

Using Classes

COMP163 Introduction to Computer
Programming

“Unless in communicating with it one says exactly what one means, trouble is bound to result.”

Alan Turing

talking about computers

Programming Assignment

- A new programming assignment has been posted on Blackboard under Assignments / Programming Assignments
- Upload your two **best** Java files to Blackboard before midnight Friday
- See the tutors in Cherry 124 or your instructor for help

Read

- The syllabus gives you the related textbook chapters for each class period
- Read the zyBooks online text and answer all participation and challenge questions by
 - Chapter 2 Sunday, August 26
 - Chapter 3 Monday, August 27
 - Chapter 4 Wednesday, August 29
 - sections 5.1-5.3 Friday, August 31

Defining Your Own Data Type

- Java provides several simple data types that are supported natively by the hardware
 - `int`, `double`, `char`, `float`
- You can make your own data types to hold more complex information

Declaring an Object

- Imagine we have a class **MyThing**
- We can declare an object of type **MyThing** just like we declare a variable to be an int or a double

```
MyThing lizard;
```

```
int moose;
```

```
MyThing newt, salamander;
```

- lizard, newt and salamander are objects of the type MyThing

Reference Variables

- When you declare a primitive data item, such as a double or int, Java reserves some memory to store the data
- When you declare an object, Java does **not** reserve space for the object until it is created
- You can instantiate a new object with the keyword **new**

Instantiating Objects

```
MyThing tiger = new MyThing();  
MyThing elephant;  
elephant = new MyThing();
```

- After the word “**new**” is a call to a constructor method that creates the object
- The constructor method may or may not have parameters

null

- `null` is a Java keyword that means “nothing”
- If you define a reference variable and do not create an object, the variable is automatically set to `null`
- If an object reference is set to the value `null`, it means there is no object

```
MyThing pig = new MyThing();
```

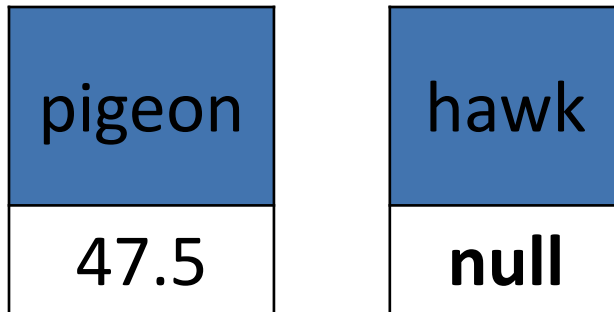
```
. . .
```

```
pig = null;
```

Reference Variables

```
double    pigeon = 47.5;
```

```
Widget    hawk;
```

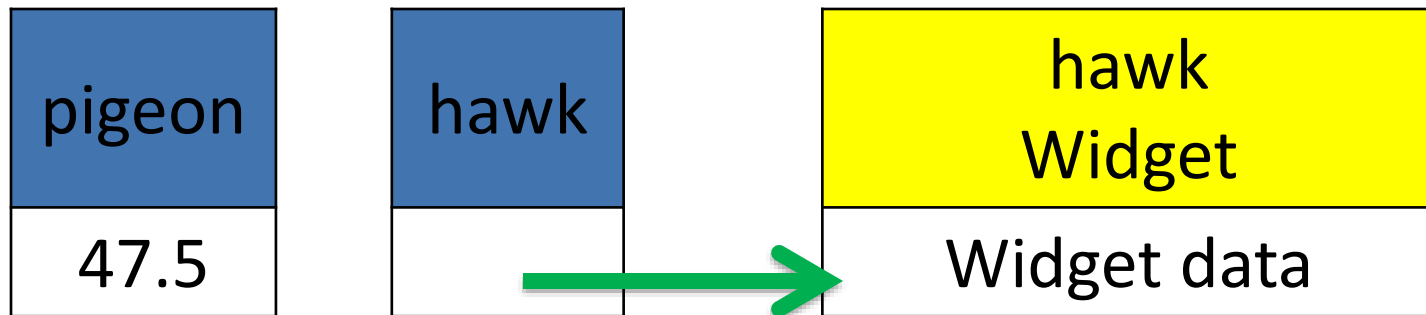


Reference Variables

```
double pigeon = 47.5;
```

```
Widget hawk;
```

```
hawk = new Widget();
```



