CS 211: Class and Object Concepts

Chris Kauffman

Week 4-2
Logistics

Goals Today

- P2 Questions
- Making your own Classes
- `toString()`, `equals()` methods
- static methods
- Access Modifiers
- Getters/Setters

Reading: Classes and Objects

- Building Java Programs Ch 8
- Lab Manual Ch 4 and 5

Career Fair!

- 11:00 a.m.- 4:00 p.m.
- Dewberry Hall, Johnson Center
- Wed 2/17: Science/Tech
- Thu 2/18: Business/Non-tech
P2 Questions

- P2 Due Sun 2/15 at 11:59pm
- Otherwise this is the last chance to discuss in person
Lab 04 Suggested Practice Problems

- **Task:** Will need to program and submit in lab
- Closed resource
- Problems are good prep
- **Practice Problems Posted** on the labs page
Questions you should be able to answer

- Here’s a method, what variables are in the call Stack when the method is run?
- What does `new` do in Java?
- How is an array of `ints` laid out in memory?
- How is an array of `String` or `OOOmelet` laid out?
- What is the purpose of the keyword `this` in Java?
printf and String.format()

Part of code distribution is PrintfDemo.java

- Create nicely formatted output with printf() and
- Create nicely formatted strings with String.format()
  - Needed for O0Omelet.toString()
  - Needed for P2’s toString() methods
- Data in fixed width / columns
- Limit floating precision to 1,2,3,4... decimal places (needed for project)
To String, or Not To String. That is not a question.

'Tis almost **always better** to endure writing a `toString()` method that prints a pretty version of the object.

**Write `toString()` for OOOmelet**

Welcome to DrJava.

```java
> OOOmelet standard = new OOOmelet();
> System.out.println(standard.toString());
3 egg 4 oz cheese omelet, cooked for 0.0 minutes

> standard.cookFor(2.3)
> System.out.println(standard)
3 egg 4 oz cheese omelet, cooked for 2.3 minutes

> OOOmelet coronary = new OOOmelet(5,12);
> coronary.addIngredient("bacon");
> coronary.cookFor(4.6785)
> System.out.println(coronary)
5 egg 12 oz cheese omelet, cooked for 4.7 minutes
```

**Notice** how the total cook time is formatted.
Exercise static Methods the Best Omelet

- static is stand-alone, independent shared by all objects
- Write code for bestOmelet(arr)

```java
public class OOOmelet{
    // Return the "best" omelet in an array; better omelets have higher
    // calorie counts as reported by the o.getBaseCalories() method. If
    // the array is empty, return null.
    public static OOOmelet bestOmelet(OOOmelet [] arr){...
}
}

Welcome to DrJava.
> OOOmelet arr[] = {new OOOmelet(3,4), new OOOmelet(2,10),
    new OOOmelet(8,2), new OOOmelet(3,3)};
> OOOmelet best = OOOmelet.bestOmelet(arr);
> best
2 eggs 10 oz cheese omelet, cooked for 0.0 minutes
> best.getBaseCalories()
1328
> OOOmelet empty[] = {};
> OOOmelet other = OOOmelet.bestOmelet(empty);
> other
null
```
Java enables *Access Control* for insides of classes

- Visibility of fields and methods to other stuff
- public, protected, none, private
- Put them in front of methods and fields
- Play with these in OOOmelet
Access Modifiers

Access Levels for Fields/Methods by other stuff

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Class</th>
<th>Package</th>
<th>Subclass</th>
<th>World</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>protected</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>no modifier</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>private</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

- Mostly concerned with public and private, read about others on your own
- Most projects will specify required public methods, maybe public fields
- Most of the time you are free to create additional private methods and fields to accomplish your task

Official docs on access modifiers

http://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html
Getter, Setter, Class Invariant

Common Java convention is to make all fields private and provide getter and setter methods to change them.

