Access modifiers and constructors

Ch 5.2 (little 5.5)
Announcements

HW submission: please ensure you follow filename and output instructions

Lab: out of 4 this week (instead of 5)

My this Friday will be changed from 12:30 - 1:30
... to ...
5:00 - 6:00
Highlights

- public vs private (access modifiers)

```java
public int x;
private int y;
```

- Constructors

```java
public class MyClass {
    public MyClass() {
    }
```
public vs private
public vs private

The **public** modifier allows any class anywhere to access the variable/method.

The **private** modifier only allows access in the class where the variable/method is defined.

(See: RunSuperDouble.java and SuperDouble.java)
public vs private

All variables should be **private**

While this means you need methods to set variables, users do not need to know how the class works

This allows an easier interface for the user (also easier to modify/update code)

(See: StringClass.java)
public vs private
Constructors

Whenever you make a Class type (with `new`) you have used a `constructor` method.

For example:

```java
Scanner input = new Scanner(System.in);
```

Constructors always have the same method name as the class and are only run (once) when you make a variable of that Class type (See: Constructor.java)
Constructors

Constructors are useful to initialize variables within a class

Some classes need additional input to be useful, like Scanner

Scanner (reader) must have System.in (keyboard) in its constructor or else it would not know where to get input from (mouse? video camera?)  
(See: Point.java)
Constructors

When you say: \texttt{int x;} ... you create a box for x (that holds an int)

When you say: \texttt{Point y;} ... things are more complicated (more on this next time)

You have one box, but you don't actually have a “point” box yet, as you need “new” and a constructor to make one
Constructors

Constructors are just normal methods, but special in a few ways:

1. They must have the same name as the class
2. They have no return type...
3. ... instead they always create an instance
4. To create an instance of a class, you MUST run a constructor
Constructors

You can easily tell how instances you have based on how many constructors (or news):

```java
Point a;
Point b;
Point c = new Point(2,3);
b=c;
c = new Point(1,6);
a=c;
```

2 constructors (2 news), thus 2 point boxes

TL;DR: One constructor per class object
Setters/getters

Some commonly created methods are ones that change or get the value of a variable.

A “getter” often simply returns the value of the variable inside the class.

```java
private int x;

public int getX() {
    return x;
}
```

You have in-fact used one of these, the String class's length() method.

(See: Getter.java)
If you only set values in the constructor and nowhere else... then it is immutable

There is no way for someone using the class to change the values, they can only create new objects with different values

Suppose you got a pet cat... you cannot change this into a dog (only way is to get another pet) (See: Pet.java)
Setters/getters

If you want to be able to change variables inside a class, you can make setter methods:

```java
private int x;

public void setX(int newX) {
    x = newX;
}
```

This is convenient if you just want to change one part of the class, but not regenerate a new copy with slightly different values (See: Pet.java (modified))