Using Methods

COMP163

"There is no reason for an individual to have a computer in their home."

Ken Olsen

Digital Equipment Corp., 1977

Quiz

- There will be a quiz in the COMP163 recitation sections from September 5 to September 7
- The quiz will cover all the material since the beginning of the semester
- The questions will be
  - Complete a program
  - Show what a program displays
- There is a sample quiz on Blackboard under course materials

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

Scientific Notation

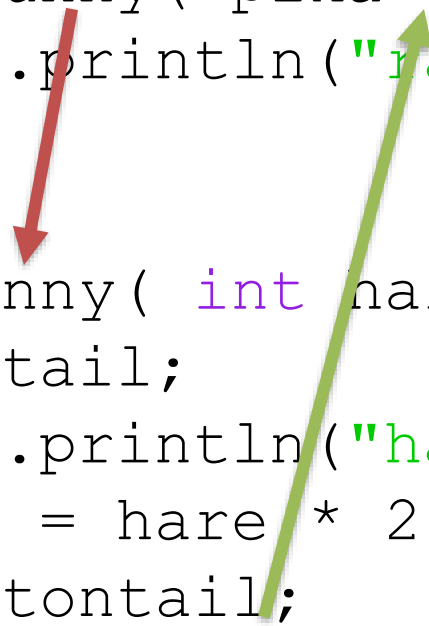
- Frequently very large or small numbers are written with an exponent, such as 6.02×10^{23}
- You can write such numbers in Java by following the simple number by “e” and the exponent, 6.02e23
- 6.626e-34 is Planck's constant
- 2.998e8 is the speed of light in m/s

Execution Flow

- Within a method, execution flows from top to bottom
- When a method is called, the program jumps into the method
- When the method ends or returns, execution resumes back in the calling program

Flow of Execution

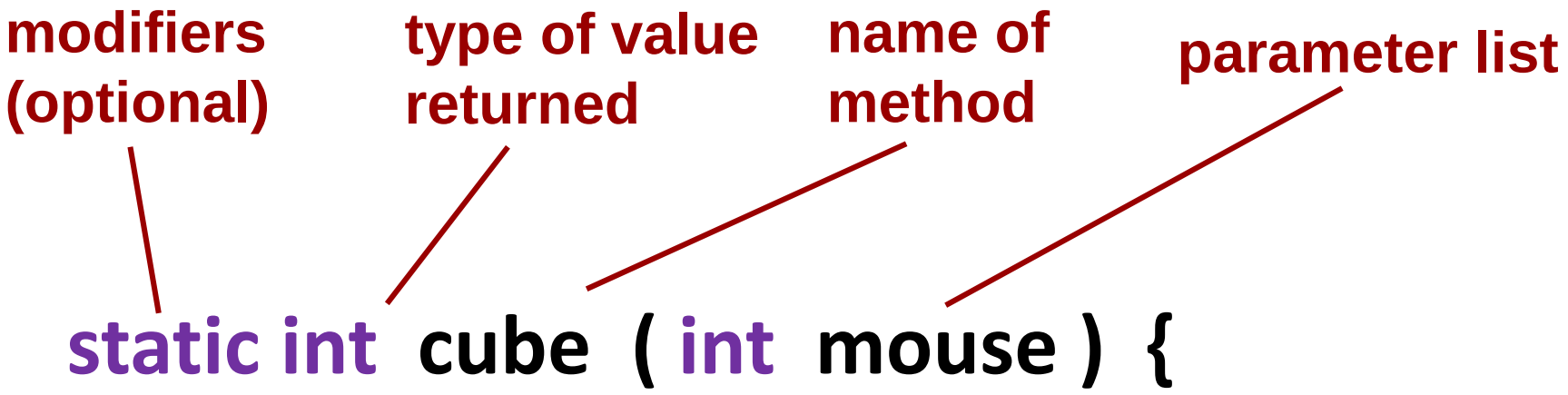
```
public class FirstMethod {
    public static void main(String[] unused) {
        int rabbit = 3, pika = 5;
        rabbit = bunny( pika ) + 7;
        System.out.println("rabbit is "+rabbit);
    }

    static int bunny( int hare ) {
        int cottontail;
        System.out.println("hare is "+hare);
        cottontail = hare * 2;
        return cottontail;
    }
}
```



Two Parts of Method

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

What is in a heading?