Reference Variables

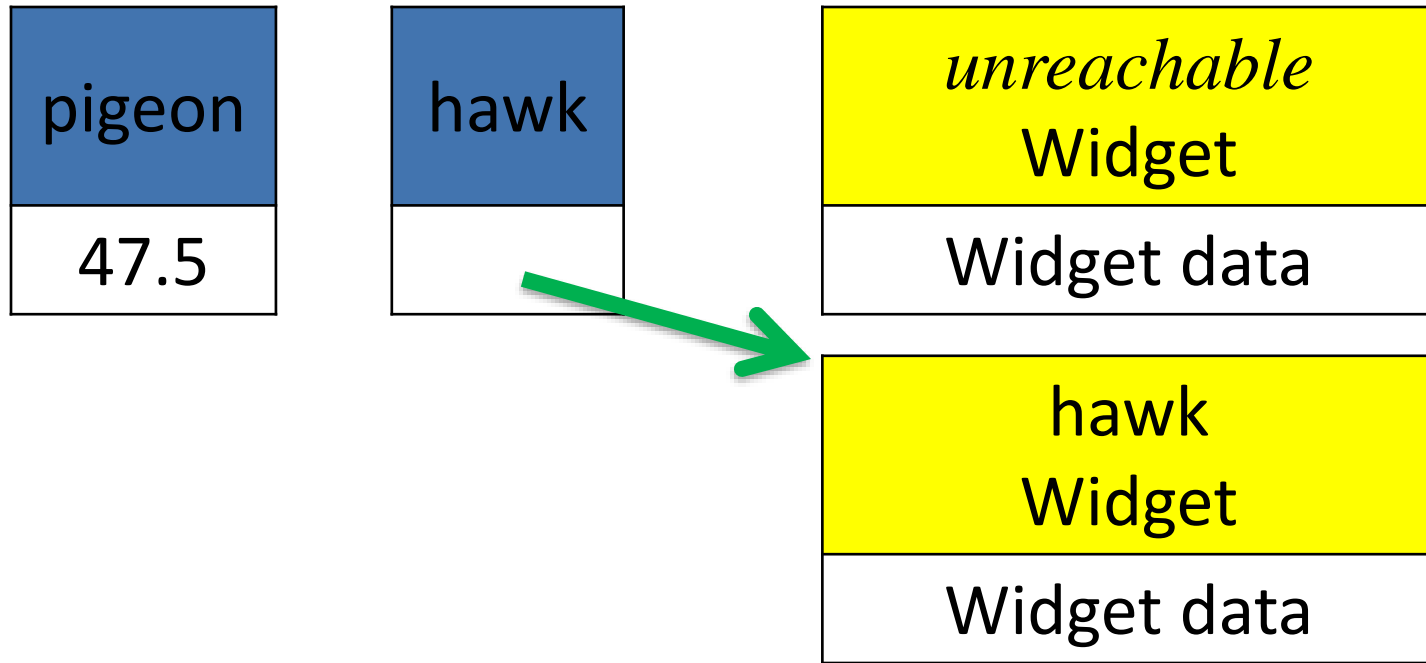
```
double pigeon = 47.5;
```

```
Widget hawk;
```

```
hawk = new Widget();
```

...

```
hawk = new Widget(); // later in program
```



Garbage Collection



- Sometimes during the execution of a Java program, an object becomes unusable

```
Widget pig = new Widget();
```

```
pig = new Widget(); // 2nd object
```

- The original pig Widget is no longer accessible and becomes wasted space
- Periodically Java performs garbage collection to reclaim unusable object space

