Welcome to CSE 121!

Simon Wu

Summer 2024

Use this QR code as one way to ask questions!

sli.do #cse121

TAs:	Trey	Hannah	Mia	Vivian	Jolie	Colton	Ziao

Announcements & Reminders

- First Quiz tomorrow in your sections! See Ed reminder.
- C1 also due tomorrow, 7/10 at 11:59 pm
- Resubmission Cycle 1 due tomorrow at 11:59 pm
- P1 will be released on Friday, due next **Thursday** (7/17)
  - note: **a big jump** from C1. **<u>start early!</u>**
  - watch out for code quality – we were a little more lenient with CQ grading for the first two assignments, but will be less lenient moving forward

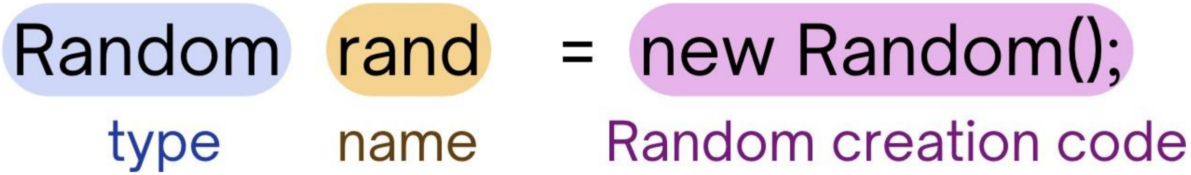
Last Time 1

- Nested for loops
  - Syntax & conventions: (i ,   j ,   k)
  - Applications: “doing the same thing for multiple iterations”

```
for (int outerLoop = 1; outerLoop <= 5; outerLoop++) {
    System.out.println("outer loop iteration #" + outerLoop);
    for (int innerLoop = 1; innerLoop <= 7; innerLoop++) {
        System.out.println("    inner loop iteration #" + innerLoop);
    }
    System.out.println(outerLoop);
}
```

Last Time 2

- Random
  - A Random object generates *pseudo*-random numbers
  - `nextInt(max)` returns random `int` value [0, max) i.e. between 0 and max-1



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

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

(PCM) Methods

Writing our own **methods** allow us to define our own statements / commands in Java!

- Naming conventions for methods are the same as variables: `camelCased`

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

Poll in with your answer!

What is the output of this program?

sli.do #cse121-6

```java
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.");
    }
}
```

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

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

Calling a method with a parameter…

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

(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>
}
```

Evaluates the expression

Returns this value to where the method is called from

Method immediately exits

Calling a method that returns a value…

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

(Recall) String Methods

Usage: `<string variable>.<method>(…)`

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

String example

```
String s = "bubblegum";
s = s.substring(7, 8).toUpperCase() + s.substring(8) + "ball";
```

Example of returns: Math class

Methods	Returns
`Math.abs(`*`value`*`)`	Absolute value of *value*
`Math.ceil(`*`value`*`)`	*value* rounded up
`Math.floor(`*`value`*`)`	*value* rounded down
`Math.max(`*`value1, value2`*`)`	Larger of the two given values
`Math.min(`*`value1, value2`*`)`	Smaller of the two given values
`Math.round(`*`value`*`)`	*value* rounded to the nearest whole number
`Math.sqrt(`*`value`*`)`	Square root of *value*
`Math.pow(`*`base, exp`*`)`	*base* to the *exp* power

Math example

```
double value = 823.577564893;
double roundedValue = (double) Math.round(value * 100) / 100;
```