2. Explain the difference between method declaration and method boyd

The line that specifies a method's name, return type, and parameters is called a method declaration. It informs the program of the method's name and the data it uses. Everything enclosed in curly brackets following the declaration is referred to as the method body. It includes the actual statements that are executed upon calling the method. The body of the method comprises the instructions it carries out, whereas the declaration explains the method's structure.

3. What type of keyword is used to change the access level of the method

The type of keywords used to change the access level of the method is public, private and protected. They are used to protect data, control how a class is used, to hide internal details and to allow or restrict access.

4. What is another word used for describing the access level of a method

Another word used for describing the access level is visibility

5. Explain the scope of each variable in the code below

var1

Declared in the main() function

Scope: Limited to the main() function

Meaning: It cannot be used by any other way, including method 1.

var2

Declared in: the main() for loop

Scope: Just within that for loop

Meaning: It is only present inside the loop body throughout each iteration of the loop.

var3

Declared in: method1()

Scope: The entire method1() method

Meaning: Useful only within method1().

var4

Declared in: method1()'s for loop

Scope: Just within that for loop

Meaning: It is limited to the body of that loop.

6. Write a method declaration for each of the following descriptions

a) A class method named getVowels that can be called by any other method requires a string parameter and returns an integer value

public static int getVowels(String text);

b) A class method named extractDigit that can be called by any other method requires an integer parameter and returns an integer value

public static int extractDigit(int num);

c) A class method named insertString that can be allied by any method requires a string parameter and an integer parameter and returns a String parameter

public static String insertString(String text, int position);

7. a) How does the compiler distinguish one method from another

The method's signature, which comprises the method's name and the quantity, kinds, and arrangement of its parameters, is how the compiler differentiates one method from another. This implies that if two methods have different parameter lists, they can have the same name. The return type does not aid the compiler in distinguishing between methods because it is not included in the method signature.

b) can two methods in the same class have the same name. Explain?

Yes, two methods within the same class can have the same name; this is known as method overloading, as long as the parameter lists of the two methods are different

8. a) What is the return statement used for?

A value can be returned to the method that called it using the return statement. A return statement causes the method to end instantly and return the result to the caller. Any method with a return type other than void must have it. Basically a return statement can be used for a calling statment

b) How many values can a return statement send back to the calling

A return statement can send back only one value to the calling method. Even though that value can be a simple type or a complex object containing many pieces of data, the return statement itself returns just one single value.

c) How is the declaration of a method returning a value is different from the declaration of a method that does not return a value

A method that returns a value must specify a return type in its declaration, such as int, double, or String, and it must include a return statement that returns a value of that type. In contrast, a method that does not return a value uses the keyword void in its declaration and does not return anything, so it does not include a return statement that sends back a value.

9. Find and explain the error in the code below:

```
public class MethodCallExample {

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

}
}

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

The mistake in the code is that the main() method is positioned outside of the class. In Java, all methods must be declared within the class body, specifically between its opening and closing braces. In the provided code, the closing brace after doSomething() comes too soon to close the class, resulting in the main() method not being included in the MethodCallExample class.

11. Determine if each of the following are true or false. If false, explain why.

a) Breaking a task down into methods is called procedural abstraction -**true**

b) A method call consists of the method declaration in an assignment statement -**false because A method call is not the same as a method declaration, and it does not consist of a declaration inside an assignment statement.**

c) A void method must return a value -**false A void method can still use a return; statement, but only to end the method early, not to return a value.**

d) An access modifier declares the return type of a method -**false An access modifier (like public, private, or protected) controls who can access the method — not what it returns.**

e) The keyword static declares a method is a class method -**true**

f) Method parameters are enclosed by braces ({}) -**false because method parameters are enclosed by normal brackets rather than curly braces**

g) Local variables can be used by any method in a class -**false Local variables only exist inside the method where they are created. They cannot be accessed by any other method in the class.**

h) The value of an argument passed to a method can be changed in an assignment statement in the method -**false When you pass a variable into a method, Java passes the value, not the original variable itself. So if you change the parameter inside the method, it does NOT change the original argument.**

i) Method overloading means that an application contains more than 10 methods - **false Method overloading has nothing to do with the number of methods in an application.**

j) The return statement is used to send a value back to the calling statement **-true**

k) The precondition of a method states the data type of the methods parameters **-false The precondition of a method does not describe the data types of its parameters.**

l) The postcondition of a method describes the way the method accomplishes its task **-false A postcondition describes what is true after the method finishes, not how the method gets the result.**