Getter/Setter for Eggs

```java
public class Omelet{
    public int eggs;
    public int ozCheese;
    ...
    public double getEggs(){
        return this.eggs;
    }
    public void setEggs(int e){
        if(this.totalCookMinutes > 0){
            throw new RuntimeException("yuck");
        }
        this.eggs = e;
    }
    ...
}
```

Questions

- Does it make sense to change the number of eggs after an omelet is cooked?
- Does it make sense to add `setCookMinutes(double)` to arbitrarily change `totalCookMinutes`?
- Why use getters/setters?
Typically Fields are private

POmelet: Private fields
Provide getters to report fields like eggs and cook time

```java
public class POmelet{
    private int eggs;
    private double totalCookMinutes;
    ...
    public double getEggs(){
        return this.eggs;
    }
    public double getTotalCookMinutes(){
        return this.eggs;
    }
    ...
}
```

Use of Getters v. Private Fields

```java
POmelet x=new POmelet(3,4);
// Correct
int eggs = x.getEggs();
// Error
x.eggs = 5; // No such symbol
x.cookFor(2.5);
// Correct
if(x.getTotalCookMinutes() > 0.0){
    ...
}
// Error
if(x.totalCookMinutes > 0.0){
    ...
}
```
Why Getters vs. Public Fields

- Simple objects can probably have public fields, direct access
  - Don’t do this as you’ll be penalized an manual inspection
- Slightly more complex objects like 000melet might get away with public fields but would allow..
  - "Uncooking" of omelets: o.totalCookMinutes = 0.0;
  - Add eggs after being cooked
  - POmelet with private fields prevents this
- Complex objects like Scanner and GeneralLE must preserve invariants: different parts must agree with each other.
  - Changing one field might screw up another one
  - Deny direct access via private fields
  - Mutation methods like next() and setX(v) keep all fields synchronized

Abstraction Up and Down

Break a problem into smaller parts. Define public methods between those parts. Think about internal details for one part at a time. Recurse for subparts as needed.
Scope and this

Name resolution rules don’t always require use of keyword this

Using this

```java
public class POmelet{
    private int eggs;
    private double totalCookMinutes;

    public int getEggs(){
        return this.eggs;
    }
    public void cookFor(double cookMinutes){
        this.totalCookMinutes += cookMinutes;
    }
}
```

Without this

```java
public class POmelet{
    private int eggs;
    private double totalCookMinutes;

    public int getEggs(){
        return eggs;
    }
    public void cookFor(double cookMinutes){
        totalCookMinutes += cookMinutes;
    }
}
```
Recall: Equality and ==

main()
{
    int li1=3, li2=3;
    boolean eq1 = (li1 == li2);  // T/F??

    Integer bi1 = new Integer(4);
    Integer bi2 = new Integer(4);
    boolean eq2 = (bi1 == bi2);  // T/F??

    OOOmelet om1 = new OOOmelet(3,4);
    OOOmelet om2 = new OOOmelet(3,4);
    boolean eq3 = (om1 == om2);  // T/F??
}

- Draw a memory diagram for the above main method
- Determine the values of eq1, eq2, eq3
x.equals(y) methods

- Provide a **deep** equality check of x to y
- What’s **deep vs shallow**?
- **All** objects have one... why?
- **Most** objects define their own
- Technical note: difference between

```java
public boolean equals(Object other)
public boolean equals(Omelet other)
```
String Equality

Show a memory Diagram

String a = new String("hello");
String b = a;
String c = new String("hello");
String d = a + "";

What is printed

System.out.println(a == b);
System.out.println(a.equals(b));
System.out.println(a == c);
System.out.println(a.equals(c));
Exercise: Equality of Omelets

public class POmelet{
    private int eggs;       // How many eggs in the omelet
    private int ozCheese;  // How many ounces of cheese
    private String extraIngredients; // Extra ingredients added
    private double totalCookMinutes; // How long the omelet has cooked

    // Define me
    public boolean equals(POmelet other){...}

    ▶ OOOmelet x and OOOmelet y
    ▶ x.equals(y) is true when
        1. x and y have equal eggs (int)
        2. and equal ozCheese (int)
        3. and equal extraIngredients (String)
    ▶ 1 and 2 are easy
    ▶ 3 is slightly trickier
    ▶ Write the equality method
        ▶ Remember that x will be this, y will be other