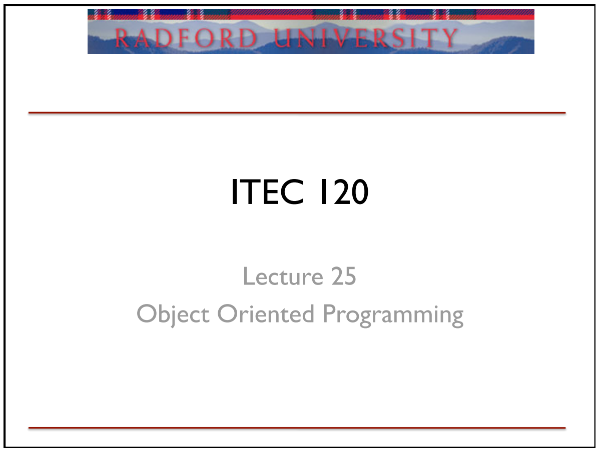
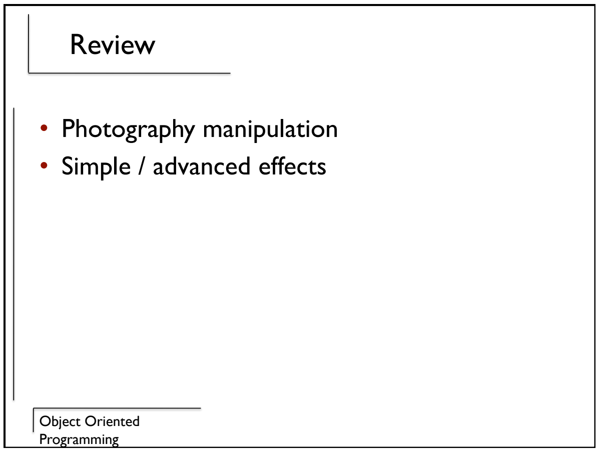


RADFORD UNIVERSITY

ITEC 120

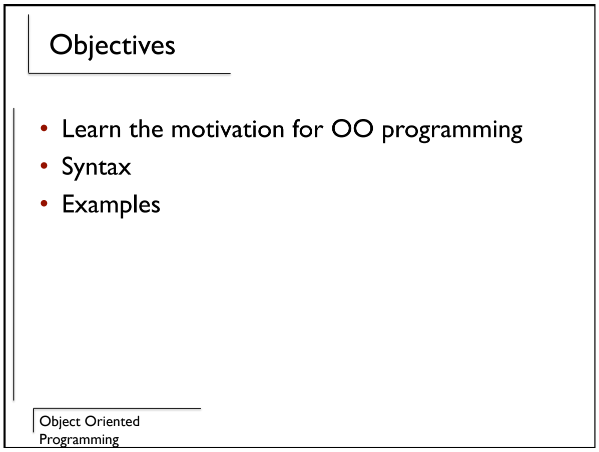
Lecture 25

Object Oriented Programming



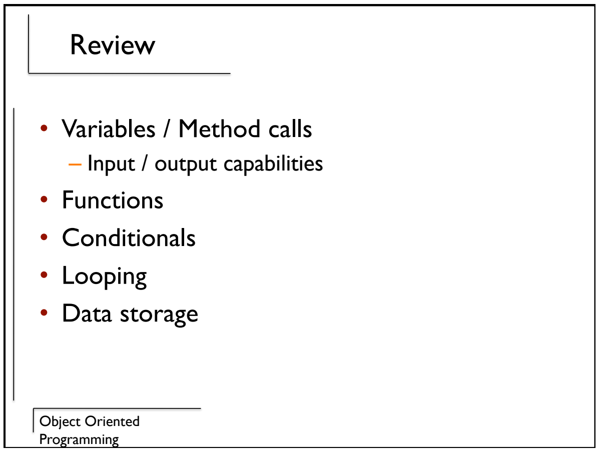
Review

- Photography manipulation
- Simple / advanced effects



Objectives

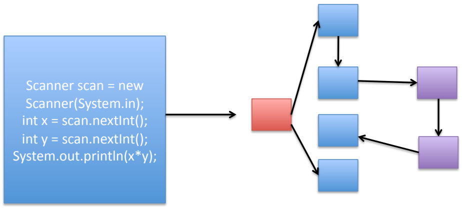
- Learn the motivation for OO programming
- Syntax
- Examples



Review

- Variables / Method calls
  - Input / output capabilities
- Functions
- Conditionals
- Looping
- Data storage

Current view



```
Scanner scan = new
Scanner(System.in);
int x = scan.nextInt();
int y = scan.nextInt();
System.out.println(x*y);
```

