LEC 06

CSE 121

Methods & Parameters

Questions during Class?

Raise hand or send here

sli.do #cse121

BEFORE WE START

Talk to your neighbors:

What's your favorite study spot on campus? Off campus?

Music: CSE 121 25sp Lecture Tunes

Instructor: Miya Natsuhara

TAs:

Chloë	Hibbah	Sushma
Ailsa	Julia	Kelsey
Johnathan	Sahej	Shayna
Christian	Ruslana	Hannah
Merav	Hanna	Zach
Judy	Maitreyi	
Janvi	Ayesha	

Agenda

- **Announcements, Reminders** ⟵
- Practice Problem
- Method, Parameter Review
- Code example

Announcements, Reminders

- Resubmission Cycle 0 (R0) due tomorrow, Thursday April 24
- P1: Election Simulator out today, due Tuesday April 29
- Quiz 0 is on Thursday, April 24 (in your registered quiz section)
  - can't make it? email Miya ASAP
- Reminder: Ed Shortcuts page!

A bit more on Quiz 0

- Taken on paper, in your registered quiz section
  - Quiz logistics in detail
- Broadly: focused on concepts, reading, writing, and debugging code
- Main topics: printing, datatypes, expressions, variables, Strings, for loops
- You get to bring one sheet of notes (8.5x11in, or standard A4)
  - We are also providing a reference sheet on the exam

Resources:

- Practice quizzes
- Review session recording
- Quiz section problems!

A bit more on P1

1. this is a big jump from C1. **Start early!**
2. P1 does *not* require you to use methods (though you can!).
   - advice: feeling shaky on writing methods? don't do it for P1
3. advice: don't put off P1 to study for Quiz 0
   - easy way to fall behind in the class

Agenda

- Announcements, Reminders
- **Practice Problem** ⟵
- Method, Parameter Review
- Code example

Last Time: Random

A `Random` **object** generates *pseudo-random* numbers.

`nextInt(max)` returns a pseudo-random `int` value from `[0, max)`
i.e. between `0` and `max-1`

```
Random rand = new Random();
```

type — name — Random creation code

```
rand.nextInt(6) + 1
```

Last Time: Random

A Random **object** generates pseudo-random numbers.

- the Random **class** is found in the java.util **package**; to use, need import java.util.*;
- we can "seed" the generator to make it behave deterministically

Method	Description
`nextInt()`	Returns a random integer
`nextInt(`***max***`)`	**Returns a random integer in the range [0, *max*), or in other words, 0 to *max*-1 inclusive**
`nextDouble()`	Returns a random double in the range [0.0, 1.0)

Last Time: Math

Calling:

`Math.<method>(…)`

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

Agenda

- Announcements, Reminders
- Practice Problem
- **Method, Parameter Review** ⟵
- Code example

PCM: Methods

Writing our own **methods** allows us to define our own commands!

(naming conventions for methods are same as variables: `camelCased`)

```
public static void myMethod() {
  /***
  Your code here
  **/
}
```

Practice: Think

sli.do #cse121

```
public class HelloGoodbye {
    public static void main(String[] args) {
        welcome();
        hello();
        goodbye();
    }

    public static void hello() {
        System.out.print("Hello! ");
        glad();
    }

    public static void goodbye() {
        System.out.println("Goodbye!");
    }

    public static void welcome() {
        System.out.print("Welcome! ");
        glad();
    }

    public static void glad() {
        System.out.println("Glad you're here.");
    }
}
```

What is the output of this program?

A.
```
Welcome! Glad you're here.
Hello! Glad you're here.
Goodbye!
```

B.
```
Welcome!
Hello!
Goodbye!
```

C.
```
Welcome! Hello! Goodbye!
```

D.
```
Welcome!
Glad you're here.
Hello!
Glad you're here.
Goodbye!
```

Practice: Pair

sli.do #cse121

```
public class HelloGoodbye {
    public static void main(String[] args) {
        welcome();
        hello();
        goodbye();
    }

    public static void hello() {
        System.out.print("Hello! ");
        glad();
    }

    public static void goodbye() {
        System.out.println("Goodbye!");
    }

    public static void welcome() {
        System.out.print("Welcome! ");
        glad();
    }

    public static void glad() {
        System.out.println("Glad you're here.");
    }
}
```

What is the output of this program?

A.
```
Welcome! Glad you're here.
Hello! Glad you're here.
Goodbye!
```

B.
```
Welcome!
Hello!
Goodbye!
```

C.
```
Welcome! Hello! Goodbye!
```

D.
```
Welcome!
Glad you're here.
Hello!
Glad you're here.
Goodbye!
```

PCM: Parameters

Definition: a value passed to a method by its caller. “Like” a variable!

```
public static void myMethod(String musicalAct) {
    System.out.print(musicalAct + " is the best!");
    ...
}
```

Calling a method with a parameter…

```
myMethod("Laufey"); // prints: Laufey is the best!
```

PCM: Scope, Redux

Our scope rules also apply to methods and parameters!

- General rule: from its **declaration to the next closing brace, }**
- a variable declared in a method only exists <u>in that method</u>!

```
public static void example(int n) {
    System.out.println("hello");
    int x = 3;
    for (int i = 1; i <= n; i++) {
        System.out.print(x);
    }
}
```

i's scope

x's scope

n's scope

Agenda

- Announcements, Reminders
- Practice Problem
- Method, Parameter Review
- **Code example** ⟵

New: Method Comments

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

```
// Randomly generates an addition problem where the
// operands are in the range 1-10 (inclusive),
// and prints the result, rounded to two decimal places.
public static void addTwoRandomNumbers() {
    Random randy = new Random();
    int num1 = randy.nextInt(10) + 1;
    int num2 = randy.nextInt(10) + 1;
    int sum = num1 + num2;
    ...
}
```

New: Class Constants

A fixed value visible (in-scope) to the whole program (the entire ***class***).

Value is set at declaration, **cannot** be reassigned – value is ***constant***.

```
public static final type NAME_OF_CONSTANT = expression;
```