Methods & GUIs

GEEN163

"There are two ways of constructing a software design. One way is to make it so simple that there are obviously no deficiencies. And the other way is to make it so complicated that there are no obvious deficiencies."

C.A.R. Hoare

First Exam

- The first exam in GEEN163 will be on Monday, September 19
- The exam will cover all material since the beginning of class
- We will review for the exam this week
- You are allowed one 8½” by 11” page of notes for the exam

Tutoring Schedule

	Monday	Tuesday	Wednesday	Thursday	Friday
11:00 AM	Prince Wynn	Prince Wynn	Prince Wynn	Prince Wynn	Prince Wynn
12:00 PM	Kaleb Holley	Prince Wynn	Hines & Holley	Wynn & Gray	Hines & Holley
1:00 PM	Kaleb Holley	Kaleb Holley	Kaleb Holley	Holley & Staggers	Kaleb Holley
2:00 PM		Kaleb Holley		Holley & Staggers	Gray
3:00 PM	Wynn & Smith	Jordyn Smith	Jordyn Smith	Wynn & Smith	Jordyn Smith
4:00 PM	Jordyn Smith	Jordyn Smith	Jordyn Smith	Wynn & Smith	Jordyn Smith
5:00 PM	Patrick Grant	Holley Jupiter	Patrick Grant	Holley Jupiter	Jordyn Smith
6:00 PM	Paul Biocco	Wagner & Biocco	Paul Biocco	Paul Biocco	Paul Biocco
7:00 PM	Paul Biocco	Wagner & Biocco	Paul Biocco	Paul Biocco	Paul Biocco
Tutors meet in Cherry 124					

Making a GUI

1. Create a class that extends JFrame and implements java.awt.event.ActionListener
2. Create component objects
3. Specify the layout manager
4. Add the component objects to the container
5. Add an action listener to the components that respond to user input
6. All the program to be visible and stop
7. Create an actionPerformed method

Usual Java GUI

```
public class MyProgram extends javax.swing.JFrame
                                implements java.awt.event.ActionListener {
        // declare GUI components here
   public MyProgram() {
        // setup GUI here
   }
   public void actionPerformed( java.awt.event.ActionEvent  dog){
        // program here
   }
   public static void main(String[] cat) {
        MyProgram fish = new MyProgram();
   }
}
```

Displaying Results in a GUI

- GUI programs usually set the text of a JLabel

```
// in the class before any methods
JLabel hawk = new JLabel();
// in the actionPerformed method
double dog = Math.sin(cat);
hawk.setText("The answer is " + dog);
```

If a JFrame GUI program calls System.out.println the program will

A. compiler error
B. run time error
C. display the data in the usual place for applications
D. do nothing

GUI Text Input

- An GUI usually gets text input from a JTextField

```
                    //  in the class before any methods
JTextField  kitten = new JTextField();
                    //  in the actionPerformed method
String bear;
bear = kitten.getText();
```

String and Numbers

- A String can contain any character on the keyboard, including the numbers

```
String myGrade = "4.0";
double myScore = 4.0;
```

- You cannot do arithmetic with Strings
- A string containing numerical characters must be converted to a double or an int for use in any calculations

Converting Strings to Numbers

- A String that contains numerical characters can be converted

```
String deuce = "2.25";
double cow = Double.parseDouble( deuce );
String four = "4";
int goat = Integer.parseInt( four );
```

- These methods will throw an exception if the string does not contain a number

Converting Numbers to Strings

- There are several methods to convert numbers to Strings
- The easy way is to concatenate a number with a string

```
double dog = 47.5;
String lizard;
lizard = "The answer is " + dog;
```

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

Starting Method

- Every Java application must have a method called main

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

- Application execution always begins with the main method

- Java GUIs need:

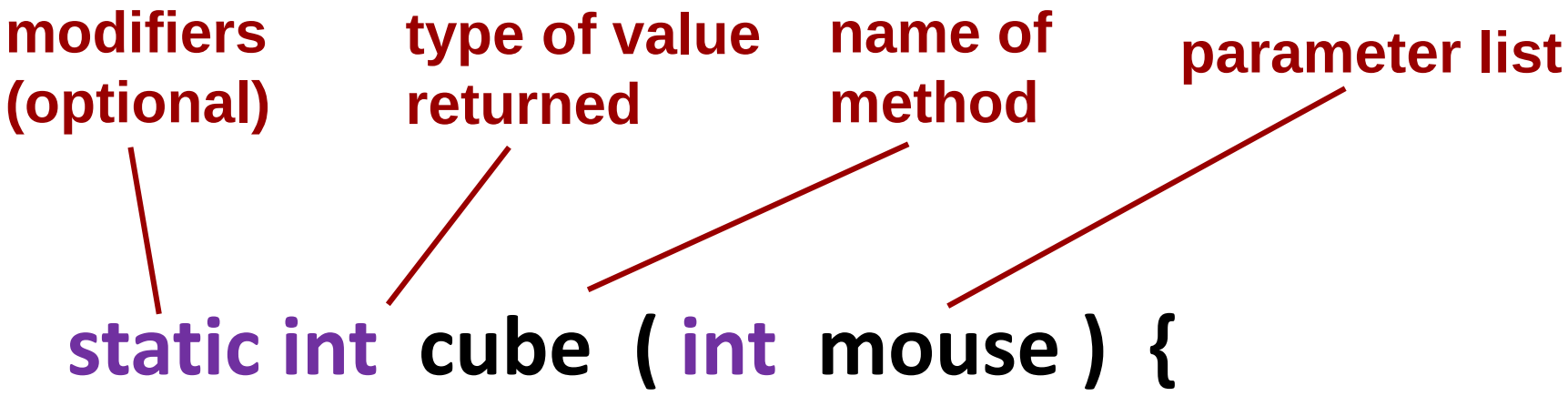
```
public void actionPerformed( java.awt.event.ActionEvent  bird)
```

- constructor method

Two Parts of Method

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

What is in a heading?

