Java Basics With Addison :3

Getting Started and Printing

* The name of the first class in a Java file should be named the same way as the file;
* **public static void main(String[] args){}:** Starts the program;
  + Every command the program contains must be written inside of those brackets.
* The primitive data types are:
  + int: 0, 1, 3…;
  + float/double: 0.25, 3.46, 0.81 etc.;
  + boolean: true or false;
  + char: A single character;
    - Ex: “a”.
* Reference types: Anything defined by classes.
  + Store the location in memory where the information about that object is stored
  + Have methods that can be called;
  + String;
  + Reference types can use .equals() or .compareTo()
* **System.out:** Outputs something;
  + **print:** Prints anything you want;
  + **println:** Doesthe same as print, but adds a new line below.
* **System.err:** Prints something as an error**;**
  + Has the same methods as System.out.
* “try” and “catch”: Error handler
  + You wrap everything that may trigger an error inside of try{}, and catch{} will handle the error in case it’s triggered;

Operations

* Additions: System.out.print(n + m);
* Subtractions: System.out.print(n - m);
* Multiplications: System.out.print(n \* m);
* Divisions: System.out.print(n / m);
  + Be careful with divisions between integers, if the result is meant to be a double, it’ll only print the integer part (before the dot);
  + System.out.print(**(double)**n / m): Divides both numbers as doubles;
  + System.out.print(n / **(double)**m): Divides both numbers as doubles;
  + System.out.print(**(double)**n / **(double)**m): Divides both numbers as doubles;
  + System.out.print(**(double) (**n / m**)**): Due to the parenthesis between the two numbers to be divided, they’re divided as integers and then the result is converted to double;
  + (double) can be used as (int) in case you want to transform a double into an integer;
* Modulus: System.out.print(n % m);
  + Negative Modulus: You’ll use the signal from the number on the left side as a reference.
    - Ex: 6 % -4 = 6 % 4.
* Order of operations:
  + Parentheses;
  + Method calls, like Math.sqrt() and Math.pow();
  + Multiplication and division;
  + Addition and subtraction;
* Boolean Operations: Comparison between two or more values;
* Order of boolean operations:
  + Parentheses;
  + Method calls;
    - Ex.: String#method().
  + Math operations;
  + Comparisons:
    - Greater than (>);
    - Lesser than (<);
    - Greater or equal than (>=);
    - Lesser or equal than (<=);
    - Equal (==);
    - Not equal (!=).
      * **Note: Only use comparisons on primitive types (int, double, Boolean, char).**
  + Not (!);
  + And (&&);
  + Or (||).
    - **Note: “&&” has priority over “||”.**

Classes

* Used to create new reference types;
* Must be in a file with the same name as the class;
  + No need to use the main method inside of those files, as they’ll be ran by a class in another file;
  + E.g.: Table must be in table.java.
* A class file must contain:
  + Instance variables: Data about the object;
    - Usually kept “private” for encapsulation;
  + Constructors: Build the class;
    - The only method types with no return type (int, boolen…);
    - public ClassName() {};
    - **“new Class()”:** Calls the constructor.
  + Getters (aka Accessors):They access private information to be used wherever they need to be used;
    - Must have a return type;
    - **“public dataType getVariable(){}”.**
  + Setters: Sets a new value to an existing variable;
    - public dataType setVariable(dataType variable){this.variable = variable};
  + toString: Converts a reference type to String so Java can print it when requested to;
* Overload: Multiple methods with a same name, but with different parameters;
  + E.g.: Table(){} and Table(int length, int height, int width){return variable};
* Signature: Name and data types of the constructor;
  + E.g.: **Table(int length, int height, int width)** is the signature of the constructor;
* **“this.”:** Differs the instance variables from the ones in the signature;
* **Methods:** Parts of code that can be called from other locations in the program to reuse code that occurs frequently;