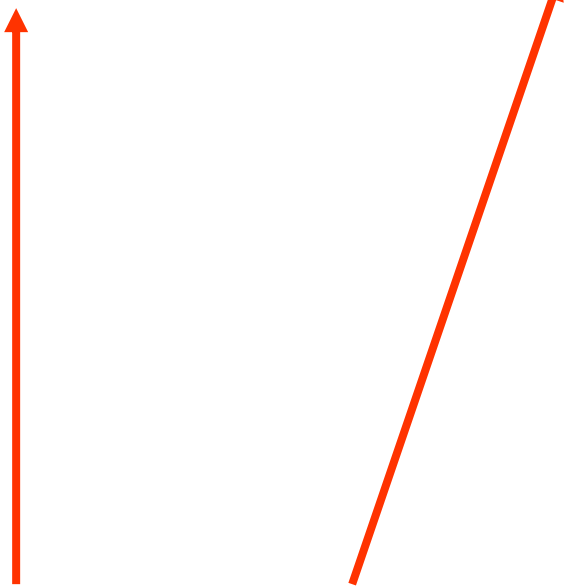
Parameters Must Match Type

- Method

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

- Calling program

```
int  cow;
String bull;
double goat;
     goat = myfunc( cow,    bull );
```

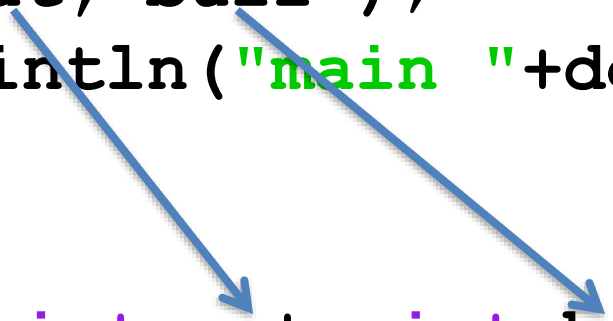


Argument Values Copied

- 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
```

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;
```

void Methods

- Some methods do not return a value
- System.out.println("Hi") prints a string but does not return a value
- Methods that do not return a value appear on a line all by themselves and are not part of an equation

Calling Methods of Classes

- Static methods are called on a class

Classname.method( )

- Math.cos(dog) is a static method

Calling Methods of Objects

- Non-static methods are called on objects, not classes

```
Scanner aardvark = new Scanner(System.in);

double dog = aardvark.nextDouble();
```

Calling Methods of this Class

- If you write a method, you can call it from within the same class without specifying a class or object

```
myMethod();
```

- If you extend an existing class, you inherit all of its methods and can call them without using a class or object name

Common Method Types

- Accessor or “get” methods return a value from the object
- Mutator or “set” methods change a value of the object’s data
- These are sometimes called *getter* and *setter* methods

What is displayed?

```
public class FirstMethod {
 public static void main(String[] leporidae) {
    int rabbit = 1, pika = 3;
    rabbit = bunny( pika ) + 4;
    System.out.print(" rabbit is "+rabbit);
 }
 static int bunny( int hare ) {
    int cottontail;
    System.out.print(" hare is "+hare);
    cottontail = hare + 2;
    return cottontail;
 }
}
```

A. rabbit is 9 hare is 3
B. hare is 3 rabbit is 9
C. rabbit is 7 hare is 2
D. hare is 2 rabbit is 4
E. none of the above

Now what is displayed?

```
public class FirstMethod {
 static int bunny( int hare ) {
    int cottontail;
    System.out.print(" hare is "+hare);
    cottontail = hare + 2;
    return cottontail;
 }
 public static void main(String[] leporidae) {
    int rabbit = 1, pika = 3;
    rabbit = bunny( pika ) + 4;
    System.out.print(" rabbit is "+rabbit);
 }
}
```

A. rabbit is 9 hare is 3
B. hare is 3 rabbit is 9
C. rabbit is 7 hare is 2
D. hare is 2 rabbit is 4
E. none of the above

Who are you?

