**Program 1**

public class HelloWorldV1  
{  
//main method  
public static void main(String[] args)  
{  
System.out.println("Hello, Virtual World!");  
System.out.println("It is a great day for programming.");  
}  
}

**Program 2**

public class HelloWorldV2  
{  
//print two lines of text  
public static void printTwoLines( )  
{  
System.out.println("Hello, Virtual World!");  
System.out.println("It is a great day for programming.");  
}  
//main method  
public static void main(String [] args)  
{  
printTwoLines();  
}  
}

**Program 3**

public class HelloWorldV3  
{  
//default constructor  
public HelloWorldV3()  
{  
}  
  
//print two lines of text  
public void printTwoLines()  
{  
System.out.println("Hello, Virtual World!");  
System.out.println("It is a great day for programming.");  
}  
  
//main method  
public static void main(String [] args)  
{  
HelloWorldV3 hello = new HelloWorldV3( );  
hello.printTwoLines();  
}  
}

In the world of programming, there are numerous ways to complete the same task; each way has it’s reasons to use it, as well as reasons to avoid it. In Java, there are three main ways to structure your code: non-procedural, procedural and object oriented.

Non-procedural programming is the figurative bottom of the ladder. It’s the easiest to read, as it follows a straight top-to-bottom approach. Take the first hello world program: it first declares the class, declares the main method, and directly prints the lines. If you were to expand this, you would simply add another print statement below the first two.

One of the main benefits of non-procedural programming is the fact that it goes top to bottom. The code at the top runs first, and you move down the line. It’s very easy to read. However, it gets cluttered and hard to read as the program grows larger and larger. This is one of the main downsides to non-procedural program.

That’s where procedural programming comes in: it improves upon the downsides of non-procedural.

Procedural programming is essentially the same; the only difference is that it separates the important code into methods. Methods are blocks of code that can be reused. In the second hello world program, the end result is the same. The only difference is that the two lines to print were put in a method that was called from the main method.

This approach is best for larger programs. If you compare the first and second programs, the procedural one is longer. That’s not good, is it? No. But think of this: if you wanted to print something twice, you would only have to add one line to the second one. In contrast, you would have to add two lines to the first one. This can add up over time, which is why procedural programming is widely used.

Finally, there is one more programming structure used in the example: object-oriented programming. OOP is another way of writing reusable code for larger programs. It involves creating a “class”, which acts as a template for as many objects as you want.

In the third example, a new object called hello is defined. Then, the method printTwoLines() is called. You could instantiate as many instances of the class as you want, making it easy to use the same code over and over.

These three ways of structuring your code all have their pros and cons. However, OOP is, in my opinion, the best option for writing larger programs. It’s very easy to read, and reduces the code needed to perform the same tasks. If you were writing a smaller program, procedural program is the easiest and most functional (pun intended) for small to mid-sized programs.