University of British Columbia
CPSC 111, Intro to Computation
2009W2: Jan-Apr 2010
Tamara Munzner

More Class Design III, Parameter/Scope Review

Lecture 32, Wed Apr 7 2010

borrowing from slides by Kurt Eiselt

http://www.cs.ubc.ca/~tmm/courses/111-10

News

- you should already have a good start on A3
  - don't wait until the last minute, it's substantial
- reminder that pair programming can only be groups of 2 (not 3 or more)
- make sure to check your ugrad account email (or forward it) to see your detailed marking report for assignments
- inform me ASAP, by end of this week at the lastest, if you have a final exam conflict/hardship

News II

- update for the 20% assignment mark breakdown
  - three main assignments are each worth 6%, not 4% as the writeups say
  - all the weekly reading questions combined are worth 2%.
- tutorials now over for the term, except Friday Apr 9 makeup sessions for Apr 2 holiday cancellation.
- final review session will be Mon Apr 26 10am-12pm, room TBA.

News: Midterm Correction Lab

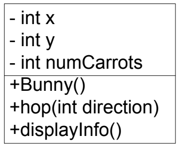
- you can earn **up to** 10% of marks that you missed back by working through what you got wrong to find correct answers
- do your new version on separate sheets of paper
  - don't mark up the original midterm
- as with all labs, if you don't finish during the time slot you can turn in at beginning of next week's lab
- pick up your midterm after class or in lab

Reading

- Last week was Chap 8
- This week is Chap 11, except 11.8.3
  - 2nd edition: Chap 13, except 13.8.3
- Weeklies due for last week either last Wed 3/31 or this Wed 4/7 (since no class Fri, Mon)
- This week's weekly due Fri as usual

Recap: Bunnies

- Bunny.java



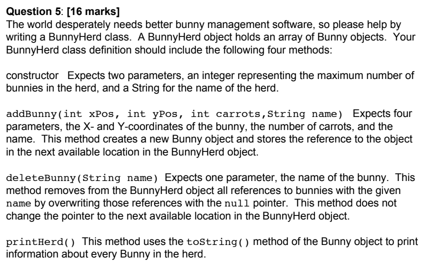
- int x
- int y
- int numCarrots

+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.

Even More Bunnies

- BunnyHerd.java

Bunnies and Interfaces

```
public interface Bunnies
{
  public void moveBunny(int direction);

}
```

Bunnies and Interfaces

```
public class BigBunny implements Bunnies
{
  private int x, y;
  private int carrots;

  public BigBunny()
  {
    x = 5;
    y = 5;
    carrots = 10;
  }

  public void moveBunny(int direction)
  {
    if (direction == 12)
    {
      y = y + 3;
      carrots = carrots - 2;
    }
```

Bunnies and Interfaces

```
    else if (direction == 3)
    {
      x = x + 3;
      carrots = carrots - 2;
    }
    else if (direction == 6)
    {
      y = y - 3;
      carrots = carrots - 2;
    }
    else if (direction == 9)
    {
      x = x - 3;
      carrots = carrots - 2;
    }
    else
    {
      System.out.println("Invalid direction");
    }
  }
}
```

Bunnies and Interfaces

```
public class LittleBunny implements Bunnies
{
  private int x, y;
  private int carrots;

  public LittleBunny()
  {
    x = 5;
    y = 5;
    carrots = 10;
  }

  public void moveBunny(int direction)
  {
    if (direction == 12)
    {
      y = y + 1;
      carrots = carrots - 1;
    }
```

Bunnies and Interfaces

```
    else if (direction == 3)
    {
      x = x + 1;
      carrots = carrots - 1;
    }
    else if (direction == 6)
    {
      y = y - 1;
      carrots = carrots - 1;
    }
    else if (direction == 9)
    {
      x = x - 1;
      carrots = carrots - 1;
    }
    else
    {
      System.out.println("Invalid direction");
    }
  }
}
```

Parameter Passing

Consider the following program:

```
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 static void method1(int x)
  {
    System.out.println("method1: x is " + x);
    x = x * x;
    System.out.println("method1: x is now " + x);
  }
}
```

Parameter Passing

Consider the following program:

```
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 static void method1(int x)
  {
    System.out.println("method1: x is " + x);
    x = x * x;
    System.out.println("method1: x is now " + x);
  }
}
```

What's the flow of control?

