HW0

(For webcast viewers): This lecture assumes you have completed homework 0. If you have not done so yet, complete HW0 and then come back.

Further Notes for Webcast Viewers

- Any time I'm live coding, I advise you to pause frequently and try to anticipate my next move. You'll probably learn more by trying to guess what I'm going to do rather than just watching me do it.

Announcements

- Project 0 is out! Due next Friday, 1/26 at 11:59 PM.
  - May work alone or in pairs.
  - Partners must have the same Java course background (either: both have taken a Java course, or neither have taken a Java course).
  - Partners must work in the same physical room at all times when working on Project 0.
  - Partners should NOT divide up the work. This is counterproductive. Project is plenty small enough to do solo.
  - We encourage you to “pair program”: One driver, one navigator.
  - See partnership guide for more.

Announcements 2

Piazza tips:

- Make sure to search for an answer before posting.
- Instructor answers are intentionally rate limited. We don't want you guys to get reliant on us for answers, and want you to talk to each other.

- **Make sure your question has enough information for someone to help.**
  - Good question: https://imgur.com/a/6wUIR
    - Screenshots! Examples of what the anonymous poster has tried. And a followup explaining the resolution for other students.
- Starting with project 0: If there's a chance a staff member might need to look at your code, make sure your most recent code is pushed to github and provide a link in your post.

Announcements 3

Example of a bad question.

- Doesn’t specify when the error is happening or what it is.
- No discussion of what the student has already tried.

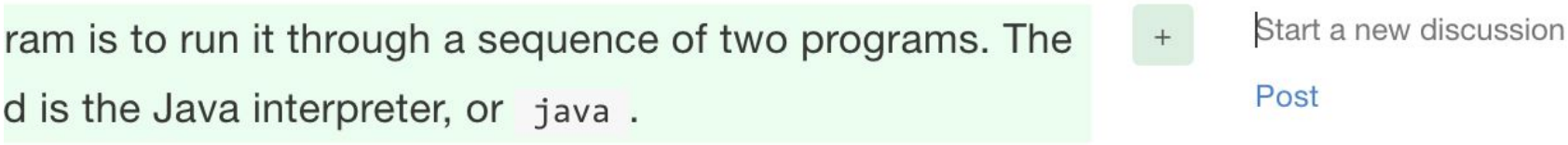
Testing Error?

In my JUnit testing, when I'm creating my own test and asserting, I write:
assertEquals(x, numYears(/*Random targetyear*/));

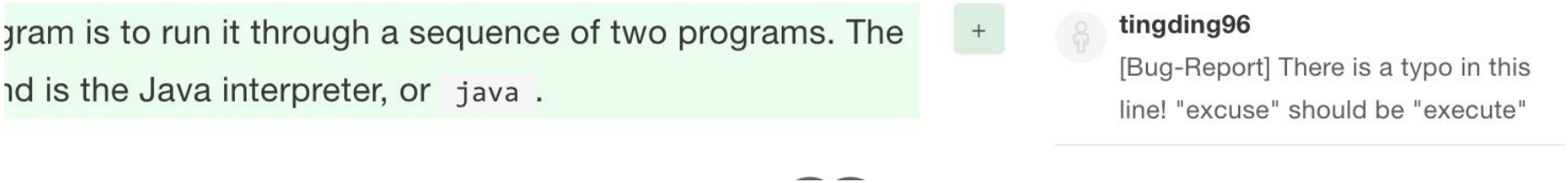
I keep getting an error, what am i doing wrong?

