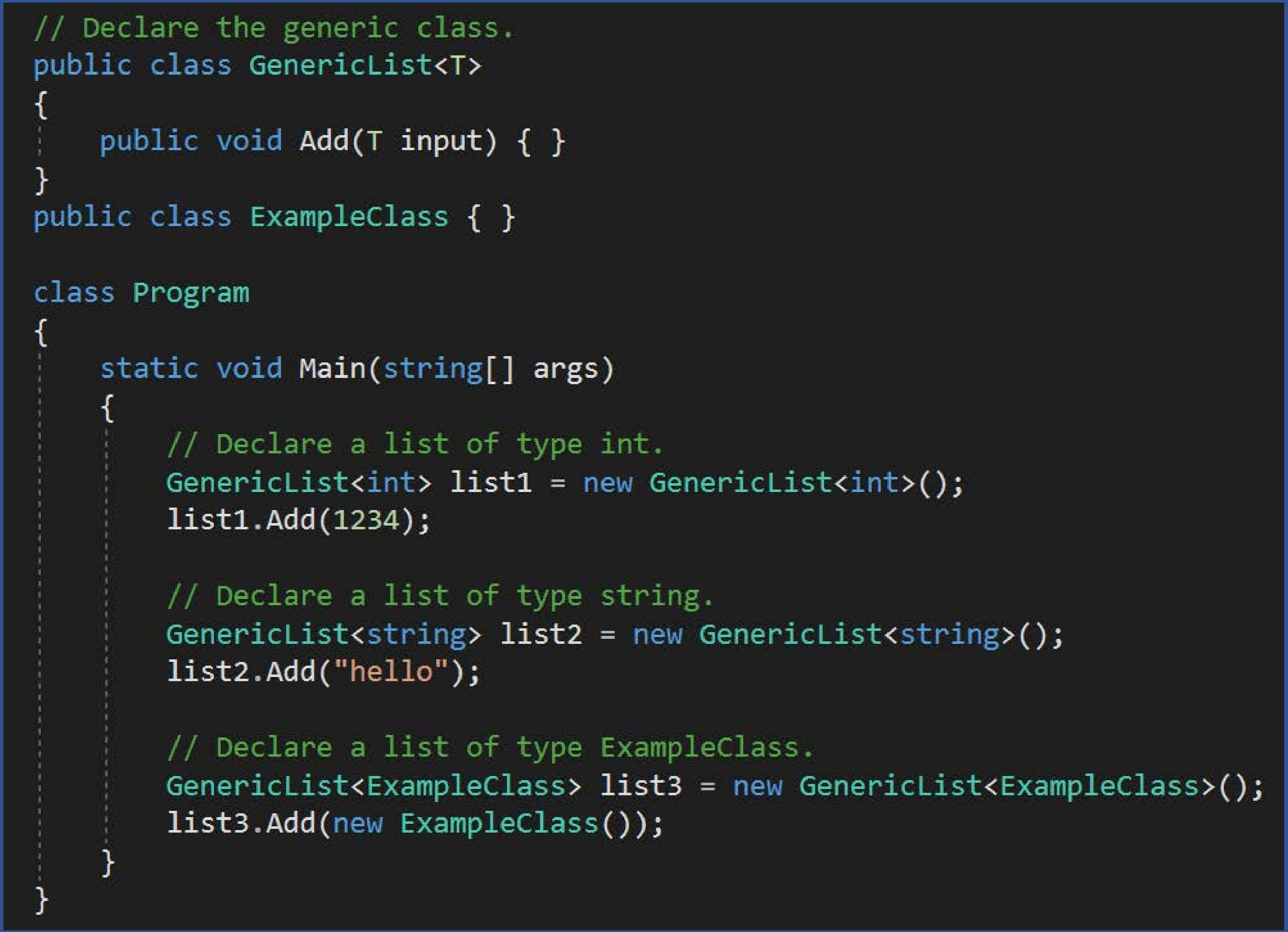
Generic

What are Generics?