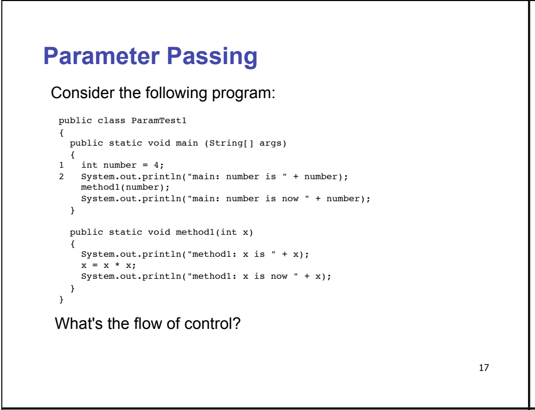
Parameter Passing

Consider the following program:

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

  public static void method1(int x)
  {
    System.out.println("method1: x is " + x);
    x = x * x;
    System.out.println("method1: x is now " + x);
  }
}
```

What's the flow of control?



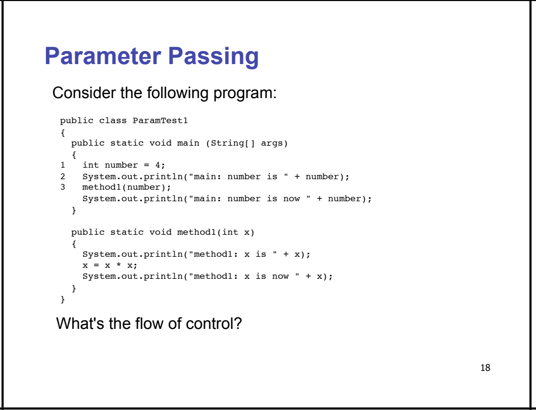
Parameter Passing

Consider the following program:

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

  public static void method1(int x)
  {
    System.out.println("method1: x is " + x);
    x = x * x;
    System.out.println("method1: x is now " + x);
  }
}
```

What's the flow of control?



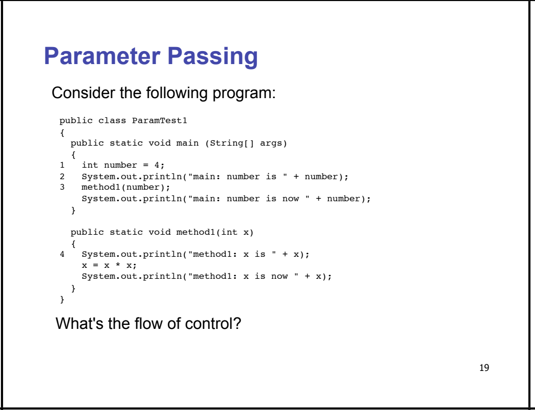
Parameter Passing

Consider the following program:

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

  public static void method1(int x)
  {
    System.out.println("method1: x is " + x);
    x = x * x;
    System.out.println("method1: x is now " + x);
  }
}
```

What's the flow of control?



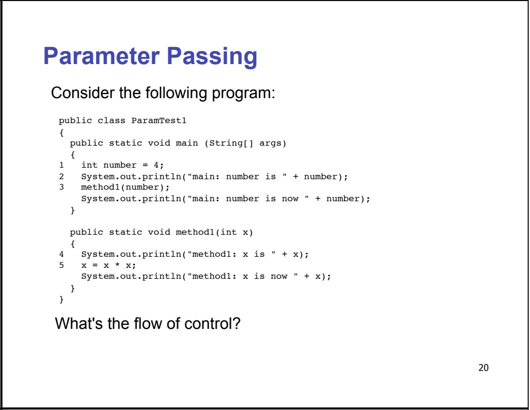
Parameter Passing

Consider the following program:

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

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
    x = x * x;
    System.out.println("method1: x is now " + x);
  }
}
```

What's the flow of control?



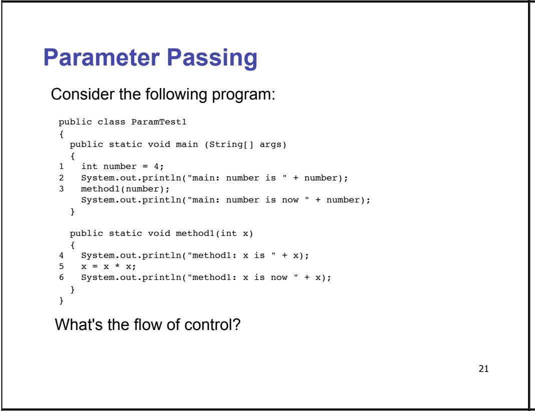
Parameter Passing

Consider the following program:

