Krishna Nigalye

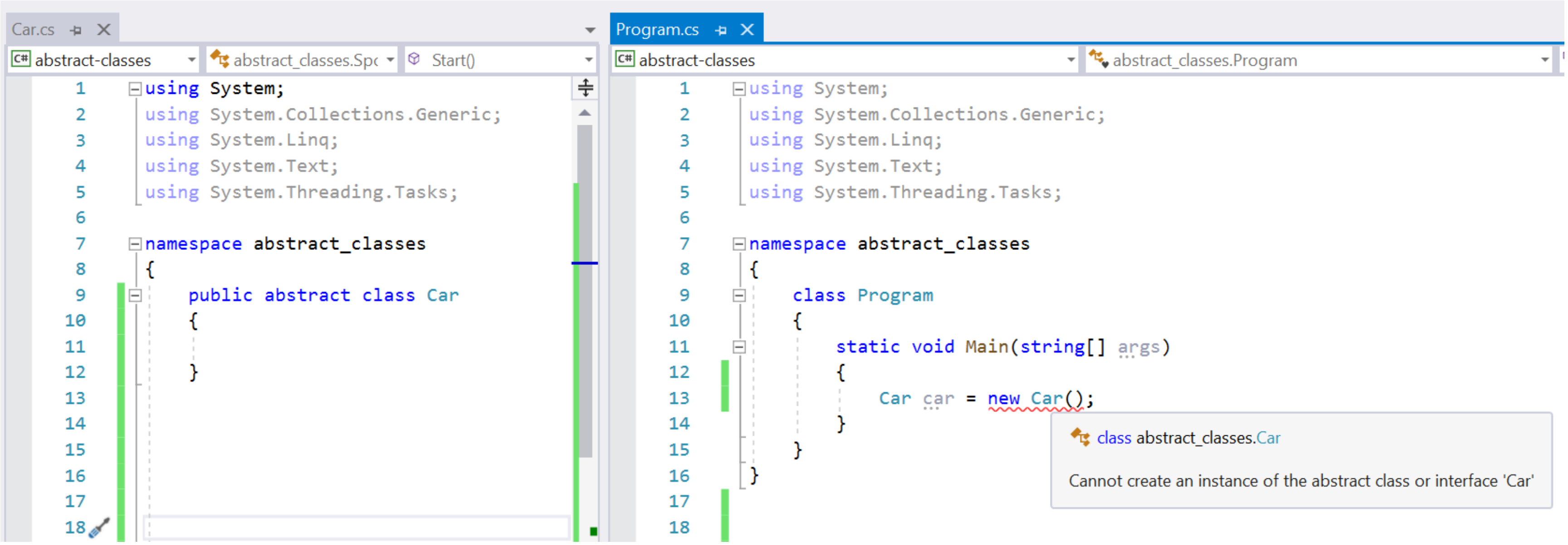
Software Developer

Learning C-Sharp

Abstract Classes

What is an Abstract class?

- By convention, an abstract class must be defined with a keyword ‘abstract’.
- You cannot create an instance of an abstract class.
- You can use Abstract Class instead of an Interface if you want to force some behaviour on Derived Classes and it also lets you define implementation of functions if you want.
- An Abstract Class can inherit from another Abstract Class.
- If you try to create an object of an abstract class, compiler will show you an error as shown.



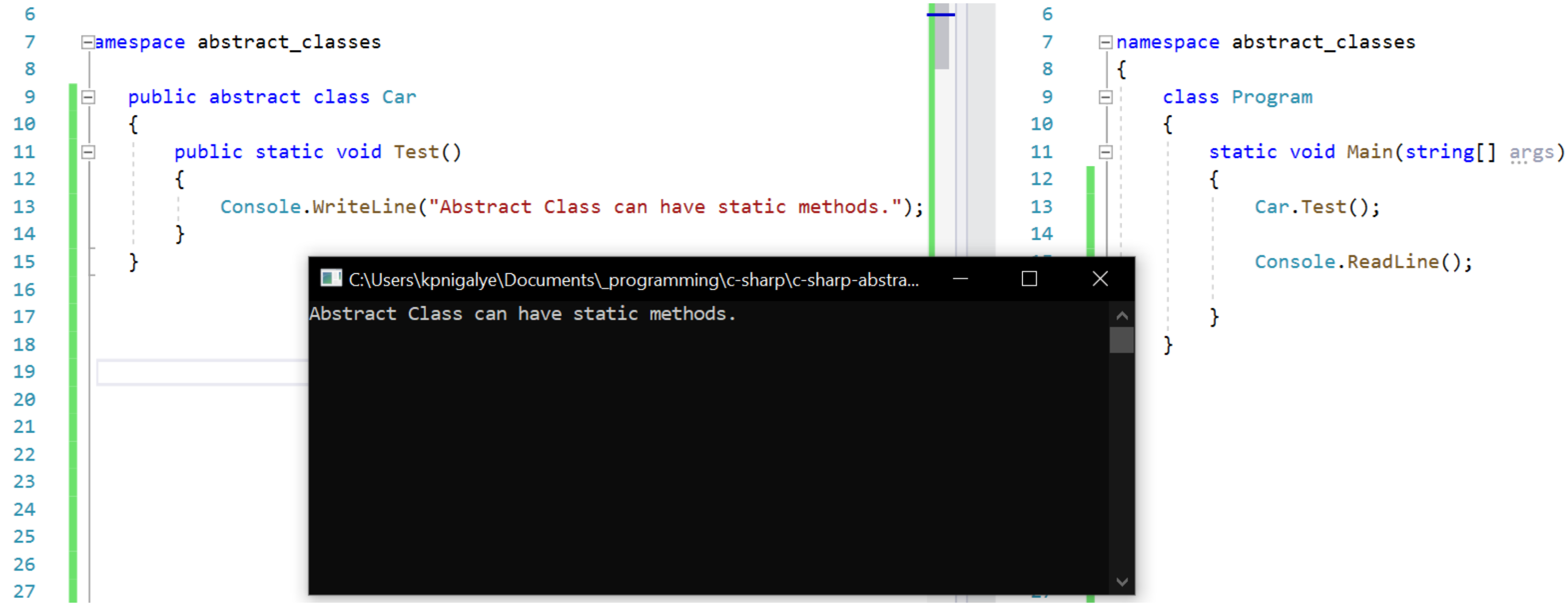
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace abstract_classes
{
    public abstract class Car
    {

    }
```

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

namespace abstract_classes
{
    class Program
    {
        static void Main(string[] args)
        {
            Car car = new Car();
        }
    }
}
```

class abstract_classes.Car

Cannot create an instance of the abstract class or interface 'Car'

- ➢ Abstract classes can have static methods.



