PFTD

- Review of classes and the object-concept
  - Ensuring progress with NBody
- Converting between arrays and ArrayList objects
  - Especially useful in our APT problems
  - Useful in being able to use java.util
  - Data conversion helps solve many problems
- ArrayList, Set, and java.util.Collection
  - Very useful in solving APT problems
  - Will help as we move toward Java Interfaces
  - Using APIs: key to many programming problems

The Object Concept

- If you make a Point a ... and a Point b ...
  - Determine distance between a and b?

```java
class Point {
    double x;
    double y;
    public Point() {
        x = y = 0;
    }
    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }
}
```
Distance Algorithmically

- Math.sqrt(DeltaX^2 + DeltaY^2)
  - Distance between (3,2) and (7,-1) is ...

- First you have to solve the problem, then you can determine how to write the code and where the code goes
  - What's responsible for determining distance?
  - Could be point, could be utility method

All methods are in a class – in Java

- Ask not what you can do to an object, but what an object can do to itself
  - A point determines how far it is from another
  - What's the alternative?

```java
class Point {
    // code not shown
    public double distanceFrom(Point a) {
    }
    public static double distanceBetween(Point a, Point b) {
    }
}
```

Object method v Class method

- Determine distance between Point a and b?
  - double d = a.distanceFrom(b);
  - double d = Point.distanceBetween(a,b);

- What will difference in code be?
  - a.x - b.x // in distanceBetween
  - this.x - b.x
  - x - b.x

The Object Concept

- Method has access to object (private) state
  - In client code we'd write a.distanceFrom(b)
  - Internally object with label 'a' is 'this' – the implicit parameter!

- In static method there is no object
  - Think Math.sqrt(.) you pass in object, but there isn't one on whom .sqrt() is invoked, e.g., 25.sqrt() could return 5 [not legal in Java]
  - Might create PointUtil.distanceBetween(a,b) if we didn't have .distanceFrom
Summary and Review

- **Classes have state and behavior**
  - State is (private) instance variables
  - Behavior is methods

- **Objects are instances of classes, think cookie-cutter**
  - Construct object: call new, invoke constructor
  - Default constructor, copy constructor, other

- **Static methods belong to class, not object**
  - Math.sqrt, don’t need to call new to invoke

High-level String[] or ArrayList<String>

- Use an array, e.g., int[] or String[] or Planet[], ...
  - When you know how many elements will be stored in the array
  - When .length tells you how many values stored
  - When you have to because API demands it, e.g., String.split

- **Advantage: fast, syntactically simple, stores primitives (e.g., int, double, char) and objects**
  - Package java.util.Arrays provides utilities

Data conversion for strings: [] returned

- Extract all "values" in white-space delimited string
  - Or CSV, comma separated values
  - Use the String.split(DELIM) method
- .split("") .split(",") .split("\s+")
  - Regular expression "\s+" is white-space
  - See javarepl.com

```java
String s = "cow dog fish bat toy";
String[] ars = s.split(" ");
---
ars is ["cow", "dog", "fish", "bat", "boy"]
```
Tradeoffs in using ArrayList?

- Advantages in using ArrayList?
  - Growable array, so no apriori limit on size
  - Works well with Set, Map, java.util.Collections
    - Example: add all elements of array to ArrayList?
  - Supports Iterator, e.g., remove during iteration

- Disadvantages in using ArrayList
  - Does not store primitives, int to Integer
  - Syntactically less "clean" than array, e.g.,
    \( a[k] += 1 \) vs \( a.set(k, a.get(k)+1) \)

---

Set concepts with java.util

- We will use TreeSet and HashSet as clients
  - Understand them from API
  - Then learn how to implement them

- For now, TreeSet is sorted, HashSet is fast
  - If we care about order of iteration ... no hash
  - If we care about ranges of elements ... no hash

- Will understand in more depth when we learn
  about .equals, .hashCode, and .compareTo

---

methods that work fast with sets

- .contains() and .add()
  - What about with ArrayList?

- .removeAll(,), .addAll(,), .retainAll(,)
  - Set difference, union, intersection
  - Work with ArrayList too, but slow

---

From Algorithm to Code for an APT

- [https://www.cs.duke.edu/csed/newapt/sandwichbar.html](https://www.cs.duke.edu/csed/newapt/sandwichbar.html)
- After reading it, solve this instance by talking to a neighbor, we'll discuss

```java
available[] = {"ham", "cheese"}
orders = [] = {"ham cheddar", "ham ham"}
```
Toward a solution


- Go over answers together after thinking about the problem and its solution
- `Arrays.asList(...)` will NOT work with array of primitives, need to make `ArrayList<Integer>` explicitly for example

Develop algorithm explicitly

- You should be able to solve an instance of the problem using your algorithm
- Think about steps conceptually, not in Java when developing algorithm (this will change as you learn more Java!)
- Try out algorithm on an instance you make up!
- Be able to explain to a friend
- Develop code!

Temptation? Be careful of the Google

- Search query `apt sandwichbar`