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

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
    System.out.println("method1: x is now " + x);
  }
}
```

What's the flow of control?



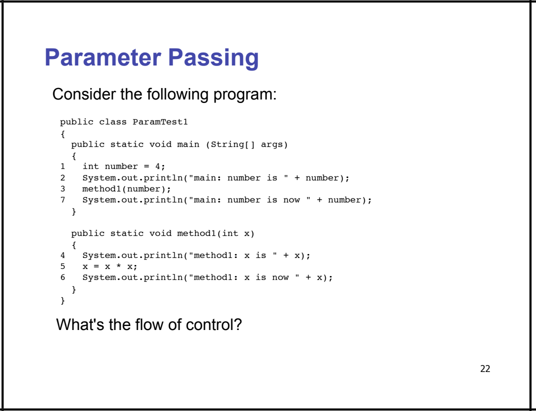
Parameter Passing

Consider the following program:

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

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

What's the flow of control?



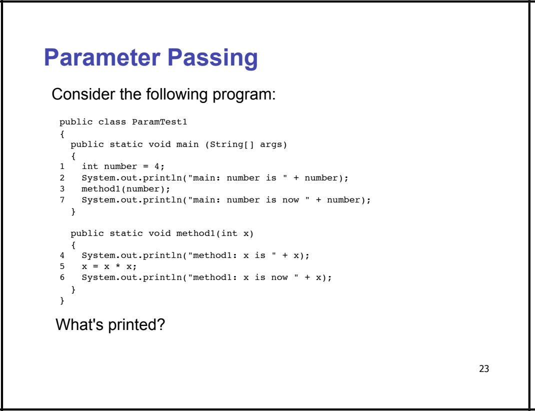
Parameter Passing

Consider the following program:

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

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

What's the flow of control?



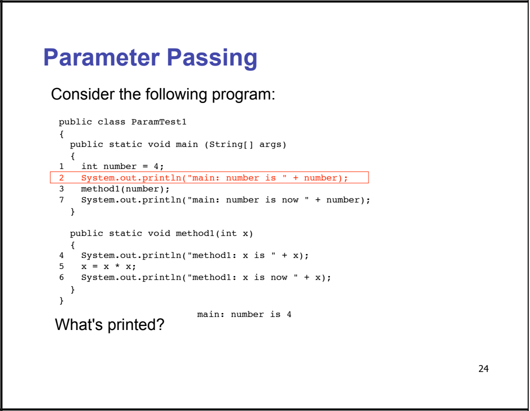
Parameter Passing

Consider the following program:

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

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

What's printed?



Parameter Passing

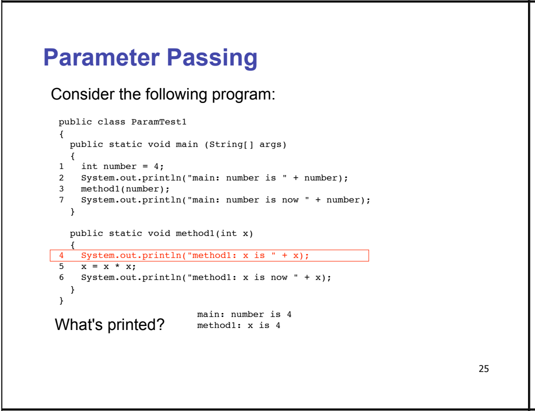
Consider the following program:

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

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

What's printed?

```
main: number is 4
```



Parameter Passing

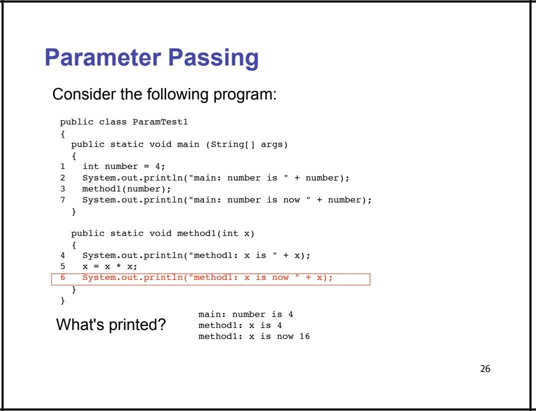
Consider the following program:

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

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

What's printed?