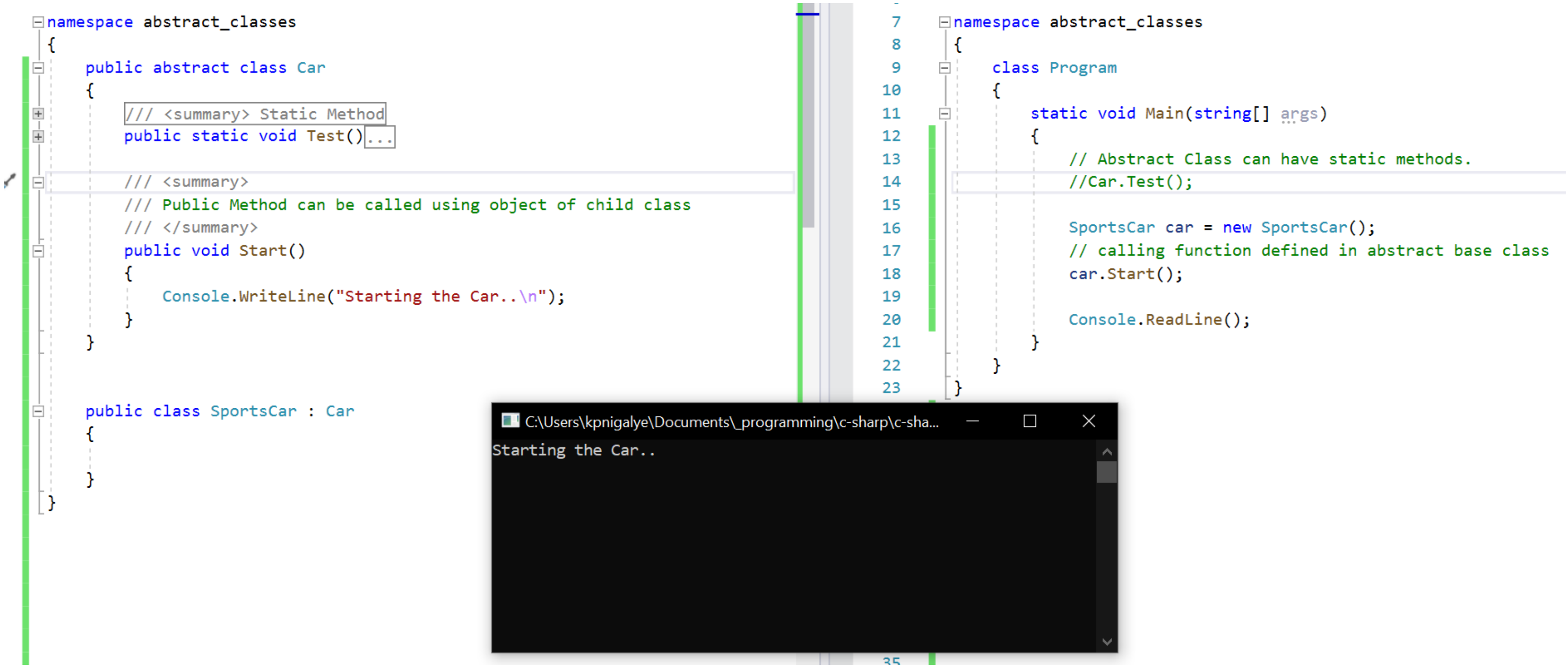
```
amespace abstract_classes

    public abstract class Car
    {
        public static void Test()
        {
            Console.WriteLine("Abstract Class can have static methods.");
        }
    }
```

C:\Users\kpnigalye\Documents_programming\c-sharp\c-sharp-abstra...

```
Abstract Class can have static methods.
```

```
namespace abstract_classes
{
    class Program
    {
        static void Main(string[] args)
        {
            Car.Test();

            Console.ReadLine();

        }
    }
```

- ➢ You can have a class deriving from an abstract class. So now you have a class ‘SportsCar’ deriving from the abstract class ‘Car’. You can call the method of abstract class using the object of derived class.



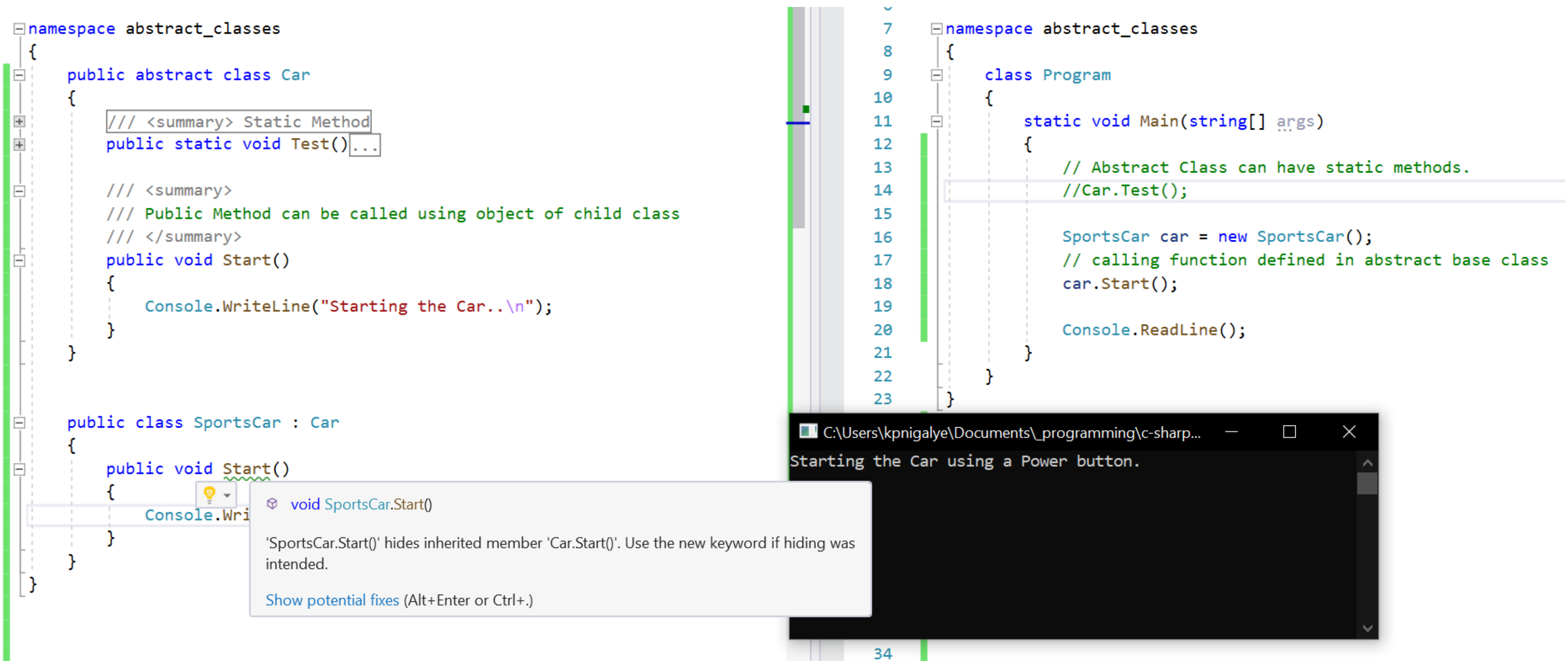
```
namespace abstract_classes
{
    public abstract class Car
    {
        /// <summary> Static Method
        public static void Test()...

        /// <summary>
        /// Public Method can be called using object of child class
        /// </summary>
        public void Start()
        {
            Console.WriteLine("Starting the Car..\n");
        }
    }


    public class SportsCar : Car
    {

    }
}
```

```
namespace abstract_classes
{
    class Program
    {
        static void Main(string[] args)
        {
            // Abstract Class can have static methods.
            //Car.Test();

            SportsCar car = new SportsCar();
            // calling function defined in abstract base class
            car.Start();

            Console.ReadLine();
        }
    }
}
```