Announcements 4

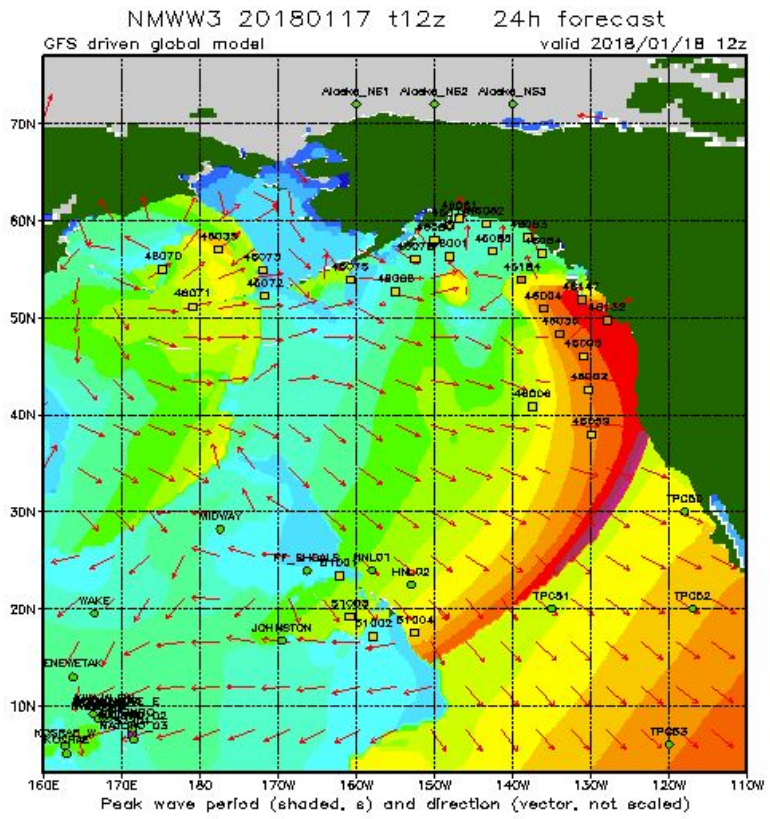
If you spot any typos or errors in the online textbook (of which there are surely many), you can comment directly on the book using the “Start a New Discussion” button:



It’s especially helpful if you include “[Bug-Report]” in your comment, e.g.:



If forecast > 18 feet, go! http://goo.gl/GzNrrv



CS61B: 2018

Using and Defining Classes

- Compilation
- Defining and Instantiating Classes
- A Closer Look at Static
- public static void main(String[] args)
- Using Libraries (e.g. StdDraw, In)

Best place to watch waves: http://goo.gl/tlDSVa

THERE IS A LIVE PIAZZA THREAD IF YOU HAVE ANY QUESTIONS MID-LECTURE. Someone will answer probably.

Compilation

Compilation

The standard tools for executing Java programs use a two step process:

- This is not the only way to run Java code.

Why make a class file at all?

- .class file has been type checked. Distributed code is safer.
- .class files are 'simpler' for machine to execute. Distributed code is faster.
- Minor benefit: Protects your intellectual property. No need to give out source.

You can learn more about all this in 61C and particularly 164.

Note: .class files are easily reversible into similar looking Java files.

Defining and Instantiating Classes

Dog

As we saw last time:

- Every method (a.k.a. function) is associated with some class.
- To run a class, we must define a main method.
  - Not all classes have a main method!

```
public class Dog {
    public static void makeNoise() {
        System.out.println("Bark!");
    }
}
```

Can't be run directly, since there is no main method.

```
public class DogLauncher {
    public static void main(String[] args) {
        Dog.makeNoise();
    }
}
```

Calls a method from another class. Can think of this as a class that tests out the Dog class.

Object Instantiation

Not all dogs are equal!

A not so good Approach

We could create a separate class for every single dog out there, but this is going to get redundant in a hurry.

```
public class MayaTheDog {
    public static void makeNoise() {
        System.out.println("arooooooooooooooo!");
    }
}
```

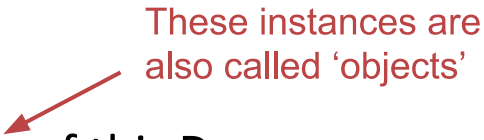
```
public class YapsterTheDog {
    public static void makeNoise() {
        System.out.println("awawawwwawwa awawaw");
    }
}
```

Object Instantiation

Classes can contain not just functions (a.k.a. methods), but also data.

- For example, we might add a `size` variable to each Dog.

Classes can be instantiated as objects.



- We’ll create a single Dog class, and then create instances of this Dog.
- The class provides a blueprint that all Dog objects will follow.