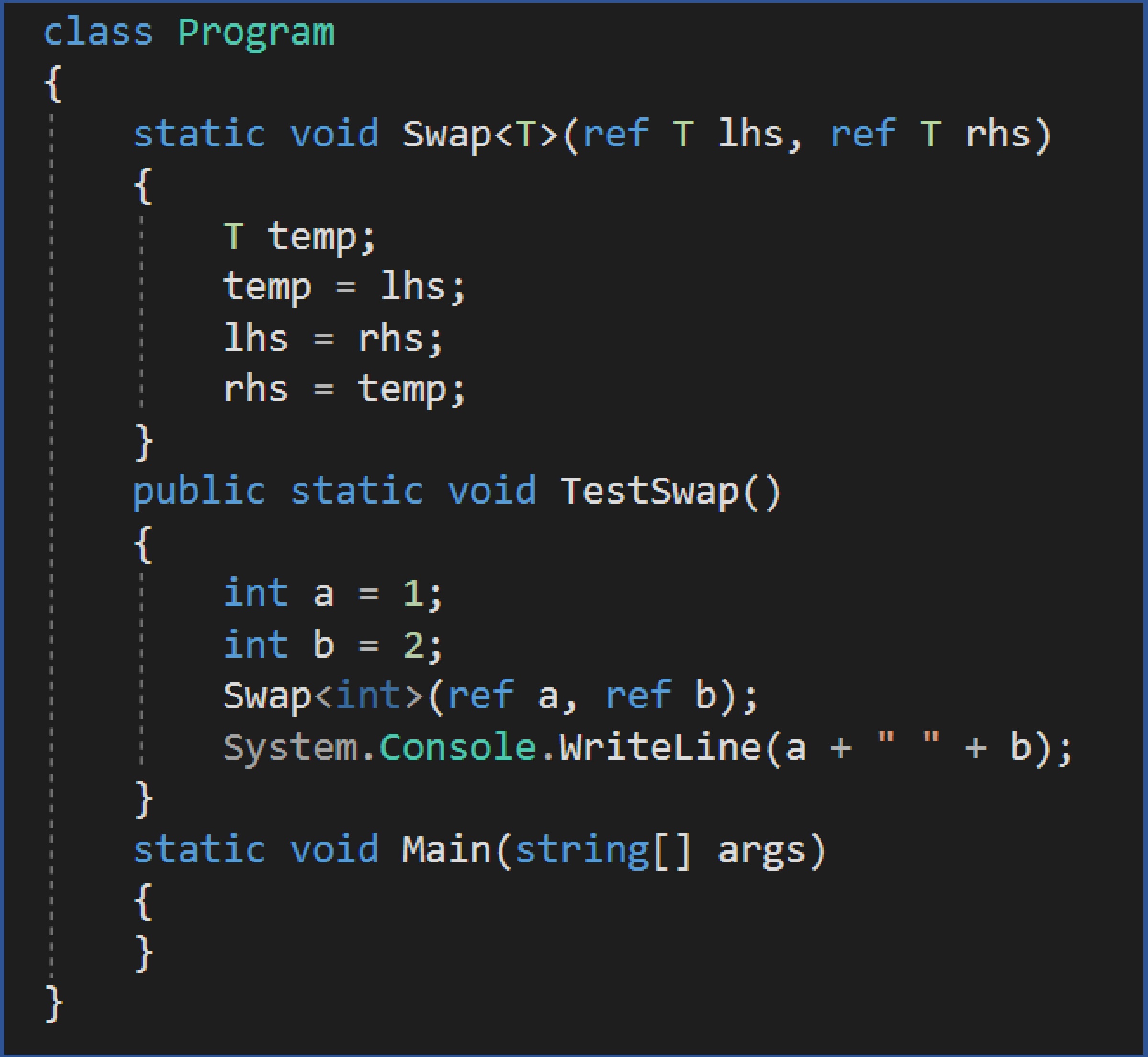
- Most powerful feature of C# 2.0.
- Allow you to define type-safe data structures, without committing to actual data types.
- Performance boost and higher quality code

