Methods

Level 1: **Group instructions** together (in curly braces) and give them a **name**. (You must use `public static void` in front)

```java
public static void printPage(){
    // what your method will do
}
```

Level 2: If you want those instructions to work with different data: pass it as an argument.

```java
public static void printPage(String drawing){
    // what your method will do
}
```

Level 3: If the method computed useful data and you want that data outside of the method, use `return`.

```java
public static int printPage(String drawing){
    // what your method will do
    int myData = 5;
    return myData;
}
```

Two parts for the method mechanism:

- **Method definition**
  - method name,
  - parameters – place holders for the actual arguments that will be passed at call time
  - instructions to be executed,
  - return value
  - *** first line (with the method name) gives the method signature: the convention for using it

- **Method call**
  - Must match **method signature**
  - This is the time when actual data is tied to the placeholder arguments in the definition.

*** One cannot work without the other:
- Definition with no call: will never run those instructions.
- Call without definition: Java will not recognize the method name.

Methods **do not communicate** with each-other. Each method is like a different classroom.

a) **Do not recognize each-others variable names**.
   (John from this room is not in the other room. If there is a John there, that is a DIFFERENT student that so happens to be named John.)
   - Draw picture
   - Check with the Debugger
   - Practice: see code where one method tries to use variables from the other.

b) **Do not run in parallel**: when one method starts, the other one is halted.
   (You cannot be in two rooms at the same time)

c) **One method does not know what the other one did**.
   (*** Even within the same method, the program does not have any memory of what it just did. It only ‘remembers’ variables that were created.)
d) A method can return at MOST one piece of data (object, array, or primitive data).
   a. If you need to return more pieces, must put them in an array or ArrayList

e) **Methods communicate only through**
   a. **arguments** (Level 2) and
   b. **returned value** (Level 3).