Side note: For E7/MATLAB folks, if you’ve ever gotten an axis using gca(), this is similar. Each axis has the same properties, e.g. they all have xTicks, etc.

Defining a Typical Class (Terminology)

```
public class Dog {
    public int weightInPounds;

    public Dog(int startingWeight) {
        weightInPounds = startingWeight;
    }

    public void makeNoise() {
        if (weightInPounds < 10) {
            System.out.println("yipyipyip!");
        } else if (weightInPounds < 30) {
            System.out.println("bark. bark.");
        } else {
            System.out.println("woof!");
        }
    }
}
```

Instance variable. Can have as many of these as you want.

Constructor (similar to a method, but not a method). Determines how to instantiate the class.

Non-static method, a.k.a. Instance Method. Idea: If the method is going to be invoked by an instance of the class (as in the next slide), then it should be non-static.

Roughly speaking: If the method needs to use "**my** instance variables", the method must be non-static.

Instantiating a Class and Terminology

```
public class DogLauncher {
    public static void main(String[] args) {
        Dog smallDog;
        new Dog(20);
        smallDog = new Dog(5);
        Dog hugeDog = new Dog(150);
        smallDog.makeNoise();
        hugeDog.makeNoise();
    }
}
```

Declaration of a Dog variable.

Instantiation of the Dog class as a Dog Object.

Instantiation and **Assignment.**

Declaration, Instantiation and **Assignment.**

Invocation of the 150 lb Dog's makeNoise method.

The dot notation means that we want to use a method or variable belonging to hugeDog, or more succinctly, a ***member*** of hugeDog.

Arrays of Objects (assuming you've done HW0!)

To create an array of objects:

- First use the `new` keyword to create the array.
- Then use `new` again for each object that you want to put in the array.

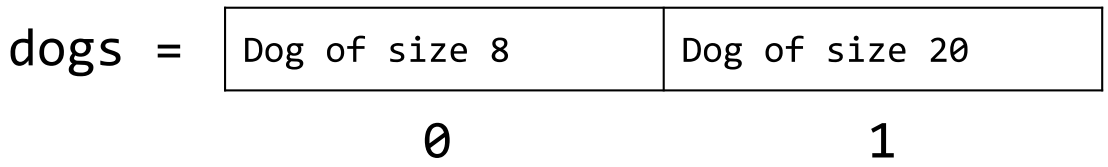
Example:

```
Dog[] dogs = new Dog[2];
dogs[0] = new Dog(8);
dogs[1] = new Dog(20);
dogs[0].makeNoise();
```

Creates an array of Dogs of size 2.

Yipping occurs.

After code runs:



Static vs. Instance Members

Static vs. Non-static

Key differences between static and non-static (a.k.a. instance) methods:

- Static methods are invoked using the class name, e.g. Dog.makeNoise();
- Instance methods are invoked using an instance name, e.g. maya.makeNoise();
- Static methods can't access "my" instance variables, because there is no "me".

Static

```
public static void makeNoise() {
    System.out.println("Bark!");
}
```

This method cannot access weightInPounds!

Invocation:

```
Dog.makeNoise();
```

Non-static

```
public void makeNoise() {
    if (weightInPounds < 10) {
        System.out.println("yipyipyip!");
    } else if (weightInPounds < 30) {
        System.out.println("bark. bark.");
    } else { System.out.println("woof!"); }
}
```

Invocation:

```
maya = new Dog(100);
maya.makeNoise();
```

Why Static Methods?

Some classes are never instantiated. For example, Math.

- `x = Math.round(5.6);`

Much nicer than:

```
Math m = new Math();
x = m.round(x);
```

Sometimes, classes may have a mix of static and non-static methods, e.g.

```
public static Dog maxDog(Dog d1, Dog d2) {
    if (d1.weightInPounds > d2.weightInPounds) {
        return d1;
    }
    return d2;
}
```

Static vs. Non-static

A class may have a mix of static and non-static ***members***.

