```java
public static void main(String[] args) {
    // function statements
    int value = 3;
    String c = "hello";
    float result = foo(value, -2.5, c);
    System.out.println(result);
}
```

arguments

Executing a method: steps

1. When you encounter a method, pause!
2. Create a *frame* to hold the method state
3. Copy argument values
4. Execute the method, line by line. Continue until
    1. you hit a return statement
    2. you run out of statements
5. Send back return value (can be nothing if function is *void*)
6. Delete the method's frame
7. Resume original function

What is different here?

```java
// Function: area
// Description: computes the area of a rectangle
// Input: width (double)
// Input: height (double)
// returns (double), the area as width * height
// side effects: none
public static double area(double width, double height) {
    return width * height;
}
```

```java
// Function: area
// Description: computes the area of a rectangle
// Input: width (double)
// Input: height (double)
// returns (none)
// Side effect: prints the area to the console
public static void area(double width, double height) {
     double a = width * height;
     System.out.println("Area is "+ a);
}
```

Warning: don’t confuse printing with returning

```java
// Function: area
// Description: computes the area of a rectangle
// Input: width (double)
// Input: height (double)
// returns (double), the area as width * height
// side effects: none
public static double area(double width, double height) {
    return width * height;
}
```

```java
// Function: area
// Description: computes the area of a rectangle
// Input: width (double)
// Input: height (double)
// returns (none)
// Side effect: prints the area to the console
public static void area(double width, double height) {
     double a = width * height;
     System.out.println("Area is "+ a);
}
```

Benefits of methods

- Split large problems into small problems
- Easier to maintain code/cleaner code
  - Only need to fix mistakes
  - DRY: Don’t repeat yourself
- Implement once, re-use in different programs
- Abstract details so user doesn’t need to worry about details

Method: distanceFromA

```
$ java LetterStats
Enter a letter: b
b is 1 away from a
Enter a letter z:
z is 25 away from a
Enter a letter h:
z is 7 away from a
```

Scope

- area of a program where a variable can be used
- Stack diagram's helpful for identifying scope
- Online demo with pythontutor.com: https://pythontutor.com/java.html#mode=edit

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.printf("Area is %d\n”, size);
    }
}
```

Method specifications

Idea: “contract” between the function user and the method implementation

- Inputs and their types
- Return type
- Description of how function behaves, including special cases and side effects

The **method signature** includes just the inputs and outputs of the function

Method Specifications

```
/**
 * Returns a random real number from a Gaussian distribution with
 *  mean &mu and standard deviation &sigma
 *
 * @param mu the mean
 * @param sigma the std
 * @ return a real number distributed according to the Gaussian distribution
 * /
public static double gaussian(double mu, double sigma) {
        return mu + sigma * gaussian();
}
```

Why have method specifications?

- Make the behavior of function clear
- Enable user to use function without having to look at the implementation

Unit testing

Verify that method is implemented correctly

Call the method with different inputs and check the results

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

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