1. So that the Java Virtual Machine can access the main method. The public access specifier allows the main method to permit a call from outside the application, whereas a specifier like private will not.
2. When the JVM makes a call to the main method no instance of a class ie an object is present, so the main method must be static in order for the main method to be called (static specifier allows the method to be accessed without an object being instantiated).
3. Parameters allow us to pass values into a method, which can then be used within the method for some purpose, such as performing a calculation. Without said parameters, we would need to define new variables and assign them every time the method is called, and we wouldn’t be able to pass values into the method.
4. A static method belongs to the class it is in, whereas a non-static method belongs to each object instantiated from the class. If a method is static, it can be accessed within a class without a need for generating an instance of the object within the main method. If it is non-static, it isn’t tied to the class so we need a working instance of the class ie an object in order to use it.
5. See diagram
6. A single point of failure is a part of a system that will stop the entire system from working if it fails. Thus, one mistake in a program for example will result in the compiler failing and throwing an error. However, in a large program this is beneficial because there would only be one point in the program that would need to be fixed for the program to compile properly and run, and the programmer would be aware that there is only one mistake.

import java.util.Scanner;

public class lesson4

{

static double one, two;

public static void main(String[]args)

{

Scanner kb = new Scanner(System.in);

System.out.println("Enter one number: ");

one = kb.nextDouble();

System.out.println("Enter second number: ");

two = kb.nextDouble();

add();

print();

}

public static double add()

{

return one+two;

}

public static void print()

{

System.out.println(one + " + " + two + " = " + add());

}

Variables one and two initialized (called to be defined throughout the program as integers)

-> Scanner defined

-> User input asked for the value of the first variable

-> User input asked for the value of the second variable

-> add() method called

-> add() method utilizes the recently acquired definitions of “one” and “two” to perform its function and return one+two

-> main method now knows what add() is supposed to represent

-> print() method is called

-> print() method performs its function utilizing the main method’s knowledge of one, two, add()

-> final print statement is printed in the main method