**News**

- Midterm 2: Thu Mar 16, 6:30pm (TODAY!)
- Woodward 2
- hour-long exam, reserve 6:30-8 time slot
  - for buffer in case of fire alarms etc
- no labs/tutorials this week
  - but one TA will be in lab during normal lab hours to answer questions

**Reading**

- This week: 9.3-9.4, 9.6-9.8
- Next week: 11.1-11.3

**Recap: Bunnies**

- Bunny.java
  - int x
  - int y
  - int numCarrots
  + Bunny()
  + hop(int direction)
  + displayInfo()

- NamedBunny.java

  + Bunny(int x, int y, int numCarrots, String name)

**Even More Bunnies**

**Question 5: [16 marks]**

The world desperately needs better bunny management software, so please help by writing a BunnyHerd class. A BunnyHerd object holds an array of Bunny objects. Your BunnyHerd class definition should include the following four methods:

- **constructor**
  - Expects two parameters, an integer representing the maximum number of bunnies in the herd, and a String for the name of the herd.

- **addBunny(int xPos, int yPos, int carrots, String name)**
  - Expects four parameters, the X- and Y-coordinates of the bunny, the number of carrots, and the name. This method creates a new Bunny object and stores the reference to the object in the next available location in the BunnyHerd object.

- **deleteBunny(String name)**
  - Expects one parameter, the name of the bunny. This method removes from the BunnyHerd object all references to bunnies with the given name by overwriting those references with the null pointer. This method does not change the pointer to the next available location in the BunnyHerd object.

- **printHerd()**
  - This method uses the toString() method of the Bunny object to print information about every Bunny in the herd.
Parameter Passing

Consider the following program:

```java
public class ParamTest1
{
    public static void main (String[] args)
    {
        int number = 4;
        System.out.println("main: number is " + number);
        method(number);
        System.out.println("main: number is now " + number);
    }
    public static void method(int x)
    {
        System.out.println("method: x is " + x);
        x = x * x;
        System.out.println("method: x is now " + x);
    }
}
```

What's the flow of control?
Consider the following program:

```java
public class ParamTest1 {
    public static void main(String[] args) {
        int number = 4;
        System.out.println("main: number is " + number);
    }
}
```

What's the flow of control?

What's printed?
Parameter Passing

Consider the following program:

```java
class ParamTest1 {
    public static void main (String[] args) {
        int number = 4;
        System.out.println("main: number is " + number);
        method(number);
        System.out.println("main: number is now " + number);
    }
}
```

What's printed? main: number is 4
method: x is 4
method: x is now 4

Parameter Passing

Consider the following program:

```java
class ParamTest1 {
    public static void main (String[] args) {
        int number = 4;
        System.out.println("main: number is " + number);
        method(number);
        System.out.println("main: number is now " + number);
    }
    public static void method(int x) {
        System.out.println("method1: x is " + x);
        x = x * x;
        System.out.println("method1: x is now " + x);
    }
}
```

What's printed? main: number is 4
method: x is 4
method: x is now 16

Parameter Passing

Consider the following program:

```java
class ParamTest1 {
    public static void main (String[] args) {
        int number = 4;
        System.out.println("main: number is " + number);
        method(number);
        System.out.println("main: number is now " + number);
    }
    public static void method(int x) {
        System.out.println("method1: x is " + x);
        x = x * x;
        System.out.println("method1: x is now " + x);
    }
}
```

Why not 16? main: number is 4
method: x is 4
method: x is now 16

Parameter Passing

Consider the following program:

```java
class ParamTest1 {
    public static void main (String[] args) {
        int number = 4;
        System.out.println("main: number is " + number);
        method(number);
        System.out.println("main: number is now " + number);
    }
    public static void method(int x) {
        System.out.println("method1: x is " + x);
        x = x * x;
        System.out.println("method1: x is now " + x);
    }
}
```

Because when the value in the int variable number is passed to method1,
Parameter Passing

Consider the following program:

```java
public class ParamTest1{
  public static void main (String[] args) {
    int number = 4;
    System.out.println("main: number is "+number);
    method1(number);
    System.out.println("main: number is now "+number);
  }
}
public class ParamTest2 {
  public static void main (String[] args) {
    int number = 4;
    System.out.println("main: number is "+number);
    method1(number);
    System.out.println("main: number is now "+number);
  }
}
```

Because when the value in the int variable `number` is passed to `method1`, what really happens is that a copy of the value (4) in `number` is assigned to the parameter `x`. It's the value in `x` that's being modified here — a copy of the value in `number`. The original value in `number` is not affected.

