Methods

GEEN163 Introduction to Computer Programming

*"Civilization is a **method** of living, an attitude of equal respect for all men."*

Jane Addams

first American woman to be awarded the Nobel Peace Prize

Homework

- The third programming assignment has been posted on Blackboard
- Programs are due by midnight on Friday, February 7, 2014
- We have not yet covered everything you need to write this program

Reading Quiz

- Read sections 4.1 – 4.4 of the textbook and then answer the reading quiz on Blackboard
- You may only attempt the quiz once
- There is no time limit
- It is due by 8:00pm on Tuesday, February 4

TuringsCraft

- Answer the questions to sections 3.10 through 4.5
- You will earn 4 points for each correct answer up to a maximum of 100 points
- Due by midnight on Monday, February 10, 2014

Schedule Update

Monday	Wednesday	Friday
	cancelled	Using Methods chapter 3
More on methods	Applet basics chapter 4	Applet programming
Graphics	review	**Exam 1**

Lab quiz on Thursday, February 13, 2014

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

Main Method

- Every Java program must have a method called main

```
public static void main(String[] args)
```

- Program execution always begins with the main method

- Any other method are subprograms and must be called from the main method or another method

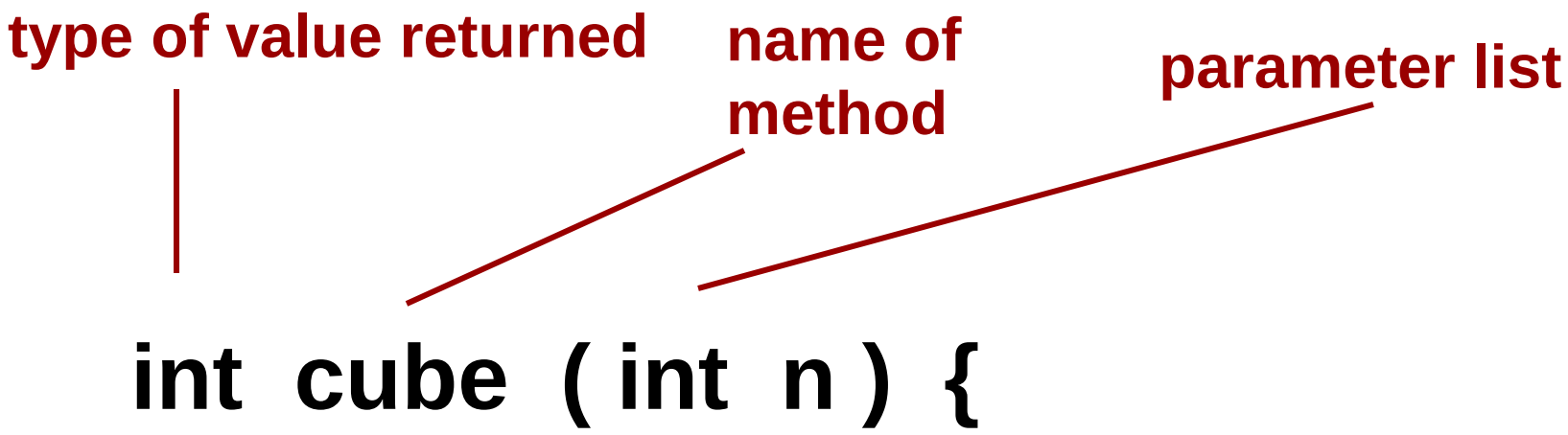
Two Parts of Method

```
static int  cube ( int  n )        {
   int dog = n * n * n ;
   return  dog ;
}
```

// heading

// body

What is in a heading?