Generic class



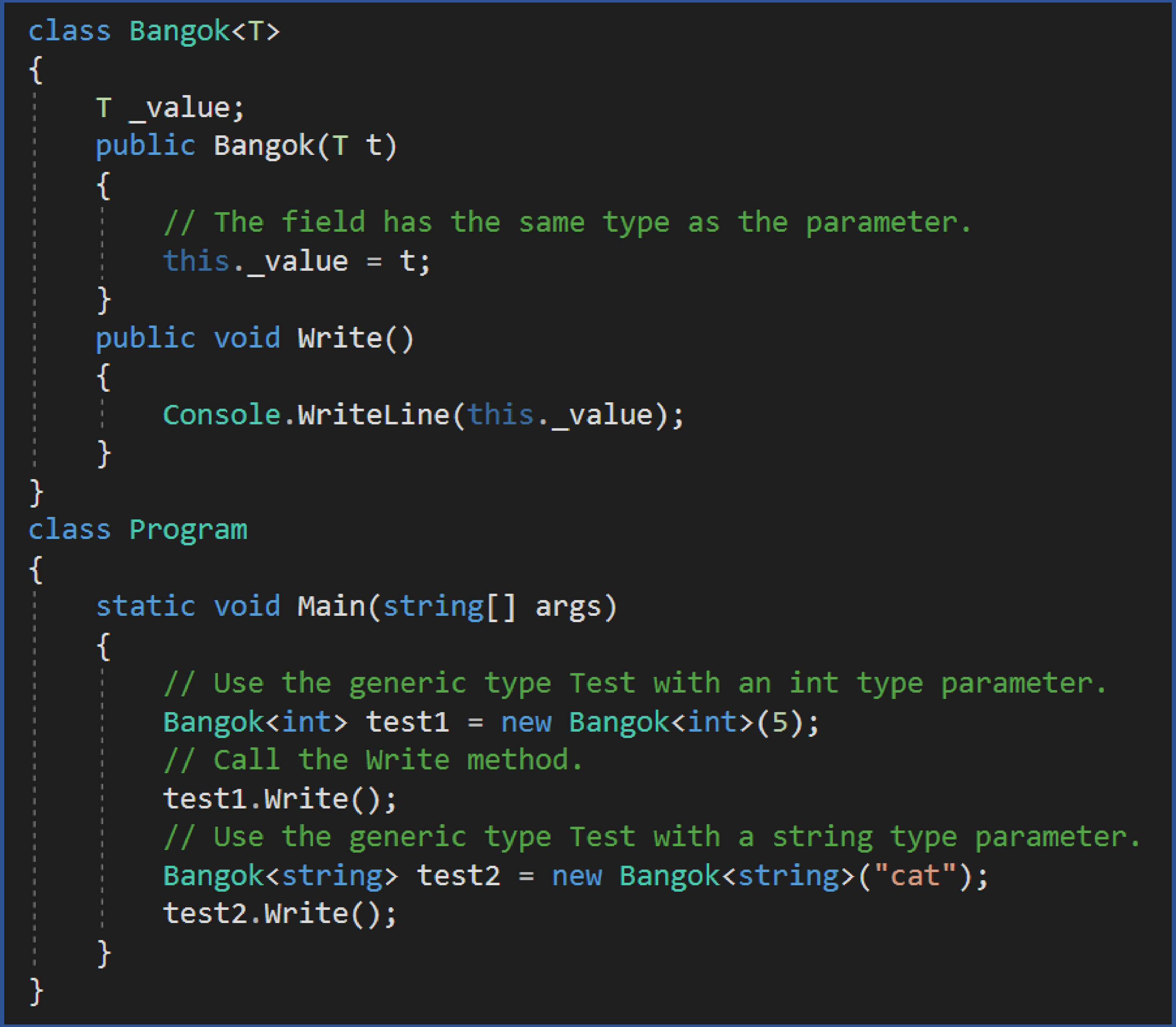
```
// Declare the generic class.
public class GenericList<T>
{
    public void Add(T input) { }
}
public class ExampleClass { }

class Program
{
    static void Main(string[] args)
    {
        // Declare a list of type int.
        GenericList<int> list1 = new GenericList<int>();
        list1.Add(1234);

        // Declare a list of type string.
        GenericList<string> list2 = new GenericList<string>();
        list2.Add("hello");

        // Declare a list of type ExampleClass.
        GenericList<ExampleClass> list3 = new GenericList<ExampleClass>();
        list3.Add(new ExampleClass());
    }
}
```

Generic Method



```
class Program
{
    static void Swap<T>(ref T lhs, ref T rhs)
    {
        T temp;
        temp = lhs;
        lhs = rhs;
        rhs = temp;
    }
    public static void TestSwap()
    {
        int a = 1;
        int b = 2;
        Swap<int>(ref a, ref b);
        System.Console.WriteLine(a + " " + b);
    }
    static void Main(string[] args)
    {
    }
}
```

Generic Example



```csharp
class Bangok<T>
{
    T _value;
    public Bangok(T t)
    {
        // The field has the same type as the parameter.
        this._value = t;
    }
    public void Write()
    {
        Console.WriteLine(this._value);
    }
}
class Program
{
    static void Main(string[] args)
    {
        // Use the generic type Test with an int type parameter.
        Bangok<int> test1 = new Bangok<int>(5);
        // Call the Write method.
        test1.Write();
        // Use the generic type Test with a string type parameter.
        Bangok<string> test2 = new Bangok<string>("cat");
        test2.Write();
    }
}
```

Exercise