C:\Users\kpnigalye\Documents_programming\c-sharp\c-sha...

```
Starting the Car..
```

- If you try to define the same ‘Start’ method in the derived class, then the compiler will allow you to run your code but it will show you a warning message.



```
namespace abstract_classes
{
    public abstract class Car
    {
        /// <summary> Static Method
        public static void Test()...

        /// <summary>
        /// Public Method can be called using object of child class
        /// </summary>
        public void Start()
        {
            Console.WriteLine("Starting the Car..\n");
        }
    }


    public class SportsCar : Car
    {
        public void Start()
        {
            Console.Wri
        }
    }
}
```

void SportsCar.Start()

'SportsCar.Start()' hides inherited member 'Car.Start()'. Use the new keyword if hiding was intended.

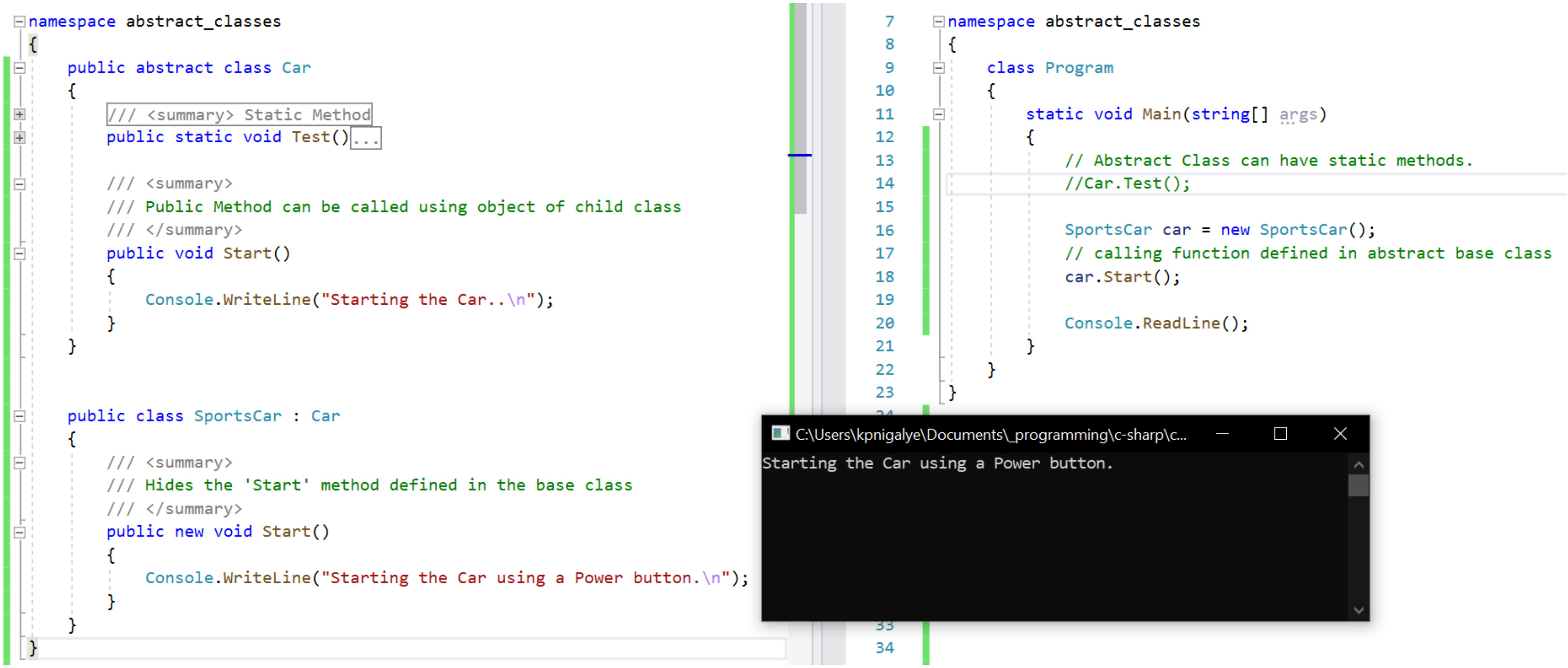
Show potential fixes (Alt+Enter or Ctrl+.)

```
namespace abstract_classes
{
    class Program
    {
        static void Main(string[] args)
        {
            // Abstract Class can have static methods.
            //Car.Test();

            SportsCar car = new SportsCar();
            // calling function defined in abstract base class
            car.Start();

            Console.ReadLine();
        }
    }
}
```

C:\Users\kpnigalye\Documents_programming\c-sharp...

```
Starting the Car using a Power button.
```

You can avoid this warning by declaring the start method in the child class by using ‘new’ keyword. You can see now the warning message has disappeared.



