Methods

In other programming languages they are called functions or subroutines.

You write a method like this:

```
//extract the first n letters from a String
String firstLetters(String s, int n) {
    String str = s.substring(0,n);
    return str;
}
```

Notes:

- `String` is the data type that is returned from the method.

- If the method does not return anything, you must specify `void`

- `firstLetters` is the name of the method

- `String s, int n` are the parameters or arguments.
  They must have a data type as well as a variable name. The variable name is local and does not have to be the same as the variables in the calling program.

- `return str;` Unless the return type is void, you have to return a value (that matches what you said you would return).

- To call the method above, you can write:
  ```
  String s2 = firstLetters("goodbye", 4);
  ```
  Note that this will not actually work since the text uses " " instead of " " (straight quotes).
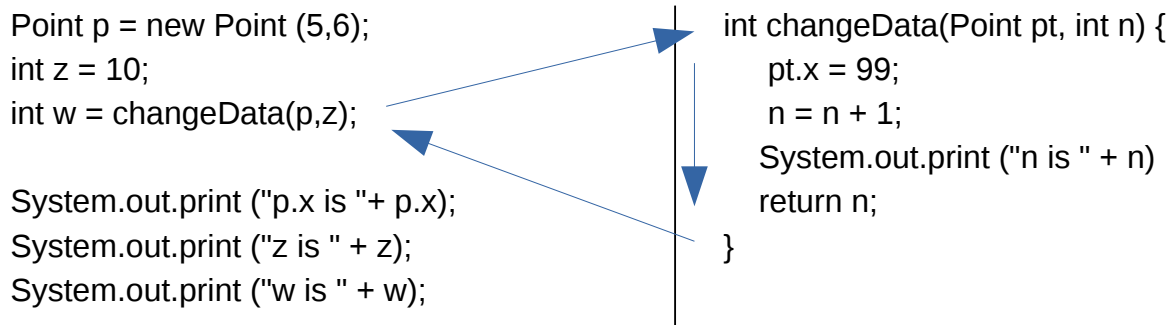
- This will also work just fine
  ```
  String text = "goodbye";
  int num = 4;
  String s = firstLetters(text, num);
  ```

- When calling the method, it doesn't matter that I use the varible num, but the method uses n.

- **The arguments for a method MUST be in the correct order.**

- **Methods can only return 1 thing.**
  So you cannot have a quadratic formula method like this because it can only return one root.

  ```
  double quadratic (double a, double b, double c) {
       ... do calculations ...
       return root1;
  }
  ```

- If you want to return more than one thing, you have to either return an array or an object.

```
Point p = new Point (5,6);                          int changeData(Point pt, int n) {
int z = 10;                                              pt.x = 99;
int w = changeData(p,z);                                n = n + 1;
                                                        System.out.print ("n is " + n)
System.out.print ("p.x is "+ p.x);                      return n;
System.out.print ("z is " + z);                     }
System.out.print ("w is " + w);
```

This will print out:

n is 11

 p.x is 99

 z is 10

 w is 11

You can see that methods only get a **COPY** of primitive data types, but **the object reference is passed** to the method, and so it gets access to the whole object and can change variables inside the object.

➔ Primitive data types cannot be changed by a method

➔ Objects can be changed by a method.

Two other topics regarding methods are

1. scope

2. method overloading (not quite as important).

I have some old pages about this in Unit 3 here: http://quarkphysics.ca/ICS3U1/javaNotes.htm