Java Building Blocks

To understand programming in Java, you must understand the basic building blocks that make up the language itself. The first thing that needs to be understood is the concept and application of ***variables***

Variables: Variables are how you store data in Java, when dealing with **primitive non-object types** (which will be discussed later) variables directly hold the data they are pointing to. When dealing with **non-primitive object types,** variables hold references to the data they are pointing to.

There are two distinct ways to declare and assign values to variables in java and depend on whether they are primitives or objects. They are as follows:

For primitive types it is as follows:

```
dataType variableName = valueOrExpressionOfSameType;
```

For object types it is as follows:

```
ObjectType variableName = valueOrExpressionOfSameType;
```

We will see concrete examples of the preceding two statements later on but for now to remember variables and what they do think of the memory balls from Inside Out, the ball itself is the variable and the memory is the data we are storing in the variable.

Primitive Data Types: In Java when you are dealing with data, there are different forms that the data can take and these forms determine what type of data can be stored. While there are many primitive data types (which I will still list for the sake of consistency the ones in bold are the ones most relevant in the actual class and the ones in italics will be relevant in an actual program):

1. **byte:** stores 8 bits of data and can hold integer values ranging from -128 to 127
2. char: stores 16 bits of data and can hold single characters ex: 'a' must use single quotes
3. ***boolean*:** stores 8 bits of data and is used to hold one of two values true or false
4. short: stores 16 bits of data and can hold integer values ranging from -32768 to 32767
5. ***int*:** stores 32 bits of data and can hold integer values ranging from -2147483648 to 2147483647
6. **float:** stores 32 bits of data and can hold floating-point values
7. ***double:*** stores 64 bits of data and can hold floating-point values
8. long: stores 64 bits of data and can hold integer values -9.22E+18 to 9.22E+18

The following is an example of how to declare a variable named myFirstInt as type int and we will assign it a value of 1004:

```
int myFirstInt = 1004;
```

We can then reassign myFirstInt to have a different value the following way:

```
myFirstInt = 1661;
```

Notice that we do not have to put the dataType when we are reassigning the variable to a different value, this is because the variable already exists you only need to put the dataType when you first declare the variable. To illustrate this the following code is identical in what it does:

```
int mySecondInt; //declaring the variable
mySecondInt = 2021; //assigning a value to the variable

int mySecondInt = 2021; //declaring and assignment
```

Now that we have covered primitive types we will now move onto Classes, Objects, and Constructors

Classes, Objects, Constructors

Classes are the main structural unit for Java programs, they house all the code for a particular file and the structure looks as follows (this should look familiar):

```
public class F2C {

     //main method
     public static void main(String[] args){
          //insert code here
     }
}
```

Files in java must follow the convention of ClassName.java so if the file you created was called 'HelloWorld.java' the name of the public class must be HelloWorld otherwise you will run into issues. When you think of a Class in relation to objects and constructors, think of the class as a blueprint for a product.

Time to make note of the main method, only briefly we will come back to it later when we talk about methods, in order for you to run a .java file directly it must have a main method, otherwise the JVM won't know where to start.

Objects are instances of a class, so think of an object as the final product that was designed using the blueprint, it therefore is an instance of that blueprint. The blueprint itself doesn't change but the specific attributes of the final product might be slightly different. For example you may have a red car or a blue car that were both designed using the same blueprint. Objects need to be instantiated into existence and can be done in the following way:

```
ClassName variableName = new ClassName(arguments);
```

Let us break down what it means to instantiate an object using the above syntax with something you are hopefully familiar with by now: The Scanner

It is important to note that the Scanner class is not available for use off the bat in java and you'll need to use a special statement to gain access, we will discuss that after we conclude with Objects and Constructors.

The Scanner is in the class named Scanner so in order to instantiate an object of type Scanner the first part of the syntax will be (note I will be naming the Scanner sc):

```
Scanner sc
```

That is the left hand side of the assignment out of the way, now time for the right hand side with only a handful of exceptions when you're dealing with objects you will always use the new keyword (since you are creating a new instance of an object) For Scanner it will be as follows:

```
= new Scanner(System.in);
//combining the previous two segments yields the following

Scanner sc = new Scanner(System.in);
```

It is important to realize that just like with primitive data types the declaration and assignment can occur on separate lines as follows:

```
Scanner sc;
sc = new Scanner(System.in);
```

For the Scanner class, it requires an argument in order to function properly, for now the only argument needed is System.in this allows for input to be collected from the user at the terminal during runtime.

Now is the perfect time to discuss constructors, constructors are special methods that are used to construct an object and are called immediately after the new keyword and can take any amount of arguments as needed. They are formatted in the following way:

```
public ClassName(Param1, Param2,...){
     //insert code here
}
```

It is important to note that there is NO return type on constructors, we will cover return types more when we talk about methods in Session 2, when you think of a

constructor think of the assembly line process in a factory, it constructs the final product (Object). To recap here is how you should view Classes, Objects, and Constructors metaphorically:

The Class is the blueprint … the design of the final product if you will. Once you have your design, you need to assemble the product on an assembly line, this is the Constructor it *constructs* the object and at the end of it all, you have your final product the Object that has just been constructed.

Import Statements

In Java there are classes that can provide functionality to our programs that are not included in the java.lang package (the set of classes that java imports automatically, no explicit declaration) the import statement syntax is as follows:

```
import java.util.Scanner;
```

Sometimes, you will need multiple classes from a package so you can use the wildcard “*”, this allows you to import all the classes within a package and can be used in the following way:

```
import java.util.*;
```

That concludes this first session, next session will focus on Strings, Methods and Object Oriented Programming.