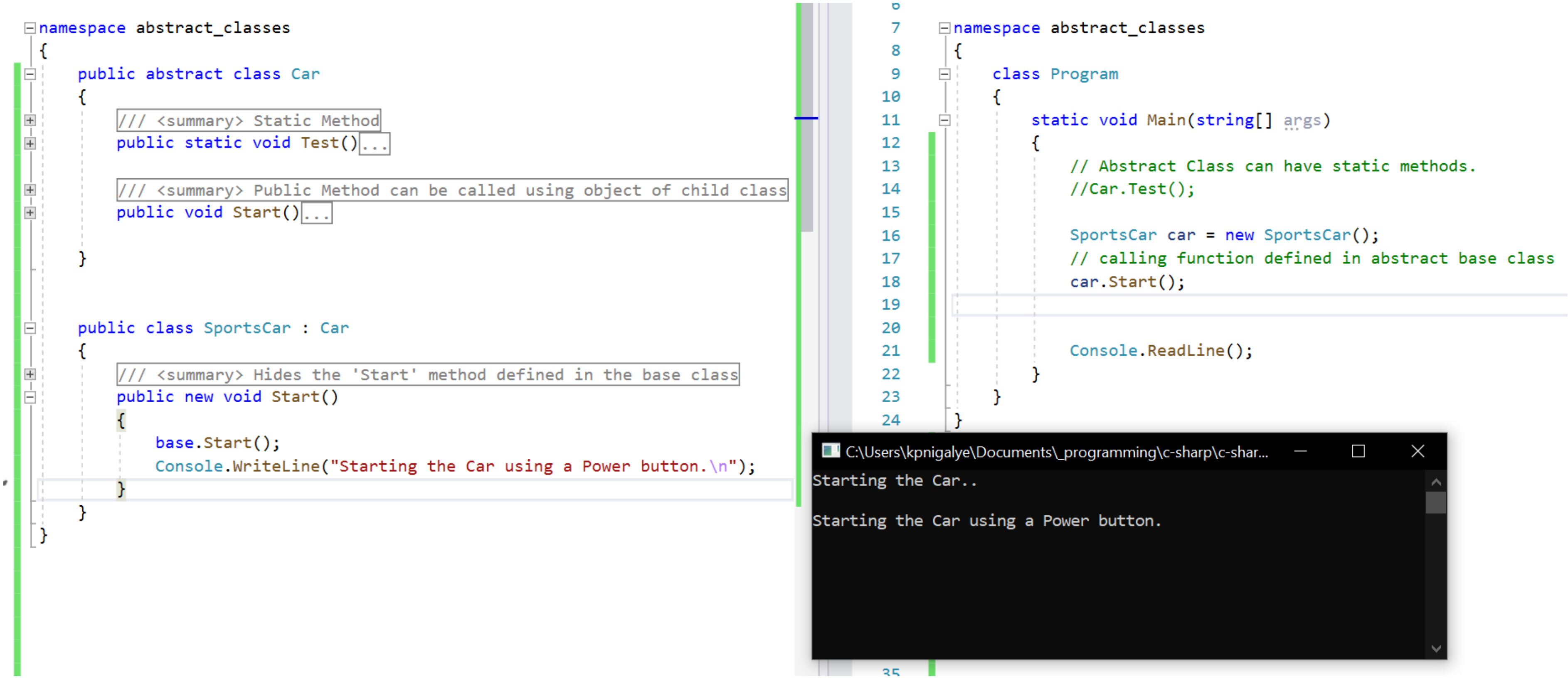
```
namespace abstract_classes
{
    public abstract class Car
    {
        /// <summary> Static Method
        public static void Test()...

        /// <summary>
        /// Public Method can be called using object of child class
        /// </summary>
        public void Start()
        {
            Console.WriteLine("Starting the Car..\n");
        }
    }


    public class SportsCar : Car
    {
        /// <summary>
        /// Hides the 'Start' method defined in the base class
        /// </summary>
        public new void Start()
        {
            Console.WriteLine("Starting the Car using a Power button.\n");
        }
    }
}
```

```
namespace abstract_classes
{
    class Program
    {
        static void Main(string[] args)
        {
            // Abstract Class can have static methods.
            //Car.Test();

            SportsCar car = new SportsCar();
            // calling function defined in abstract base class
            car.Start();

            Console.ReadLine();
        }
    }
}
```

C:\Users\kpnigalye\Documents_programming\c-sharp\c...

```
Starting the Car using a Power button.
```

- ➢ You can call the same method defined in the base class from parent class using 'base' keyword.



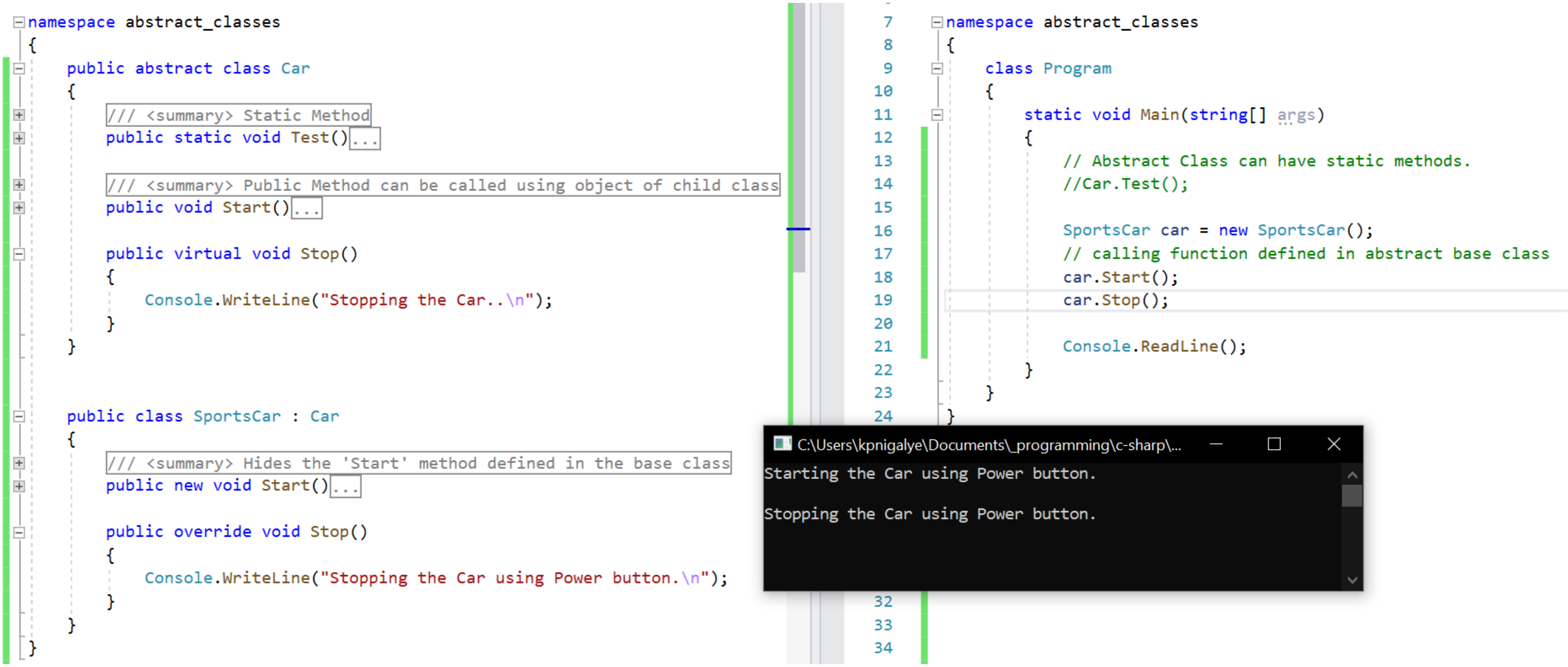
```
namespace abstract_classes
{
    public abstract class Car
    {
        /// <summary> Static Method
        public static void Test()...

        /// <summary> Public Method can be called using object of child class
        public void Start()...

    }


    public class SportsCar : Car
    {
        /// <summary> Hides the 'Start' method defined in the base class
        public new void Start()
        {
            base.Start();
            Console.WriteLine("Starting the Car using a Power button.\n");
        }
    }
}
```

```
namespace abstract_classes
{
    class Program
    {
        static void Main(string[] args)
        {
            // Abstract Class can have static methods.
            //Car.Test();

            SportsCar car = new SportsCar();
            // calling function defined in abstract base class
            car.Start();


            Console.ReadLine();
        }
    }
}
```

