

Miscellaneous Topics

A "but wait, there's more!" presentation

Random Vocab Words

- OOP
 - Object Oriented Programming
- Encapsulation
 - connecting variables with methods in a class
- Coupling
 - level of dependency between classes
- Cohesion
 - how "focused" code is

Integer Limits

- The Integer class defines two constants for the maximum and minimum int

Integer.MAX_VALUE

Integer.MIN_VALUE

```
System.out.println("The biggest int is: " + Integer.MAX_VALUE);
```

Integer Limit Use

- Finding the biggest integer in a list

```
public int biggestInt(List<Integer> list)
{
    int biggest = Integer.MIN_VALUE;
    for(int x: list)
        if(x > biggest)
            biggest = x;
    return biggest;
}
```

Sharing Data

- Let's say we want to keep track of how many BoxBug's we make...

```
public class BoxBug extends Bug
{
    private int boxBugCount = 0;

    public BoxBug(){
        boxBugCount++;
    }

    public int getCount() {
        return boxBugCount;
    }
}
```

Uh oh!

- Each BoxBug gets their own count variable

```
public static void main(String[] args)
{
    BoxBug alice = new BoxBug();
    BoxBug bob = new BoxBug();
    BoxBug clark = new BoxBug();

    System.out.println(clark.getCount()); // 1
}
```

Solution: static variable

```
public class BoxBug extends Bug
{
    private static int boxBugCount = 0;

    public BoxBug(){
        boxBugCount++;
    }

    public static int getCount() {
        return boxBugCount;
    }
}
```

Static == Shared

- Each BoxBug shares the same count variable!

```
public static void main(String[] args)
{
    BoxBug alice = new BoxBug();
    BoxBug bob = new BoxBug();
    BoxBug clark = new BoxBug();

    System.out.println(BoxBug.getCount()); //3
}
```

static methods are usually referenced by the **class** name

Static Variables

- Shared between all instances of a class
- Also called Class variables

One Last Thing

- Let's say we want to make two classes that have references to each other
 - A student has a way to contact a teacher
 - A teacher has a way to contact a student

A little code

```
public static void main(String[] args)
{
    Student stud = new Student();
    Teacher teach = new Teacher(stud);
}
```

```
public class Student
{
    private Teacher t;

    public Student() {
    }
}
```

```
public class Teacher
{
    private Student s;

    public Teacher(Student theS) {
        s = theS
        //need to connect student
        //to teacher!
    }
}
```

Add a setter!

```
public static void main(String[] args)
{
    Student stud = new Student();
    Teacher teach = new Teacher(stud);
}
```

```
public class Student
{
    private Teacher t;

    public Student() {
    }
}
```

```
public void setTeacher(Teacher theT) {
    t = theT;
}
}
```

```
public class Teacher
{
    private Student s;
```

```
public Teacher(Student theS) {
    s = theS
    //how to call setTeacher?
}
}
```

Add a setter!

```
public static void main(String[] args)
{
    Student stud = new Student();
    Teacher teach = new Teacher(stud);
}
```

```
public class Student
{
    private Teacher t;

    public Student() {
    }
}
```

```
public void setTeacher(Teacher theT) {
    t = theT;
}
}
```

```
public class Teacher
{
    private Student s;
```

```
public Teacher(Student theS) {
    s = theS
    s.setTeacher(??????);
}
}
```

Add a setter!

```
public static void main(String[] args)
{
    Student stud = new Student();
    Teacher teach = new Teacher(stud);
}
```

```
public class Student
{
    private Teacher t;

    public Student() {
    }
}
```

```
public void setTeacher(Teacher theT) {
    t = theT;
}
}
```

```
public class Teacher
{
    private Student s;
```

```
public Teacher(Student theS) {
    s = theS
    s.setTeacher(this);
}
}
```

Refers to the
current Teacher



This Again...!

- Do you remember this code?

```
Clock you = new Clock(12, 30);  
Clock me = new Clock(other);
```

```
public Clock(Clock other) {  
    hours = other.hours;  
    minutes = other.minutes;  
}
```

What's the name
of **you** in this
constructor?

This Again...!

- Do you remember this code?

```
Clock you = new Clock(12, 30);  
Clock me = new Clock(other);
```

```
public Clock(Clock other) {  
    hours = other.hours;  
    minutes = other.minutes;  
}
```

What's the name
of **you** in this
constructor?

other

This Again...!

- Do you remember this code?

```
Clock you = new Clock(12, 30);  
Clock me = new Clock(other);
```

```
public Clock(Clock other) {  
    hours = other.hours;  
    minutes = other.minutes;  
}
```

What's the name
of **me** in this
constructor?

This Again...!

- Do you remember this code?

```
Clock you = new Clock(12, 30);  
Clock me = new Clock(other);
```

```
public Clock(Clock other) {  
    hours = other.hours;  
    minutes = other.minutes;  
}
```

What's the name
of **me** in this
constructor?

this

This is another common use of this

```
public class Clock
{
    private int hours, minutes;

    public Clock(Clock other) {
        this.hours = other.hours;
        this.minutes = other.minutes;
    }
}
```

Our variables always have someone they belong too (how sweet)

Other place you might see this

```
public class Clock
{
    private int hours, minutes;

    public Clock(int hours, int minutes) {
        this.hours = hours;
        this.minutes = minutes;
    }
}
```

Distinguishes between instance fields and
parameters with the same name