Last Time – Custom Point Class

- We wrote our own Point class, consisting of two fields and a couple of methods.
- When we wrote our own class, we basically defined our own new “type”, Point (not JDK's Point).
- Once we defined the class, we could use that class to instantiate multiple Point objects.
- So, we had a single Point class, and multiple instances/Point objects we created from it.
Object Creation

- Objects in a program are created as the program executes.
- Once objects are created, we call methods on them.
- A Java class allows us to define new types of objects that we can create.
- Can also use classes to create “client” code for these objects.
Object Terminology

- **Object instance** – this is the same as saying “an object”. An instance is a single, “tangible” object we have instantiated from a class.
  - Ex. Scanner input = new Scanner(System.in);
  - input refers to a Scanner-type object.

- **Object state** – The set of values that's stored in an object:
  - Ex. A Point object has x and y coordinates.
Terminology

- **field** – A variable defined in a class outside any methods (and without “static”). A field holds part of the object state. All the fields together define the object state.

- **method** – Two different kinds:
  - An object or “instance” method (no “static”) provide actions on their objects.
  - static methods just provide actions on whatever they are given (don't need instance to call from).
Method Calls

- **method call** – To call an object method, you need a reference (variable) to the object you want to work with.

- After this reference variable, you put a single “.”, then the name of the method, with parameters inside the parentheses.

- To call a static method, just need method name plus params.
Method Calls

• If calling a static method in another class, you need the class name, then a “.”, then the method name
  – Ex: Math.max(x, y)

• Note that a “.” does not mean we're calling an object method
  – If the line starts with a lowercase letter and has a dot, it's usually an object call - “p1.getX()”
General Object Methods

- **Constructor** – this is the code in an object class for creating new instances. This is like a special method (no return type...)

- A constructor has special syntax – it must be named exactly the same as the class, and can't have a return type.
General Object Methods

- **accessor** (or “getter”) - These methods are used to access the fields of an object, kind of like peeking at the current state of the object.

- We usually name accessors something like “getFieldName”, and it has the same return type as the field.

- Ex: `p1.getX()` returns the x-coord. of a point object (double in JDK, int in our Point type).
General Object Methods

- **mutator** (or “setter”) - A method used to “mutate” or change the value of a field. Thus, a mutator method can change the state of the object it is called on.

- Usually name mutators “setFieldName”, and they don't return anything (void return type).

- Ex: scanner.useRadix(16) will change the scanner object to use hex.
Encapsulation

- The use of getter and setter methods tie in closely with the idea of **encapsulation**.
- Getters allow us to access the current state of an object without providing a means to change state.
- Setters allow us to dictate exactly which fields (parts of the object state) may be changed, and in which way.