C:\Users\kpnigalye\Documents_programming\c-sharp\c-shar...

```
Starting the Car..

Starting the Car using a Power button.
```

- When you declare a method in a base class as ‘Virtual’, a derived class can provide its own implementation by declaring the method using ‘override’ keyword.



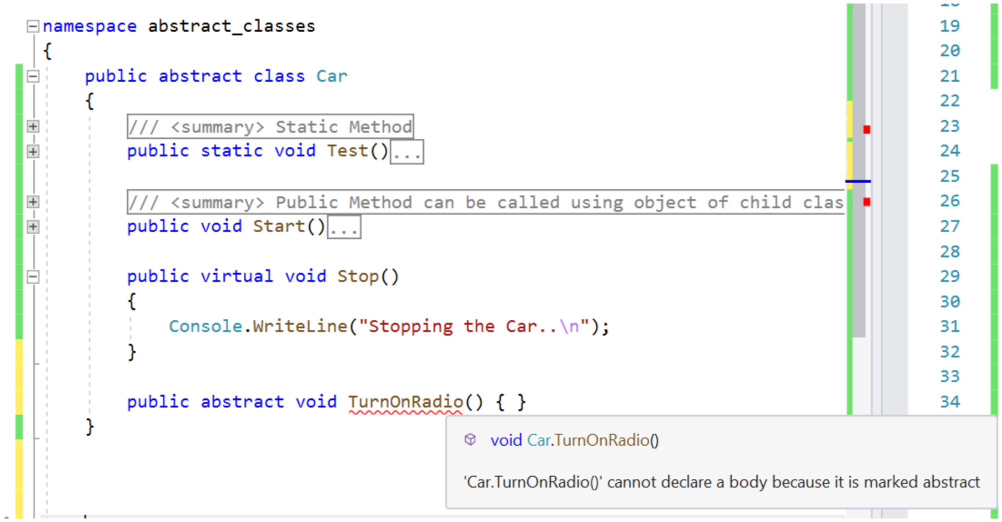
```
namespace abstract_classes
{
    public abstract class Car
    {
        /// <summary> Static Method
        public static void Test()...

        /// <summary> Public Method can be called using object of child class
        public void Start()...

        public virtual void Stop()
        {
            Console.WriteLine("Stopping the Car..\n");
        }
    }


    public class SportsCar : Car
    {
        /// <summary> Hides the 'Start' method defined in the base class
        public new void Start()...

        public override void Stop()
        {
            Console.WriteLine("Stopping the Car using Power button.\n");
        }
    }
}
```

```
namespace abstract_classes
{
    class Program
    {
        static void Main(string[] args)
        {
            // Abstract Class can have static methods.
            //Car.Test();

            SportsCar car = new SportsCar();
            // calling function defined in abstract base class
            car.Start();
            car.Stop();

            Console.ReadLine();
        }
    }
}
```

C:\Users\kpnigalye\Documents_programming\c-sharp\...

```
Starting the Car using Power button.

Stopping the Car using Power button.
```

➢ Base class can have a method defined as 'abstract'. When you declare a method as 'abstract', compiler won't allow you to write its implementation.

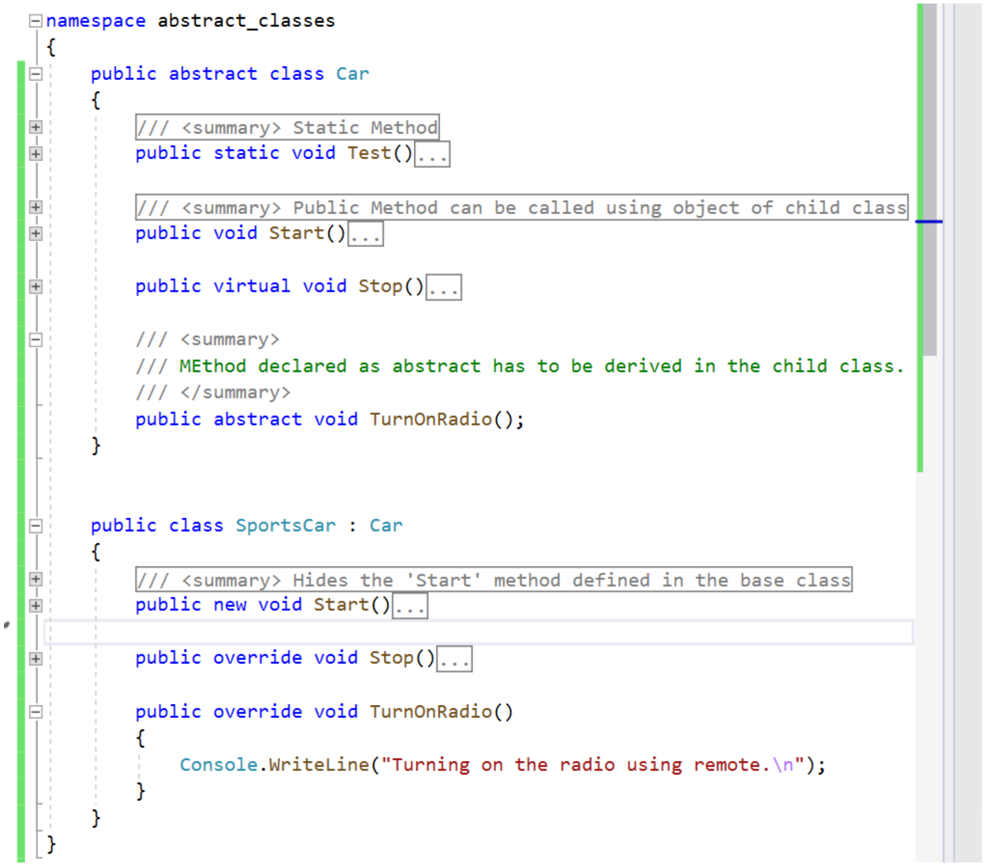


```
namespace abstract_classes
{
    public abstract class Car
    {
        /// <summary> Static Method
        public static void Test()...

        /// <summary> Public Method can be called using object of child clas
        public void Start()...

        public virtual void Stop()
        {
            Console.WriteLine("Stopping the Car..\n");
        }

        public abstract void TurnOnRadio() { }
    }
```

