1. Method Overriding (Runtime Polymorphism)

```
using System;

class Animal {
    public virtual void Eat() {
        Console.WriteLine("Animal eats");
    }
}

class Dog : Animal {
    public override void Eat() {
        Console.WriteLine("Dog eats bones");
    }
}

class Program {
    static void Main() {
        Animal myDog = new Dog(); // Polymorphism
        myDog.Eat(); // Outputs: Dog eats bones
    }
}
```

2. Abstract Class

```
using System;

abstract class Shape {
    public abstract void Draw(); // No body
}

class Circle : Shape {
    public override void Draw() {
        Console.WriteLine("Drawing Circle");
    }
}

class Program {
    static void Main() {
        Shape s = new Circle();
        s.Draw();
    }
}
```

3. Exception Handling

```
using System;

class Program {
    static void Main() {
        try {
            int a = 10, b = 0;
            int c = a / b;
        }
        catch (System.Exception ex) {
            Console.WriteLine("Error: " + ex.Message);
        }
        finally {
            Console.WriteLine("Cleanup code here.");
        }
    }
}
```

4. Properties (Encapsulation)

```
using System;

class Student {
    private int _age;

    public int Age {
        get { return _age; }
        set {
            if (value > 0)
                _age = value;
            else
                Console.WriteLine("Invalid Age");
        }
    }
}

class Program {
    static void Main() {
        Student s = new Student();
        s.Age = 20; // Uses set
        Console.WriteLine(s.Age); // Uses get
    }
}
```

5. Static Constructor

```
using System;

class Demo {
    static Demo() {
        Console.WriteLine("Static Constructor"); // Runs once
    }
    public Demo() {
        Console.WriteLine("Instance Constructor"); // Runs every new object
    }
}

class Program {
    static void Main() {
        Demo d1 = new Demo();
        Demo d2 = new Demo();
    }
}
```

6. Jagged Arrays

```
using System;

class Program {
    static void Main() {
        int[][] jagged = new int[2][]; // Declare

        jagged[0] = new int[] { 1, 2 };       // Row 0
        jagged[1] = new int[] { 3, 4, 5 };    // Row 1

        Console.WriteLine(jagged[1][2]); // Access element (Outputs 5)
    }
}
```

7. Generic List (`List<T>`)

```
using System;
using System.Collections.Generic;

class Program {
    static void Main() {
```

```
        List<string> names = new List<string>();
        names.Add("Alice");
        names.Add("Bob");

        foreach (string name in names) {
            Console.WriteLine(name);
        }
    }
}
```

8. ADO.NET Connection & Reading (C#)

```
using System;
using System.Data.SqlClient;

class Program {
    static void Main() {
        string cs = "Data Source=.;Initial Catalog=MyDb;Integrated
Security=True";

        using (SqlConnection con = new SqlConnection(cs)) {
            con.Open();
            SqlCommand cmd = new SqlCommand("SELECT * FROM Users", con);
            SqlDataReader rdr = cmd.ExecuteReader();

            while (rdr.Read()) {
                Console.WriteLine(rdr[0].ToString()); // Print first column
            }
        } // Connection closes automatically here
    }
}
```

9. VB.NET Inheritance

```
Public Class Parent
    Public Sub Greet()
        Console.WriteLine("Hello from Parent")
    End Sub
End Class

Public Class Child
    Inherits Parent
End Class
```

```
Module Program
    Sub Main()
        Dim c As New Child()
        c.Greet()
    End Sub
End Module
```

10. VB.NET Select Case

```
Module Program
    Sub Main()
        Dim grade As Char = "A"

        Select Case grade
            Case "A"
                Console.WriteLine("Excellent")
            Case "B"
                Console.WriteLine("Good")
            Case Else
                Console.WriteLine("Invalid")
        End Select
    End Sub
End Module
```

11. ASP.NET Validation Control

```
<form id="form1" runat="server">

    Name: <asp:TextBox ID="txtName" runat="server"></asp:TextBox>

    <asp:RequiredFieldValidator
        ID="rfv1"
        runat="server"
        ControlToValidate="txtName"
        ErrorMessage="Name is required!">
    </asp:RequiredFieldValidator>

    <asp:Button ID="btnSubmit" runat="server" Text="Submit" />

</form>
```