2.
Method declaration determines what the method is, things such as what it returns, its name, and its parameters.

Ex.

```
public int totalnum(int a, int b) {
    return a + b;
}
```

The method body is what the method actually does (the code inside of the braces)

Ex.

```
public int totalnum(int a, int b) {
    return a + b;
}
```

3.
Access Modifiers
The keyword public can be changed to adjust the access level of a method. This can be changed to words such as private, protected, final and more. These are known as access modifiers.

4.
Visibility
Another word used for describing the access level of a method is its visibility, for example of the method, was public its visibility would be public.

5.
For the code:

```
public class ScopeExample {
    public static void method1() {
        int var3;
        for (int var4 = 0; var4 < 2; var4++) {
            var3 += 1;                      }
           }
    }

    public static void main(String[] args) {
        int var1;
```

```
        for (int var2 = 0; var2 < 5; var2++) {
            method1();
        }
    }
```

Scope of var3
The variable var3 is accessible within method1, and is usable in loops and if statements that would be in that method (although that doesn't happen here) (also note it can't be used in the method main)

Scope of var4
The variable var4 is declared within the for loop in method1 meaning it is only accessible within the loop.

Scope of var1
The variable var1 is accessible within the method main and can be used anywhere in that method such as loops and if statements that would be in that method (although that doesn't happen here) (also note it can't be used in method1)

Scope of var2
The variable var2 is declared within the for loop in the method main meaning it is only accessible within the loop.

6.
a)

```
public static int getVowels(String input) {
    int vowelcount = 0;
    // This is where the code to get the number of vows would go
    return vowelcount;
}
```

b)

```
public static int extractDigit(int number) {
// code to extract a digit (for example you may want the last digit of a number)
        int lastdigit = number / 10; // Example code for if you want the tens place number
        (note: if the number inputted is 9 the result will be 0)
        return lastdigit;
}
```

c)

```
public static String insertString(String name, int age) {
```

```
    // some code to do with the string and what this method will return
    return name;
}
```

7.

a)

The compiler can distinguish between methods using something called overloading. This looks at each method's signatures which include its name, parameters, and the order the parameters are in. Together these things help the compiler distinguish methods from one another.

b)

Yes, they can have the same name as long as they have different signatures. The compiler will still be able to tell these two methods apart by their parameters, and the parameters order.

Ex.

```
void display(int a)

void display(double a)
```

These are different because of their parameters, the first one takes in an int, whereas the second one takes in a double.

8.

a)

Return statements can be used to exit a method, but also to store information that can be used in different parts of the program. For example, if a method does calculations and returns a value of 15, whenever a different part of the program calls on the method it will return the value of 15 for the code to use.

b)

Generally, a return statement can only return one value which can be in any type (Ex. int, string float, etc.). There are, however, ways to work around this with things such as arrays and objects.

c)

A return statement that returns a value must indicate what type of value it is returning (Ex. int, float, string, etc.), however, a return statement that does not return a value, is used in a void method and is used to exit a method.

9.

```
public class MethodCallExample {

        public static int doSomething() {
                return(5);
        }
}

        public static void main(String[] args) {
        int num;
        doSomething();
        num = doSomething();
}
```

-the main error here is that the main method is placed outside the public class (the braces separate it) this causes an error when executing the program. (to fix this simply eliminate that brace and move it to the end of the code to include the main method.

Corrected

```
public class MethodCallExample {

   public static int doSomething() {
      return 5; // Note you could remove the parentheses around 5
   }

   public static void main(String[] args) {
      int num;
      doSomething();
      num = doSomething();
num
      System.out.println("The value of num is: " + num); // Optional: Added to print the
value of num
   }
} // moved the brace down here to include the main method within the public class
```