Parameter Passing

Will this program behave differently? Why or why not?

```java
public class ParamTest2 {
  public static void main (String[] args) {
    int number = 4;
    System.out.println("main: number is "+number);
    method1(number);
    System.out.println("main: number is now "+number);
  }
}
```

What's printed?

```
main: number is 4
method1: number is 4
method1: number is now 16
```

Parameter Passing

Consider the following program:

```java
public class ParamTest1{
  public static void main (String[] args) {
    int x[0] = 4;
    System.out.println("foo ");
    System.out.println("x[0] "+x[0]);
  }
}
```

What's printed?

```
foo
x[0] 4
```

Parameter Passing

Now consider this program.

```java
public class Ptest{  
  public static void main(String[] args)  
  {    
      int x[0] = 4;    
      System.out.println("x[0] "+x[0]);    
      x[0] = x[0] * x[0];    
      System.out.println("method1: x is now: "+x[0]);  
  }  
}
```

What's printed?

```
method1: x is now: 16
```
Parameter Passing

Now consider this program.

```java
class Ptest {
    public static void main(String[] args) {
        int[] foo = new int[1];
        foo[0] = 4;
        System.out.println("main: foo is now: " + foo[0]);
        method1(foo);
    }
    public static void method1(int[] x) {
        System.out.println("method1: x is now: " + x[0]);
        x[0] = x[0] * x[0];
        System.out.println("method1: x is now: " + x[0]);
    }
    public static void method2(int[] x) {
        System.out.println("method2: x is now: " + x[0]);
    }
}
```

What's printed?

```
main: foo is now: 4
method1: x is now: 4
method1: x is now: 16
```

Parameter Passing

Now consider this program.

```java
class Ptest {
    public static void main(String[] args) {
        int[] foo = new int[1];
        foo[0] = 4;
        System.out.println("main: foo is now: " + foo[0]);
        method1(foo);
    }
    public static void method1(int[] x) {
        System.out.println("method1: x is now: " + x[0]);
        x[0] = x[0] * x[0];
        System.out.println("method1: x is now: " + x[0]);
    }
    public static void method2(int[] x) {
        System.out.println("method2: x is now: " + x[0]);
    }
    public static void method3(int[] x) {
        System.out.println("method3: x is now: " + x[0]);
    }
}
```

What's printed?

```
main: foo is now: 4
method1: x is now: 4
method1: x is now: 16
method2: x is now: 4
method3: x is now: 16
```

Parameter Passing

Now consider this program.

```java
class Ptest {
    public static void main(String[] args) {
        int[] foo = new int[1];
        foo[0] = 4;
        System.out.println("main: foo is now: " + foo[0]);
        method1(foo);
    }
    public static void method1(int[] x) {
        System.out.println("method1: x is now: " + x[0]);
        x[0] = x[0] * x[0];
        System.out.println("method1: x is now: " + x[0]);
    }
    public static void method2(int[] x) {
        System.out.println("method2: x is now: " + x[0]);
    }
}
```

What's printed?

```
main: foo is now: 4
method1: x is now: 4
method1: x is now: 16
method2: x is now: 4
```

Parameter Passing

Now consider this program.

```java
class Ptest {
    public static void main(String[] args) {
        int[] foo = new int[1];
        foo[0] = 4;
        System.out.println("main: foo is now: " + foo[0]);
        method1(foo);
    }
    public static void method1(int[] x) {
        System.out.println("method1: x is now: " + x[0]);
        x[0] = x[0] * x[0];
        System.out.println("method1: x is now: " + x[0]);
    }
    public static void method2(int[] x) {
        System.out.println("method2: x is now: " + x[0]);
    }
    public static void method3(int[] x) {
        System.out.println("method3: x is now: " + x[0]);
    }
}
```

What's printed?

```
main: foo is now: 4
method1: x is now: 4
method1: x is now: 16
method2: x is now: 4
method3: x is now: 4
```

Parameter Passing

Now consider this program.

```java
class Ptest {
    public static void main(String[] args) {
        int[] foo = new int[1];
        foo[0] = 4;
        System.out.println("main: foo is now: " + foo[0]);
        method1(foo);
    }
    public static void method1(int[] x) {
        System.out.println("method1: x is now: " + x[0]);
        x[0] = x[0] * x[0];
        System.out.println("method1: x is now: " + x[0]);
    }
    public static void method2(int[] x) {
        System.out.println("method2: x is now: " + x[0]);
    }
    public static void method3(int[] x) {
        System.out.println("method3: x is now: " + x[0]);
    }
}
```

What's printed?

```
main: foo is now: 4
method1: x is now: 4
method1: x is now: 16
method2: x is now: 4
method3: x is now: 4
```