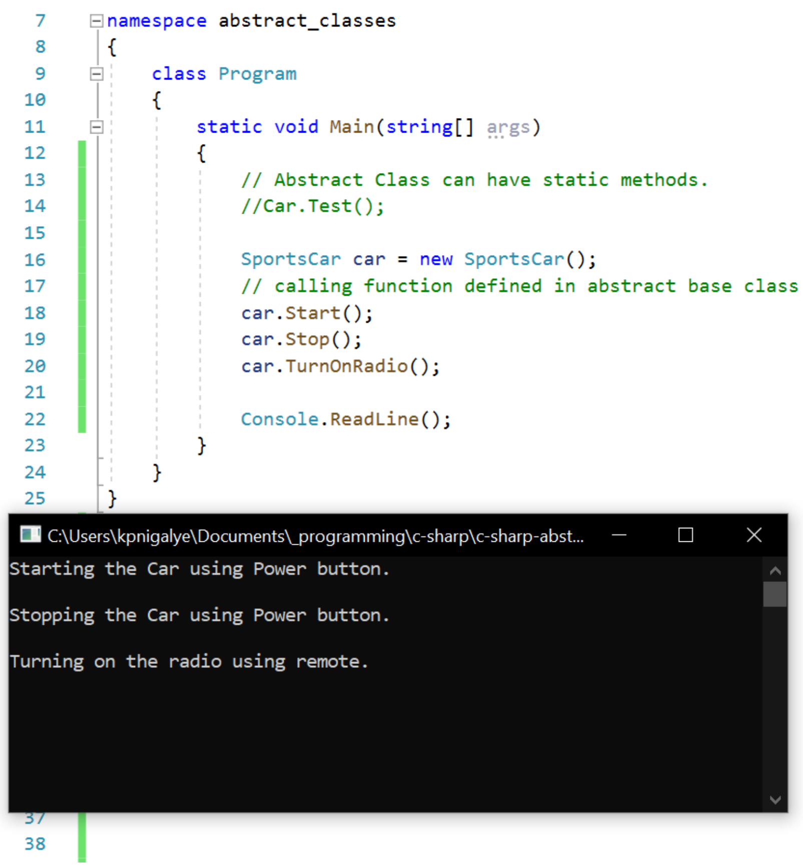
void Car.TurnOnRadio()

'Car.TurnOnRadio()' cannot declare a body because it is marked abstract

➢ When you mark a method in base class as 'abstract', you have to provide its implementation in derived class by using 'override' keyword.



```
namespace abstract_classes
{
    public abstract class Car
    {
        /// <summary> Static Method
        public static void Test()...

        /// <summary> Public Method can be called using object of child class
        public void Start()...

        public virtual void Stop()...

        /// <summary>
        /// MEthod declared as abstract has to be derived in the child class.
        /// </summary>
        public abstract void TurnOnRadio();
    }


    public class SportsCar : Car
    {
        /// <summary> Hides the 'Start' method defined in the base class
        public new void Start()...

        public override void Stop()...

        public override void TurnOnRadio()
        {
            Console.WriteLine("Turning on the radio using remote.\n");
        }
    }
}
```

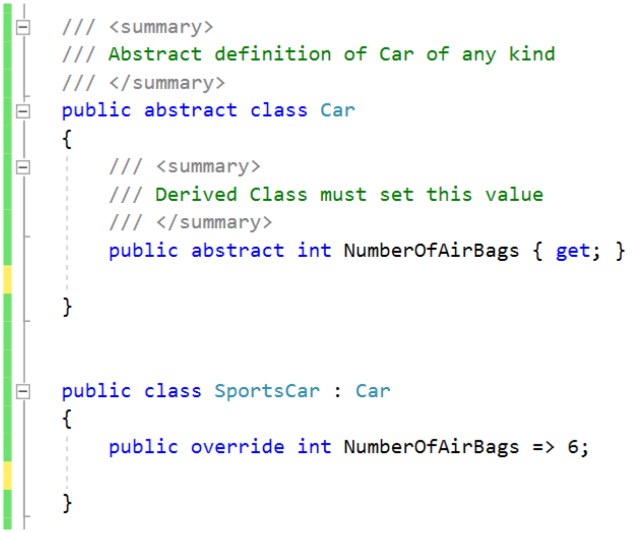


```
namespace abstract_classes
{
    class Program
    {
        static void Main(string[] args)
        {
            // Abstract Class can have static methods.
            //Car.Test();

            SportsCar car = new SportsCar();
            // calling function defined in abstract base class
            car.Start();
            car.Stop();
            car.TurnOnRadio();

            Console.ReadLine();
        }
    }
}
```

C:\Users\kpnigalye\Documents_programming\c-sharp\c-sharp-abst...