Do you have the clicker with this code?

AD20AF22

String Methods

- There are many methods that can be called on objects of the String class
- length() Returns the length of this string

```
String lecture = "COMP163";
int howLong = lecture.length();     // 7
```

Searching Strings

- You can search a string to see if it contains the another string
- indexOf(String str) Returns the index within this string of the first occurrence of the specified String or -1 if not found

indexOf Example

- The position of a character in a string starts counting at zero

```
String major =   "Computer Science";
                 //   0123456789012345
```

```
int where = major.indexOf( "put" );      // 3
where = major.indexOf( "fail" );         // -1
```

Taking Strings Apart

- substring(int beginIndex, int endIndex) Returns a new string that is a substring of this string. The substring begins at beginIndex and extends to the character at position **endIndex - 1**. Thus the length of the substring is
  endIndex - beginIndex
- charAt(int index) Returns the char value at the specified index

substring Example

```
String major =   "Computer Engineering";
                 //                1111111111
                 //   01234567890123456789
```

String result = major. substring(11, 14);

result has the value "gin"

Changing Case

- toUpperCase() returns a copy of the String with all of the letters in upper case
- toLowerCase() returns a copy of the String with all of the letters in lower case

```
String gnu = "Today is 1/27/17";
String wildebeest = gnu.toUpperCase();
System.out.println(gnu +"\n"+wildebeest);
```

```
Today is 1/27/17
TODAY IS 1/27/17
```

What is the value of result?

```
String major =      "Computer Engineering";
                //   01234567890123456789
String result = "S" + major. substring( 13, 17 );
```

A. neer
B. Smajor
C. Sneeri
D. Sneer
E. S

Miscellaneous String Method

- trim() Returns a copy of the string, with leading and trailing whitespace omitted

```
String hair = "  What ever    ";
              // 01234567890123
String bald = hair.trim();
// bald is "What ever";
```

Interesting String Methods

- length
- indexOf
- substring
- charAt
- toUpperCase
- toLowerCase
- trim

Work Together

- Write a line of Java to find the location of a space in

```
String bulldog = "Aggie Pride";
int space = ?
```

- Write Java to find the length of bulldog

```
int len = ?
```

Possible Solutions

- Write a line of Java to find the location of a space in

```
String bulldog = "Aggie Pride";
int space = bulldog.indexOf( " " );
```

- Write Java to find the length of bulldog

```
int len = bulldog.length;
```

Work Together

- Given a String with two English words separated by a space, write some Java to copy the second English word to another String

```
String words = "some  thing";
```

Possible Solution

- Given a String with two English words separated by a space, write some Java to copy the second English word to another String

```
String words = "some  thing";
int len = words.length();
int space = words.indexOf( " " );
String second = words.substring(space+1, len);
```

Mixed Mode Arithmetic

- In general, you should avoid mixing int and double values in an equation
- Sometimes you need to, so you must be aware of the rules of mixed mode equations

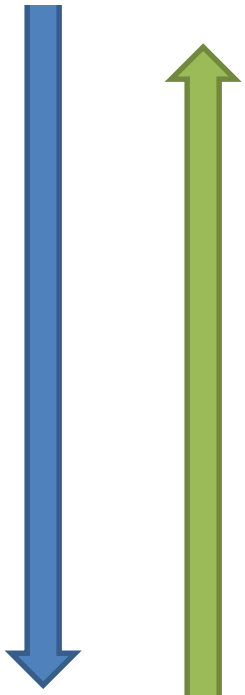
Data Conversion

- Sometimes it is convenient to convert data from one type to another
- For example, in a particular situation we may want to treat an integer as a double value
- These conversions do not change the type of a variable or the value that's stored in it – they only convert a value as part of a computation

