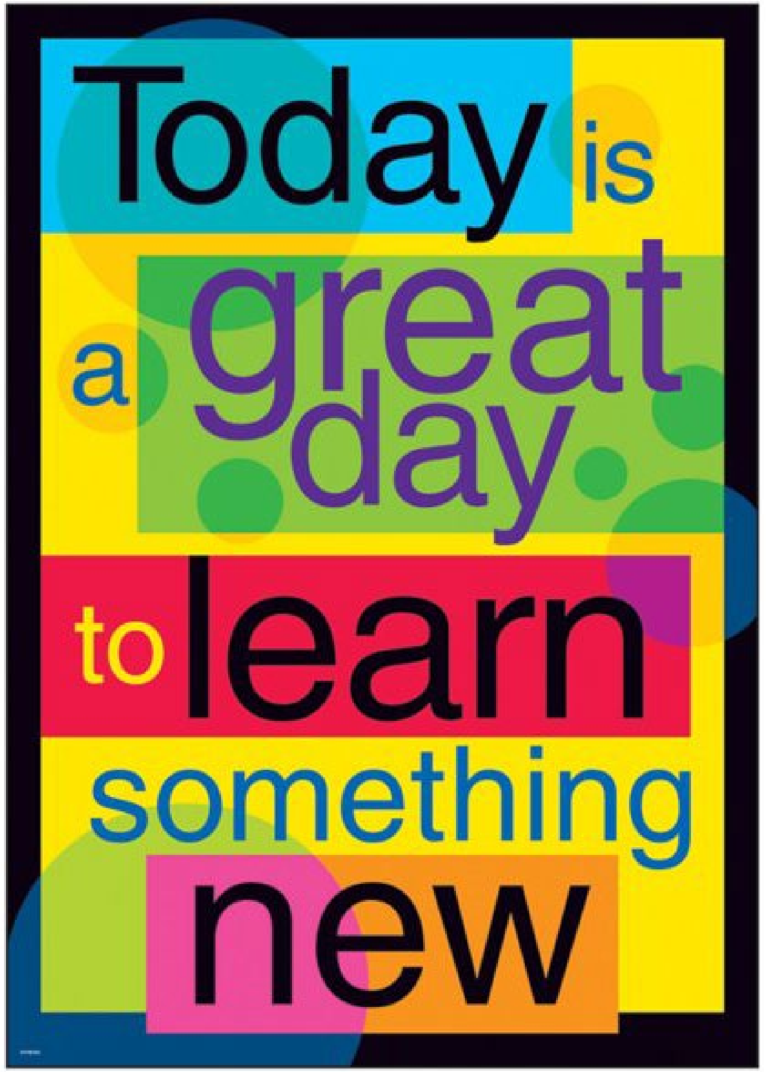
Objects and Methods

Colorado State University
Department of Computer Science

Slides Originally Created by Albert Lionelle (Albert.Lionelle@colostate.edu), updated by Marcia Moraes (marcia.moraes@colostate.edu)

Announcements

- Reminder – readings are due **before** lecture
  - You don’t have to do all of it - challenge problems can be challenging…
  - You can return to them.
  - We start off each lecture with a quiz from your reading!



Todo:
Busy Week!
(readings + labs)
Lab projects start!

Recall Activity - Attendance

Grab a paper, write your name, as it is in our Canvas course, and your answers to the following questions. Turn this as your attendance for today's lecture.

What is a method?

Which of the following are valid method "signatures"