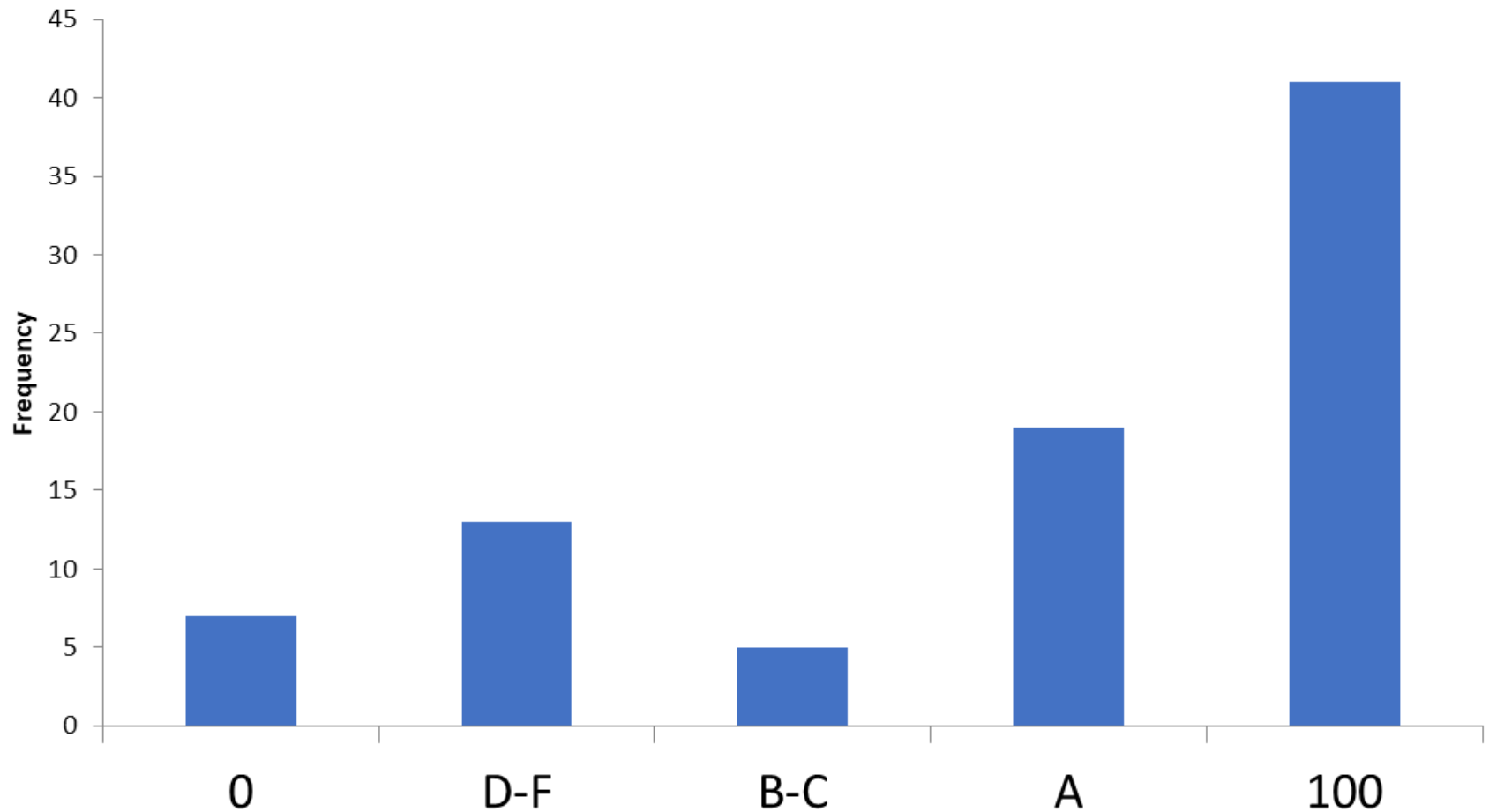
Name Rules

- By tradition, the names of classes start with an upper case letter and rest is lower case
- The name of a variable (object or primitive) is all lower case
- If you create a variable name from two English words, capitalize the first letter of the second English word

