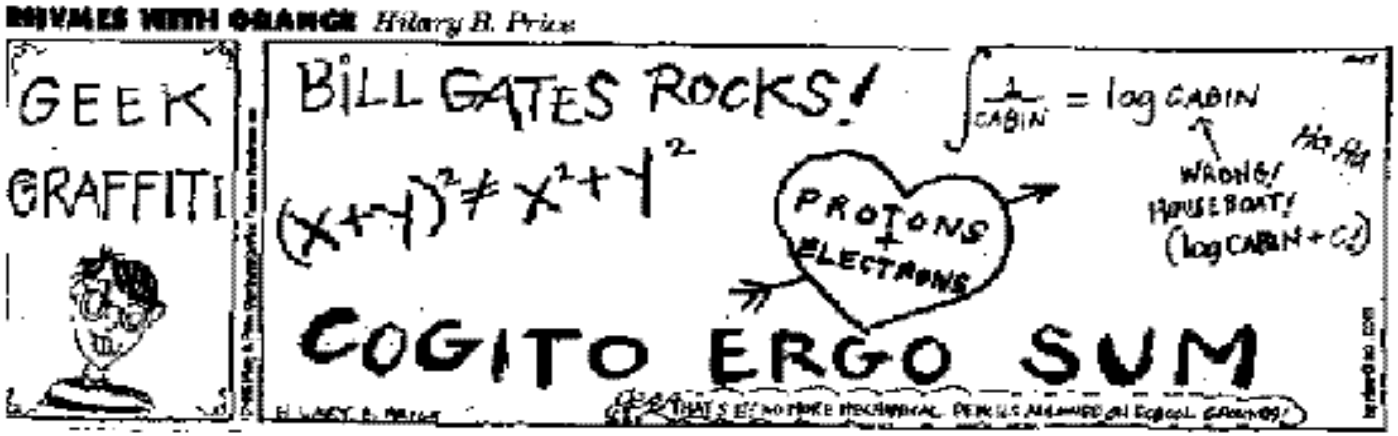
ORF 201

COMPUTER METHODS FOR PROBLEM SOLVING

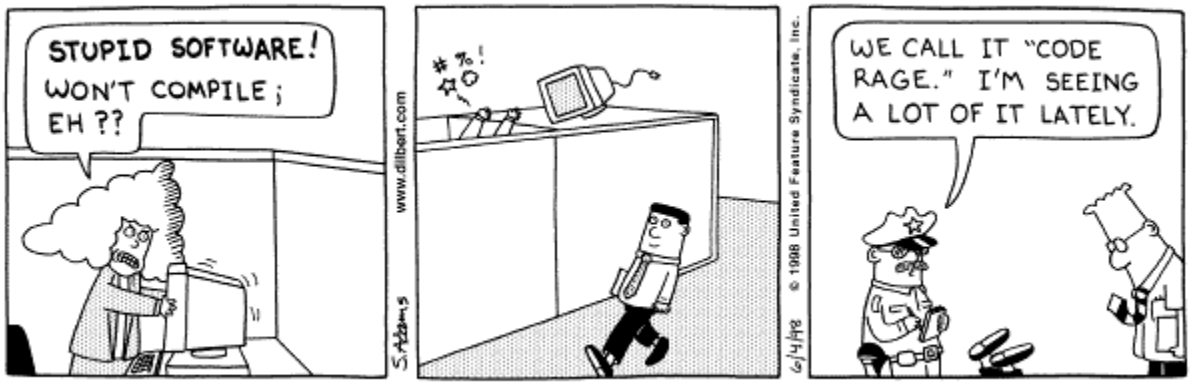
Lecture 8

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`**.