```
main: number is 4
method1: x is 4
```



Parameter Passing

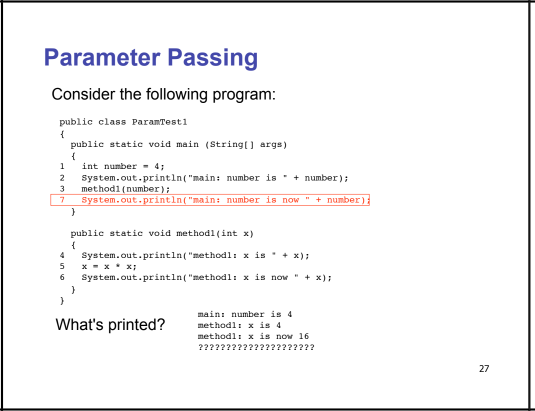
Consider the following program:

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

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

What's printed?

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



Parameter Passing

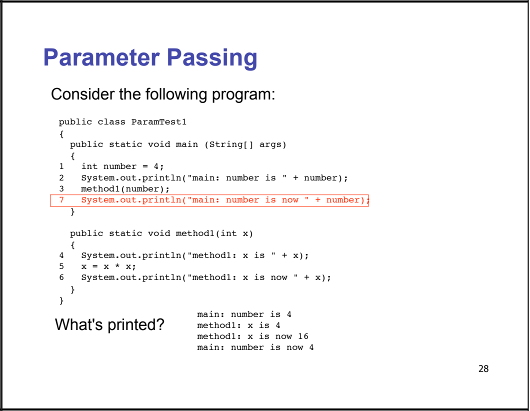
Consider the following program:

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

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

What's printed?

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



Parameter Passing

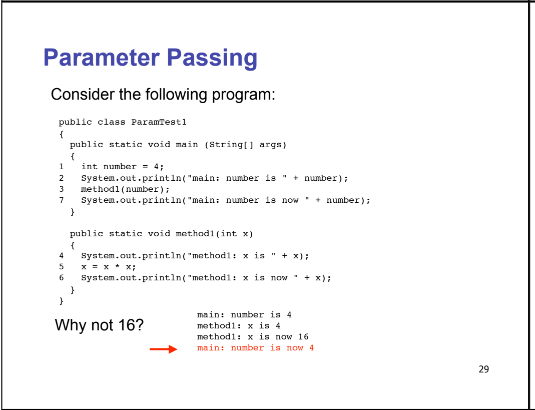
Consider the following program:

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

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

What's printed?

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



Parameter Passing

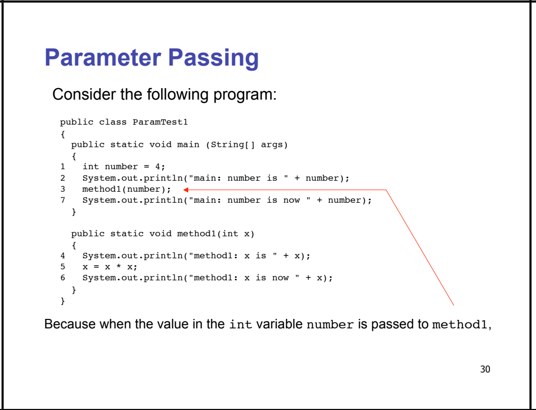
Consider the following program:

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

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

Why not 16?

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



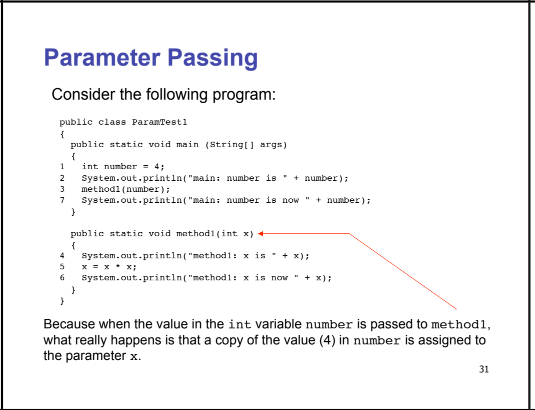
Parameter Passing

Consider the following program:

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

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

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



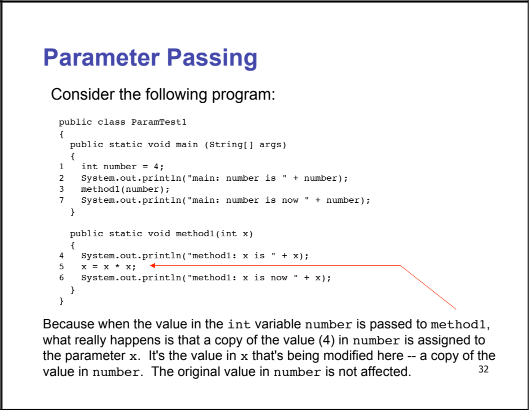
Parameter Passing

Consider the following program:

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

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

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`.



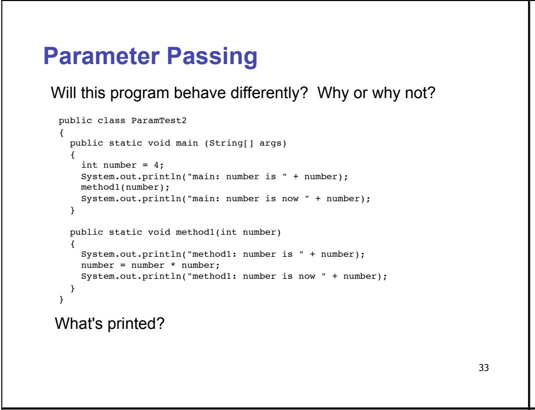
Parameter Passing

Consider the following program:

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

  public static void method1(int x)
  {
4   System.out.println("method1: x is " + x);
5   x = x * x;
6   System.out.println("method1: x is now " + x);
  }
}
```

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.



