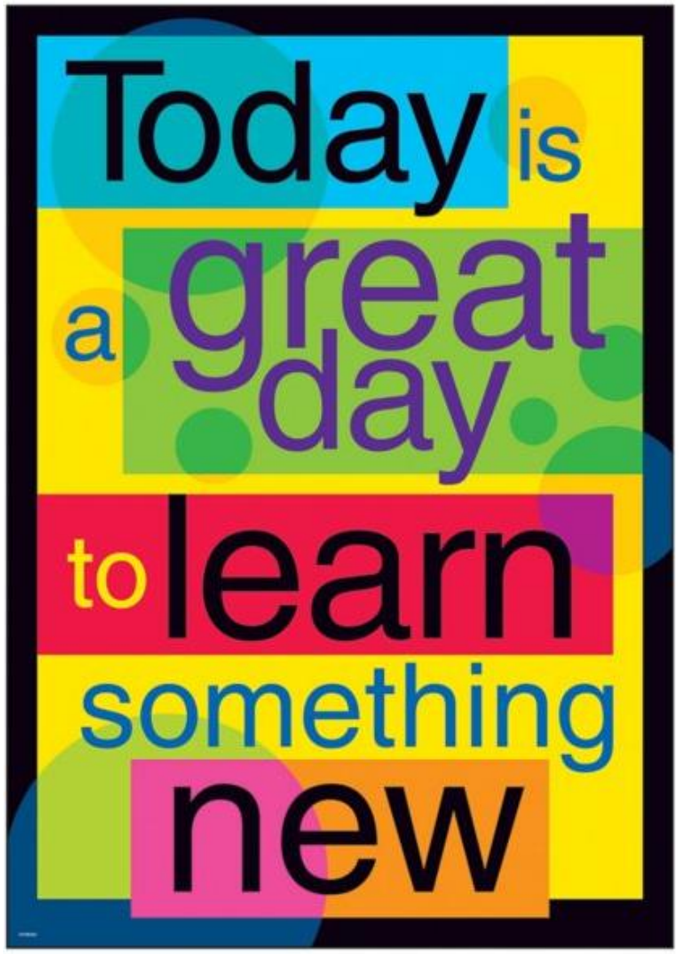
Objects and Methods

Colorado State University
Department of Computer Science
Slides Originally Created by Albert Lionelle (Albert.Lionelle@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 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"

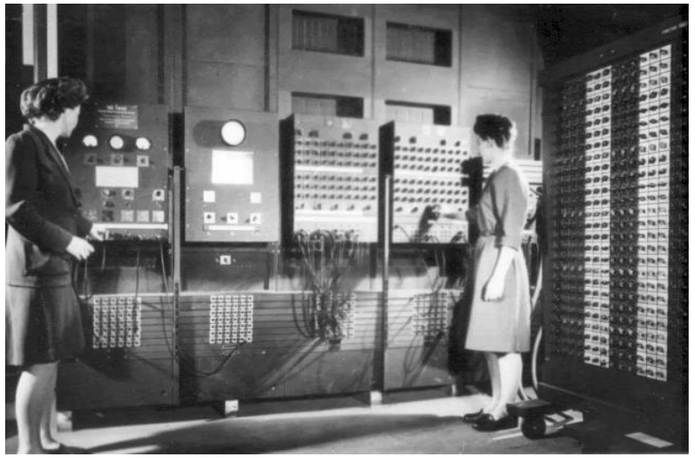
A. public double calcArea(double width, double height) {}

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

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

D. All listed

E. None listed



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

Reusable Code

```
public static void main(String[] args) {
     double quotient, remainder;
     int value1 = 12;
     int value2 = 30;
     quotient = value1 / value2;
     remainder = value1 % value2;
     System.out.println(value1 + " * " + value2 + " = " + quotient);
     System.out.println(value1 + " / " + value2 + " = " + remainder);
     int value3 = 100;
     int value4 = 5;
     quotient = value3 / value4;
     remainder = value1 % value4;
     System.out.println(value3 + " * " + value4 + " = " + quotient);
     System.out.println(value3 + " / " + value4 + " = " + remainder);
     int value5 = 1000;
     double value6 = 0.52;
     quotient = value5 / value6;
     remainder = value5 % value6;
     System.out.println(value1 + " * " + value2 + " = " + quotient);
     System.out.println(value1 + " / " + value2 + " = " + remainder);
 }
```

ERROR!