Parameter Passing

Now consider this program.

```java
class Ptest {
    public static void main(String[] args) {
        int[] foo = new int[1];
        foo[0] = 4;
        System.out.println("main: foo is now: " + foo[0]);
        method1(foo);
    }
    public static void method1(int[] x) {
        System.out.println("method1: x is now: " + x[0]);
        x[0] = x[0] * x[0];
        System.out.println("method1: x is now: " + x[0]);
    }
    public static void method2(int[] x) {
        System.out.println("method2: x is now: " + x[0]);
    }
    public static void method3(int[] x) {
        System.out.println("method3: x is now: " + x[0]);
    }
}
```

What's printed?

```
main: foo is now: 4
method1: x is now: 4
method1: x is now: 16
method2: x is now: 4
method3: x is now: 4
```

Parameter Passing

Now consider this program.

```java
class Ptest {
    public static void main(String[] args) {
        int[] foo = new int[1];
        foo[0] = 4;
        System.out.println("main: foo is now: " + foo[0]);
        method1(foo);
    }
    public static void method1(int[] x) {
        System.out.println("method1: x is now: " + x[0]);
        x[0] = x[0] * x[0];
        System.out.println("method1: x is now: " + x[0]);
    }
    public static void method2(int[] x) {
        System.out.println("method2: x is now: " + x[0]);
    }
    public static void method3(int[] x) {
        System.out.println("method3: x is now: " + x[0]);
    }
}
```

What's printed?

```
main: foo is now: 4
method1: x is now: 4
method1: x is now: 16
method2: x is now: 4
method3: x is now: 4
```
Parameter Passing

Now consider this program.

```java
public class Ptest {
    public static void main(String[] args) {
        int[] foo = new int[1];
        foo[0] = 4;
        System.out.println("main: foo is now: "+ foo[0]);
        method(foo);
        System.out.println("main: foo is now: "+ foo[0]);
    }
}
```

```java
public static void method(int[] x) {
    System.out.println("method: x is now: "+ x[0]);
    x[0] = x[0] + x[0];
    System.out.println("method: x is now: "+ x[0]);
}
```

What's in foo? Is it the int[] array object? No, it's the reference, or pointer, to the object.

Parameter Passing

Now consider this program.

```java
public class Ptest {
    public static void main(String[] args) {
        int[] foo = new int[1];
        foo[0] = 4;
        System.out.println("main: foo is now: "+ foo[0]);
        method(foo);
        System.out.println("main: foo is now: "+ foo[0]);
    }
}
```

```java
public static void method(int[] x) {
    System.out.println("method: x is now: "+ x[0]);
    x[0] = x[0] + x[0];
    System.out.println("method: x is now: "+ x[0]);
}
```

What's in foo? Is it the int[] array object? No, it's the reference, or pointer, to the object.

Parameter Passing

Now consider this program.

```java
public class Ptest {
    public static void main(String[] args) {
        int[] foo = new int[1];
        foo[0] = 4;
        System.out.println("main: foo is now: "+ foo[0]);
        method(foo);
        System.out.println("main: foo is now: "+ foo[0]);
    }
}
```

```java
public static void method(int[] x) {
    System.out.println("method: x is now: "+ x[0]);
    x[0] = x[0] + x[0];
    System.out.println("method: x is now: "+ x[0]);
}
```

What's in foo? Is it the int[] array object? No, it's the reference, or pointer, to the object. A copy of that reference is passed to method and assigned to x. The reference in foo and the reference in x both point to the same object.

Parameter Passing

Now consider this program.

```java
public class Ptest {
    public static void main(String[] args) {
        int[] foo = new int[1];
        foo[0] = 4;
        System.out.println("main: foo is now: "+ foo[0]);
        method(foo);
        System.out.println("main: foo is now: "+ foo[0]);
    }
}
```

```java
public static void method(int[] x) {
    System.out.println("method: x is now: "+ x[0]);
    x[0] = x[0] + x[0];
    System.out.println("method: x is now: "+ x[0]);
}
```

What's in foo? Is it the int[] array object? No, it's the reference, or pointer, to the object. A copy of that reference is passed to method and assigned to x. The reference in foo and the reference in x both point to the same object.

Parameter Passing

- Passing primitive types (int, double, boolean) as parameter in Java
  - "pass by value"
  - value in variable is copied
  - copy is passed to method
  - modifying copy of value inside called method has no effect on original value outside called method
  - modifying aka mutating

Parameter Passing

Now consider this program.

```java
public class Ptest {
    public static void main(String[] args) {
        int[] foo = new int[1];
        foo[0] = 4;
        System.out.println("main: foo is now: "+ foo[0]);
        method(foo);
        System.out.println("main: foo is now: "+ foo[0]);
    }
}
```

```java
public static void method(int[] x) {
    System.out.println("method: x is now: "+ x[0]);
    x[0] = x[0] + x[0];
    System.out.println("method: x is now: "+ x[0]);
}
```

What's in foo? Is it the int[] array object? No, it's the reference, or pointer, to the object. A copy of that reference is passed to method and assigned to x. The reference in foo and the reference in x both point to the same object.