Stage 1: One thing to worry about

Stage 2: Functions and organizing code

Scalability

- Current view only goes so far
- How to manage thousands of lines of code?
- Can only do so much with functions…

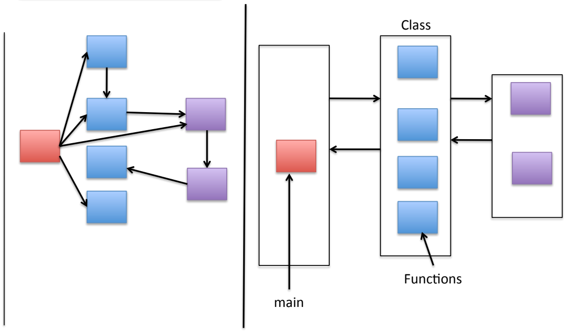
Examples

- Sound
  - Array of samples and code that can play it
- Pictures
  - Handling file formats
  - Displaying to the screen
  - Organizing data in general
- pBot
  - iPod application for teaching programming
  - GUI
  - Program storage
  - Simulator
  - Interpreter

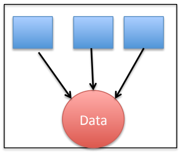
Shift

- Different view of programs
- A new paradigm
- Designed to handle complexity

OO



Last piece



```
public class Example
{
    private int x;
    public void init()
    {
        x=3;
    }
    public void second()
    {
        x=x+4;
    }
    public void third()
    {
        x=x=-2
    }
}
```

Syntax

Old

```
public class A
{
    public int square(int a, int b)
    {
        return a*b;
    }
}
```

Transitional

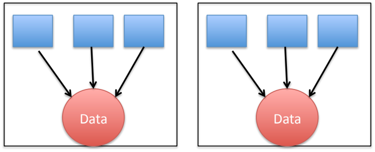
```
public class Pokemon
{
    public String name;
    public String type
    public int HP;
    public int damage;
}
```

```
Pokemon pika = new Pokemon();
pika.name = “Pikachu”
pika.type = “Electric”
pika.hp = 10
pika.damage=2
```

Steps

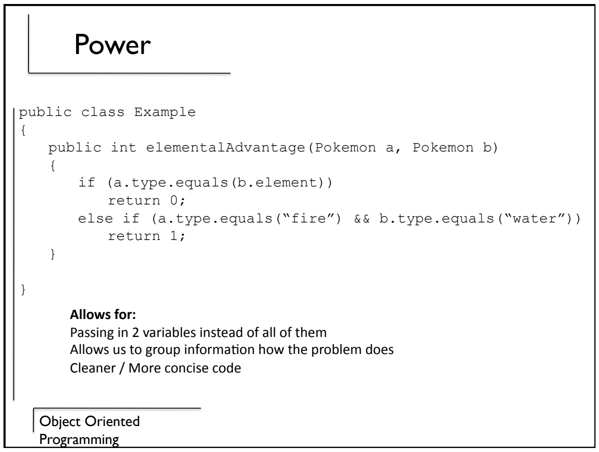
- What variable holds the class?
- What variables are inside of the previous variable?

```
Example2 a = new Example2();
Example2 b = new Example2();
```



```
a.Data=4;
b.Data=5;
```

Class Example2



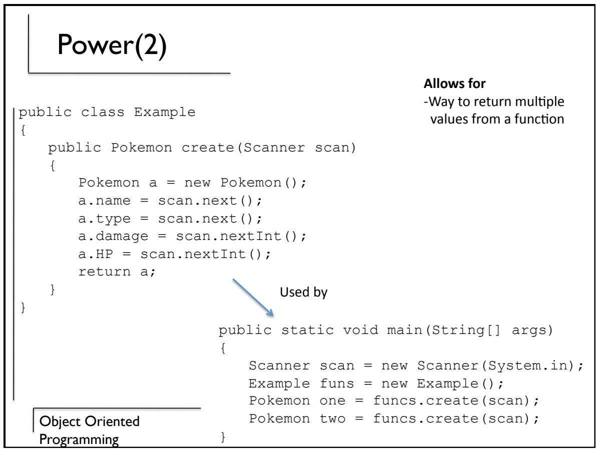
Power

```
public class Example
{
   public int elementalAdvantage(Pokemon a, Pokemon b)
   {
      if (a.type.equals(b.element))
         return 0;
      else if (a.type.equals("fire") && b.type.equals("water"))
         return 1;
   }

}
```

Allows for:
Passing in 2 variables instead of all of them
Allows us to group information how the problem does
Cleaner / More concise code

Object Oriented Programming



Power(2)