ERROR!

ERROR!

Is this a good way to reuse code?

Methods

- Are ways to modularize / reduce the code
- Small, repeatable blocks
- **One** idea at a time

Scope | Memory Space | Return type | Identifier | Parameters

```
public static void longDivision(int value1, int value2) {
    // block of code in here

}
```

Tip: Thinks of static as **s**hared memory space.
If the method is **s**elf contained, it should be **s**tatic

Return Types

- Return Types
  - **Critically important**
  - Can return a primitive, array or an object
  - void is how you say the method returns nothing

- Whenever you call a method
  - Assume the return type is what is returned
  - Think of replacing the method name with the answer

```
public static String getName() {
    return "Melody Pond";
}


public static void main(String args[]) {

    System.out.println("Hello, my name is " + getName());


}
```

The program output is

Hello, my name is Melody Pond

Notice: Melody Pond simply replaces ‘getName()’

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!

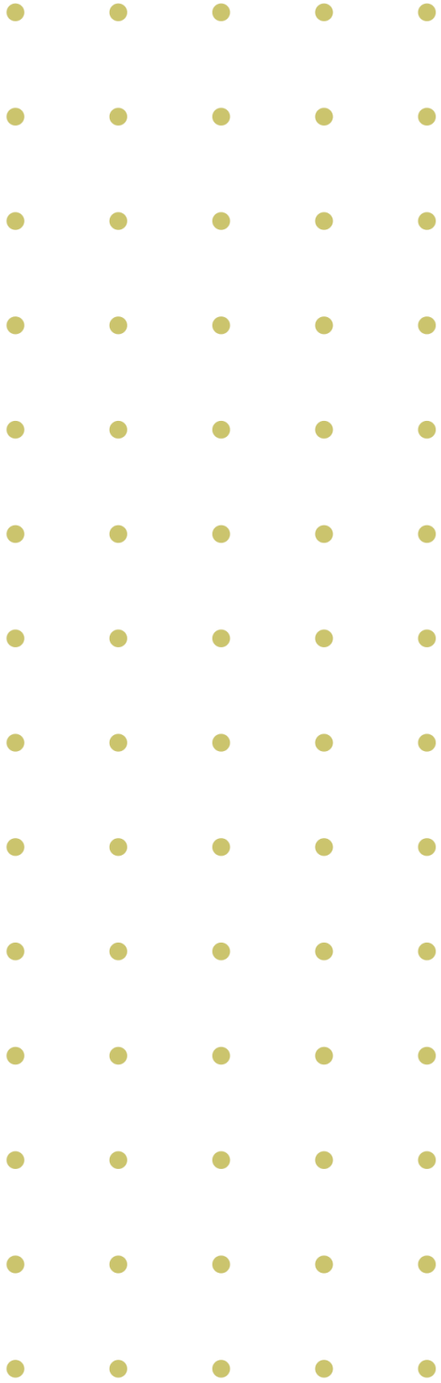
```
public static void longDivision(int value1, int value2) {
    // pseudocode here

}
```

Putting it together

```
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);
}
```



