**Technology Training at Winhe SEA  
Exercise – 4**

7) What do mean by the scope of a variable. Explain with examples.

In C#, **when we declare a variable inside a class, the variable can be accessed within the class**. This is known as class level variable scope. Class level variables are known as fields, and they are declared outside of methods, constructors, and blocks of the class.

A variable scope refers to the availability of variables in certain parts of the code.

In C#, a variable has three types of scope:

* Class Level Scope
* Method Level Scope
* Block Level Scope

Class Level Variable Scope

when we declare a variable inside a class, the variable can be accessed within the class. This is known as **class level variable scope**.

Class level variables are known as fields and they are declared outside of methods, constructors, and blocks of the class.

For example,

using System;

namespace VariableScope {

class Program {

// class level variable

string str = "Class Level";

public void display() {

Console.WriteLine(str);

}

static void Main(string[] args) {

Program ps = new Program();

ps.display();

Console.ReadLine();

}

}

}

Method Level Variable Scope

When we declare a variable inside a method, the variable cannot be accessed outside of the method. This is known as **method level variable scope**.

For example,

using System;

namespace VariableScope {

class Program {

public void display() {

string str = "inside method";

// accessing method level variable

Console.WriteLine(str);

}

static void Main(string[] args) {

Program ps = new Program();

ps.display();

Console.ReadLine();

}

}

}

Block Level Variable Scope

When we declare a variable inside a block (for loop, while loop, if else), the variable can only be accessed within the block. This is known as **block level variable scope**.

For example

using System;

namespace VariableScope {

class Program {

public void display() {

for(int i=0;i<=3;i++) {

}

Console.WriteLine(i);

}

static void Main(string[] args) {

Program ps = new Program();

ps.display();

Console.ReadLine();

}

}

}

8) Why do you think adhering to the general convention and standard  
practices is important as a software engineer.

If programming standards in software engineering are followed, **the code is consistent and can be easily maintained**. This is because anyone can understand it and can modify it at any point in time. It becomes really easy to locate and correct bugs in the software if the source code is written in a consistent manner.

The reason why coding standards are important is that **they help to ensure safety, security, and reliability**. Every development team should use one. Even the most experienced developer could introduce a coding defect — without realizing it. And that one defect could lead to a minor glitch

**Advantages of implementing Coding Standards**  
  
Makes it easier to detect errors. Make code simpler, more readable, and easier to maintain. Boost programmer efficiency and generates faster results.

9) What are good coding practices in programming.

* Commenting & Documentation
* Consistent Indentation
* Avoid Obvious Comments
* Code Grouping
* Consistent Naming Scheme
* DRY Principle
* Avoid Deep Nesting
* Limit Line Length
* File and Folder Organization
* Consistent Temporary Names
* Capitalize SQL Special Words
* Separation of Code and Data
* Alternate Syntax Inside Templates
* Object-Oriented vs. Procedural
* Use Meaningful Names for Variables and Functions
* Code Refactoring

10) What do you mean by fall-down effect in relation to the switch  
statement in Java.

Java Programming Technologies Object Oriented Programming. Following rules govern the fall through the behavior of switch statement. **When the variable being switched on is equal to a case, the statements following that case will execute until a break statement is reached**.

Following rules govern the fall through the behavior of switch statement.

* When the variable being switched on is equal to a case, the statements following that case will execute until a break statement is reached.
* When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
* Not every case needs to contain a break. If no break appears, the flow of control will fall through to subsequent cases until a break is reached.

11) What is the difference between break and continue statements.  
Explain using a code sample.

The break statement terminates the loop and transfers execution to the statement immediately following the loop.

The continue statement causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.When the break statement is encountered inside a loop, the loop is immediately terminated and program control resumes at the next statement following the loop.

The continue statement in C# works somewhat like the break statement. Instead of forcing termination, however, continue forces the next iteration of the loop to take place, skipping any code in between.

Example

The following is the complete code to use continue statement in a while loop −

using System;

namespace Demo {

   class Program {

      static void Main(string[] args) {

         /\* local variable definition \*/

         int a = 10;

         /\* loop execution \*/

         while (a > 20) {

            if (a == 15) {

               /\* skip the iteration \*/

               a = a + 1;

               continue;

            }

            Console.WriteLine("value of a: {0}", a);

            a++;

         }

         Console.ReadLine();

      }

   }

}

The following is an example of break statement –

Example

using System;

namespace Demo {

   class Program {

      static void Main(string[] args) {

         /\* local variable definition \*/

         int a = 10;

         /\* while loop execution \*/

         while (a < 20) {

            Console.WriteLine("value of a: {0}", a);

            a++;

            if (a > 15) {

               /\* terminate the loop using break statement \*/

               break;

            }

         }

         Console.ReadLine();

      }

   }

}