**Methods:**

Methods: instructions, action, verb functions

Static Methods vs Non-Static Methods

^Class Method ^Object/Instance Method

Non-Static Methods

* Task that an object can preform
* Specific to an object
* Specific to variables

Ex: getters, setters, person dance, Jack jump

Static Methods

* Called on a class
* Help methods when executing a main method
  + Handle subtasks
  + Enhance reusability
* Handle generic calculations

Ex: printing, finding avg of numbers, square root

Implementing Static Methods

public static <return type> <MethodName>(parameters) {

//Method Body

}

Modifier:

* Public – called outside of the author class
* Private – can only call within the class

Return Type

* Any data type that should be given back to the caller when the method is executed
* Void = doesn’t return anything
* Method writer must return the value of the type
* Can have multiple return statements, but once one is execute, the method breaks

Method Name

* Follow variable naming rules
* Methods can share the same name but must have different parameters

Parameter List

* Zero 🡪 a lot
* Identifies the type of value required of the method

Calling Static Methods

* From within the same class
  + Method (arguments);
* From outside of the class
  + Class.method(arguments);

Parameters/Arguments

* Don’t have to have the same name

Ex:

public static void method(int x, double y) {} //double y and int x are parameters

psvm{

method(5,2.8); //2.8 and 5 are arguments

}

Scope and Lifetime

* **Scope:** Where you can use a variable
  + useable you can use the variable within the body it was declared
* **Lifetime:** When the variable exists in a program

Pass-by-value

* When passing a primitive as an argument, you only pass the value not the variable itself (“copying the variable”)
* psvm() {

int x=5;

method(x);

sout(x);

**}**

psv methodA(int y) {

y = 7;

y++

}

^^This just makes x equal to 7 only during methodA, then increments it, then prints x, which is still equal to 5 in the main method.