A. public double calcArea(double width, double height) {}

B. Public static my_method(int x, boolean y) {}

C. public static void main(String args[]);

D. All listed

E. None listed

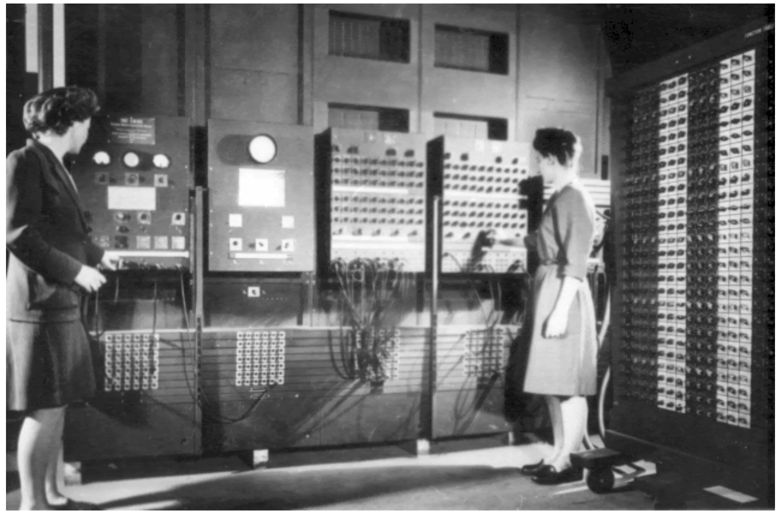
Programming == Problem Solving

- You look at the problem to solve
  - Clarify the problem and constraints
- Break it up into *smaller* parts (Divide)
- Outline the steps needed
  - Solve each step (Conquer)
- Reassemble the pieces (Glue)
- Completed program

What if we want to do the same set of instructions again?

REUSE CODE

```
public class BasicCalculations {
   public static void main(String[] args) {
      double value1, value2, sum, subtraction;
      double multiplication, division;
      value1 = 12;
      value2= 30;
      sum = value1 + value2;
      subtraction = value1 - value2;
      multiplication = value1 * value2;
      division = value1/value2;
      System.out.println(value1 + " + " +
                      value2 + " = " + sum);
      System.out.println(value1 + " - " +
                     value2 + " = " + subtraction);
      System.out.println(value1 + " * " +
                     value2 + " = " + multiplication);
      System.out.println(value1 + " / " +
                     value2 + " = " + division);
   }
}
```



Programmers Betty Jean Jennings (left) and Fran Bilas (right) operate ENIAC's main control panel By United States Army (Image from http://ftp.arl.army.mil/~mike/comphist/) [Public domain], via Wikimedia Commons

Methods: Reusable Code

The ENIAC women pioneered reusable code

Methods

- Are ways to modularize / reduce the code

- Methods are designed to implement a specific function in our program
  - Small / Repeatable blocks

- Methods are defined inside a class

- We can call a method as many times as we want

Methods

- When we build a method inside a class that has a main method, the method need to have the following format definition:
  - public static <typeOfReturn> methodName(<listOfParameters>)
    - public static: access mode is public, can be called without restrictions
      - static: method belongs to the class (will talk about this later in detail)
    - <typeOfReturn>
      - void: no return
      - Any class or primitive type
    - <listOfParameters>
      - list of parameters separated by comma (,)
      - Each parameter needs to have it type and name

Methods

- **public static void** main(String [] args)
  - access mode: public static
    - static: method belongs to the class
  - void: method does not have any return
  - main: name of the method
  - String [] args: array of Strings (we will talk about this later)

Methods Activity

```
import java.util.Scanner;

public class IdentifyingMethods {

    public static int module(int num1, int num2){
        return num1%num2;
   }

  public static double average(int num1, int num2, int num3){
      return (num1 + num2 + num3)/3.0;
   }

 public static void end(){
    System.out.println("End of the Program.");
    System.out.println("Goodbye!");
 }
#to be continued in the next slide
```

- Identify:
  - Name of the class
  - For each method
    - Access mode
    - Return
    - Name
    - List of parameters

Methods

```
#continued from previous slide

public static void main(String args []){
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter 3 integer numbers: ");
        int value1 = scanner.nextInt();
        int value2 = scanner.nextInt();
        int value3 = scanner.nextInt();
        double avg = average(value1, value2, value3);
        System.out.println("Average of the values entered: " + avg);
        System.out.println("Module " + value1 + "%" + value2 + " = " + module(value1,value2));
        end();
    }
}
```

Let's understand this main method!

Methods

```
Scanner scanner = new Scanner(System.in);
System.out.println("Enter 3 integer numbers: ");
```

Creates and object scanner from the class Scanner

Print to the console the message “Enter 3 integer numbers:” and goes to the next line.

```
int value1 = scanner.nextInt();
int value2 = scanner.nextInt();
int value3 = scanner.nextInt();
```

Uses the method nextInt from Scanner class to read an int number.

To call a method using an object:

nameObject.nameMethod(<parameters>)

Example: scanner.nextInt()

Scanner – is the class (S capitalized – indicates class)

scanner – is the object

Methods

Calling method average. Since average expects 3 int values as parameters we need to send 3 int values to the method. The method return is stored in the avg variable.

```
double avg = average(value1, value2, value3);
System.out.println("Average of the values entered: " + avg);
```

Concatenates the String with the avg variable, prints this new String and go to the next line.

