Domain 1: Java Fundamentals

Java File

- needs main method
  - ↳ declared as static
- String Args []
  - ↳ not used often
  - ↳ used when needing text input for example command line
- example of boilerplate:

```
class Main {
    public static void main (String args[]){}
}
```

- main → name of function
- void → doesn't need return
- static → able to be called w/out instance
- public → can be called anwhere

Compiling / Running

i. javac Main
   - ↳ compiles the file, creates another final run file that must be called

ii. java Main
   - ↳ runs the compiled file

ii.a. java Main 9 8
   - ↳ runs w/ 2 inputs (ints) in this scenario this input is not taken from the user, but is taken from the command line arguments that are entered

Input / Output (i.)

Output →

- System.out.
  - print → prints next to eact other
  - println → prints on a new line

Input / Output (ii.)

Input

- built in Scanner class → import from .utils tho
- example

```
import java.util.Scanner;
class Main{
public static void main (String args[]){
        Scanner in = new Scanner(System.in);
        System.out.println("PROMPT");
        String name = in.nextLine();
    }
}
```

- in this example the input is stored in name

Type Spacing

Blocks

```
- if ( ) {
    var x = m;
}
```

← var x can now only be used within the {} of the if statement
- using outside would cause issue

Methods

```
- psum (sa[]){
    var x = m;
}
```

← now x can be used ANYWHERE inside the function

Class

```
- cM{
    m2(){}
    n2(){}
}
```

← Any var inside cM will be allowed to be call var

☆ static ___ ___ = ___ ;
(type, name, value)

→ don't nee instance
non static → value changes per class

Commenting / Javadoc

Commenting => easy way to take notes
- /*
  * regular multi-line comment
  */
- // regular single line comment

Javadoc -> special documentation
- /**
  * Main class
  * @param args
  * @return a welcome message
  */

} param, return would have syntax highlighting