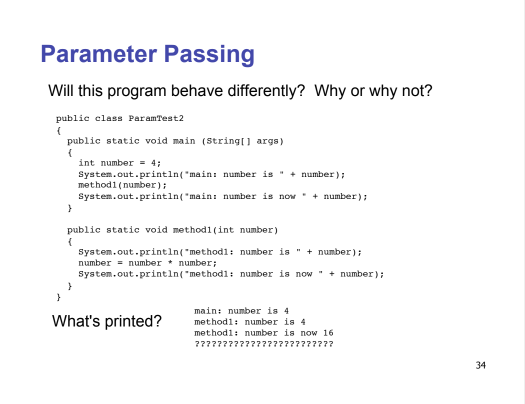
Parameter Passing

Will this program behave differently? Why or why not?

```
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);
  }

  public static void method1(int number)
  {
    System.out.println("method1: number is " + number);
    number = number * number;
    System.out.println("method1: number is now " + number);
  }
}
```

What's printed?



Parameter Passing

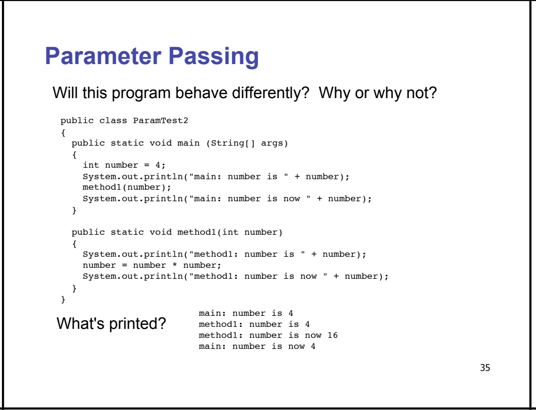
Will this program behave differently? Why or why not?

```
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);
  }

  public static void method1(int number)
  {
    System.out.println("method1: number is " + number);
    number = number * number;
    System.out.println("method1: number is now " + number);
  }
}
```

What's printed?

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



Parameter Passing

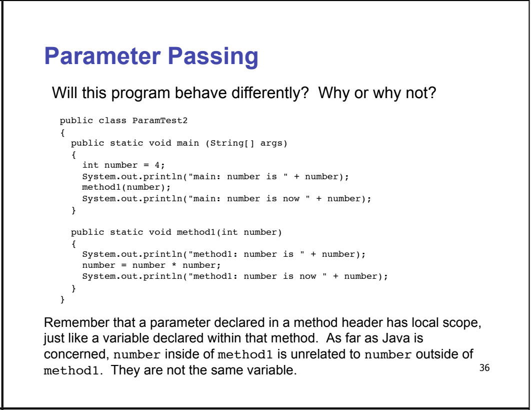
Will this program behave differently? Why or why not?

```
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);
  }

  public static void method1(int number)
  {
    System.out.println("method1: number is " + number);
    number = number * number;
    System.out.println("method1: number is now " + number);
  }
}
```

What's printed?

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



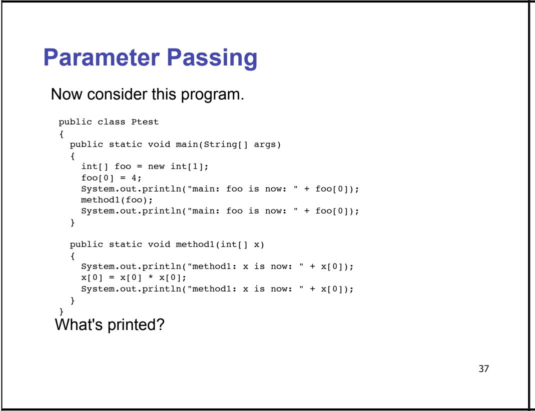
Parameter Passing

Will this program behave differently? Why or why not?

```
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);
  }

  public static void method1(int number)
  {
    System.out.println("method1: number is " + number);
    number = number * number;
    System.out.println("method1: number is now " + number);
  }
}
```

Remember that a parameter declared in a method header has local scope, just like a variable declared within that method. As far as Java is concerned, `number` inside of `method1` is unrelated to `number` outside of `method1`. They are not the same variable.



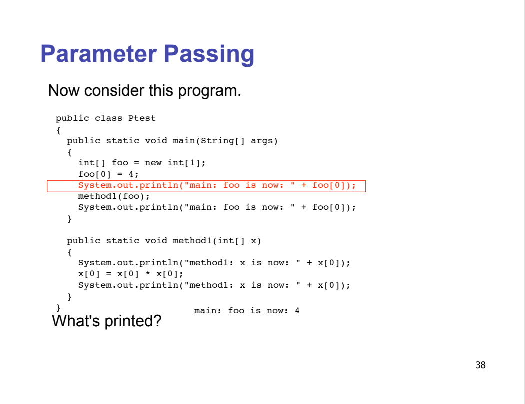
Parameter Passing

Now consider this program.

```
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]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  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]);
  }
}
```

What's printed?