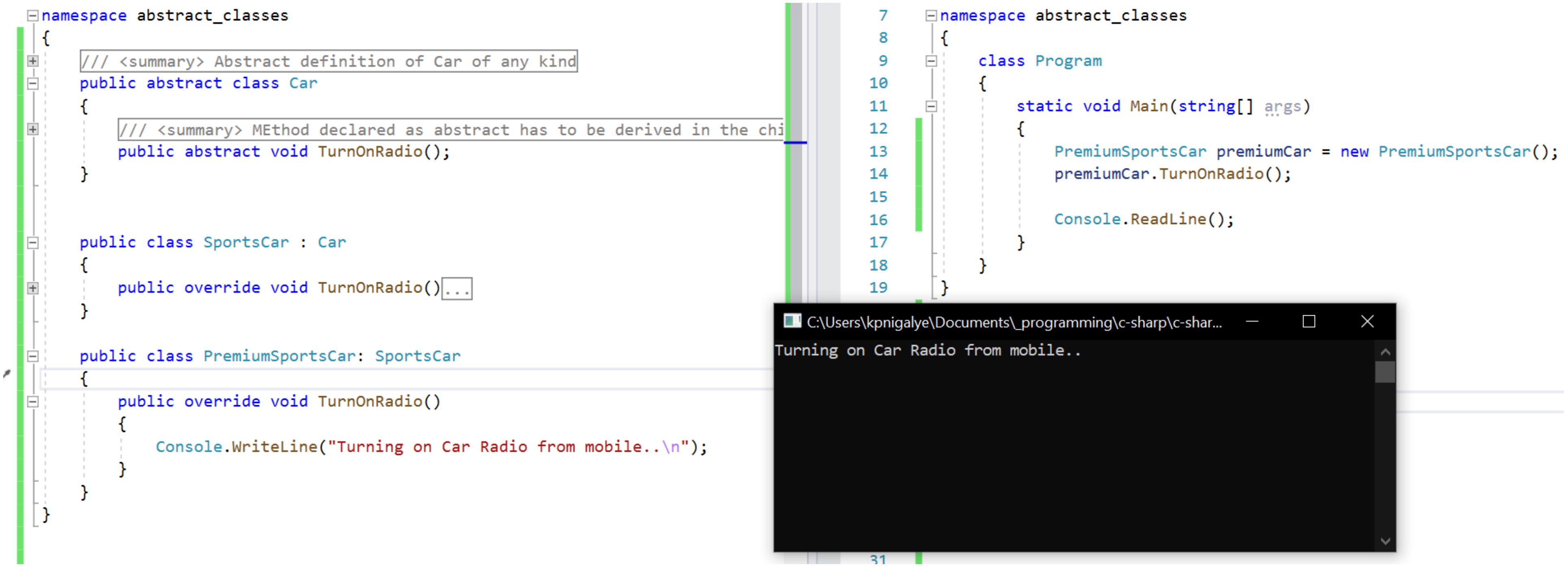
```
Starting the Car using Power button.

Stopping the Car using Power button.

Turning on the radio using remote.
```

- You can also have a property in abstract class defined as ‘abstract’. It will become mandatory for derived class to set a value for this property using ‘override’ keyword.



```
/// <summary>
/// Abstract definition of Car of any kind
/// </summary>
public abstract class Car
{
    /// <summary>
    /// Derived Class must set this value
    /// </summary>
    public abstract int NumberOfAirBags { get; }

}


public class SportsCar : Car
{
    public override int NumberOfAirBags => 6;

}
```

- If you have a class say ’PremiumSportsCar’ which is deriving from ‘SportsCar’, then you can further override the method defined in the ‘SportsCar’ class using ‘override’ keyword.



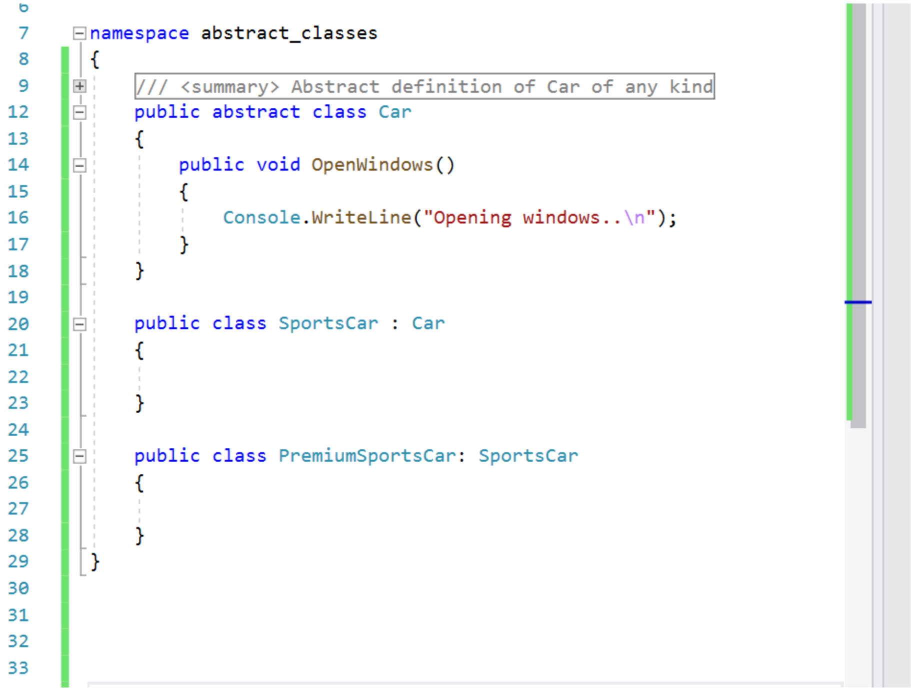
```
namespace abstract_classes
{
    /// <summary> Abstract definition of Car of any kind
    public abstract class Car
    {
        /// <summary> MEthod declared as abstract has to be derived in the chi
        public abstract void TurnOnRadio();
    }


    public class SportsCar : Car
    {
        public override void TurnOnRadio()...
    }

    public class PremiumSportsCar: SportsCar
    {
        public override void TurnOnRadio()
        {
            Console.WriteLine("Turning on Car Radio from mobile..\n");
        }
    }
}
```

```
namespace abstract_classes
{
    class Program
    {
        static void Main(string[] args)
        {
            PremiumSportsCar premiumCar = new PremiumSportsCar();
            premiumCar.TurnOnRadio();

            Console.ReadLine();
        }
    }
}
```

