Lecture 8
Java is Pointerless?
Hints on Programming

• Don’t wait until the last day to start.
• Develop incrementally. Start with a working program. Make small changes. Check that the program still works.
Example of incremental development

- Come prepared, having read the assignment, having read the textbook, having listened in lecture, having prepared the outline below.

- Copy working version of `Integra.java` from `orf201`.

- Add a method `func()` to evaluate a function. For starters, let the function be \( f(x) = x^2 \).

- Call `func()` from `integrate()` with a few choice values and print out the values computed to make sure `func()` works properly.

- Write code in `paint()` to draw the function defined by `func()`.
Example of incremental development

- Write a method `rect()` to compute the area using the rectangle rule. Put a call to this method in `integrate()`. Use `System.out.println` to print out the answer. Run with `lowx = 0, highx = 1, n = 1`. Compare this answer against a hand calculation. Try some other simple cases.

- Make a new method called `trap()`. Copy the code in `rect()` to `trap()`. Modify the code just copied so that it implements the trapezoidal rule. Test as above.

- Add code to `paint()` to illustrate the approximating rectangles. Test with small values of `n`.

- Add code to `paint()` to illustrate the approximating trapezoids. Test with small values of `n`.

- Change function implemented in `func()` to the complicated exponential function given in the assignment.
Memory Layout: Variables and Arrays

Declare:
```java
int n;
double[] x;
```

Initialize Variable:
```java
n = 12;
```

Initialize Array:
```java
x = new double[n];
```

Initialize Array Elements:
```java
for (j=0; j<n; j++) {
    x[j] = j*j;
}
```
Declare:

```
Zip z;
```

Instantiate Class Variable:

```
z = new Zip();
```

Initialize Components:

```
z.zip = "90210";
z.lat = 34.09;
z.lon = 118.41;
```
Memory Layout: Arrays of Class Variables

Declare:

```csharp
Zip[] zlist;
```

Initialize Array Variable:

```csharp
zlist = new Zip[100];
```

Initialize Array Elements:

```csharp
for (j=0; j<100; j++) {
    zlist[j] = new Zip();
}
```

Initialize Components:

```csharp
for (j=0; j<100; j++) {
    zlist[j].zip = Console.in.ReadInt();
    zlist[j].lat = Console.in.ReadDouble();
    zlist[j].lon = Console.in.ReadDouble();
}
```
Java Has Pointers

This makes an instance of an integer.

These only make pointers to objects of the type mentioned.
What does `new` do?

`new` makes an *instance* and gives a *pointer* to it.

```java
x = new double[12];
z = new Zip();
zlist = new Zip[10];
```

`zlist` is now an array of *Zips*. But each of the elements is just a pointer that still needs to be told where to point:

```java
for (int j=0; j<10; j++) {
    zlist[j] = new Zip();
}
```

Now, finally, the individual fields of `zlist[j]` can be used.