- A variable or method defined in a class is also called a member of that class.
- Static members are accessed using class name, e.g. Dog.binomen.
- Non-static members **cannot** be invoked using class name: ~~Dog.makeNoise()~~
- Static methods must access instance variables via a specific instance, e.g. d1.

```
public class Dog {
    public int weightInPounds;
    public static String binomen = "Canis familiaris";

    public Dog(int startingWeight) {
        weightInPounds = startingWeight;
    }

    public static Dog maxDog(Dog d1, Dog d2) {
        if (d1.weightInPounds > d2.weightInPounds)
            return d1;
        return d2;
    }

    public void makeNoise() {
        if (weightInPounds < 10)
            System.out.println("yipyipyip!");
        else if (weightInPounds < 30)
            System.out.println("bark.");
        else
            System.out.println("woof. woof.");
    }
```

Question: Will this program compile? If so, what will it print?

```
public class DogLoop {
    public static void main(String[] args) {
        Dog smallDog = new Dog(5);
        Dog mediumDog = new Dog(25);
        Dog hugeDog = new Dog(150);

        Dog[] manyDogs = new Dog[4];
        manyDogs[0] = smallDog;
        manyDogs[1] = hugeDog;
        manyDogs[2] = new Dog(130);

        int i = 0;
        while (i < manyDogs.length) {
            Dog.maxDog(manyDogs[i], mediumDog).makeNoise();
            i = i + 1;
        }
    }
}
```

< 10:
yip

< 30:
bark

>=30:
woof

Answer to Question

Won’t go over in live lecture. Use the visualizer to see the solution at this link.

- Or if you’re watching this video and can’t find the slides, the link is: http://goo.gl/HLzN6s

public static void main(String[] args)

One Special Role for Strings: Command Line Arguments

```
public class ArgsDemo {
    /** Prints out the 0th command line argument. */
    public static void main(String[] args) {
        System.out.println(args[0]);
    }
}
```

```
jug ~/Dropbox/61b/lec/usingDefiningClasses
$ java ArgsDemo hello some args
hello
```

ArgsSum Exercise

Goal: Create a program ArgsSum that prints out the sum of the command line arguments, assuming they are numbers.

- Search engines are our friend!

One Special Role for Strings: Command Line Arguments

```
public class ArgsSum {
    /** Prints out the sum of arguments, assuming they are
    *  integers.
    */
    public static void main(String[] args) {
        int index = 0;
        int sum = 0;
        while (index < args.length) {
            sum = sum + Integer.parseInt(args[index]);
            index = index + 1;
        }
        System.out.println(sum);
    }
}
```

How'd we know to do this? We Googled "convert string integer java".

```
$ java ArgsSum 1 2 3 4
10
```

Using Libraries (e.g. StdDraw, In)

Java Libraries

There are tons of Java libraries out there.

- In 61B, we will provide all needed libraries. These include (but are not limited to):
  - The built-in Java libraries (e.g. Math, String, Integer, List, Map)
  - The Princeton standard library (e.g. StdDraw, StdAudio, In)

As a programmer, you'll want to leverage existing libraries whenever possible.

- Saves you the trouble of writing code.
- Existing widely used libraries are (probably) will probably be less buggy.
- … but you'll have to spend some time getting acquainted with the library.

Java Libraries

As a programmer, you'll want to leverage existing libraries whenever possible.