Parameter Passing

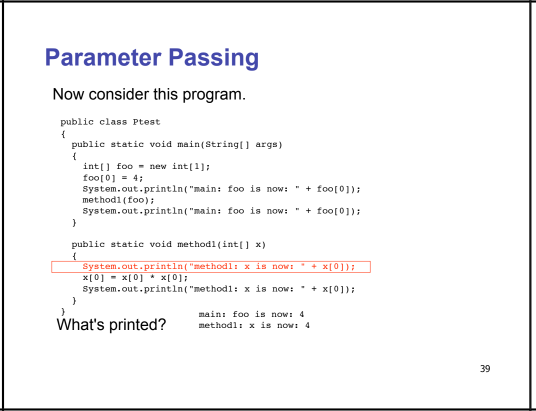
Now consider this program.

```
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]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  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]);
  }
}
```

What's printed?

```
main: foo is now: 4
```



Parameter Passing

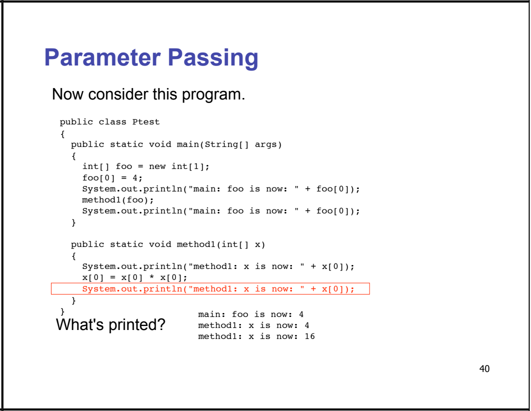
Now consider this program.

```
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]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  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]);
  }
}
```

What's printed?

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



Parameter Passing

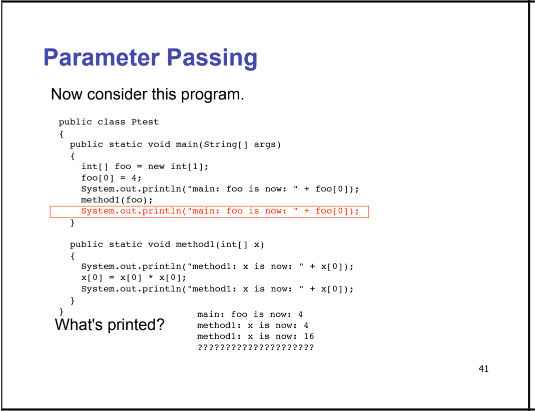
Now consider this program.

```
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]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  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]);
  }
}
```

What's printed?

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



Parameter Passing

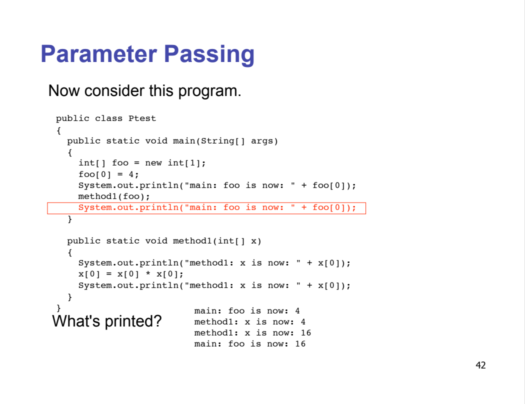
Now consider this program.

```
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]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  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]);
  }
}
```

What's printed?

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



Parameter Passing

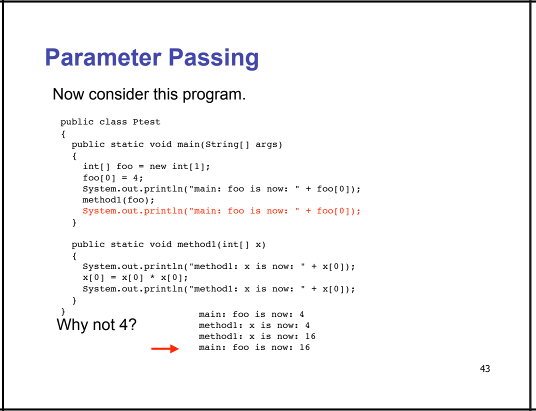
Now consider this program.

```
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]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  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]);
  }
}
```

What's printed?

