A Deeper Dive into Parameters

1. **What is the difference between a class variable and a class constant? When would you use each?**

- A class variable can be changed, a class constant cannot.
- Use a class variable when you have data you want to share with the whole class **that can/will change while the program is running.**
- Use a class constant when you have data you want to share with the whole class **that cannot/should not be changed.**

2. What does it mean to “parameterize” a method? When/why would you want to do this?

- Set up a method so that custom values can be given to it (“passed to it”) at runtime.
- Allows generalizing the behavior of the method.
- A parameter is a value given to a method when it is called.

3. What is the difference between “declaring a parameter” and “passing a parameter?”

- **Declaring a parameter** means you create a variable inside the parentheses in the header of the method. This variable will “receive” and hold the value passed to the method when it is called.

- **Passing a parameter** means when you call a method, you give it specific value(s) (parameters) for it to use and customize how the method will behave.

4. What is the difference between a class variable and a parameter? When would you use each?

- A class variable is available to all methods in a class.
- A parameter is available to only the method it is associated with.
- Use a class variable **when you want to share info with more than one method.**
- Use a parameter **when you want to generalize/customize a method, or when you have info to share with only one method.**

5. What does it mean to “overload a method?” How does it relate to “method signature?

- Define two or more methods with the same name but different parameters.

- **Method signature** is the name of the method along with its number and type of parameters.

- You overload a method by creating one with the same name but different method signatures…
    - i.e. same name different number and/or type of parameters.

6. In a method, if you change the value of a parameter does it change the value back in the code that called the method? Why or why not?

- No it does not.
- Because when you call a method passing a value, a copy of that value is made and placed in the parameter variable declared in the method header.
- We call this “pass by value.”

7. Research online: what is the difference in Java between “pass by value” and “pass by reference?”

- **Pass by value** means when you call a method and pass a value to it, a copy of that value is made for the method to work with. If the method changes the passed value, it does NOT affect the original value.

- **Pass by reference** means when you call a method and pass a value to it, the actual value is given to the method so that if the method changes the passed value, the actual value is changed.

- **Java is exclusively a PASS BY VALUE language!**

8. What is the difference between “calling a method” and “passing a value to a method?” How are they related?

- **Calling a method** simply means causing that method to execute/be used.
- **Passing a value to a method** means when you call a method you give it a value to work with.
- They are related in that you have to call a method in order to pass it a value.