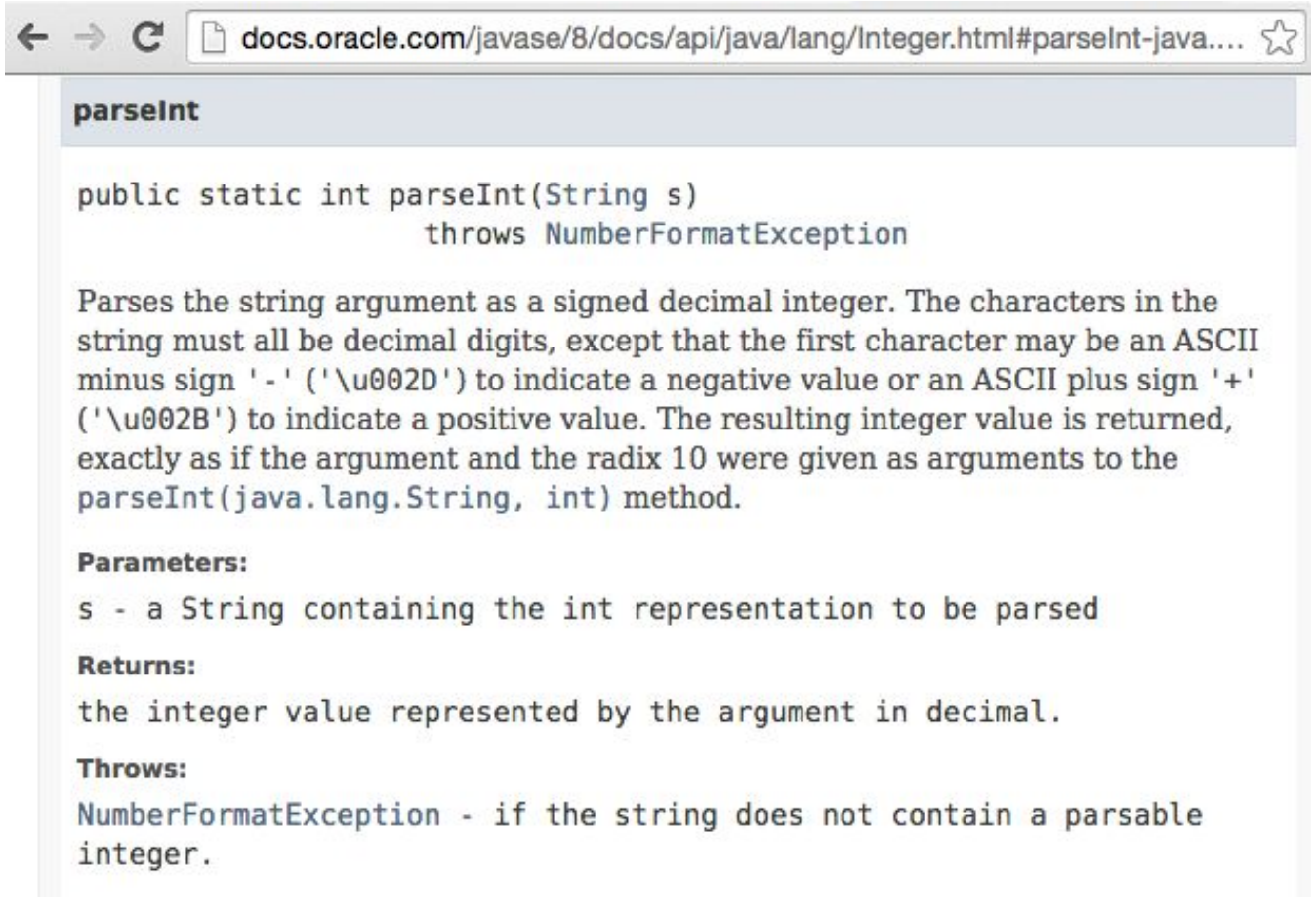
Best ways to learn how to use an unfamiliar library:

- Find a tutorial (on the web, youtube, etc.) for the library.
- Read the documentation for the library (Java docs often very good).
- Look at example code snippets that use the library.

In 61B, please don't use new libraries downloaded from the web (won't work with our grader). Use the provided libraries only.

- Won't really come up until we get to project 2.

Library Documentation Example



The Princeton Standard Library

We'll be using a great library courtesy of my old colleagues at Princeton, mostly Kevin Wayne: http://introcs.cs.princeton.edu/java/stdlib/

Makes various things much easier:

- Getting user input.
- Reading from files.
- Making sounds.
- Drawing to the screen.
- Getting random numbers.

Make sure to see the example code for project 0!

Proj0 Libraries Demo (time permitting)

Citations

Dog videos:

Maya the malamute:
https://www.youtube.com/watch?v=D07rb5KsiSE

Mystery alien dog:
https://www.youtube.com/watch?v=jeQcGjprcCM

Dog Marriage:

- Jasmine, K.T.