Announcements

Midterm info:

- Midterm info, including what room you should go to, is available here: https://piazza.com/class/j9j0udrxjjp758?cid=1105.
- Exam is 8 - 10 PM, tonight.
- Use Piazza for questions, or email cs61b@berkeley.edu if you don't get a response fast enough.
  - Please email us ASAP if you are expected alternative accommodations of any kind and haven't heard back from us.
- Midterm covers everything up to and including 2/7 (subtype polymorphism and HoFs).
- Reminder: Midterm will be pretty hard (sry).

CS61B

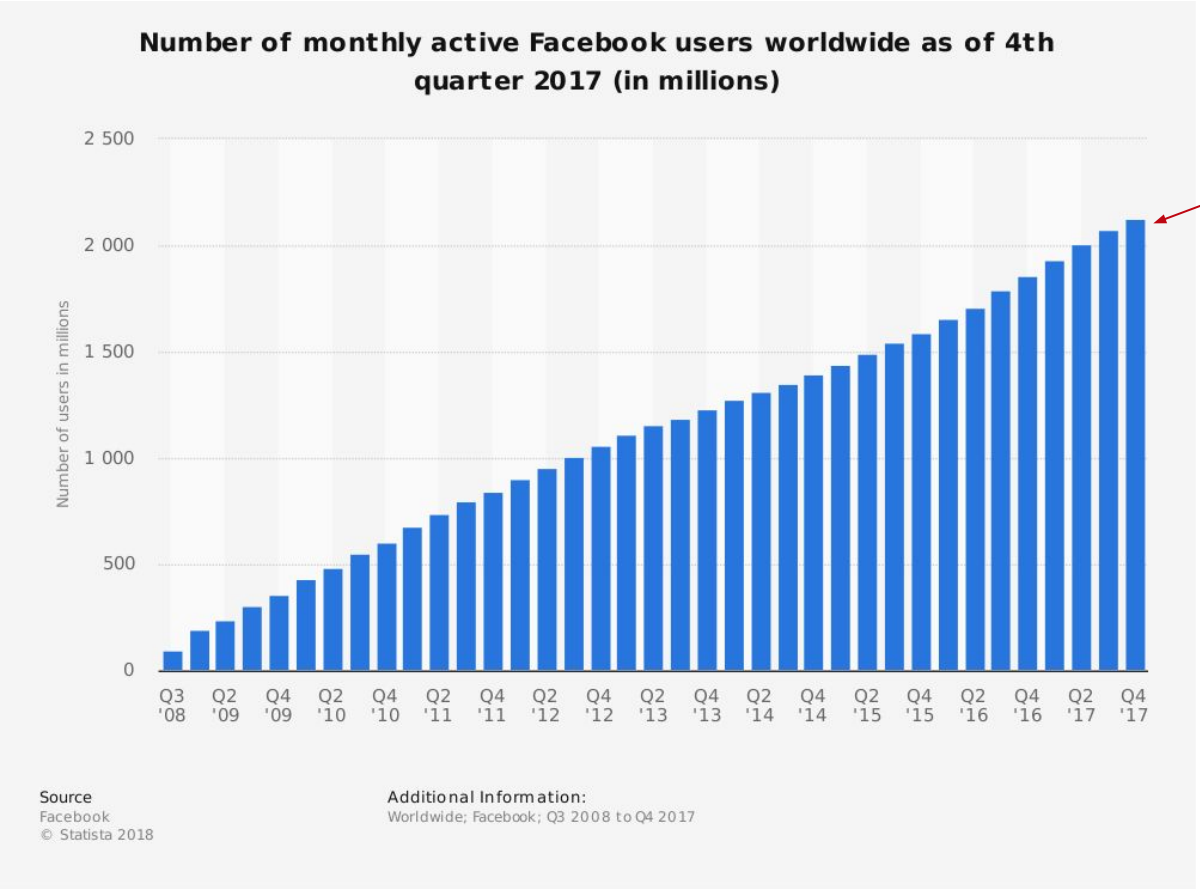
Lecture 12: Libraries

- Programming in the Real World
- Review

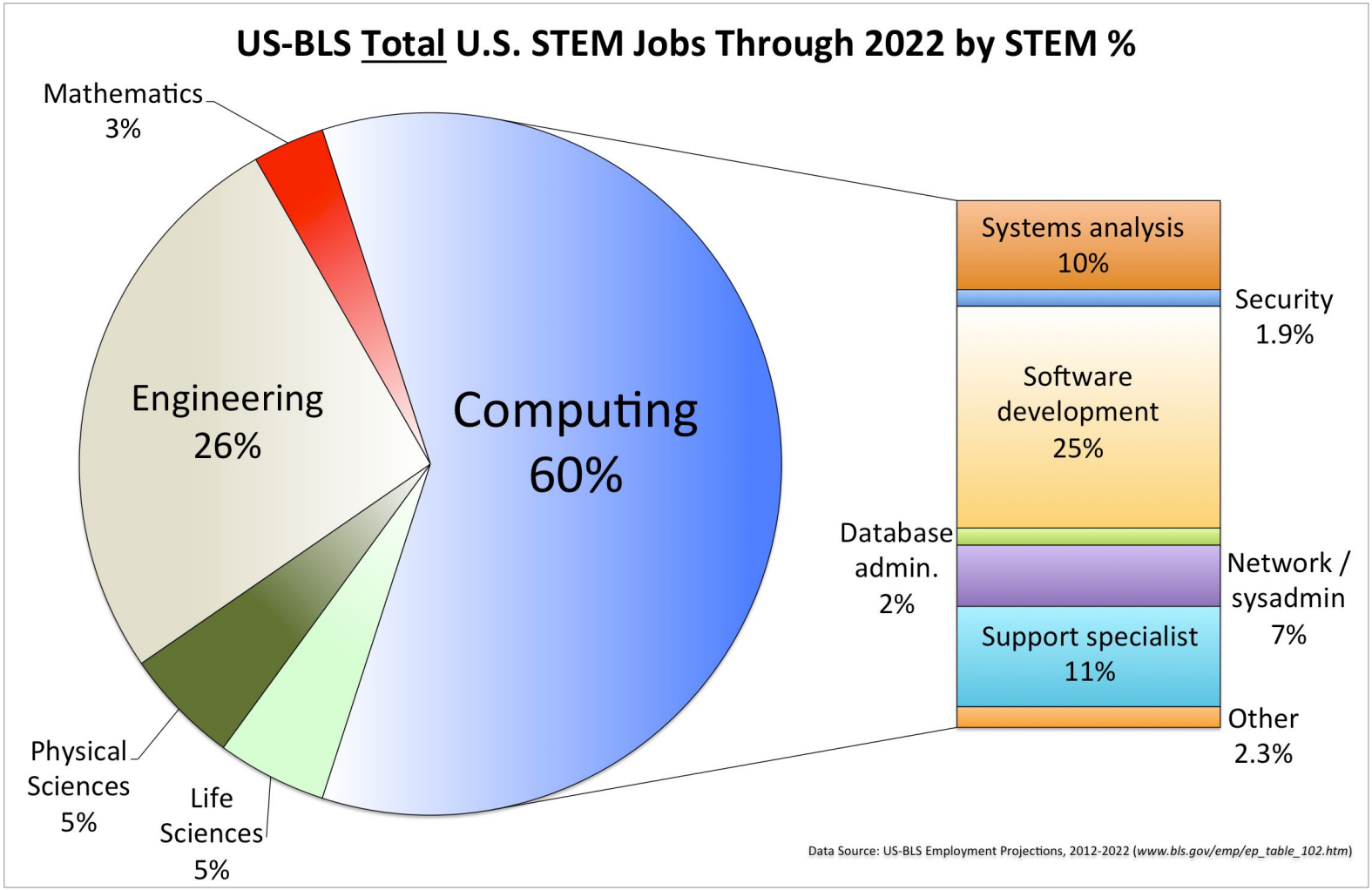
Programming in the Real World

Tesla Assembly Line (short clip)

Facebook Monthly Active Users



Over 2 billion monthly active users.


US-BLS Total U.S. STEM Jobs Through 2022 by STEM %
Mathematics 3%
Engineering 26%
Computing 60%
Physical Sciences 5%
Life Sciences 5%
Systems analysis 10%
Security 1.9%
Software development 25%
Database admin. 2%
Network / sysadmin 7%
Support specialist 11%
Other 2.3%
Data Source: US-BLS Employment Projections, 2012-2022 (www.bls.gov/emp/ep_table_102.htm)

Current Economic Upheaval

From an April 2015 McKinsey Global Institute report: Four trends are leading to a dramatic upheaval in society:

- Urbanization
- Technological Change
- Aging Global Populace
- Shift from North American and European Trading Hubs

“Compared with the Industrial Revolution, we estimate that this change is happening ten times faster and at 300 times the scale.” [Link]

And at the Middle… You

Even if you are not a CS major, I believe this course is a major step on transforming you into someone who will be driving this change.