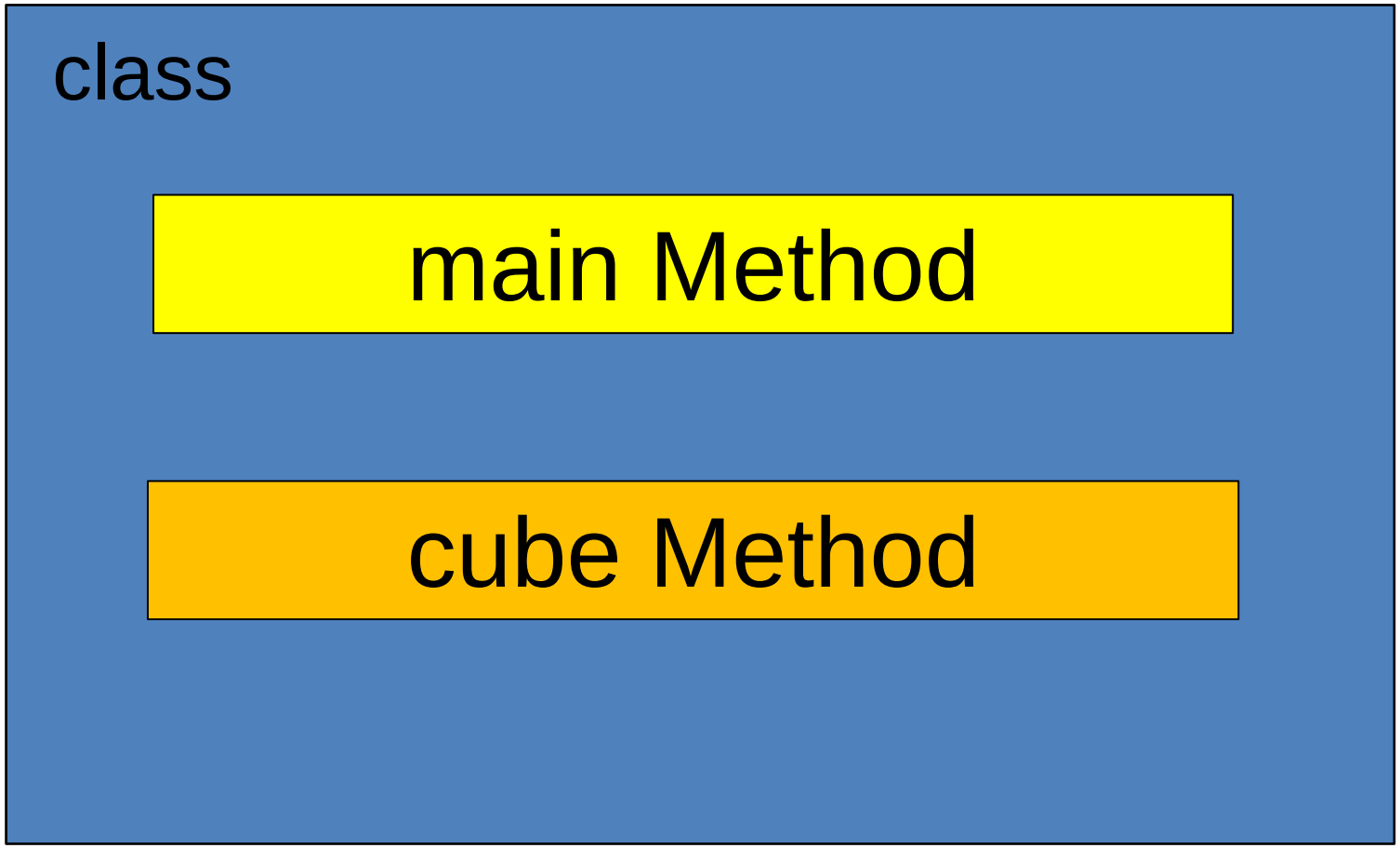


Good Name

- You should give your methods names the indicate what they do
- Method names should be in lower case letter
- The first letter of a second English word should be capitalized

```
cube          cubeRoot
printName     getText
average       characterAt
```

Methods inside a Class



Methods inside a Class

```
public class Example {

    public static void main(String[] dog) {
                    // main method body
    }

    static double cube( double cat ) {
                    // cube method body
    }

}
```

Method Calls

- One method calls or executes another by using the name of the called method together with **(** parenthesis**)** containing an argument list

- A method call temporarily transfers control from the calling method to the called method