Methods

Will print the return of the module method. Since module needs two integer parameters, we sent value1 and value2.

```
System.out.println("Module " + value1 + "%" + value2 + " = " +
module(value1,value2));
```

```
end();
```

Calls the end method, which is a void method, meaning that there is no return

Quick Practice Pseudocode

- As a group, block out / outline what you need to do for the **longDivision** method.
  - It needs to print both the quotient and the remainder of value 1 long divided by value 2
  - This outline is called pseudocode, and often done in *comments* for example
    - // multiple value1 and value2 together – store in answer
    - // Print hello doc, the answer is _answer_
  - Focus on major "sub tasks" of the method task
  - Most methods should have one task, with a couple small things needed to accomplish that task
    - That is it!

```
public static void longDivision(int value1, int value2) {
    // pseudocode here

}
```

Called a method stub!

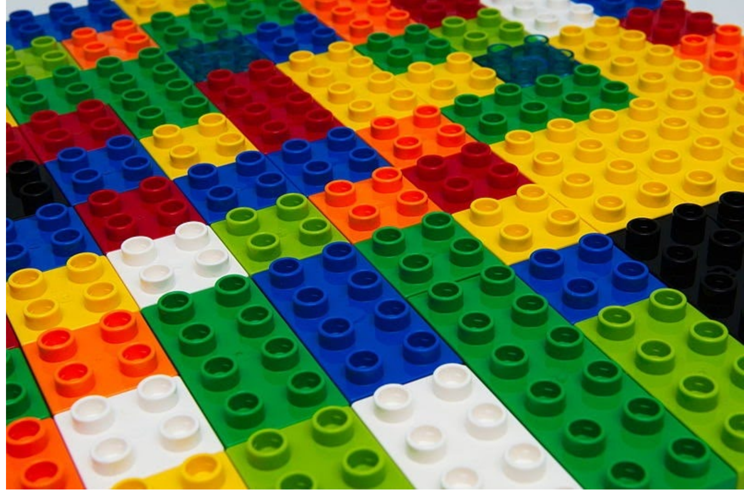
Putting it together

```java
public static void longDivision(int value1, int value2) {
    int quotient = value1 / value2;
    int remainder = value1 % value2;
    System.out.println(value1 + " / " + value2 + " = " + quotient);
    System.out.println(value1 + " % " + value2 + " = " + remainder);
}

public static void main(String[] args) {
   longDivision(12, 30);
   longDivision(100, 5);
   longDivision(1000, 52);
}
```

