Logistics

Reading from Eck
Ch 5 on Objects/Classes

Goals
- Finish arrays of objects
- Static fields
- Non-static methods

Lab08: Simple object definitions
- Stock object
- Methods in same java file

Project
- Spec up
- Due a week from Wed
Static/Non-static Stuff so far

- The keyword static in Java roughly translates to "belongs to the whole class and all objects"
- So far we have written the following

static methods

```java
public class MyClass{
    public static int doSomething(...){
        ...
    }
}
```

- Nothing special about them, invoked with `MyClass.doSomething(..)`
- Must pass in all parameters to the methods

Non-static fields

```java
public class Thing{
    int part1;
    double part2;
    String part3;
}
```

- Each Thing has its own part1, part2, part3
- 4 Things means 12 pieces of data, 4 ints, 4 doubles, 4 String references
Static Class Fields

- A static field indicates there is only 1 memory location for the entire class, NOT one per object
- Closest thing Java has to a *global variable*
- Seen examples of static fields from some classes
  ```java
double pie = Math.PI;
double natbase = Math.E;
PrintStream ps = System.out;
```
- Syntax static establish a static field is simple
  ```java
public class Mixed{
    public static int e; // static field
    public String f; // non-static field
}
```
- Every instance of a Mixed has its own f
- There is only one integer e, accessible via Mixed.e
Demo of Static vs Non-Static Fields

```java
class OnlyStatic{
    public static int a; // both static
    public static String b;
}
class OnlyNon{
    public int c; // both non-static
    public String d;
}
class Mixed{
    public static int e; // one static
    public String f; // one non-static
}

public class StaticFields{
    public static void main(String args[]){
        OnlyStatic.a = 5; OnlyStatic.b = "bb";
        OnlyNon x = new OnlyNon();
        x.c = 10; x.d = "dd";
        OnlyNon y = new OnlyNon();
        y.c = 15; y.d = "dddd";
        Mixed.e = 20;
        Mixed z = new Mixed();
        z.f = "ff";
        Mixed w = new Mixed();
        w.f = "ffff";
    }
}
```
Exercise: Recap what we learned about static fields

1. What’s the difference between a static and a non-static field?
2. How many of each kind of field are gotten when calling `new`?
3. Draw a quick diagram of the following.

```java
class Thing {
    public int red;
    public double blue;
    public static int green;

    public static void main(String args[]) {
        Thing x = new Thing();
        Thing y = new Thing();

        x.red = 5;
        y.blue = 7.0;

        // DRAW HERE

        Thing.green = 9;
        Thing.red = 10;
    }
}
```
Non-static Methods

- static roughly means *class-level*, as in belonging to the entire class
- Non static roughly means *instance-level*, as in associated with a specific instance/object
- Non-static methods are ALWAYS invoked with a specific object/instance

String s = "hello";
String t = "goodbye";

int len1 = s.length(); // 5
int len2 = t.length(); // 7

- During a the execution of a non-static method, the keyword this refers to the object on which the method is running
Compare: Static vs Non-static Method Defs/Calls

### Static

```java
1 public class Omelet{
2    int eggs;
3    int cheese;
4    double cookedFor;
5    String extras;
6
7    static void cookFor(Omelet om,
8        double time){
9        om.cookedFor += time;
10    }
11    static void addEgg(Omelet om){
12        om.eggs++;
13    }
14 }
15 main(){
16    Omelet standard = new Omelet();
17    int x = 5;
18    Omelet.addEgg(standard);
19    Omelet.cookFor(standard, 2.5);
20 }
```

### Non-static

```java
1 public class 000melet{
2    int eggs;
3    int cheese;
4    double cookedFor;
5    String extras;
6
7    void cookFor(double time){
8        this.cookedFor += time;
9    }
10    void addEgg(){
11        this.eggs++;
12    }
13 }
14 main(){
15    000melet standard = new 000melet();
16    int x = 5;
17    standard.addEgg();
18    standard.cookFor(2.5);
19 }
```

Examine 000melet.java to see full implementation
this variable: reference to current object

- Variable `this` is automatically created in non-static methods.
- Gets filled in with the value of the object being operated on.

```
standard.addEgg();
coronary.addEgg();
```

during `addEgg()`, this will refer to `OOOmelet standard`
during `addEgg()`, this will refer to `OOOmelet coronary`
Constructors

- Objects usually have necessary fields initialized at creation.
- Special method called a constructor.
- Method name is always identical to class name, return type is omitted.
- CK commonly uses `this.field = param;` to initialize fields.

```java
public class OOOmelet{
    ...
    // Constructor to initialize fields to given values. cookedFor is always initialized to 0.0.
    public OOOmelet(int eggs, int cheese, String extras){
        this.eggs = eggs; // set field eggs to parameter eggs
        this.cheese = cheese; // set field cheese to parameter cheese
        this.extras = extras; // set field extras to parameter extras
        this.cookedFor = 0.0; // always set cookedFor to 0.0
    }
    ...
}

public class OOOmeletMain{
    public static void main(String args[]){
        OOOmelet small = new OOOmelet(2,5,"ham"); // smallish OOOmelet
        OOOmelet large = new OOOmelet(5,8,"bacon"); // largeish OOOmelet
        ...
    }
}
```
Exercise: Draw a Memory Diagram

- Show the OOOMeletMain.java and OOOMelet.java.exercise
- Running the main() method, trace execution
- Draw memory diagrams of what things look like at the numbered locations
- **Note:** May hit some locations more than once
- **Important:** Don’t forget the automatic this variable in non-static methods
Name Binding Resolution Mechanics
Easy Printing: *toString()* method