Returning a Value

- Some methods can return a value to the calling program
- After the Math.cos method computes the cosine, it returns it to your program
- The returned value of a method can be used in an equation

```
double dog = 5.0 * Math.cos( cat ) + 47.0;
```

```
import java.util.Scanner;
public class Example1 {
    public static void main(String[] args) {
      double a, b, result;
      Scanner keys = new Scanner(System.in);
      a = keys.nextDouble();
      b = keys.nextDouble();
      result = cube( a );
      System.out.println("cube is "+result);
      result = cube( b );
      System.out.println("cube is "+result);
    }

    static double cube( double x ) {
       double triple = x * x * x;
       return triple;
    }
}
```

Methods must be written

A. inside the main method
B. inside the class
C. outside the class
D. none of the above

Calling a Method with Parameters

- Method calls have the name of the method followed by the arguments enclosed in parenthesis
- The calling program must supply exactly the same number of arguments of the same type as the method
- The names of the arguments in the main program can be different from the names of the parameters in the method

Arguments and Parameters

Arguments	Parameters
Always appear in a method call within the calling program	Always appear in the method heading

Method Call Syntax

MethodName (*Argument List*)

- The argument list is a way for methods to communicate with each other by passing information
- The argument list can contain 0, 1, or more arguments, separated by commas, depending on the method

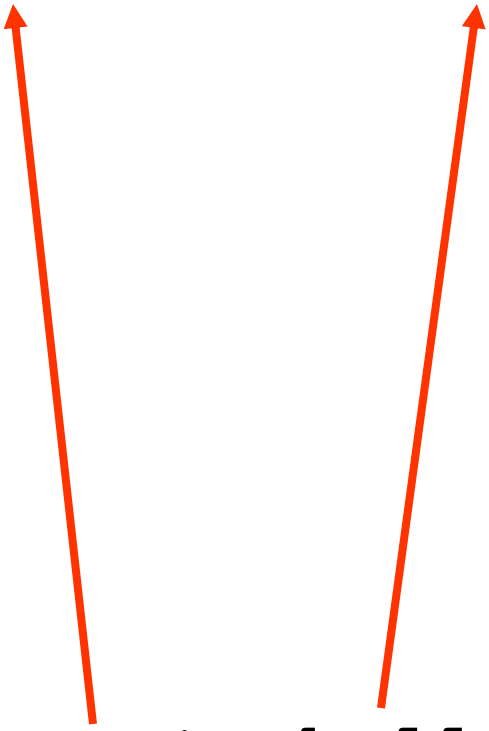
Parameter Types Must Match

Method

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

main program

```
int goat;
double bull;
int rat;
     rat = myfunc( goat, bull );
```



Variable Type

- In the method header, you need to specify the variable type

```
int myFunc( int trout, double salmon)
```

- When you call the method, you do **not** need to specify the type of the parameters

```
int fish, cow = 47;
fish = myFunc( cow, 3.13 );
```

Alternate Names

- Some Java texts use the term ***"actual parameters"*** for arguments

- Those books then refer to parameters as ***"formal parameters"***

When you call a method, the variable names in the main program must match the variable names in the method header.

A. True

B. False

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   a = 5, b = 47, c;
        c = example( a, b );
        System.out.println("main"+c);
     }

     static int example( int x, int y ) {
        System.out.println("example"+(x+y));
        return x + y;
     }
}
```

What is displayed by this program?

```
int dog = 5, cat = 7;
tryit( cat, dog );
       -----   -----   -----   -----
void tryit( int cow, int bull) {
   System.out.println("cow="+cow+ "bull="+bull);
}
```

A. cow=5 bull=7

B. cow=7 bull=5

C. dog=5 cat=7

D. none of the above

Follow the Execution

- Java executes lines sequentially through a method
- When a method is called, execution jumps to the method
- Methods may appear before or after the main method

What is displayed?