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



Parameter Passing

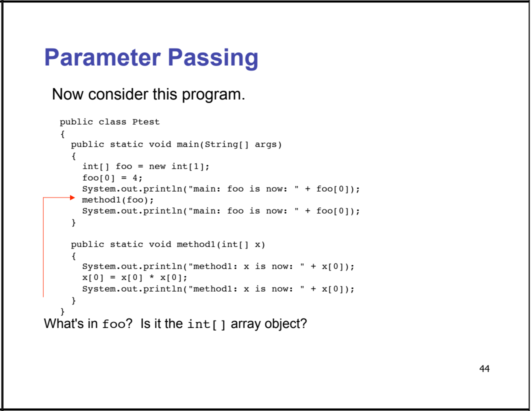
Now consider this program.

```
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]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  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]);
  }
}
```

Why not 4?

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



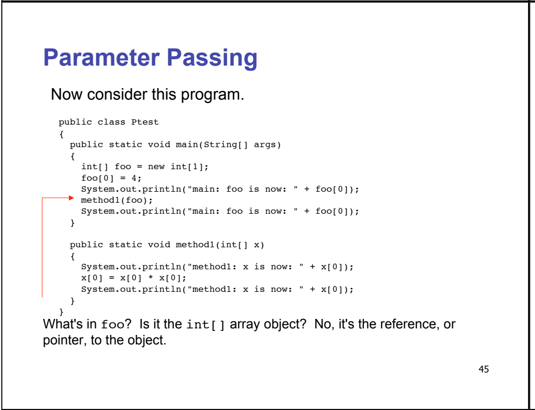
Parameter Passing

Now consider this program.

```
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]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  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]);
  }
}
```

What's in `foo`? Is it the `int[]` array object?



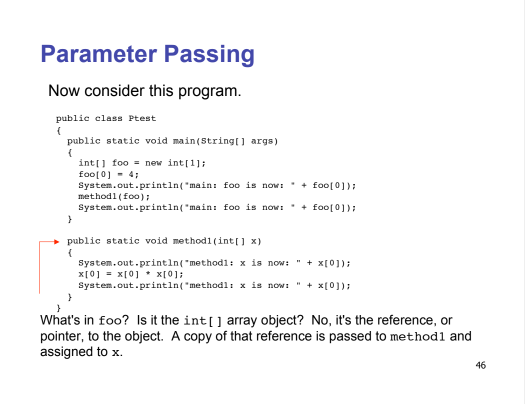
Parameter Passing

Now consider this program.

```
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]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  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]);
  }
}
```

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



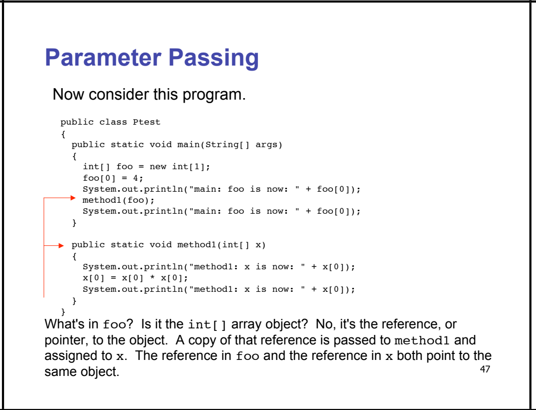
Parameter Passing

Now consider this program.

```
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]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  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]);
  }
}
```

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 `method1` and assigned to `x`.



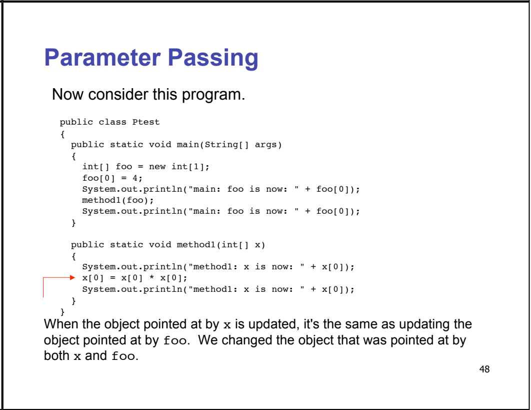
Parameter Passing

Now consider this program.

```
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]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  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]);
  }
}
```

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 `method1` 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.

```
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]);
    method1(foo);
    System.out.println("main: foo is now: " + foo[0]);
  }

  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]);
  }
}
```

When the object pointed at by `x` is updated, it's the same as updating the object pointed at by `foo`. We changed the object that was pointed at by both `x` and `foo`.