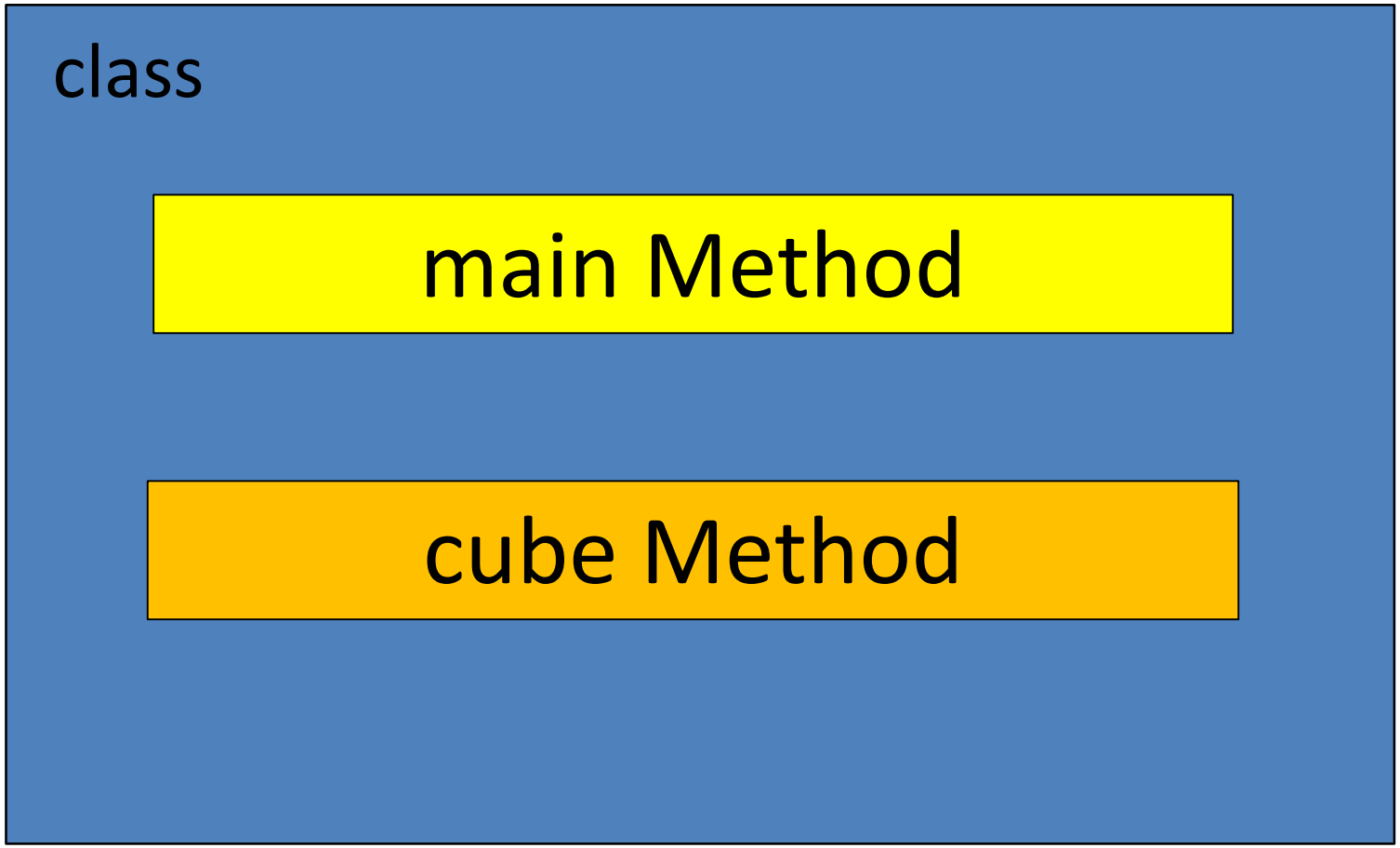


Good Name

- 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.sqrt method computes the cosine, it returns it to your program
- The returned value of a method can be used in an equation

```
double cat = 9.0;
double dog = 2.0 * Math.sqrt( cat ) + 7.0;
double dog = 2.0 * 3.0 + 7.0;
double dog = 6.0 + 7.0;
double dog = 13.0;
```

```
import java.util.Scanner;
public class Example1 {
    public static void main(String[] args) {
      double cat, dog, gerbil;
      Scanner keys = new Scanner(System.in);
      System.out.println("Enter two numbers");
      cat = keys.nextDouble();
      dog = keys.nextDouble();
      gerbil = cube( cat );
      System.out.println("cube is "+ gerbil);
      gerbil = cube( dog );
      System.out.println("cube is "+ gerbil);
    }

    static double cube( double goat ) {
       double aardvark = goat * goat * goat;
       return aardvark;
    }
}
```