MIT Technology Review's "Biggest Tech Failures of 2016/2017"

Volkswagen's "defeat device": Volkswagen created what may be the most expensive few lines of code ever written. The software, called a "defeat device" and cooked up by the German automaker to fool U.S. emissions testers, concealed that its diesel cars emit as much as 40 times the allowed pollution [cost to VW: $14,700,000,000].

Facebook's News Feed: "Pope Francis Shocks World, Endorses Donald Trump for President, Releases Statement." That never happened. Yet this headline was among a tsunami of falsehoods spread on Facebook, the 1.7 billion-member social network [now 2+ billion].

EOS Tokens: By December it had raised $700 million by selling... EOS tokens, to investors in what's called an initial coin offering. An ICO is [an] alternative to a stock market debut... Often, the promise is that buyers of the virtual coins can redeem them for a company's product. But EOS tokens give the buyer the right to nothing specific. The coins, the company says, have no "uses, purpose, attributes, functionalities, or features."

Pre-Midterm Q&A

Pre-Midterm Q&A

I'll take whatever questions you have and answer them cold, so I can show not just how to do it, but how I personally approach the problem.

- Give an example of when you get runtime errors with inheritance.
  - NullPointerExceptions, IndexOutOfBounds???
  - ClassCastException
  - `Animal a = (Dog) new Animal();  //runtime error`
  - Would this cause a compile error if:
    - Dog a = (Dog) new Animal(); // no compile error! Yes, runtime error!
  - Another example:
    - Someone ahs defined: public List getList()
    - We use it by sayign: ArrayList a = (ArrayList) getList();
      - Compile error? No.
      - Runtime error? Maybe. Depends on what kind of getList makes.

Pre-Midterm Q&A

Comparing strings for equality using == vs .equals:

- Why does == seem to work sometimes correctly but not others.
- String a = “horse”;
- String b = “horse”;
- Boils down to whether two separate ACTUAL strings in memory are created, or just one, and they point at the same object.
  - The rule for when “string interning” occurs is not in my brain.
- Can this happen with int?
  - No, all ints are the same
- Can this happen with Integer?
  - Probably. Pretty sure. Use .equals!
- This is similar to the issue we ahd in lab 3, and the issue that was in the “correct” solution under test for proj1gold.
- Are the 8 primitives immune to interning black magicery? Yes! Ok to use ==.

Pre-Midterm Q&A

Past example exam: compared equality of two objects. There was a risk that they were both null, in which case, using .equals would be bad.

- Because of nullopintexception

Meta-question on exam: can we assume that objects in a list are not `null`. No, but we will often say “it is ok to assume they are not null”.

Pre-Midterm Q&A

Let's talk about non-destructive methods which involve change or creation, but that don't use new.

- Let's try to come up with a problem. incrList from lab 2:

```
/* Non-destructively increment the current list by x, don't use new. */
public IntList incrList(int x) {
}
```

No this isn't possible.

- **Observation: Can technically create arrays without new, e.g. int[] a = {1, 3, 4};**
-

Pre-Midterm Q&A

Difference between:

- Integer a = 5;
- int a = 5;
- This will be in a lecture about autoboxing/unboxing. Don’t worry about it for now.
- Observation: There was an old exam that did new Integer(5).
  - The Integer class has a single instance variable that holds an int.
  -

Pre-Midterm Q&A

Do we need to know bit operations? No.

What can static methods access and not access? For a class X:

- Static methods of X can access other static members of X.
- Static methods of X cannot access instance members of “the current X”, i.e. you can’t do this.value, and can’t do this this.addFirst().
- Static methods CAN instantiate Xs.

When you declare a instance variable and don’t specify public or private, what is it?

- “Package Private” and we’ll discuss next week I think. Handy for proj2.
-

Pre-Midterm Q&A

```
Class Dog extends Animal {
}

Dog d = new Dog();
((Animal) d).makeNoise(d);
```

- Does the type-casting affect both ds, or just the left one.
  - Just the left one!
- What class do we check at compile time for a method?
  - Animal: Does it have a makeNoise method that accepts Dogs?
  - Dynamic type is unchanged in any way by casting always, forever.
- When passing a

Pre-Midterm Q&A

```
Class Dog extends Animal {
}
Dog d = new Dog();
((Animal) d).makeNoise(d);
```

