Review of Assignments

- Sudoku – subsquares
- More compile errors than previous assignments
- Many logic errors - pseudocode, trace variables
- Several errors in output

Array variables are references

```
int [ ] a = new int[4];
```

Homogeneous Collections

Object Oriented Programming

Chapter 8
November 8, 2010
Heterogeneous Collections

With arrays, we could refer to an element of the array by an index.

How can we refer to the variables inside an object?

Class Definition (Version 1)

```java
public class BigBird{
    String name = "Big Bird";
    int age = 6;
    double height = 8.167;
    String color = "yellow";
}
```

Instantiate

```java
BigBird bb = new BigBird();
```
**Instantiate Objects**

```
BigBird
  name
  "Big Bird"
  age
  6
  height
  8.167
  color
  "yellow"

new BigBird();

bb1
  name
  "Big Bird"
  age
  6
  height
  8.167
  color
  "yellow"

new BigBird();

bb2
  name
  "Big Bird"
  age
  6
  height
  8.167
  color
  "yellow"

new BigBird();

bb3
  name
  "Big Bird"
  age
  6
  height
  8.167
  color
  "yellow"

new BigBird();
```

**Class Definition (Version 2)**

```java
Muppet.java

class Muppet{
  String name;
  int age;
  double height;
  String color;

  public Muppet(String n, int a, double h, String c){
    name = n;
    age = a;
    height = h;
    color = c;
  }
}
```

**Instantiate: Big Bird**

```
Muppet bb = new Muppet("Big Bird", 6, 8.167, "yellow");
```

**Instantiate: Oscar the Grouch**

```
Muppet oscar = new Muppet("Oscar", 43, 3.85, "green");
```

http://muppet.wikia.com/wiki/Oscar_the_Grouch
**Access to Fields**

- bb
  - name: "Big Bird"
  - age: 6
  - height: 8.167
  - color: "yellow"

**Direct Access to Fields**

(Not good Object Oriented Design!)

```
Muppet bb = new Muppet("Big Bird", 6, 8.167, "yellow");
System.out.println(bb.name + " is " + bb.age + " years old.");
System.out.println(bb.name + " is a " + bb.color +" muppet that is " + bb.height + " inches tall.");
```

Big Bird is 6 years old.
Big Bird is a yellow muppet that is 8.167 inches tall.

**Object Oriented Design**

"**Object-oriented** design is the process of planning a system of interacting objects for the purpose of solving a software problem."

(http://en.wikipedia.org/wiki/Object-oriented_design)

- Collection of heterogeneous objects
- Collection of state and behavior
- Encapsulation

**Class Definition (Version 3)**

```
Muppet.java

class Muppet{
    String name;
    int age;
    double height;
    String color;
    public Muppet(String name, int age, double height, String color){
        this.name = name;
        this.age = age;
        this.height = height;
        this.color = color;
    }
    public String getName(){
        return name;
    }
}
```
Indirect Access to Fields
(Closer, but still not good Object Oriented Design!)

```java
Muppet bb = new Muppet("Big Bird", 6, 8.167, "yellow");
System.out.println(bb.getName() + " is " + bb.age + " years old.");
System.out.println(bb.getName() + " is a "+ bb.color + " muppet that is " + bb.height + " inches tall.");
```

Variable Access
(Halfway there)

```java
Muppet bb = new Muppet("Big Bird", 6, 8.167, "yellow");
System.out.println(bb.ageToString());
System.out.println(bb.getName() + " is a "+ bb.color + " muppet that is " + bb.height + " inches tall.");
```

Class Definition (Version 4)

```java
Muppet.java
class Muppet{
  String name; int age;
  double height;
  String color;
  public Muppet(String n, int a, double h, String c){
    name = n;
    age = a;
    height = h;
    color = c;
  }
  public String ageToString(){
    return "age" + age + " years old.";
  }
  public String descriptionString(){
    return "description" + description + " muppet that is " + height + " inches tall.";
  }
}
```

Class Definition (Version 5)

```java
class Muppet{
  String name; int age;
  double height;
  String color;
  public Muppet(String n, int a, double h, String c){
    name = n;
    age = a;
    height = h;
    color = c;
  }
  public String ageToString(){
    return "age" + age + " years old.";
  }
  public String descriptionString(){
    return "description" + description + " muppet that is " + height + " inches tall.";
  }
}
```
Access to Fields through Methods

```java
Muppet bb = new Muppet("Big Bird", 6, 8.167, "yellow");
System.out.println(bb.ageToString());
System.out.println(bb.descriptionString());
```

Change to the output format

Instead, let’s say “The <color> muppet, <name>, is <height> inches tall.”

```java
Muppet bb = new Muppet("Big Bird", 6, 8.167, "yellow");
System.out.println(bb.name + " is " + bb.age + " years old.");
System.out.println("muppet that is " + bb.height + " inches tall.");
```

Changes effect all Objects

```java
public String descriptionString(){
    return name + " is a " + color + " muppet that is " + height + " inches tall.";
}
```

Encapsulation

```java
class Muppet{
    private String name;
    private int age;
    private double height;
    private String color;

    public Muppet(String n, int a, double h, String c){
        name = n;
        age = a;
        height = h;
        color = c;
    }
}
```
Access to Private Variables

```java
Muppet oscar = new Muppet("Oscar", 43, 3.85, "green");
System.out.println(oscar.name + " is " + oscar.age + " years old.");
```

Mutators

```java
public void setColor(String c){
    color = c;
}
```

```java
public void incrementAge(){
    age = age + 1;
}
```

Keyword this

```java
class Muppet{
    String name;

    public Muppet(String name){
        this.name = name;
    }
}
```

Variable that belongs to the object
Parameter from the method call