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

Alternate Names

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

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

No Food or Drinks in the Lab

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

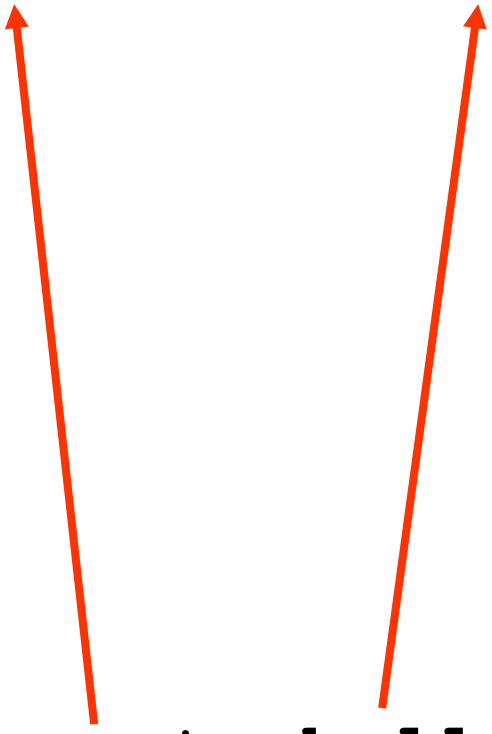
Parameter Types Must Match

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 );
```

When you call a method, the variables in the main program must match _____ in the method header

A. The variable name but not the type
B. The variable type but not the name
C. Both the variable name and type
D. Neither the variable name or type
E. All the above

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 sequentially through a method one line at a time
- 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");
  }
  public static void main(String[] args) {
    kermit();
    rouge();
    System.out.println("Gray");
  }
  static void kermit(){
    System.out.println("Green");
  }
}
```

A. Blue
   Scarlet
   Gray
   Green

B. Green
   Scarlet
   Gray

C. Green
   Blue
   Scarlet
   Gray

D. Gray
   Green
   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

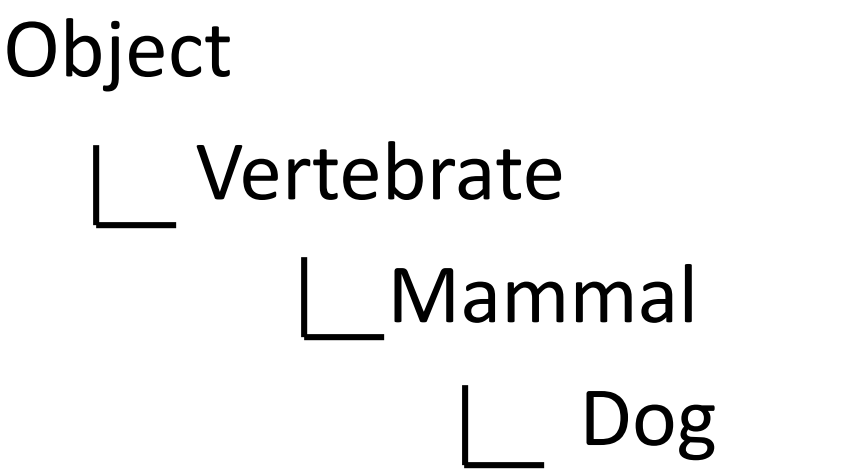
```
I want an A
Good Bye
```

Inheritance

- A new class can inherit all the data and methods of another class
- The child class can add extra features or change some features of the parent class
- A child class can call all of the methods of the parent class and all ancestors

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 Mammal extends Vertebrate {
```

Big Computer Science Concepts

- Inheritance
- Methods
- Constructor methods
- Creating objects
- Types of variables
- Strings are different from numbers

First Exam

- The first exam in GEEN163 will be on Monday, September 19
- The exam will cover all material since the beginning of class
- We will review for the exam this week
- You are allowed one 8½” by 11” page of notes for the exam

Tutoring Schedule

	Monday	Tuesday	Wednesday	Thursday	Friday
11:00 AM	Prince Wynn	Prince Wynn	Prince Wynn	Prince Wynn	Prince Wynn
12:00 PM	Kaleb Holley	Prince Wynn	Hines & Holley	Wynn & Gray	Hines & Holley
1:00 PM	Kaleb Holley	Kaleb Holley	Kaleb Holley	Holley & Staggers	Kaleb Holley
2:00 PM		Kaleb Holley		Holley & Staggers	Gray
3:00 PM	Wynn & Smith	Jordyn Smith	Jordyn Smith	Wynn & Smith	Jordyn Smith
4:00 PM	Jordyn Smith	Jordyn Smith	Jordyn Smith	Wynn & Smith	Jordyn Smith
5:00 PM	Patrick Grant	Holley Jupiter	Patrick Grant	Holley Jupiter	Jordyn Smith
6:00 PM	Paul Biocco	Wagner & Biocco	Paul Biocco	Paul Biocco	Paul Biocco
7:00 PM	Paul Biocco	Wagner & Biocco	Paul Biocco	Paul Biocco	Paul Biocco
Tutors meet in Cherry 124					