1. VIRTUAL

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;

// VIRTUAL
//
// CALLBACK EXAMPLE USING VIRTUAL METHODS AS CONTRACT AND INHERITANCE AS CALLBACK-IMPLEMENTATION


namespace X03_Callbacks
{
    public class ProgressReporter
    {
        public virtual void ReportProgress(int percentDone)
        {
           // No implementation here
        }
    }

    public static class Calculator
    {
        public static int SomeLengthyCalculation(ProgressReporter pr)
        {
            for (int i = 0; i < 100; i++)
            {
                // Sleep 1/10 second - simulates a step in the calculation
                Thread.Sleep(100);

                pr.ReportProgress(i);
            }
            return 42;
        }
    }


    //  ^
    //  |    Implementation of Calculation. Implementors don't know anything about
    //  |    the context their code is called in (language, UI-System, etc.).
    ////////////////////////////////////////////////////////////////////////////////////
    //  |    User Code using the Calculation. User code cannot change the calculation's
    //  |    implementation but needs to report the progress to the user.
    //  V

    public class UserProgressReporter : ProgressReporter
    {
        public override void ReportProgress(int percentDone)
        {
            Console.WriteLine($"Calculating. {percentDone}% already done.");
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Starting the calculation");
            var result = Calculator.SomeLengthyCalculation(new UserProgressReporter());
            Console.WriteLine($"The result is: {result}.");

            Console.ReadKey();
        }
    }
}
```

2. INTERFACE

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;

// INTERFACE
//
// CALLBACK EXAMPLE USING AN INTERFACE AS CONTRACT AND IMPLEMENTATION AS CALLBACK-IMPLEMENTATION

namespace X03_Callbacks
{
    public interface IProgressReporter
    {
        void ReportProgress(int percentDone);
    }

    public static class Calculator
    {
        public static int SomeLengthyCalculation(IProgressReporter pr)
        {
            for (int i = 0; i < 100; i++)
            {
                // Sleep 1/10 second - simulates a step in the calculation
                Thread.Sleep(100);

                pr.ReportProgress(i);
            }
            return 42;
        }
    }


    //  ^
    //  |    Implementation of Calculation. Implementors don't know anything about
    //  |    the context their code is called in (language, UI-System, etc.).
    ////////////////////////////////////////////////////////////////////////////////
    //  |    User Code using the Calculation. User code cannot change the calculation's
    //  |    implementation but needs to report the progress to the user.
    //  V

    public class UserProgressReporter : IProgressReporter
    {
        public void ReportProgress(int percentDone)
        {
            Console.WriteLine($"Calculating. {percentDone}% already done.");
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Starting the calculation");
            var result = Calculator.SomeLengthyCalculation(new UserProgressReporter());
            Console.WriteLine($"The result is: {result}.");

            Console.ReadKey();
        }
    }
}
```

3. DELEGATE

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;

// DELEGATE
//
// CALLBACK EXAMPLE USING A DELEGATE AS CONTRACT AND A METHOD AS IMPLEMENTATION

namespace X03_Callbacks
{
    // Declares the _DATA TYPE_ ProgressReporter. Variables of that type can
    // hold a method
    public delegate void ProgressReporter(int percentDone);

    public static class Calculator
    {
        public static int SomeLengthyCalculation(ProgressReporter pr)
        {
            for (int i = 0; i < 100; i++)
            {
                // Sleep 1/10 second - simulates a step in the calculation
                Thread.Sleep(100);

                pr(i);
            }
            return 42;
        }
    }


    //  ^
    //  |    Implementation of Calculation. Implementors don't know anything about
    //  |    the context their code is called in (language, UI-System, etc.).
    ////////////////////////////////////////////////////////////////////////////////
    //  |    User Code using the Calculation. User code cannot change the calculation's
    //  |    implementation but needs to report the progress to the user.
    //  V

    class Program
    {
        static void ReportProgress(int percentDone)
        {
            Console.WriteLine($"Calculating. {percentDone}% already done.");
        }

        static void Main(string[] args)
        {
            Console.WriteLine("Starting the calculation");
            var result = Calculator.SomeLengthyCalculation(ReportProgress);
            Console.WriteLine($"The result is: {result}.");

            Console.ReadKey();
        }
    }
}
```

4. DELEGATE THREADED

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

// DELEGATE THREADED
//
// COMPLEX CALLBACK EXAMPLE USING DELEGATES

namespace X03_Callbacks
{
    // Declares the _DATA TYPE_ ProgressReporter. Variables of that type can
    // hold a method
    public delegate void ProgressReporter(int percentDone);
    public delegate void ResultReceiver(int result);

    public class Calculator
    {
        public ProgressReporter PR;
        public ResultReceiver RR;

        public void StartSomeLengthyCalculation()
        {
            // Start the calculation in a different thread and immediately return to caller.
            new Task(DoCalculate).Start();
        }

        private void DoCalculate()
        {
            for (int i = 0; i < 100; i++)
            {
                // Sleep 1/10 second - simulates a step in the calculation
                Thread.Sleep(100);

                PR(i);
            }
            RR(42);
        }
    }


    //  ^
    //  |    Implementation of Calculation. Implementors don't know anything about
    //  |    the context their code is called in (language, UI-System, etc.).
    ////////////////////////////////////////////////////////////////////////////////
    //  |    User Code using the Calculation. User code cannot change the calculation's
    //  |    implementation but needs to report the progress to the user.
    //  V

    class Program
    {
        static void ReportProgress(int percentDone)
        {
            Console.WriteLine($"Calculating. {percentDone}% already done.");
        }

        static void ReceiveResult(int result)
        {
            Console.WriteLine($"The result is {result}.");
        }

        static void Main(string[] args)
        {
            var calc = new Calculator();
            calc.PR = ReportProgress;
            calc.RR = ReceiveResult;

            Console.WriteLine("Starting the calculation");
            calc.StartSomeLengthyCalculation();
```

```
            Console.WriteLine("We are here but the calculation is not done yet!!");
            Thread.Sleep(1000);
            Console.WriteLine("How long might the calculation take??");
            Thread.Sleep(2000);
            Console.WriteLine("Still not done?");
            Thread.Sleep(4000);
            Console.WriteLine("Seems to take hours!!!");


            Console.ReadKey();
        }
    }
}
```