- Does the type-casting affect both ds, or just the left one.
  - Just the left one!
- What class do we check at compile time for a method?
  - Animal: Does it have a makeNoise method that accepts Dogs?
  - Dynamic type is unchanged in any way by casting always, forever.
- ((Animal) d).makeNoise(d);: When checking to see if a method exists in Animal, do we check for a makeNoise(Dog) method or a makeNoise(Animal) method? Yes. In particular, the compiler sees if ANY method exists that accepts Dog. If it happens to have both, then it'll use makeNoise(Dog) because this is the most specific version. THIS IS TOTALLY DISTINCT FROM DYNAMIC METHOD SELECTION. Has nothing to do with

Pre-Midterm Q&A

```
Class Dog extends Animal {
}
Dog d = new Dog();
((Animal) d).makeNoise(d);
```

- ((Animal) d).makeNoise(d);: When checking to see if a method exists in Animal, do we check for a makeNoise(Dog) method or a makeNoise(Animal) method? Yes. In particular, the compiler sees if ANY method exists that accepts Dog. If it happens to have both, then it'll use makeNoise(Dog) because this is the most specific version. THIS IS TOTALLY DISTINCT FROM DYNAMIC METHOD SELECTION. Has nothing to do with overriding.
- If animal has only makeNoise(ANimal), and Dog has both makeNoise(Animal) and makeNoise(Dog) which gets called?
  - **The compiler checks, see that makeNoise(Animal) exists, and RECORDS THIS AS ****THE**** method signature.**

Pre-Midterm Q&A

In a subclass, the no-argument super class constructor is automatically called if you don't explicitly call it, so why bother using super class constructor explicitly?

- If you want to use a different super class constructor, e.g. super(int a, int b)
- What if the super class doesn't have any construcotr?
  - IMPOSSIBLE. If you don't give a constructor, you get a no-argument constructor for free.
  - **HOWEVER: If you create a one or more argument constructor, your free no-argument constructor is not given.**
  - Sorry. I don't know why they did this. It seems like a dumb thing to memorize.
- If super class doesn't have any no-argument constructor, then compile error result.

Pre-Midterm Q&A

Class with two variables:

```
public class Dog {
    int size;
    Static String scientificName = “canis familiaris”;
}

Public class Poodle extend Dog {
        Poodle() {
           this.scientificName = “Poodle familiaris”;
        }
}
```

Pre-Midterm Q&A

Class with two variables:

```
public class Dog {
    int size;
    String scientificName = “canis familiaris”;
}

Public class Poodle {
    Sttring scientifcName // this is “HIDING”
}
```

Pre-Midterm Q&A

How does nested class access work?

```
Public class Outer {
   Public int outvar;
    Public class Inner {
        public int invar;
    }
}
```

In the example above: Anybody can create an Outer, anybody can create an Inner, and anybody can access an Inner's var variabel. And an Inner can access this.outvar.

Pre-Midterm Q&A

How does nested class access work?

```
Public class Outer {
   Public int outvar;
    private class Inner {
        public int invar;
    }
}
```

In the example above, Inner is private. At this point, the access modifiers to all of inner's members are irrelevant.

- An Outer **CAN access** an Inner's variables. The reason is that "Inner" is a subordinate slave of Outer, and thus has no personal possessions.
- We can change the italicized public to anything and have no observable

Pre-Midterm Q&A

How does nested class access work?

```
Public class Outer {
   private int outvar;
    private class Inner {
        public int invar;
    }
}
```

In the example above, can an Inner access an Outer's outvar.

- I think so? [but i might be wrong] [95% correctness guess]

Pre-Midterm Q&A

How does nested class access work?

```
Public class Outer {
   private int outvar;
    private static class Inner {
        public int invar;
    }
}
```

What could I do so that an Inner can't use outvar?

- Make Inner static: I cannot access my enclosing instance's outvar.
-

Pre-Midterm Q&A

How does nested class access work?

```
Public class Outer {
   private int outvar;
    private class Inner {
        public int invar;
    }
}
```

Can you instantiate a non-static inner class without an outer instance?

- Inner inner = (new Outer).new Inner(); [??? maybe]

Pre-Midterm Q&A

Default methods?

- Interfaces can provide implementation inheritances (oh no) to their subclasses with default methods.
- They can be overridden just like regular methods.
- The quirky thing about them is that they are written as part of the INTERFACE not as part of a specific implementation.
- In default, you can certainly use new

Public interface List61B

```
Default public void doAThing() {
  List61B S = new SLList(); //OK EVEN IF SLList extends List61B
```

can we use this in default method? Sure!

Pre-Midterm Q&A

Why can't we reassign this?

```
Public class Dog {
  Public void f() {
      This = new Dog(); }
}
```

```
Dog d = new Dog();
d.f();
```