Coupling Ideas Together: Objects

Objects are Building Blocks

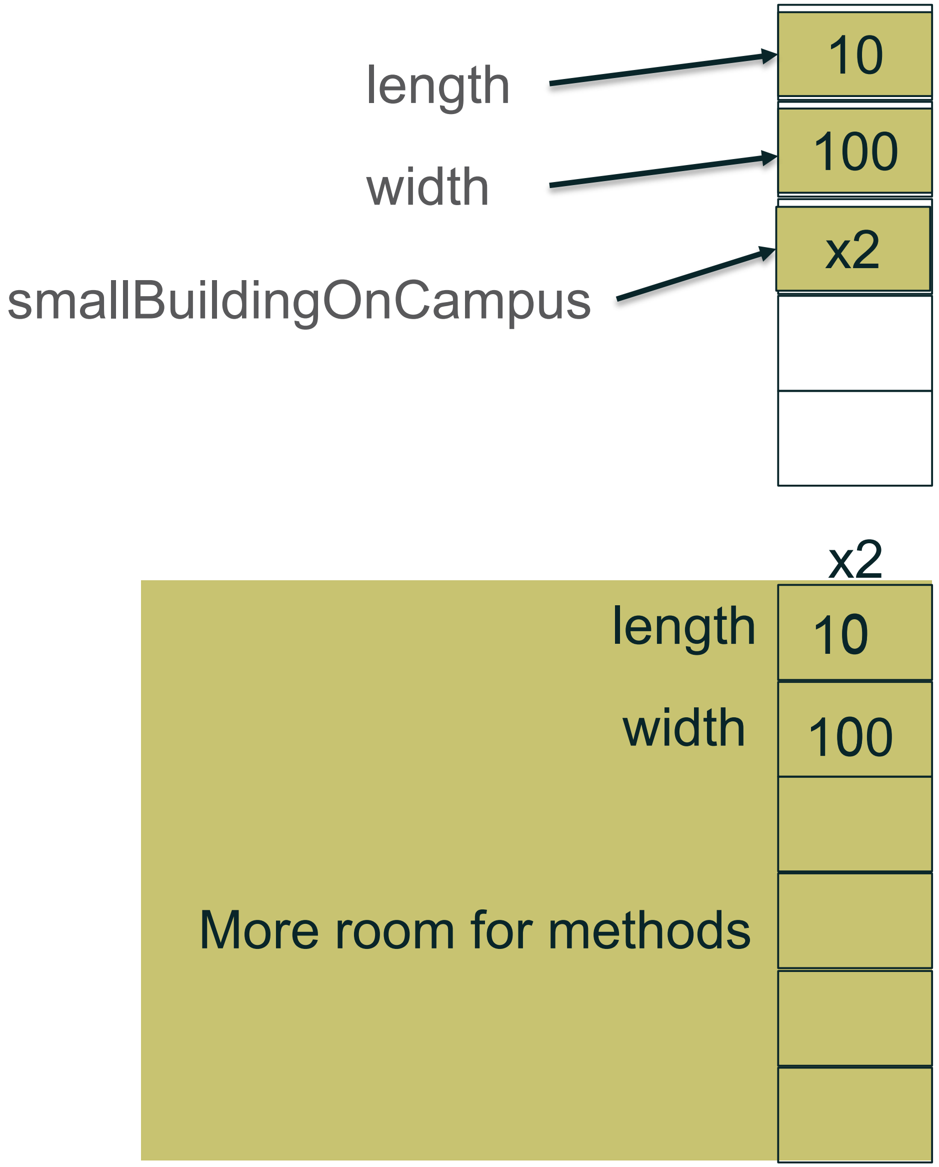


© Ralf Roletschek - Published with permission. Read full copy information on wikicommons.

- Think of LEGOs
  - Blocks
  - Assembled in different ways - creates new and interesting things
- Objects contain information in a logical order
- Most objects use the **new** keyword
  - `MyCoolObject obj = new MyCoolObject(); // this reserves room in memory`
  - `obj.myCoolMethod();`
- We will keep coming back to this
  - Important to know - methods belong to Objects
  - Even methods that **you** write

Memory Example

```java
public static void main(String[] args) {
  int length = 10;
  int width = 100;
  Rectangle smallBuildingOnCampus = new Rectangle();
  smallBuildingOnCampus.setLength(length);
  smallBuildingOnCampus.setWidth(width);
  System.out.println(smallBuildingOnCampus.getArea());
}
```



length → 10

width → 100

smallBuildingOnCampus → x2

x2

length 10

width 100

More room for methods

Rectangle? Is a Class

- In Java (an Object Oriented Programming – OOP - language) - everything must be in a class!

- You can create Objects out of classes
  - use the new keyword
  - Rectangle myHouse = new Rectangle();
  - Scanner scnr = new Scanner(System.in);

- new Reserves memory for that ‘instance’ / object

Rectangle Class

```java
public class Rectangle {
    private int width;  // instance variables
    private int length;

    public void setLength(int length) {
        this.length = length;
    }

    public void setWidth(int w) {
        width = w;
    }

    public int calculateArea() {
        return width * length;
    }
}
```

- **Instance variables**
  - Represent the data (attribute)
- **private**
  - Means that only the class can access those values directly
- **public**
  - Others can access public methods
- **this –** keyword
  - Means "this object/instance"
  - Helps keep track of which variable
  - Common practice
    - But not required

Class Constructor

```
    public Rectangle() {
        width = 0;
        this.length = 0;
    }

    public Rectangle(int w, int l) {
        width = w;
        length = l;
    }
```

OR

```
    public Rectangle() {
        setWidth(0);
        setLength(0);
    }

    public Rectangle(int w, int l) {
        setWidth(w);
        setLength(l);
    }
```

- Special method that has the name of the class
- No return not even void
- Can be overloaded
  Meaning that we have more then one implementation for the method
  Same name with different parameters
- Rectangle() – no parameters
- Rectangle(int w, int l) – with parameters
- You can call methods inside of the constructor
- Usually, you call mutators (sets) methods, if the class has them defined

Class Instance Methods

```
    public void setLength(int length) {
        this.length = length;
    }

    public void setWidth(int w) {
        width = w;
    }

    public int calculateArea() {
        return width * length;
    }
```

- **static methods**
  Belongs to the class / self-contained

- **instance methods**
  Need to access instance variables
  Uses the data in the object
  Unique to that instance

Use Tables!