You create objects using the Java keyword

- A. new
- B. object
- C. class
- D. public

zyBooks Chapter 1 Questions



Deadline extended until midnight on Tuesday, August 28

Class Content

A class contains:

- **Data** – classes can contain multiple values
 - Variables in a class are called **instance variables** or **fields**
- **Actions** – classes have **methods** that can do things relating to the data

Class Data

- A class or object can contain multiple data values often called **fields** or **instance variables**
- Data can be primitive data types, such as double or int, or other objects
- A class encapsulates data values that form a logical entity, such as
 - Information about a student in class
 - Data received from a network
 - A clickable button in a program

Methods of a Class

- A class can have methods which define actions the class can take on its data
- A Java program has a **main** method that is the first to execute when the program starts
- Not every class has a main method
- A program may be comprised of multiple classes

Static or non-Static

- Methods and data of a class can be either static or non-static
- The keyword **static** specifies a static item
- Static data or methods belong to the class
 - All objects share static data and methods
- Non-static data or methods belong to an object
 - Each object has its own copy

Many Objects of a Class

- You can make many objects of a class
- Each class exists only once in a program
- There is only one copy of static data in the class
- Each object has its own copy of the object data
- If you had a Book class, you might make many objects of the Book with different titles

Static Variables

- Class variables can be specified as static

```
public class electric {  
    static int cow; // static class variable  
    int goat;      // non- static class variable  
}
```
- There is only one copy of a static variable shared by all objects of that class
- There is a separate copy of non-static variables for each object

Multiple Objects

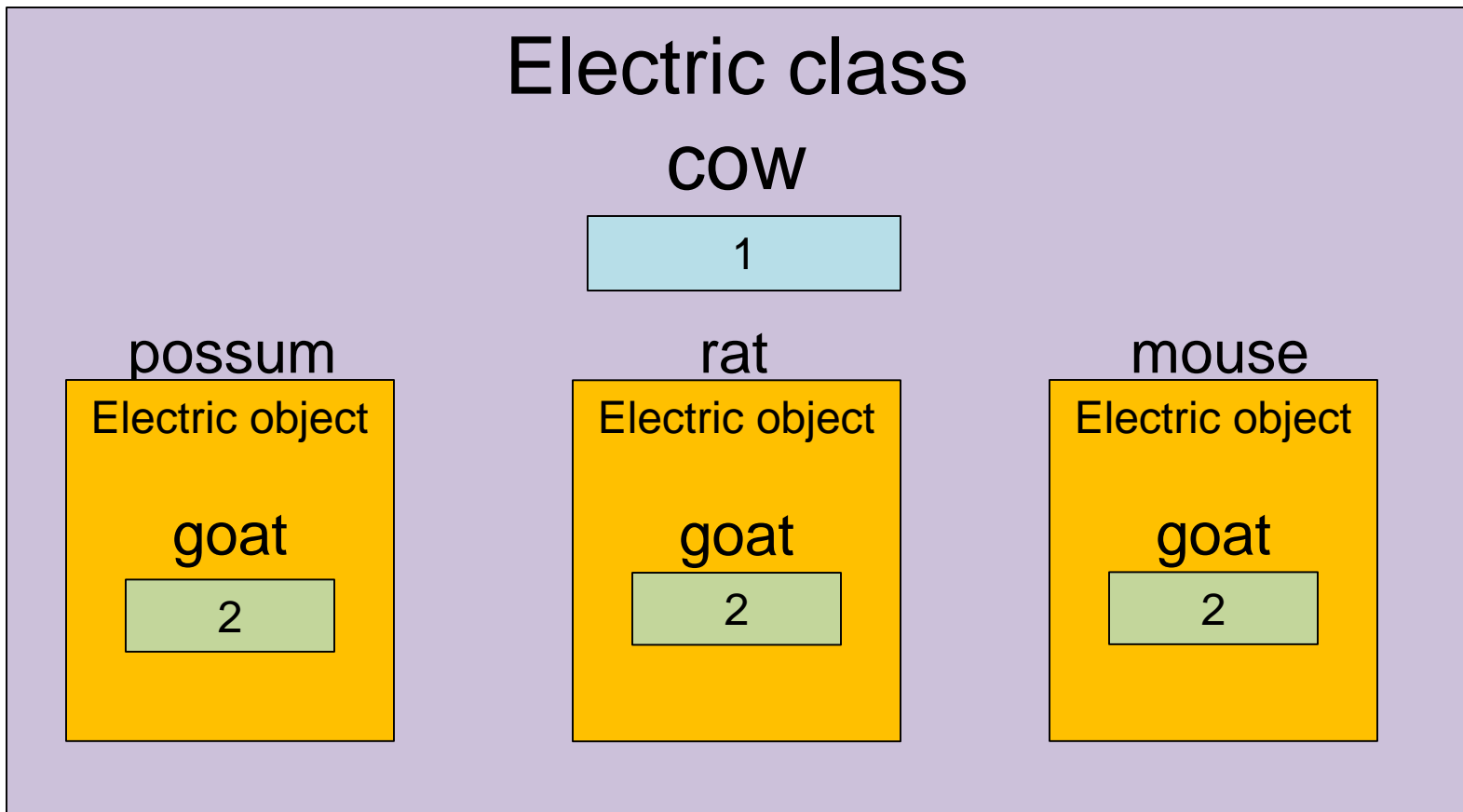
```
public class Electric {  
    static int cow = 1;  
    int goat = 2;  
}
```

```
Electric possum = new Electric();  
Electric rat     = new Electric();  
Electric mouse  = new Electric();
```