5. EVENT THREADED

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

// EVENT THREADED
//
// COMPLEX CALLBACK EXAMPLE USING EVENTS

namespace X03_Callbacks
{
    // Declares the _DATA TYPE_ ProgressReporter. Variables of that type can
    // hold a method
    public delegate void ProgressReporter(int percentDone);
    public delegate void ResultReceiver(int result);

    public class Calculator
    {
        public event ProgressReporter PR;
        public event ResultReceiver RR;

        public void StartSomeLengthyCalculation()
        {
            // Start the calculation in a different thread and immediately return to caller.
            new Task(DoCalculate).Start();
        }

        private void DoCalculate()
        {
            for (int i = 0; i < 100; i++)
            {
                // Sleep 1/10 second - simulates a step in the calculation
                Thread.Sleep(100);

                PR(i);
            }
            RR(42);
        }
    }


    //  ^
    //  |    Implementation of Calculation. Implementors don't know anything about
    //  |    the context their code is called in (language, UI-System, etc.).
    ////////////////////////////////////////////////////////////////////////////////
    //  |    User Code using the Calculation. User code cannot change the calculation's
    //  |    implementation but needs to report the progress to the user.
    //  V

    class Program
    {
        static void ReportProgress(int percentDone)
        {
            Console.WriteLine($"Calculating. {percentDone}% already done.");
        }

        static void OtherProgressReporter(int percentDone)
        {
            if (percentDone % 10 == 0)
                Console.WriteLine($"============= ANOTHER TENTH OF THE WORK DONE =============");
        }

        static void ReceiveResult(int result)
        {
            Console.WriteLine($"The result is {result}.");
        }

        static void Main(string[] args)
        {
```

```
            var calc = new Calculator();
            calc.PR += ReportProgress;
            calc.PR += OtherProgressReporter;
            calc.RR += ReceiveResult;

            Console.WriteLine("Starting the calculation");
            calc.StartSomeLengthyCalculation();
            Console.WriteLine("We are here but the calculation is not done yet!!");
            Thread.Sleep(1000);
            Console.WriteLine("How long might the calculation take??");
            Thread.Sleep(2000);
            Console.WriteLine("Still not done?");
            Thread.Sleep(4000);
            Console.WriteLine("Seems to take hours!!!");



            Console.ReadKey();
        }
    }
}
```

6. EVENT LAMBDA

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

// EVENT THREADED
//
// COMPLEX CALLBACK EXAMPLE USING EVENTS

namespace X03_Callbacks
{
    // Declares the _DATA TYPE_ ProgressReporter. Variables of that type can
    // hold a method
    public delegate void ProgressReporter(int percentDone);
    public delegate void ResultReceiver(int result);

    public class Calculator
    {
        public event ProgressReporter PR;
        public event ResultReceiver RR;

        public void StartSomeLengthyCalculation()
        {
            // Start the calculation in a different thread and immediately return to caller.
            new Task(DoCalculate).Start();
        }

        private void DoCalculate()
        {
            for (int i = 0; i < 100; i++)
            {
                // Sleep 1/10 second - simulates a step in the calculation
                Thread.Sleep(100);

                PR(i);
            }
            RR(42);
        }
    }


    //  ^
    //  |    Implementation of Calculation. Implementors don't know anything about
    //  |    the context their code is called in (language, UI-System, etc.).
    ////////////////////////////////////////////////////////////////////////////////
    //  |    User Code using the Calculation. User code cannot change the calculation's
    //  |    implementation but needs to report the progress to the user.
    //  V

    class Program
    {
        static void Main(string[] args)
        {
            int theResult = 0;
            var calc = new Calculator();
            calc.PR += delegate (int done) { Console.WriteLine($"Calculating. {done}% already done.");
};
            calc.PR += percent => { if (percent % 10 ==0) Console.WriteLine($"============= ANOTHER
TENTH OF THE WORK DONE ============="); };
            calc.RR += r => theResult = r;

            Console.WriteLine("Starting the calculation");
            calc.StartSomeLengthyCalculation();
            Console.WriteLine("We are here but the calculation is not done yet!!");
            Thread.Sleep(1000);
            Console.WriteLine("How long might the calculation take??");
            Thread.Sleep(2000);
            Console.WriteLine("Still not done?");
            Thread.Sleep(4000);
```

```
            Console.WriteLine("Seems to take hours!!!");

            Thread.Sleep(12000);
            Console.WriteLine($"Lets see if its there: theResult is {theResult}");

            Console.ReadKey();
        }
    }
}
```