```
public class Colors {
  static void aqua() {
    System.out.println("Blue");
  }
  static void rouge() {
    System.out.println("Scarlet");
  }
  static void sun(){
    System.out.println("Yellow");
  }
  public static void main(String[] args) {
    sun();
    rouge();
    System.out.println("Gray");
  }
}
```

A. Blue
   Scarlet
   Yellow
   Gray

B. Yellow
   Scarlet
   Gray

C. Yellow
   Blue
   Scarlet
   Gray

D. Gray
   Yellow
   Scarlet

E. none of the above

More About Methods

- Methods should perform a single task or function
- It is not considered good practice for a method to be very long
- Every method has a return type
- If the return type is not void, the method returns a value to the calling program

Two Kinds of Returns

Value-Returning	Void
Always returns a single value to its caller and is called from within an expression	Never returns a value to its caller, and is called as a separate statement

A void Method Call

```
public class VoidMeth {
    public static void main(String[] args) {
      displayMessage( 'A' ) ;   //void method call
      System.out.println("Good Bye");
    }

    static void  displayMessage( char grade) {
      System.out.println("I want an "+ grade );
    }
}
```

This program displays

```
I want an A
Good Bye
```

Extending Classes

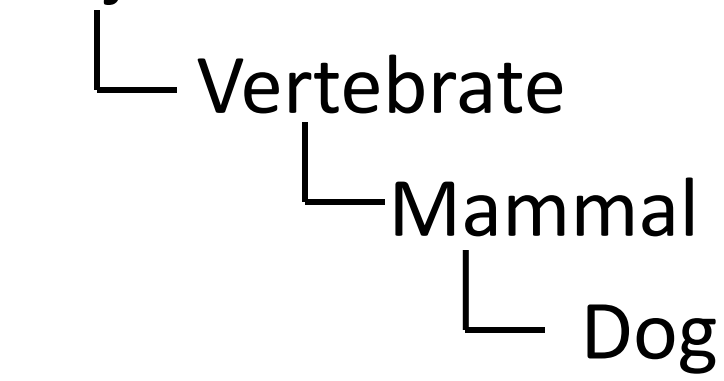
- One of the great advantages of Object Oriented programming is the ability to extend an existing class
- You can create a new class with all of the features of an existing class and then add extra methods or data
- You can extend any class, including the system library classes
- You do not need the source to extend a class

Inheritance

- A new class can inherit all the data and methods of another class
- A child class inherits from a parent class
- Any class can be a parent class
- The child class can add extra features or change some features of the parent class
- A class can inherit from a class that inherits from another class to any depth

Ancestors

- Imagine you have a class called Mammal
- A Dog class might inherit from Mammal
- The Mammal class might inherit from the class Vertebrate
- All classes inherit from the class **O**bject



extends

- In Java you indicate that one class extends or inherits from another by using the **extends** keyword

```
public class NewClass extends OldClass {
```

Inheritance Example

- Assume a class Mammal with methods
  - getWeight( ) and setColor( )
- You can create a class Dog that extends the class mammal with the method
  - getWagRate( )
- Objects of the Dog class can call any of the methods

```
Dog  fido = new Dog( );
double pounds = fido.getWeight();
double wags   = fido.getWagRate();
```

Is A

- Many experts recommend that class B extend class A only if class B "is a" class A
- Dog can extend Mammal because Dog "is a" Mammal
- If Dog extends the Mammal class, an object of the Dog class can be used anywhere an object of the Mammal class could be used

Homework

- The third programming assignment has been posted on Blackboard
- Programs are due by midnight on Friday, February 7, 2014
- We have not yet covered everything you need to write this program

Reading Quiz

- Read sections 4.1 – 4.4 of the textbook and then answer the reading quiz on Blackboard
- You may only attempt the quiz once
- There is no time limit
- It is due by 8:00pm on Tuesday, February 4

TuringsCraft

- Answer the questions to sections 3.10 through 4.5
- You will earn 4 points for each correct answer up to a maximum of 100 points
- Due by midnight on Monday, February 10, 2014

Schedule Update

Monday	Wednesday	Friday
	cancelled	Using Methods chapter 3
More on methods	Applet basics chapter 4	Applet programming
Graphics	review	**Exam 1**

Lab quiz on Thursday, February 13, 2014