- There are three Electric objects
- Each object has its own copy of goat
- There is only one cow variable

Static and Instance Variables

- There is only one cat variable, but there is a dog for every object



static and non-static Method Calls

- Static methods are called on the class, not a specific object
- `Math.sqrt()` and `System.in` are static
- non-static methods are called on an object of the class, not the class itself

```
java.util.Scanner keyboard =  
    new java.util.Scanner(System.in) ;  
myVar = keyboard.nextDouble() ;
```

Period

- You can reference a class or object's data or methods by putting a period after the class or object name

- method of the Math class

```
Math.sin ( dog ) ;
```

- data field of Math class

```
Math.PI ;
```

- method of the keyboard Scanner object

```
keyboard.nextDouble ( ) ;
```

The method `weight` is

```
Widget goat = new Widget();  
int pounds = goat.weight();
```

- A. static
- B. non-static
- C. both
- D. neither

Math class

- The Math class contains many methods for computing mathematical functions
- There is little data in the Math class
- The Math class does have two constants, `Math.PI` and `Math.E`
- The Math class has only static methods
- You never need to make a Math object

Scanner class

- The `java.util.Scanner` class is used to read numbers from an input source
- You have to create a Scanner object
- When you create a Scanner object, you specify where the input is coming from
- The methods we call on Scanner objects are non-static methods

String

- **String** is a Java class
- The String class is special in that you can create a string object just by putting letters inside of double quotes

```
String dog = "Bow Wow";
```

```
String dog = new String("Bow Wow");
```

The main method of your program

- A. is a method of your program's class
- B. is a method of your program's object
- C. all the above
- D. none of the above

Function Concept in Math

Function definition

$$f(x) = 5x - 3$$

Parameter of function

Name of function

When $x = 1$, $f(x) = 2$ is the returned value.

When $x = 4$, $f(x) = 17$ is the returned value.