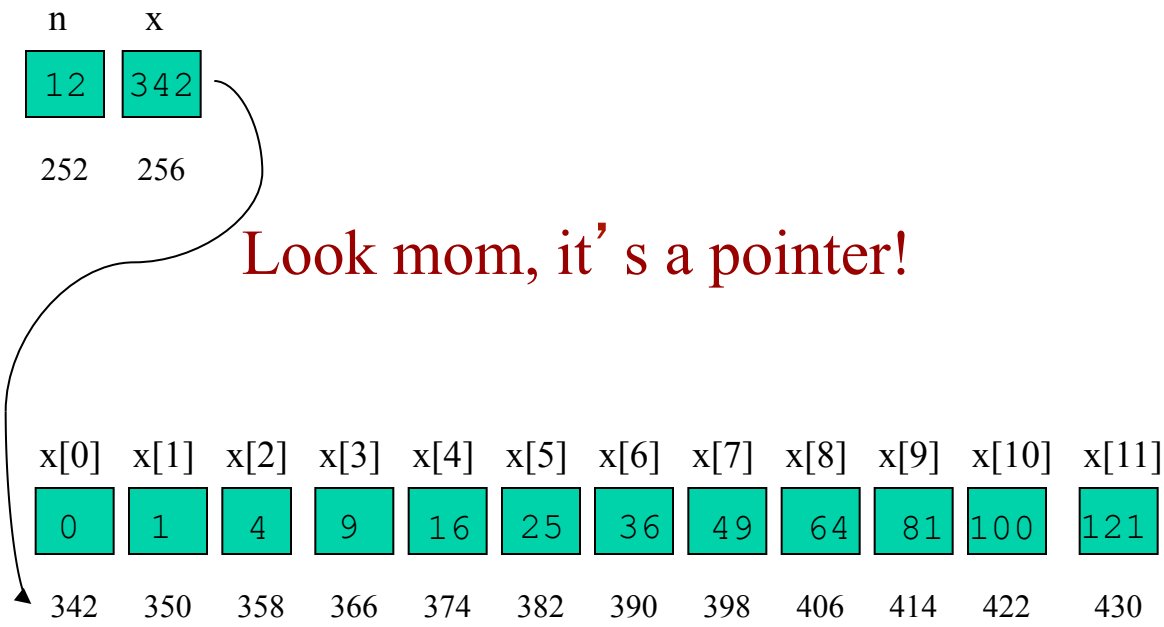
- Add code to **`paint()`** to illustrate the approximating trapezoids. Test with small values of **`n`**.

- Change function implemented in **`func()`** to the complicated exponential function given in the assignment.

Memory Layout: Variables and Arrays

Declare:

```
int n;
double[] x;
```



Initialize Variable:

```
n = 12;
```

Initialize Array:

```
x = new double[n];
```

Initialize Array Elements:

```
for (j=0; j<n; j++) {
    x[j] = j*j;
}
```

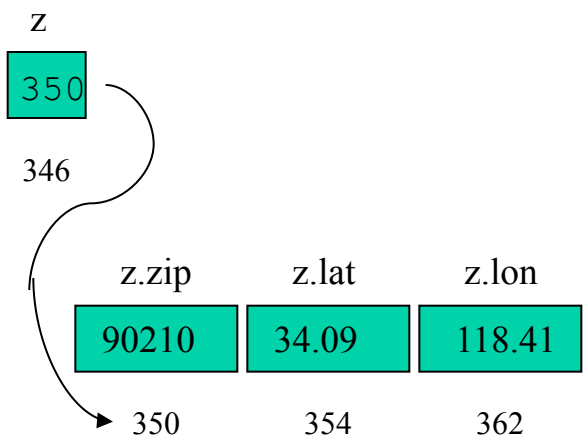
Memory Layout: Class Variables

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:

```
Zip[] zlist;
```

Initialize Array Variable:

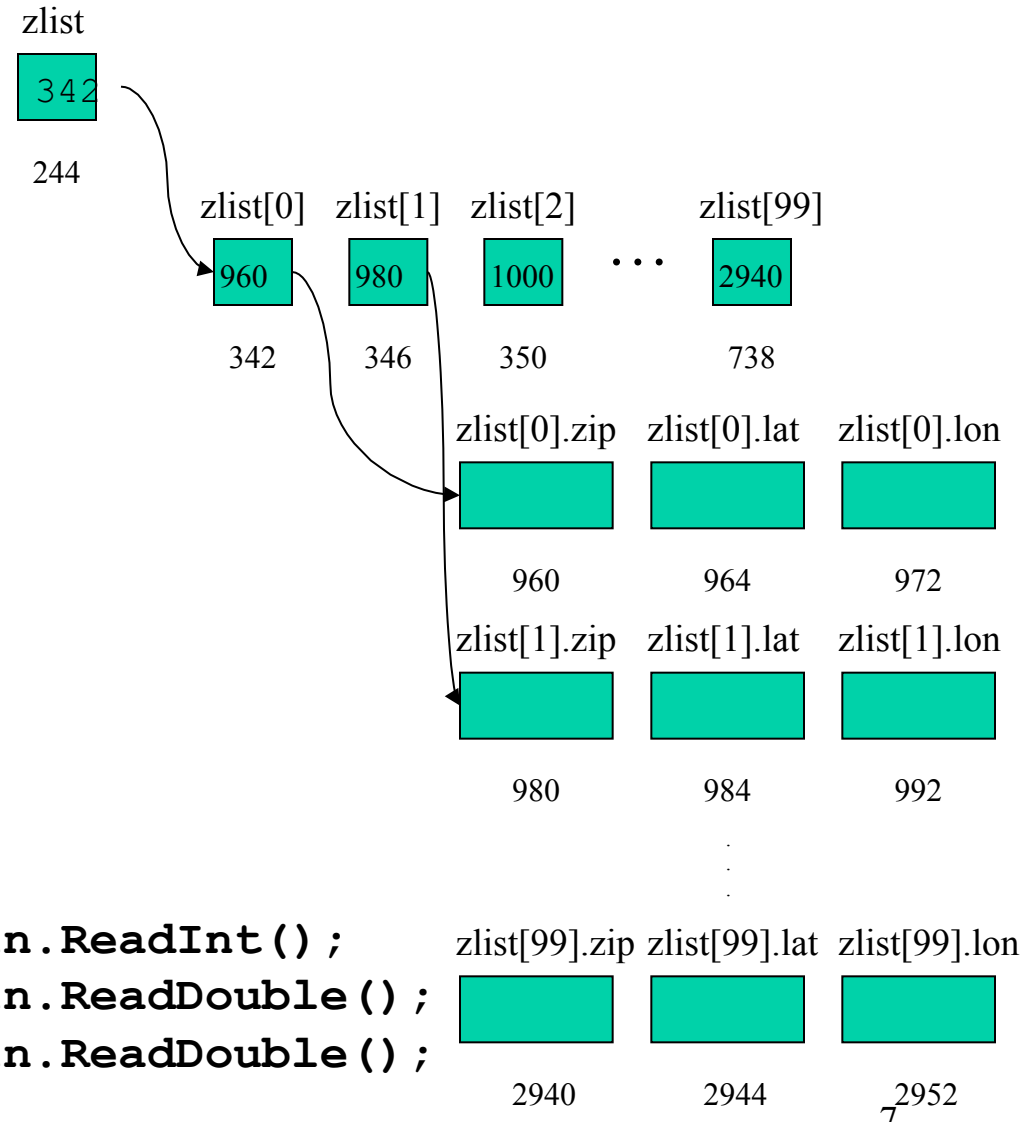
```
zlist = new Zip[100];
```

Initialize Array Elements:

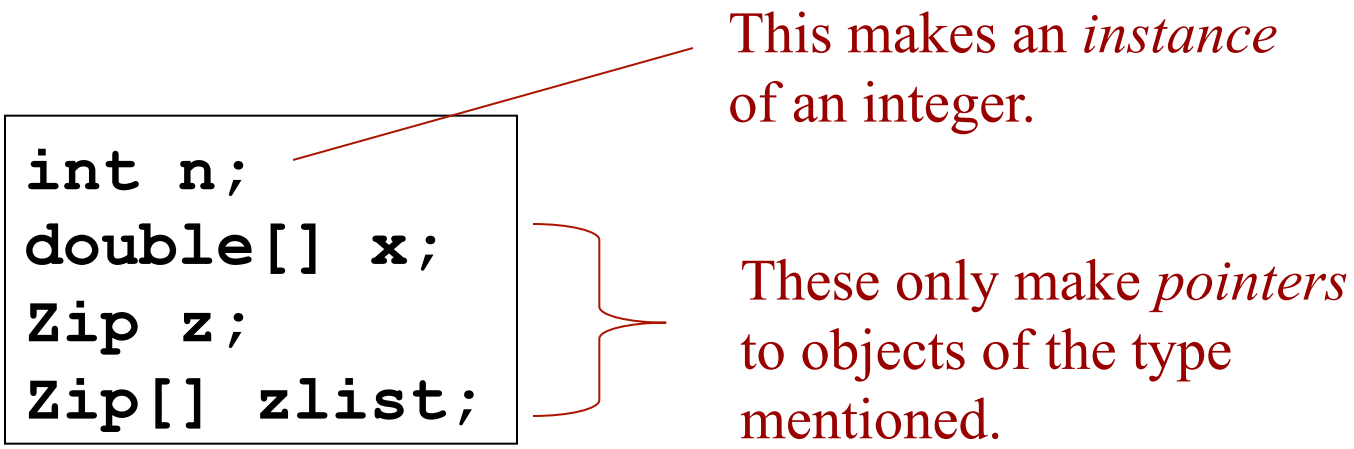
```
for (j=0; j<n; j++) {
    zlist[j] = new Zip();
}
```

Initialize Components:

```
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



What does `new` do?

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

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

`x` is now an array of doubles and using **`x[j]`** is now allowed.

`z` is now an instance of **`Zip`**. It is now possible to put things into the individual fields.

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

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

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