

Static Methods

AP Computer Science

Credit: Slides are modified with permission from Barry Wittman at Elizabethtown College

This work is licensed under an [Attribution-NonCommercial-ShareAlike 3.0 Unported](#) License

Methods

Adding is Great, but...

- There are operations beyond +, -, *, /, and % you want to do with numbers
- **Java** has those built-in because the computer can do those directly
- A number of other operations can be done by calling **methods**

Methods

- A method is a piece of **Java** code packaged up so you can reuse it
- Usually, a method will take some input and give some output
- `System.out.println()` is an example of a method
- Using a method (calling a method) always requires parentheses

Method Example with `sin()`

- The `sin()` method allows you to find the sine of an angle (in radians)
- This method is inside the `Math` class
- The answer it gives back is of type `double`
- To use it, you might type the following:

```
double value = Math.sin( 2.4 );
```

Method Syntax

Unless the method is inside your class, you must supply a class name and a dot

If your method takes input, you put it inside the parentheses, if not, you leave them empty

```
result = class.method( input );
```

You can store the result of the method, as long as the variable matches the type the method returns

Next, you must give the method name you are calling

More Math Methods

Returns	Name	Job
<code>int</code>	<code>abs(int a)</code>	Returns the absolute value of an int value.
<code>double</code>	<code>abs(double a)</code>	Returns the absolute value of a double value.
<code>double</code>	<code>pow(double a, double b)</code>	Returns the value of the first argument raised to the power of the second.
<code>double</code>	<code>sqrt(double a)</code>	Returns the double value closest to the true mathematical square root of the argument value.
<code>double</code>	<code>random()</code>	Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.

Here is a link to the entire [Math library](#)

random()

- The only tricky one from the previous slide is the random() method.
- random() returns a value between 0.0 and 1.0 exclusive of 1.0

```
double rand1 = 0.0;
int rand2 = 0;
int rand3 = 0;
int rand4 = 0;
int rand5 = 0;
rand1 = Math.random(); //value between 0.0 and 0.99..
rand2 = (int)(Math.random() + 5); //result is?
rand3 = (int)(Math.random() * 10); //result is?
rand4 = (int)(Math.random() * 10 + 5); //result is?
rand5 = (int)(Math.random() * 20 + 20); //result is?
```


Static Methods

Why Methods?

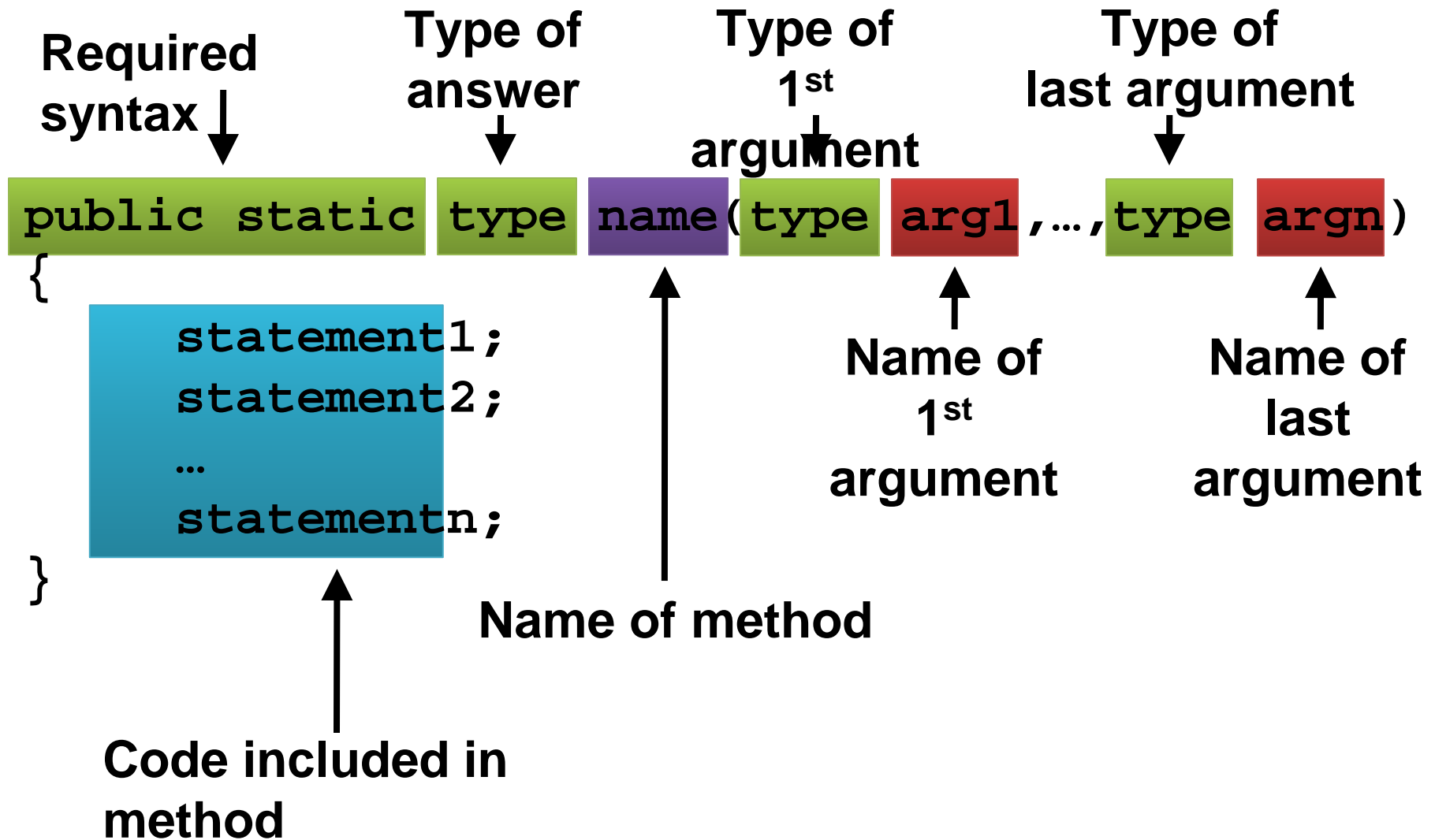
- All of the code we have written so far has been inside of the `main()` method
- What about a big program?
- The `main()` method is going to get really long and difficult to read

