Methods

Called functions in some languages
Examples

- Have already seen lots of methods from String
  - String s = "eat my shorts";
  - int lengthOfString = s.length();

- Have even seen methods in JavaScript
  - just called functions

- Now let’s make our own methods.
Start With Classes

To create our own methods, must explain classes

Classes are separate programs.

- Examples
  - Every program we’ve written.
    - public class MyHomework
      {
        ...
      }
  - Random class.
    - someone else just wrote a program that deals with generating random #’s.
    - public class Random
      {
        ...
      }

They’re more than that, but take OO programming to learn the extra-cool side of classes.
Classes Without “main”

- Not all classes need a “main()”
  - “main()” tells the program where to start.
  - But if you are using String class and myHomework, can’t start in both places!
    - The String class does not have a “main()”
    - But you put a “main” in your program.
      - So Java interpreter knows to start with your program.
      - General rule: only one “main”!

Rule can be broken, but advanced topic, and don’t recommend it.
Methods Live In Classes

If peaked inside the String class would see methods called
- length
- charAt
- substring
- etc.

Their code looks similar to the “main()”
- public int length()
  
  { …

Notice this business of “returning” something
- Like a math function.

Why int?
Because this tells us what it returns.
It returns an integer that is the length.
Methods Are Similar to Math Functions

In math we say
- \( f(x) = x^2 \)
  - To “run” this function we pass it a value
    - e.g., \( f(2) \)
  - It returns a value
    - e.g., 4.

- \( f(x, y) = x^2 + y^2 \)
  - \( f(4, 1) \) returns 17
  - \( f(3, 5) \) returns 34
Language “Functions”

- Can do the same thing in Java, C, etc.
  - but call them **methods**
  - \( f(x) = x^2 \) becomes

```java
public class MethodExample {
    public int f(int x) {
        int theSquare = x * x;
        return theSquare;
    }
}
```

- This means the method takes an integer as an argument (or parameter).
- Program will crash if I try to give it a double, boolean, or anything else.
- Says that the method will return an integer. Won’t compile if try to return anything else.

Here is where it returns the value (when done calculating).
Meaningful Names

- Can name our method anything.
  - “f” is so uninformative.
  - How about “squareTheNumber”?

```java
public class MathStuff
{
    public int squareTheNumber(int x)
    {
        int theSquare = x*x;
        return theSquare;
    }
}
```

- Ahh, much better.
- Now when I use it, I’ll know what it does.
How Do You Use Methods?

1. Put them in their own class.
2. Then call them with a “.”
   - just like calling the String methods

3. But need to **instantiate** the class!
   - Just fancy language for same thing we do with basic types
     - int n = 3;
     - char c = ‘A’;
Similarly, suppose your method is in class called MathStuff

MathStuff stuff = new MathStuff();

variable name

assignment

“value” assigned!
type
Running The Method

Now can say
- `stuff.squareTheNumber()`

Might have more than one method in the same class.
- e.g., `cubeTheNumber()`
- Then could also say
  - `stuff.cubeTheNumber()`
public class MathStuff
{
    public int squareTheNumber(int x)
    {
        int theSquare = x*x;
        return theSquare;
    }
}

public class Example
{
    public static void main(String[] args)
    {
        System.out.println("About to use my own method…");
        int i = 3;
        MathStuff m = new MathStuff();
        System.out.println(m.squareTheNumber(i));
        System.out.println(m.squareTheNumber(5));
    }
}

Notice method defined with an x.
But I gave method an i.
That’s ok! It will take **ANY** integer.
Code for Multiple Methods

```java
public class MathStuff {
    public int squareTheNumber(int x) {
        int theSquare = x * x;
        return theSquare;
    }

    public int cubeTheNumber(int n) {
        return n * n * n;
    }
}
```

Note the `x` and `n`. Can use any variable name.

I returned the calculation without any intermediate variables. That’s ok.
Local Variables

Will this work?

```java
public class MathStuff {
    public int squareTheNumber(int x) {
        int theSquare = x*x;
        return theSquare;
    }

    public int weird(int n) {
        return n*n*n + x;
    }
}
```

NO! The x and n have local “scope”. They only work within the `{…}` where they are defined.

So this x is undefined!
public class AnotherExample {

    public static void main(String[] args) {
        int i = 3, j = 5;
        MathStuff m = new MathStuff();
        int k = m.squareTheNumber(i) + m.cubeTheNumber(j);

        //What number gets printed?
        System.out.println(k);
    }

}
Methods In Main Class

No rule that says you can’t put methods in the class that has the main.

- General rules:
  - Group similar methods into the same class.
    - class called Math has sin, cos, square, etc.
    - class called Car has accelerate, fillWithGas, turn, etc.
    - class called Chalkboard has … (you tell me)

- Usually keep class with main free of other methods.
  - clearer
  - but sometimes simpler for small programs to put method in same class
Example of Method in the Class With the Main

```java
public class InsultUser {
    public static void main(String[] args) {
        InsultUser insult = new InsultUser();
        insult.printInsult();
    }

    public void printInsult() {
        System.out.println("You look like a computer science professor.");
    }
}
```

Notice that there is nothing returned, and nothing passed in!
Your Turn

Help me write code that

1. Has one method.
   -- Prints n asterisks “*” where n is a parameter.
2. Call this method from the main().
3. Use this method to print out.

****
***
**
*

Just like the lab, but now made easier with a method.