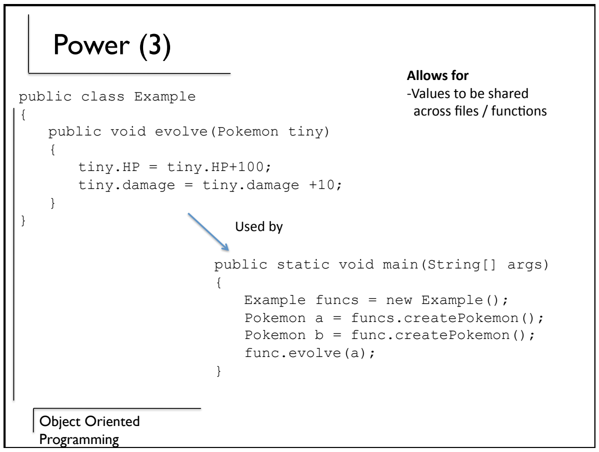
Allows for
-Way to return multiple values from a function

```
public class Example
{
   public Pokemon create(Scanner scan)
   {
      Pokemon a = new Pokemon();
      a.name = scan.next();
      a.type = scan.next();
      a.damage = scan.nextInt();
      a.HP = scan.nextInt();
      return a;
   }
}
```

Used by

```
public static void main(String[] args)
{
   Scanner scan = new Scanner(System.in);
   Example funs = new Example();
   Pokemon one = funcs.create(scan);
   Pokemon two = funcs.create(scan);
}
```

Object Oriented Programming



Power (3)

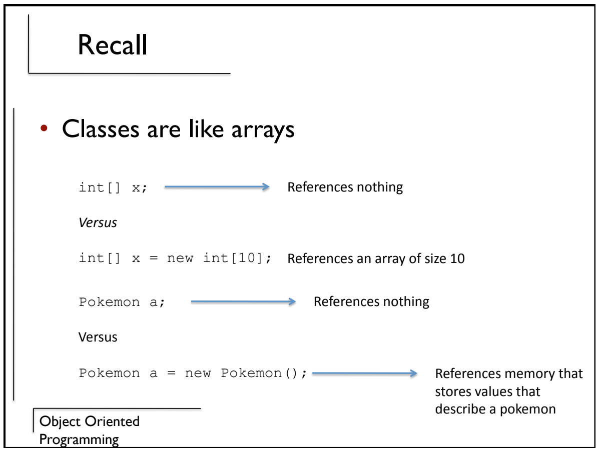
Allows for
-Values to be shared across files / functions

```
public class Example
{
   public void evolve(Pokemon tiny)
   {
      tiny.HP = tiny.HP+100;
      tiny.damage = tiny.damage +10;
   }
}
```

Used by

```
public static void main(String[] args)
{
   Example funcs = new Example();
   Pokemon a = funcs.createPokemon();
   Pokemon b = func.createPokemon();
   func.evolve(a);
}
```

Object Oriented Programming



Recall

- Classes are like arrays

`int[] x;` → References nothing

Versus

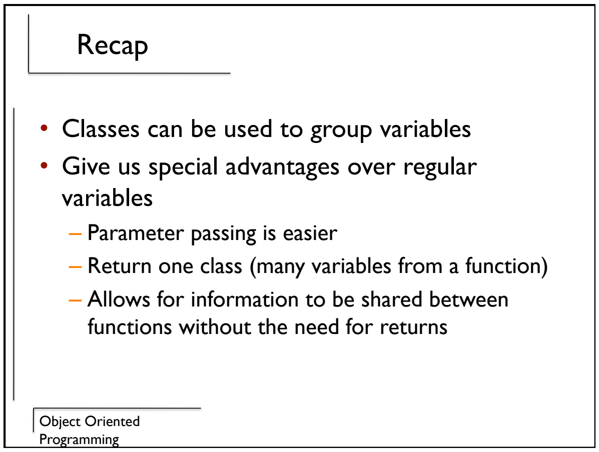
`int[] x = new int[10];` References an array of size 10

`Pokemon a;` → References nothing

Versus

`Pokemon a = new Pokemon();` → References memory that stores values that describe a pokemon

