

LEC 08

CSE 121

More Methods, Parameters, and Returns

Questions during Class?

Raise hand or send here

sli.do #cse121



BEFORE WE START

*Talk to your neighbors:**What's your favorite book?***Respond on sli.do!**Music: 🍂 [CSE 121 26au Lecture Tunes](#) 🍂**Instructor:** Miya Natsuhara and Anjali Pal

TAs:	Samrutha	Tamsyn	Owen	Jennie
	William	Cedric	Jameson	Anisha
	Samvedha	Anum	Paul	TJ
	Syreeta	Navya	Gurbax	Ivory
	Ryan	Gene	Tony	Sarah W
	Anant	Margot	Lucas	Avery
	Henry	Sarah K	Alex	
	Reese	Ethan	Jesse	
	Savannah	Evelyn	Matthew	

Agenda (1/4)

- **Announcements, Reminders** ←
- Methods, Parameters Warmup
- Returns Review
- Finding Primes Example

Announcements, Reminders

- P0 and CQ0 Grades now visible on Ed + Gradebook
 - See [Ed Announcement](#)
- CQ 1 Today through next Wednesday
 - Sign up for a slot by **11:59pm the night before**
 - There are more than enough slots for everyone, but there won't be if everyone tries to wait until the last day to sign up
- PQ 1 Tuesday, November 3rd
- cse121-instructors@cs.washington.edu if you need to reach Miya and me

Agenda (2/4)

- Announcements, Reminders
- **Methods, Parameters Warmup** ←
- Returns Review
- Finding Primes Example



Practice: Think

sli.do

#cse121

What is the output of this program?

```
public static void main(String[] args) {  
    int x = 1;  
    int y = 2;  
    int z = 5;  
  
    mystery(z, y, x);  
  
    mystery(z + x, y + x, y * 2);  
}  
  
public static void mystery(int x, int z, int y) {  
    z += x;  
    System.out.println(z + " and " + (y - x));  
}
```

A. 7 and -4
7 and -2

B. 6 and 1
7 and 1

C. 5 and -4
9 and -2

D. 7 and -4
9 and -2



Practice: Pair

sli.do

#cse121

What is the output of this program?

```
public static void main(String[] args) {  
    int x = 1;  
    int y = 2;  
    int z = 5;  
  
    mystery(z, y, x);  
  
    mystery(z + x, y + x, y * 2);  
}  
  
public static void mystery(int x, int z, int y) {  
    z += x;  
    System.out.println(z + " and " + (y - x));  
}
```

A. 7 and -4
7 and -2

B. 6 and 1
7 and 1

C. 5 and -4
9 and -2

D. 7 and -4
9 and -2

Agenda (3/4)

- Announcements, Reminders
- Methods, Parameters, Warmup
- **Returns Review** ←
- Finding Primes Example

PCM: Returns

Returns allow us to send values *out of a method*

```
public static <type> myMethod(int num) {  
    System.out.print(num + " is the best!");  
    ...  
    return <value of correct type>  
}
```

return does 3 things:

1. **Evaluate** the expression
2. **Return** this value to where the method was called
3. **Exit** the method

Calling a method that returns a value...

```
<type> result = myMethod(42);
```

Recall: Math

Method	Description
<code>Math.abs(value)</code>	Returns the absolute value of value
<code>Math.ceil(value)</code>	Returns value rounded up
<code>Math.floor(value)</code>	Returns value rounded down
<code>Math.max(value1, value2)</code>	Returns the larger of the two values
<code>Math.min(value1, value2)</code>	Returns the smaller of the two values
<code>Math.round(value)</code>	Returns value rounded to the nearest whole number* note: need to cast result to int (it's complicated!)
<code>Math.sqrt(value)</code>	Returns the square root of value
<code>Math.pow(base, exp)</code>	Returns base raised to the exp power

Recall: String Methods

Method	Description
<code>length()</code>	Returns the length of the string.
<code>charAt(i)</code>	Returns the character at index <code>i</code> of the string
<code>indexOf(s)</code>	Returns the index of the first occurrence of <code>s</code> in the string; returns <code>-1</code> if <code>s</code> doesn't appear in the string
<code>substring(i, j)</code> or <code>substring(i)</code>	Returns the characters in this string from <code>i</code> (inclusive) to <code>j</code> (exclusive); if <code>j</code> is omitted, goes until the end of the string
<code>contains(s)</code>	Returns whether or not the string contains <code>s</code>
<code>equals(s)</code>	Returns whether or not the string is equal to <code>s</code> (case-sensitive)
<code>equalsIgnoreCase(s)</code>	Returns whether or not the string is equal to <code>s</code> ignoring case
<code>toUpperCase()</code>	Returns an uppercase version of the string
<code>toLowerCase()</code>	Returns a lowercase version of the string

Recall: chaining methods in expressions

b	u	b	b	l	e		g	u	m
0	1	2	3	4	5	6	7	8	9

`s.substring(7, 8).toUpperCase() + s.substring(8) + "ball"`

`"g".toUpperCase() + s.substring(8) + "ball"`

`"G" + s.substring(8) + "ball"`

`"G" + "um" + "ball"`

Remember: Method Comments

Each method you write (except main) should have a short comment!

```
// Behavior:  
// - Calculates net profit using monthly income and daily spending.  
// Parameters:  
// - income: user's income this month (non-negative)  
// - spending: amount spent each day this month (non-negative)  
// Returns:  
// - int: the net profit or loss. Positive if profit, negative if loss.  
public static int calculateNetExpenses(int income, int spending) {  
    return income - (spending * DAYS_IN_MONTH);  
}
```



Practice: Think

sli.do

#cse121

What value is returned from this method?

```
public static int returnExample() {  
    for (int i = 0; i < 5; i++) {  
        return i;  
    }  
    return -1;  
}
```

A. -1

B. 0

C. 4

D. 5



Practice: Pair

sli.do

#cse121

What value is returned from this method?

```
public static int returnExample() {  
    for (int i = 0; i < 5; i++) {  
        return i;  
    }  
    return -1;  
}
```

A. -1

B. 0

C. 4

D. 5

Agenda (4/4)

- Announcements, Reminders
- Methods, Parameters Warmup
- Returns Review
- **Finding Primes Example**



Common Problem-Solving Strategies

- **Analogy** – Is this similar to another problem you've seen?
- **Brainstorming** – Consider steps to solve problem before jumping into code
 - Try to do an example "by hand" → outline steps
- **Solve sub-problems** – Is there a smaller part of the problem to solve?
- **Debugging** – Does your solution behave correctly?
 - What is it doing?
 - What do you expect it to do?
 - What area of your code controls that part of the output?
- **Iterative Development** – Can we start by solving a different problem that is easier?