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

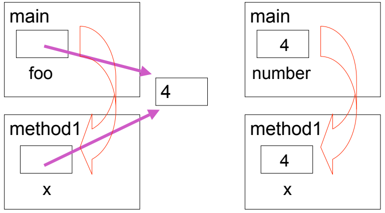
Parameter Passing

- 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

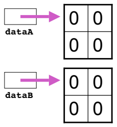
prim as parameter:
copy of value



Midterm Q4 from 04W2

```
int[][] dataA = { { 0, 0 }, { 0, 0 } };
int[][] dataB = { { 0, 0 }, { 0, 0 } };
process( dataA, dataB );
```

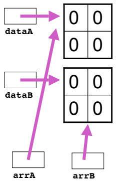
```
public void process( int[][] arrA, int[][] arrB )
{
   int row;
   int col;
   int[][] arrC = { { 1, 1, 1 }, { 1, 1, 1 } };
   arrA = arrC;
   for( row = 0; row < arrB.length; row++ )
   {
      for( col = 0; col < arrB[ row ].length; col++
   )
      {
         arrB[ row ][ col ] = row + col;
 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



Midterm Q4 from 04W2

```
int[][] dataA = { { 0, 0 }, { 0, 0 } };
int[][] dataB = { { 0, 0 }, { 0, 0 } };
process( dataA, dataB );
```

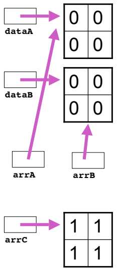
```
public void process( int[][] arrA, int[][] arrB )
{
   int row;
   int col;
   int[][] arrC = { { 1, 1, 1 }, { 1, 1, 1 } };
   arrA = arrC;
   for( row = 0; row < arrB.length; row++ )
   {
      for( col = 0; col < arrB[ row ].length; col++
   )
      {
         arrB[ row ][ col ] = row + col;
 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



Midterm Q4 from 04W2

```
int[][] dataA = { { 0, 0 }, { 0, 0 } };
int[][] dataB = { { 0, 0 }, { 0, 0 } };
process( dataA, dataB );
```

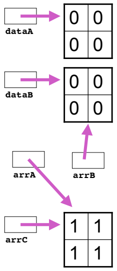
```
public void process( int[][] arrA, int[][] arrB )
{
   int row;
   int col;
   int[][] arrC = { { 1, 1, 1 }, { 1, 1, 1 } };
   arrA = arrC;
   for( row = 0; row < arrB.length; row++ )
   {
      for( col = 0; col < arrB[ row ].length; col++
   )
      {
         arrB[ row ][ col ] = row + col;
 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



Midterm Q4 from 04W2

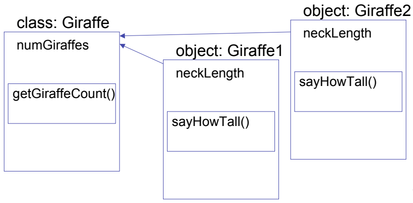
```
int[][] dataA = { { 0, 0 }, { 0, 0 } };
int[][] dataB = { { 0, 0 }, { 0, 0 } };
process( dataA, dataB );
```

```
public void process( int[][] arrA, int[][] arrB )
{
   int row;
   int col;
   int[][] arrC = { { 1, 1, 1 }, { 1, 1, 1 } };
   arrA = arrC;
   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
  - nonstatic fields belong to instantiated object
- Static methods can only use static fields
  - nonstatic 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

```
public class CokeMachine4
{
  private int numberOfCans;

  public CokeMachine4()
  {
    numberOfCans = 2;
    System.out.println("Adding another machine to your empire");
  }

  public int getNumberOfCans()
  {
    return numberOfCans;
  }

  public void reloadMachine(int loadedCans)
  {
    numberOfCans = loadedCans;
  }
```

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

Variable Scope

```
public class CokeMachine4
{
  private int numberOfCans;

  public CokeMachine4()
  {
    numberOfCans = 2;
    System.out.println("Adding another machine to your empire");
  }

  public double getVolumeOfCoke()
  {
    double totalLitres = numberOfCans * 0.355;
    return totalLitres;
  }

  public void reloadMachine(int loadedCans)
  {
    numberOfCans = loadedCans;
  }
```

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

Variable Scope

```
public class CokeMachine4
{
  private int numberOfCans;

  public CokeMachine4()
  {
    numberOfCans = 2;
    System.out.println("Adding another machine to your empire");
  }

  public int getNumberOfCans()
  {
    return numberOfCans;
  }

  public void reloadMachine(int loadedCans)
  {
    numberOfCans = loadedCans;
  }
```

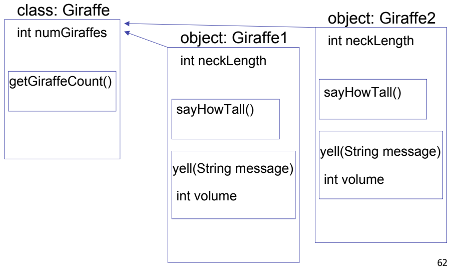
- loadedCans is method parameter
  - scope is method: also local scope
  - just like variable declared within parameter
  - accessed only within that method

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?