Data Conversion

- ***Widening conversions*** are safest because they tend to go from a small data type to a **larger** one (such as an `int` to a `double`)
- ***Narrowing conversions*** can lose information because they tend to go from a large data type to a smaller one (such as a `double` to an `int`)
- In Java, data conversions can occur in three ways:
  - assignment conversion
  - promotion
  - casting

Number Range



Widening

Narrowing

short -32768 to 32767

int -2147483648 to 2147483647

long -9223372036854775808 to 9223372036854775807

float $-3.4028235 \times 10^{38}$ to 3.4028235×10^{38}

double $-1.7976931348623157 \times 10^{308}$ to $1.7976931348623157 \times 10^{308}$

Assignment Conversion

- *Assignment conversion* occurs when a value of one type is assigned to a variable of another
- Example:

```
int dollars = 20;
double money;
money = dollars;      // money is 20.0
```

- **Only widening conversions can happen via assignment**
- Note that the value or type of **`dollars`** did not change

No Narrowing Assignments

- You cannot assign a variable to a larger value

```
int       smallstuff;
double    biggie = 47.0;
smallstuff = biggie; // this generates an error
```

- Although in this case, 47 will fit happily in an int, Java does not know what value will be in the double when you do the assign

Type Promotion

- When possible, avoid mixing data types. Expressions with integers should contain only integers.
- Sometimes you will need to mix data types. When an operation is performed on mixed data types, the lower operand is promoted to the type of the higher operand.

Promotion order

- If either operand is of type **double**, the other is converted to **double**
- Otherwise, if either operand is of type **float**, the other is converted to **float**
- Otherwise, if either operand is of type **long**, the other is converted to **long**
- Otherwise, both operands are converted to type **int**

Promotion Examples

```
double mouse, rat = 1.0;
int    vole = 2, shrew = 1;

mouse = rat / vole + shrew;    // 1.5
mouse = shrew / vole + rat;    // 1.0
```

What is the value of cow?

```
double  cow = 3.0, goat = 5.0;
int         sheep = 7;
cow = sheep / 2 + goat;
```

A. 8.5

B. 8.0

C. 3.0

D. 5.0

E. 7.0

What is the value of cow?

```
double  cow = 3.0, goat = 5.0;
int         sheep = 7;
cow = sheep / 2.0 + goat;
```

A. 8.5

B. 8.0

C. 3.0

D. 5.0

E. 7.0

Casting

- *Casting* is the most powerful, and dangerous, technique for conversion
- To cast, the desired type is put in parentheses in front of the value being converted

```
int total = 50;
double result = (double)total / 6;
```

- Without the cast, the fractional part of the answer would be lost
- You can cast a single variable or an expression

Casting Dangers

- Both widening and narrowing conversions can be accomplished by explicitly casting a value
- Note that a narrowing conversion may cause data to be lost

```
long big = 123456789012L;
int small = (int)big;
```

- An int cannot hold such a large number. The program will not generate an error, it will just give the wrong answer (small is -1097262572)

Type Casting is Explicit Conversion of Type

(int)4.8	has value	4
(double)5	has value	5.0
int s = 7, f = 4;		
s / f	has value	1
(double)s / (double)f	has value	1.75
(double)s / f	has value	1.75
(double)(s / f)	has value	1.0
7.0 / (double)f	has value	1.75

Narrowing Casting

- You can assign a variable to a larger value by casting

```
int      smallstuff;
double   biggie = 47.0;
smallstuff = (int)biggie;     // 47
```

What is the value of cow?

```
double  cow = 3.0, goat = 5.0;
int         sheep = 7;
cow = (double)sheep / 2 + goat;
```

A. 8.5

B. 8.0

C. 3.0

D. 5.0

E. 7.0

Quiz

- There will be a quiz in the COMP163 recitation sections from September 5 to September 11
- The quiz will cover all the material since the beginning of the semester
- The questions will be
  - Complete a program
  - Show what a program displays
- There is a sample quiz on Blackboard under course materials

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