Idea of a method

- Methods allow you to package up some code to run over and over
- Methods usually take some input (like numbers or `strings`) so that they can be customized
- Methods **often** give back an answer (like the square root of a number)

Declaration of Static Methods

Anatomy of a static method



Simple method example

- Given two integers, find the smaller:

```
public static int min(int a, int b)
{
    if( a < b )
        return a;
    else
        return b;
}
```

Why `static`?

- `static` methods are methods not connected to a particular object
- They perform a simple or specific task
- We are going to focus on `static` methods until after we cover objects
- For now, the definition of every method will include the `static` keyword

Value returning methods

- It is possible to divide methods into two types:
 - Value returning methods
 - Void methods
- Value returning methods give an answer:

```
int small = min(x, y);
```


Void methods

- Void methods are declared with `void` as their return type

```
public static void help(int times)
{
    for( int i = 0; i < times; i++ )
        System.out.println( "Help!" );
}
```

- Void methods only do something
- If you try to save the value they return, there will be a compiler error

return statements

- Like most code in Java, the code inside of a method executes line by line
- You can put in `return` statements
- **A method will stop executing and jump back to wherever it was called from when it hits a `return`**
- The `return` statement is where you put the value you want to return back to the caller

Calling Static Methods

Calling methods

- Defining a method is only half the game
- You have to **call** methods to use them
- Calling a method means giving a method the **parameters** (or **arguments**) it needs and then waiting for its answer
- You have done some method calls
 - `System.out.println()`
- You can call your own methods the same way

Calling syntax

- Proper syntax for calling a static method gives first the name of the class the method is in, a dot, the name of the method, then the arguments

```
Class.name(arg1, arg2, arg3);
```

- If the method is in the same class as the code calling it, you can leave off the **Class**.
- If it is a method returning a value, you can store the value into a variable of the right type

Scope

- Variables from outside of the method do not exist unless they have been passed in as parameters

```
public static int add(int x, int y)
{
    int z = x + y;
    return z;
}
```

- No matter how complex a program is, inside this method, only **x**, **y**, and **z** variables exist


Binding

- A magical process called **binding** happens which copies the values from the calling code into the parameters of the method
- The calling code can use variables with the same names, with different names, or even literals
- The method does not change the values of the variables in the original code
- Remember, it only has copies of the values

Binding example

```
int a = 10;  
int x = 3;  
int y = add( 5, a ); //y contains 15 now
```

```
public static int add(int x, int y)  
{  
    int z = x + y; //5 + 10  
    return z;  
}
```



- No connection between the different **x**'s and **y**'s

Static method rules

1. Start a method with the keywords `public`
`static`
2. Next is the return type
3. Then the name of the method
4. Then, in parentheses, the arguments you want to give to the method
5. Inside braces, put the body of the method
6. Include a `return` statement if you are wanting to return a value