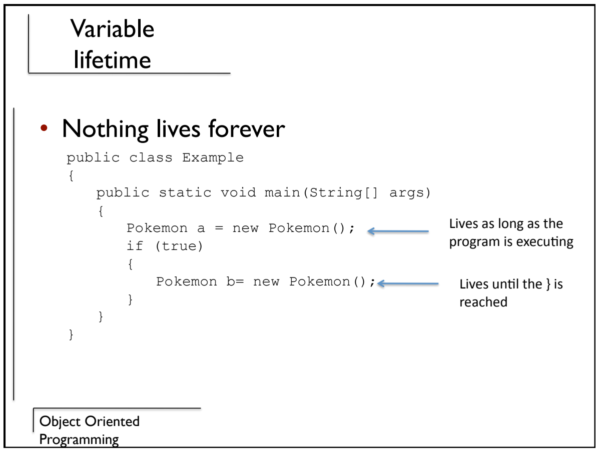
Object Oriented Programming



Recap

- Classes can be used to group variables
- Give us special advantages over regular variables
  - Parameter passing is easier
  - Return one class (many variables from a function)
  - Allows for information to be shared between functions without the need for returns

Object Oriented Programming



Variable lifetime

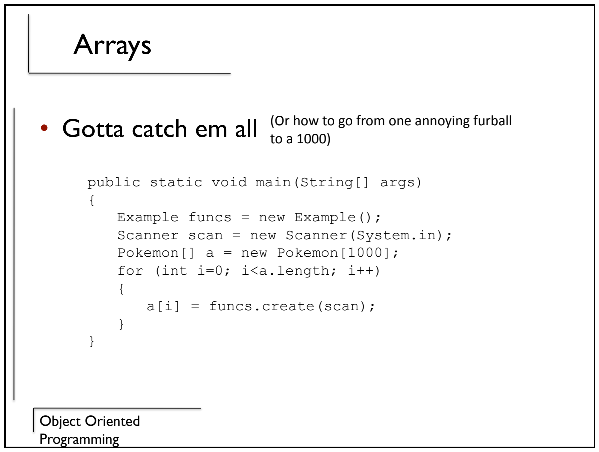
- Nothing lives forever

```
public class Example
{
   public static void main(String[] args)
   {
      Pokemon a = new Pokemon();
      if (true)
      {
         Pokemon b= new Pokemon();
      }
   }
}
```

Lives as long as the program is executing

Lives until the } is reached

Object Oriented Programming

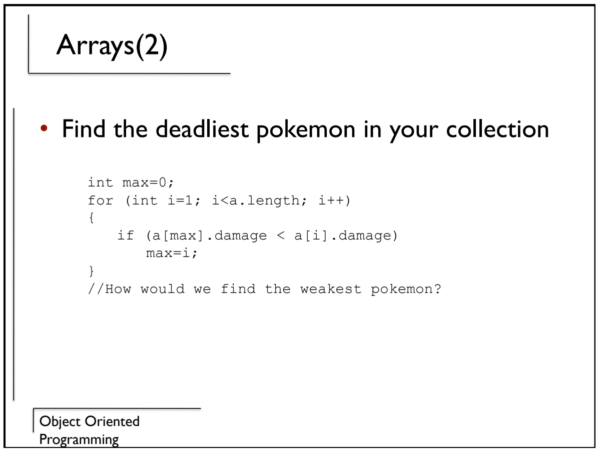


Arrays

- Gotta catch em all (Or how to go from one annoying furball to a 1000)

```
public static void main(String[] args)
{
   Example funcs = new Example();
   Scanner scan = new Scanner(System.in);
   Pokemon[] a = new Pokemon[1000];
   for (int i=0; i<a.length; i++)
   {
      a[i] = funcs.create(scan);
   }
}
```

Object Oriented Programming

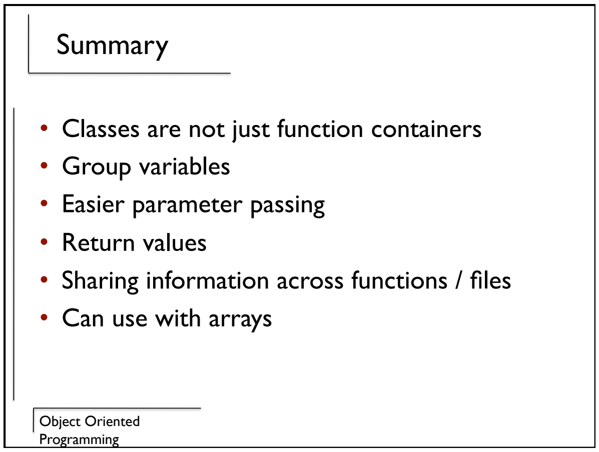


Arrays(2)

- Find the deadliest pokemon in your collection

```
int max=0;
for (int i=1; i<a.length; i++)
{
   if (a[max].damage < a[i].damage)
      max=i;
}
//How would we find the weakest pokemon?
```

Object Oriented Programming



Summary

- Classes are not just function containers
- Group variables
- Easier parameter passing
- Return values
- Sharing information across functions / files
- Can use with arrays

Object Oriented
Programming