The result of the function is determined by the function definition and by the values of the parameters

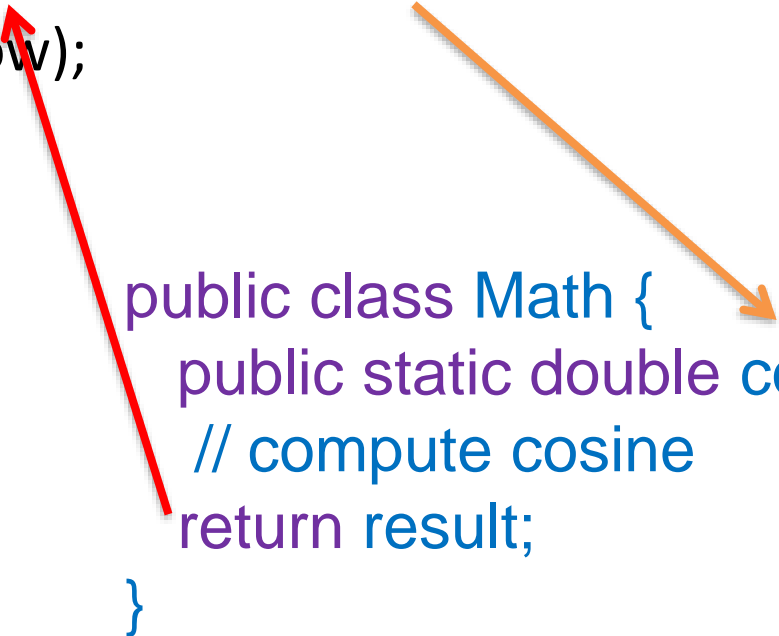
Would a rose by any other name ...

- In Java, the name **method** and **function** are sometimes used interchangeably
- A **method** refers to a function in an Object-Oriented language, like Java
- The words **procedure** and **subroutine** are also names for methods, usually void methods

Transfer of Execution

```
public class Demo {  
    public static void main( String[ ] stuff) {  
        double theta = 1.047;  
        double cow = 2.0*Math.cos(theta);  
        System.out.println(cow);  
    }  
}
```

```
public class Math {  
    public static double cos(double xyz) {  
        // compute cosine  
        return result;  
    }  
}
```



Keeping it Simple

- It is good programming practice to divide large programs into smaller manageable methods
- A method is a part of a program that performs a specific function
- Methods should be kept small enough to be understandable

Same Function, Different Data

- Methods are particularly useful when you are going to perform the same operation several times with different data
- The author of the `Math.cos` method did not know what angle you were going to use
- You can call the `Math.cos` method to get the cosine of any angle

```
cow = Math.cos( frog ) + Math.cos( toad );
```

Your programs

- A. Always contain methods
- B. Can use methods from other programs
- C. Can have many methods
- D. All of the above
- E. None of the above

Using Methods and Classes

- Many valuable Java classes and methods have been written
- The standard Java library has many classes and more libraries are available
- You can create your own classes and methods
- Using existing classes and methods allows you to create exciting programs with far less effort

Method Parameters

- Data can be passed to a method

```
double dog = 25.0, cat;  
cat = Math.sqrt( dog );
```

- Data passed to a method is called a parameter or argument

Multiple Parameters

- Method can have any number of parameters
- Some methods, such as `nextInt()` have no parameters
- Other methods have multiple parameters

```
double rat = 3.0, mouse = 2.0, vole;  
vole = Math.pow( mouse, rat );
```

- Parameters are separated by commas

Example Method

```
public class FirstMethod {  
    public static void main(String[] unused) {  
        int rabbit = 3, pika = 5;  
        rabbit = bunny( pika ) + 7;  
        System.out.println("rabbit is "+rabbit);  
    }  
  
    static int bunny( int hare ) {  
        int cottontail;  
        System.out.println("hare is "+hare);  
        cottontail = hare * 2;  
        return cottontail;  
    }  
}
```

Value Returning Methods

- Some methods return a value
- For example, `Math.sqrt(dog)` returns the square root of `dog`
- Methods that return a value are generally used in an equation

```
cat = Math.sqrt( dog ) * 47.9;
```

Methods in Equations

- When a method returns a value, the value logically replaces the method call in an equation
- The normal rules of priority apply
- The parameters to a method are always inside parenthesis, so if the parameter is an equation, it is calculated before calling the method