- Every time you are:
  - In a new method
  - See a **new** keyword
- Draw a table

```
int small = 5;
Rectangle one = new Rectangle();
one.setLength(10);
one.setWidth(10);

Rectangle two = new Rectangle();
two.setWidth(small);

small = 12;
```

Current Method	
small	~~5~~ 12
one	@rec.one
two	@rec.two

@rec.one	
length	10
width	10

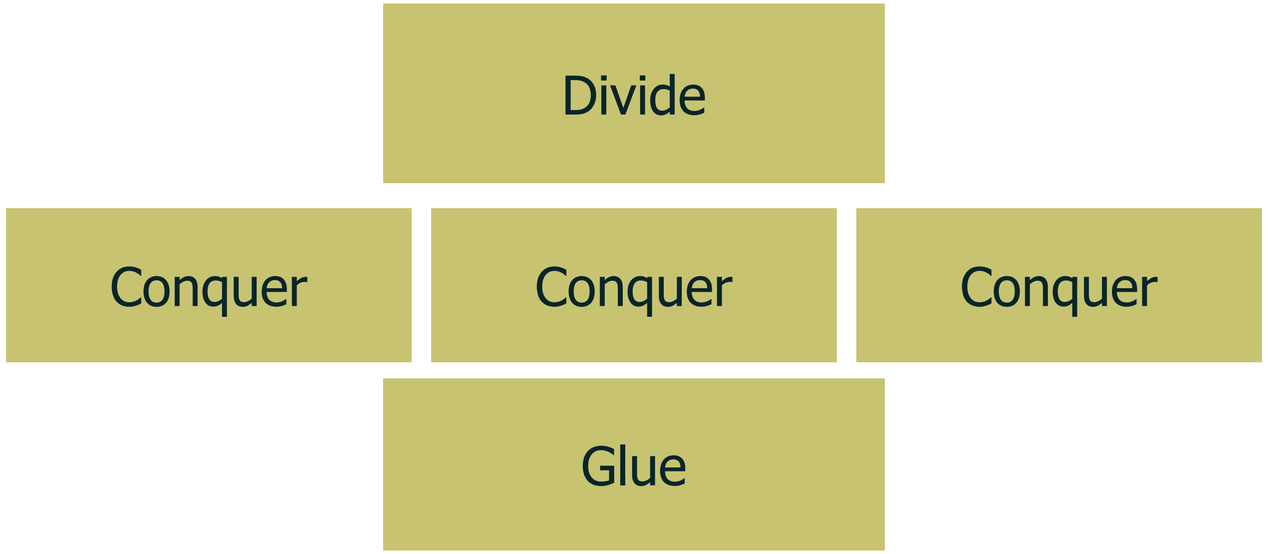
@rec.two	
length	0
width	5

Why Methods and Objects?

Fun Fact:
Software Engineers, Andy Hunt and Dave Thomas, are credited with first using the the term for coding in the **The Pragmatic Programmer**

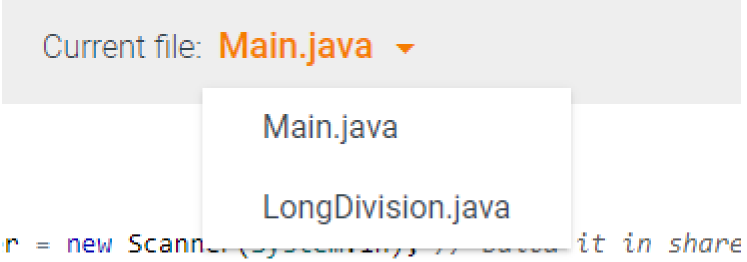
DRY Code

- Code should be DRY
  - **D**on’t **R**epeat **Y**ourself
- Code should be
  - Reusable
  - Small Snippets
- Reusable code
  - Only write once
  - Use in multiple applications
- Java
  - Objects are blocks of information, with reusable code / methods
  - Methods are blocks of reusable code
    - Ideally, no more than 20 instructions
  - **CLUE**: If you are cutting and pasting code - it should be a method
    - Really, that happens



Coding Practice

- Go canvas to access In Class: Long Division
- We will build a long division object, that the main method will call
- Notice – **two** classes!
- Time pending, you should build the memory tables for your code!


Current file: Main.java
Main.java
LongDivision.java
r = new Scann
it in share