Welcome to CS 61BL!

Quote of the week: “Speak casually, but never think casually.”
Let’s introduce the staff

- Your TAs! Say hello to them if you see them around!
Let’s introduce the staff

- Me...
- NOT a professor. A Berkeley alumnus.
- Call me Joseph, or Joe, or Joey...
- Technical interests: NLP, machine learning, AI
- Office hours: M 4-6, Tu 2-4, W 12-2 in 329 Soda
What is this class?

- A lab-based course
- The sequel to CS 61A
- A class about **data structures** and **programming methodology**
A lab-based course

- Learn by doing!
- Lecture isn’t very useful...
- Collaboration over competition
The sequel to CS 61A

- 61A (or some equivalent) is a *required prereq*

- Not just how to program, but how to program *well*

- Expect more work than CS 61A

- Homework now graded on correctness. But no more autograders...

- For other course policies (grading, etc.) please see the course webpage
What you know about Java

- All code appears in a class

```java
public class Counter {
    // Code goes here
}
```
Here, we define a method (function) inside the class

```java
public class Counter {
    public void printOne() {
        System.out.println(1);
    }
}
```
What you know about Java

- The equivalent Python. Talk to your partner. What does this code do if we were to run it?

```python
class Counter:
    def print_one(self):
        print(1)
```

```java
public class Counter {
    public void printOne() {
        System.out.println(1);
    }
}
```
What you know about Java

- Code outside a definition tells Python to actually do something

```python
class Counter:
    def print_one(self):
        print(1)

c = Counter()
c.print_one()
```
What you know about Java

- `main` tells Java to actually do something

```java
class Counter:
    def print_one(self):
        print(1)

c = Counter()
c.print_one()
```

```python
def print_one(self):
    print(1)

c = Counter()
c.print_one()
```

```java
public class Counter {
    public void printOne() {
        System.out.println(1);
    }

    public static void main(String[] args) {
        Counter c = new Counter();
        c.printOne();
    }
}
```
To understand our Java programs, it will be helpful to draw them
The code we’ll be drawing

public class Counter {

    public void printNumber(int num) {
        int x = 3;
        System.out.println(num);
        num = 10;
    }

    public static void main(String[] args) {
        int x = 5;
        Counter c = new Counter();
        c.printNumber(x);
    }
}
main
public class Counter {

    public void printNumber(int num) {
        int x = 3;
        System.out.println(num);
        num = 10;
    }

    public static void main(String[] args) {
        int x = 5;
        Counter c = new Counter();
        c.printNumber(x);
    }
}
public class Counter {

    public void printNumber(int num) {
        int x = 3;
        System.out.println(num);
        num = 10;
    }

    public static void main(String[] args) {
        int x = 5;
        Counter c = new Counter();
        c.println(num); // This line is incorrect.
    }
}
main

int x

5

Counter c
public class Counter {

    public void printNumber(int num) {
        int x = 3;
        System.out.println(num);
        num = 10;
    }

    public static void main(String[] args) {
        int x = 5;
        Counter c;
        new Counter();
        c = new Counter();
        c.printNumber(x);
    }
}
main

int x

5

Counter c
public class Counter {

    public void printNumber(int num) {
        int x = 3;
        System.out.println(num);
        num = 10;
    }

    public static void main(String[] args) {
        int x = 5;
        Counter c;
        new Counter();
        c = new Counter();
        c.printNumber(x);
    }
}
```java
main

int x

5

Counter c

Counter
```
public class Counter {

    public void printNumber(int num) {
        int x = 3;
        System.out.println(num);
        num = 10;
    }

    public static void main(String[] args) {
        int x = 5;
        Counter c;
        new Counter();
        c = new Counter();
        c.printNumber(x);
    }
}
Primitive

main

int x

5

Counter c

Reference

Object

Counter
public class Counter {

    public void printNumber(int num) {
        int x = 3;
        System.out.println(num);
        num = 10;
    }

    public static void main(String[] args) {
        int x = 5;
        Counter c = new Counter();
        c.printNumber(x);
    }
}
public class Counter {

    public void printNumber(int num) {
        int x = 3;
        System.out.println(num);
        System.out.println(this);
        num = 10;
    }

    public static void main(String[] args) {
        int x = 5;
        c.printNumber(x);
    }
}
The equivalent Python has `self` to refer to the object that called the method. Does Java have such a thing?

class Counter:
    def print_number(self, num):
        x = 3
        print(num)
        num = 10

x = 5
c = Counter()
c.print_number(x)
And now...