Equation Evaluation

```
double root, a = 2.0, b = -4.0, c = -6.0;
```

```
root = (-b + Math.sqrt( b*b - 4.0*a*c ) ) / (2 * a);
```

```
root = (-b + Math.sqrt( 16.0 - 4.0*a*c ) ) / (2 * a);
```

```
root = (-b + Math.sqrt(16.0 - 8.0*c) ) / (2 * a);
```

```
root = (-b + Math.sqrt(16.0 + 48.0) ) / (2 * a);
```

```
root = (-b + Math.sqrt( 64.0 ) ) / (2 * a);
```

```
root = (-b + 8.0) / (2 * a);
```

```
root = 12.0 / (2 * a);
```

```
root = 12.0 / 4.0;
```

```
root = 3.0;
```

What is the value of cow?

```
double dog = 25.0, goat = 2.0;  
double cow;  
cow = (3.0 + Math.sqrt( dog )) /goat;
```

- A. 2.0
- B. 4.0
- C. 15.5
- D. 25.0
- E. Hand full of magic beans

void methods

- A method that does not return a value is said to be a **void** method
- **void** methods usually appear on a line by themselves and not in equations

```
System.out.println("println is a void method");
```

Two Parts of Method

```
static int cube ( int cat )    { // heading  
    int dog = cat*cat*cat ;  
    return dog ;              } // body  
}
```

What is in a heading?

**modifiers
(optional)**

**type of value
returned**

**name of
method**

parameter list

static int cube (int mouse) {



Parameters Must Match Type

- Method

```
double myfunc( int cat, String dog) {  
... }
```

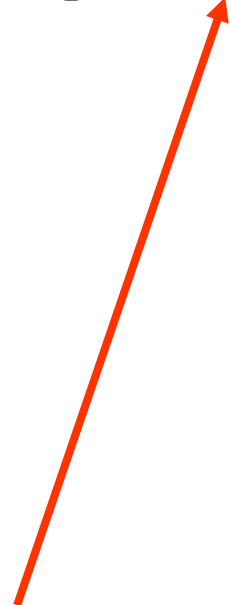
- Calling program

```
int cow;
```

```
String bull;
```

```
double goat;
```

```
goat = myfunc( cow, bull );
```



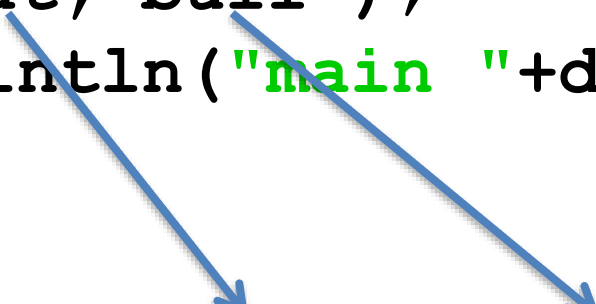
Argument Values Copied

- When a method call is executed, the values of the argument variables (or constants) are copied to the parameter variables
- The method uses a copy of the argument variables

Parameter Values Copied

```
public class PassParm {  
    public static void main(String[] args ) {  
        int cat = 5, bull = 7, dog = 1;  
        dog = doIt( cat, bull );  
        System.out.println("main "+dog);  
    }  
  
    static int doIt( int ant, int bug ) {  
        System.out.println("doIt"+ant+" "+bug);  
        return ant + bug;  
    }  
}
```

Displays: doIt 5 7
 main 12



Read

- The syllabus gives you the related textbook chapters for each class period
- Read the zyBooks online text and answer all participation and challenge questions by
 - Chapter 2 Sunday, August 26
 - Chapter 3 Monday, August 27
 - Chapter 4 Wednesday, August 29
 - sections 5.1-5.3 Friday, August 31

Programming Assignment

- A new programming assignment has been posted on Blackboard under Assignments / Programming Assignments
- Upload your two best Java files to Blackboard before midnight Friday.
- See the tutors in Cherry 124 or your instructor for help