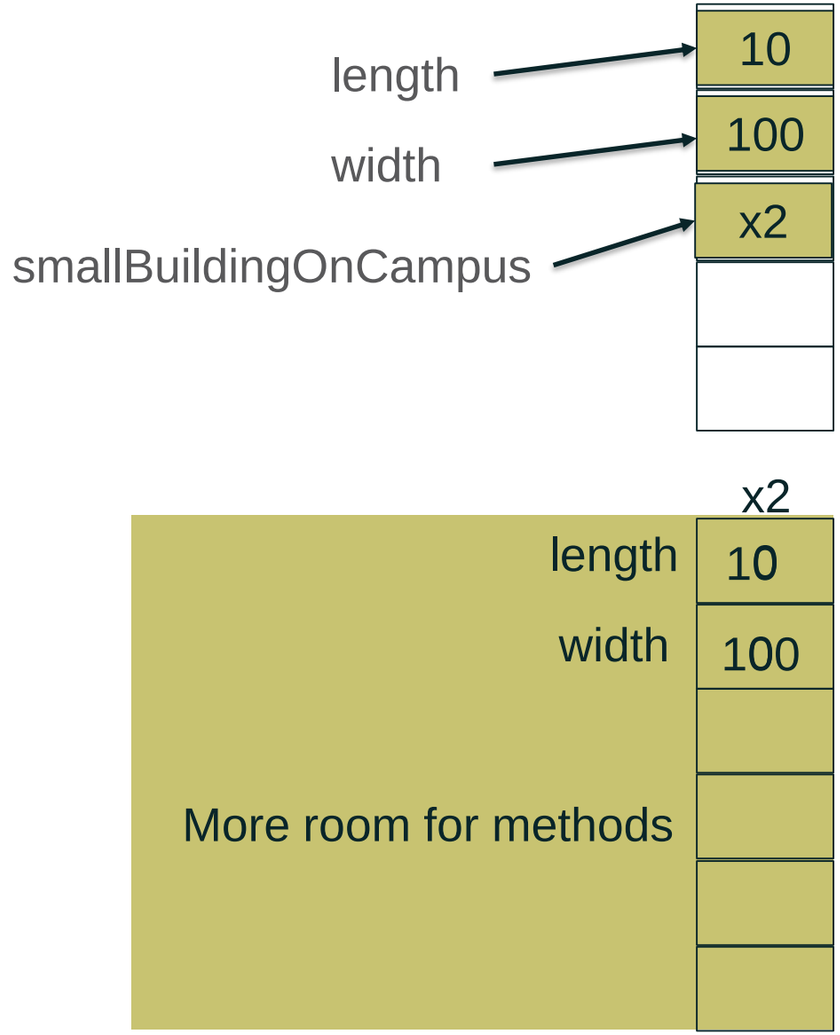
Coupling Ideas Together: Objects

Objects are Building Blocks

- 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

```
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());
}
```



Rectangle?

```
public class Rectangle {
    int width = 0;  // instance variables
    int length = 0;

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

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

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

- **this –** keyword
  - Means "this object/instance"
- Helps keep track of which variable
- Common practice
  - But not required

Instance Methods

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

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

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

- static methods
  - Shared / often Self contained (remember **S**)
- 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
- <u>Draw a table</u>

```
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

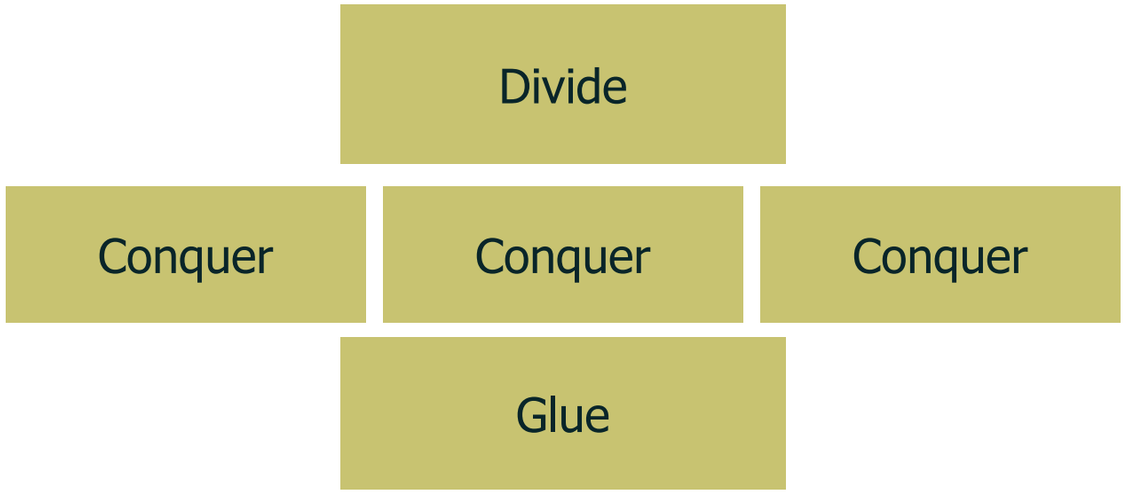
@rec.one	
length	10
width	10

@rec.two	
length	0
width	5

Why Methods and Objects? DRY Code

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

- 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!