- A break.
- It’s a long two hours.
Objects can have variables inside

```java
public class Counter {
    int myValue = 0;

    public void increment(int x) {
        this.myValue += x;
    }

    public static void main(String[] args) {
        int x = 5;
        Counter c = new Counter();
        c.increment(x);
        System.out.println(c.myValue);
    }
}
```
- Objects can have variables inside

```java
public class Counter {
    int myValue = 0;

    public void increment(int x) {
        this.myValue += x;
    }

    public static void main(String[] args) {
        int x = 5;
        Counter c = new Counter();
        c.increment(x);
        System.out.println(c.myValue);
    }
}
```
Objects can have variables inside

```java
public class Counter {
    int myValue = 0;

    public void increment(int x) {
        this.myValue += x;
    }

    public static void main(String[] args) {
        int x = 5;
        Counter c = new Counter();
        c.increment(x);
        System.out.println(c.myValue);
    }
}
```
Objects can have references to other objects inside

```java
public class Counter {
    int myValue = 0;
    Counter other;

    public static void main(String[] args) {
        Counter c = new Counter();
        c.other = new Counter();
        c.other.myValue = 1;
    }
}
```
The Stack
(frames stacked on top of each other)

The Heap
(objects messily connected together)

null

Counter

Counter other

int myValue

1

0

Counter

Counter other

int myValue
Often we abbreviate diagrams
• **Stack and heap diagrams...**

  • Are not literally how your computer’s memory works

  • They are useful models for understanding what your program does

• In this class, we are mainly concerned with the heap (leave stack frame craziness for 61A!)
• Additional points...

  • References do not point to other references, only to objects

  • Objects do not contain other objects, only primitives and references

  • A new object is created *only* if there is a call to `new`
Quiz time!

(More quizzes, even in lecture?!)  

Be chill. It’s worth 1 ec point. And you can work with your partner.

One sometime in every lecture...
Your turn!

- Draw everything by the end of the `main` method.

```java
public class Thing {
    int myValue;
    Thing myThang;

    public static void main(String[] args) {
        Thing firstThing = new Thing();
        int num = 2;
        Thing currentThing = firstThing;
        currentThing.myValue = num;
        while (num > 0) {
            num--;
            currentThing.myThang = new Thing();
            currentThing.myThang.myValue = num;
            currentThing = currentThing.myThang;
        }
    }
}
```
public class Thing {
    int myValue;
    Thing myThang;

    public static void main(String[] args) {
        Thing firstThing = new Thing();
        int num = 2;
        Thing currentThing = firstThing;
        currentThing.myValue = num;
        while (num > 0) {
            num--;
            currentThing.myThang = new Thing();
            currentThing.myThang.myValue = num;
            currentThing = currentThing.myThang;
        }
    }
}
Now that we have primitives, objects, and references, we have almost all of Java.

The next major piece is the array.
Arrays and lists

- You may remember the list from Python

- An array is like a list, but more limited
  - It can only store objects of one type!
  - It is a fixed size.
Declaring an array

- Declare an int variable like so...

  ```java
  int x = 3;
  ```

- Declare an array of ints like so...
  
  `[]` tells you it’s an array type...

  ```java
  int[] arr = new int[4];
  ```

- You can put things in it like so:

  ```java
  arr[2] = 10;  
  ```

  The third thing in `arr` is now 10
An array itself is an object, so it has a reference to it:

`int[] arr = new int[4];`
We can change things in the array

```java
int[] arr = new int[4];
arr[3] = 7;
```
An array of objects starts out full of null

```java
Thing[] things = new Thing[2];
```
Thing[] things

null

null
When we put objects inside, we just get references to the object

```java
Thing[] things = new Thing[2];
things[0] = new Thing();
```
Let’s talk about the cheating policy.

(Sorry)
What constitutes cheating?

- For **in-lab quizzes** and **exams**, the normal policy: you’re totally on your own.

- For **labs** and **group projects**, you can share *everything* within your partnership/group.

- For **labs**, high-level collaboration is allowed across-partnership during lab time only. Can discuss ideas, but no direct sharing of code.

- For **group projects**, essentially no collaboration is allowed outside your group.
What constitutes cheating?

- Do **NOT** host your code publicly online (such as on Github — use BitBucket if you don’t have a private)

- Don’t look up answers to lab exercises online, but you can look up general how-to Java (in fact, this is encouraged)

- If you get ideas from another partnership, or if you take significant code from online, *please provide a citation as a comment.*
The exam times are

- Friday, 10 July, 7-9 pm
- Friday, 31 July, 7-9 pm
- Friday, 14 August 3-6 pm

If you have conflicts, please email me ASAP. Before the end of this week. If you don’t I cannot guarantee you a make-up.

Please provide a reason, and exactly what time it takes up
Introducing the first project! (Released Monday)