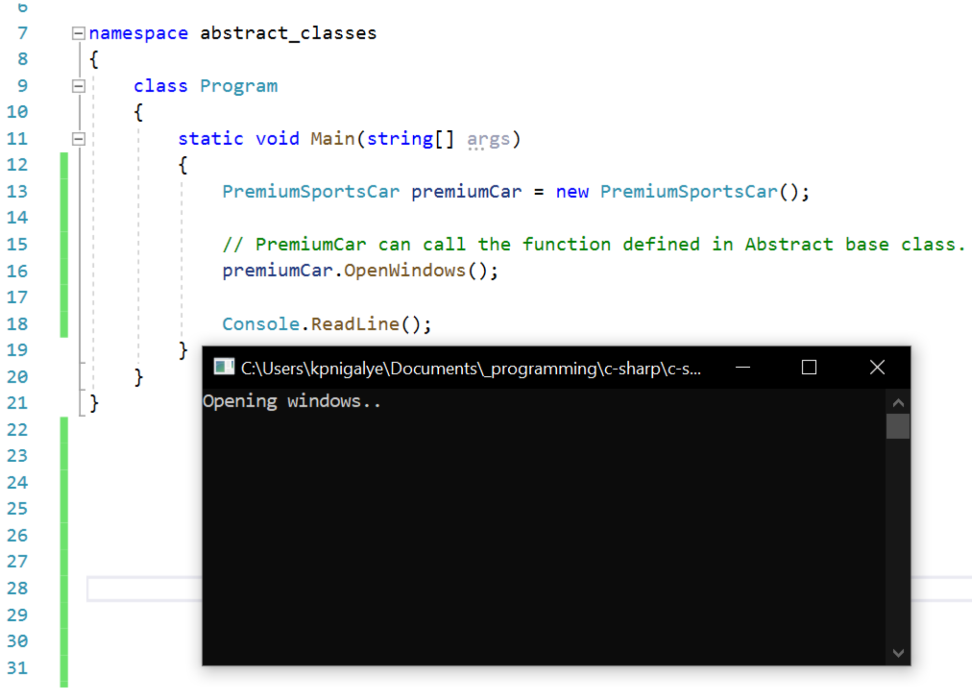
C:\Users\kpnigalye\Documents_programming\c-sharp\c-shar...

```
Turning on Car Radio from mobile..
```

- ’PremiumSportsCar’ which is deriving from ‘SportsCar’ which in turn derives from abstract base class ‘Car’ can call a function defined in ‘Car’ class.



```
namespace abstract_classes
{
    /// <summary> Abstract definition of Car of any kind
    public abstract class Car
    {
        public void OpenWindows()
        {
            Console.WriteLine("Opening windows..\n");
        }
    }

    public class SportsCar : Car
    {

    }

    public class PremiumSportsCar: SportsCar
    {

    }
}
```

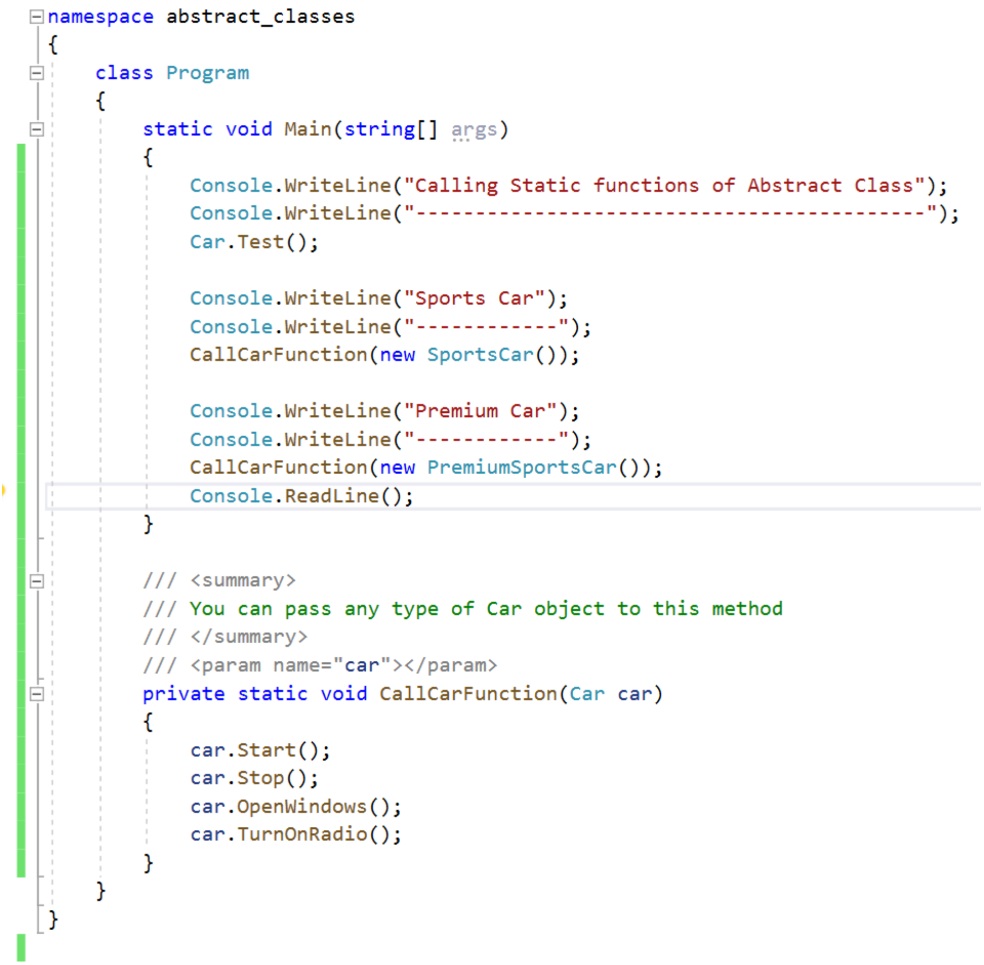


```
namespace abstract_classes
{
    class Program
    {
        static void Main(string[] args)
        {
            PremiumSportsCar premiumCar = new PremiumSportsCar();

            // PremiumCar can call the function defined in Abstract base class.
            premiumCar.OpenWindows();

            Console.ReadLine();
        }
    }
}
```

C:\Users\kpnigalye\Documents_programming\c-sharp\c-s...

```
Opening windows..
```

- You can assign a reference of derived class to the Car class object and it will work the same way. Check out the ‘CallCarFunction’ method defined which calls the functions related to Car objects.



```
namespace abstract_classes
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Calling Static functions of Abstract Class");
            Console.WriteLine("-------------------------------------------");
            Car.Test();

            Console.WriteLine("Sports Car");
            Console.WriteLine("------------");
            CallCarFunction(new SportsCar());

            Console.WriteLine("Premium Car");
            Console.WriteLine("------------");
            CallCarFunction(new PremiumSportsCar());
            Console.ReadLine();
        }

        /// <summary>
        /// You can pass any type of Car object to this method
        /// </summary>
        /// <param name="car"></param>
        private static void CallCarFunction(Car car)
        {
            car.Start();
            car.Stop();
            car.OpenWindows();
            car.TurnOnRadio();
        }
    }
}
```