Parameter Passing

Now consider this program.

```java
public class Ptest {
    public static void main(String[] args) {
        int[] foo = new int[1];
        foo[0] = 4;
        System.out.println("main: foo is now: "+ foo[0]);
        method(foo);
        System.out.println("main: foo is now: "+ foo[0]);
    }
}
```

```java
public static void method(int[] x) {
    System.out.println("method: x is now: "+ x[0]);
    x[0] = x[0] + x[0];
    System.out.println("method: x is now: "+ x[0]);
}
```

What's in foo? Is it the int[] array object? No, it's the reference, or pointer, to the object. A copy of that reference is passed to method and assigned to x. The reference in foo and the reference in x both point to the same object.

Parameter Passing

Now consider this program.

```java
public class Ptest {
    public static void main(String[] args) {
        int[] foo = new int[1];
        foo[0] = 4;
        System.out.println("main: foo is now: "+ foo[0]);
        method(foo);
        System.out.println("main: foo is now: "+ foo[0]);
    }
}
```

```java
public static void method(int[] x) {
    System.out.println("method: x is now: "+ x[0]);
    x[0] = x[0] + x[0];
    System.out.println("method: x is now: "+ x[0]);
}
```

What's in foo? Is it the int[] array object? No, it's the reference, or pointer, to the object. A copy of that reference is passed to method and assigned to x. The reference in foo and the reference in x both point to the same object.

Parameter Passing

Now consider this program.

```java
public class Ptest {
    public static void main(String[] args) {
        int[] foo = new int[1];
        foo[0] = 4;
        System.out.println("main: foo is now: "+ foo[0]);
        method(foo);
        System.out.println("main: foo is now: "+ foo[0]);
    }
}
```

```java
public static void method(int[] x) {
    System.out.println("method: x is now: "+ x[0]);
    x[0] = x[0] + x[0];
    System.out.println("method: x is now: "+ x[0]);
}
```

What's in foo? Is it the int[] array object? No, it's the reference, or pointer, to the object. A copy of that reference is passed to method and assigned to x. The reference in foo and the reference in x both point to the same object.
Parameter Passing

- Passing object as parameter in Java
  - "pass by reference"
- Objects could be huge, so do not pass copies around
- Pass copy of the object reference
- Object reference aka pointer
- Modifying object pointed to by reference inside calling method does affect object pointed to by reference outside calling method
- Both references point to same object

Parameter Passing Pictures

Object as parameter: copy of pointer made

Prim as parameter: copy of value

Midterm Q4 from 04W2

```java
public void process( int[][] arrA, int[][] arrB ) {   int row;   int col;   int[][] arrC = new int[1][1];   arrC[0][0] = arrB[0][0];   for( row = 0; row < arrB.length; row++ ) {     for( col = 0; col < arrB[row].length; col++ ) {       arrB[row][col] = row + col;     }   }
}
```

Midterm Q4 from 04W2

```java
public void process( int[][] arrA, int[][] arrB ){   int row;   int col;   int[][] arrC = new int[1][1];   arrC[0][0] = arrB[0][0];   for( row = 0; row < arrB.length; row++ ) {     for( col = 0; col < arrB[row].length; col++ ) {       arrB[row][col] = row + col;     }   }
}
```

Midterm Q4 from 04W2

```java
public void process( int[][] arrA, int[][] arrB ){   int row;   int col;   int[][] arrC = new int[2][2];   for( row = 0; row < arrB.length; row++ ) {     for( col = 0; col < arrB[row].length; col++ ) {       arrB[row][col] = row + col;     }   }
}
```
Review: Static Fields/Methods

- Static fields belong to whole class
- Static fields can only use static fields
- Static methods can use either nonstatic or static fields

Review: Variable Scope

- Scope of a variable (or constant) is that part of a program in which value of that variable can be accessed

Variable Scope

- numberOfCans variable declared inside class but not inside particular method
  - scope is entire class: can be accessed from anywhere in class

Variable Scope

- totalLitres declared within a method
  - scope is method: can only be accessed from within method
  - variable is local data: has local scope

Variable Scope

- loadedCans is method parameter
  - scope is method: also local scope
  - just like variable declared within parameter
  - accessed only within that method

Variable Types

- Static variables
  - declared within class
  - associated with class, not instance

- Instance variables
  - declared within class
  - associated with instance
  - accessible throughout object, lifetime of object

- Local variables
  - declared within method
  - accessible throughout method, lifetime of method

- Parameters
  - declared in parameter list of method
  - accessible throughout method, lifetime of method
Variable Types

- Static?
- Instance?
- Local?
- Parameters?

Questions?