Explicit Interface Implementation

Explicit Implementation

Implement Interface

Explicitly Implement Interface

Class with No Interface

Declaration

```
public class Catalog : ISaveable
{
  public string Save()
  {
    return "Catalog Save";
  }

  // Other members not shown
}
```

Usage

```
Catalog catalog = new Catalog();
catalog.Save(); // "Catalog Save"
```

Standard Interface Implementation

Declaration

```
public interface ISaveable
{
  string Save();
}
public class Catalog : ISaveable
  {
    public string Save()
    {
      return "Catalog Save";
    }

    // Other members not shown
  }
```

Usage

```
Catalog catalog = new Catalog();
catalog.Save(); // "Catalog Save"

ISaveable saveable = new Catalog();
saveable.Save(); // "Catalog Save"
```

Explicit Interface Implementation

Declaration

```
public class Catalog : ISaveable
{
   public string Save()
   {
     return "Catalog Save";
   }
   string ISaveable.Save()
   {
     return "ISaveable Save";
   }
   // Other members not shown
 }
```

Concrete Type

```
Catalog catalog = new Catalog();
catalog.Save(); // "Catalog Save"
```

Interface Variable

```
ISaveable saveable = new Catalog();
saveable.Save(); // “ISaveable Save"
```

Cast to Interface

```
((ISaveable)catalog).Save();
//  “ISaveable Save"
```

Explicit Interface Implementation

Declaration

```
public class Catalog : ISaveable
{
   string ISaveable.Save()
   {
     return “ISaveable Save";
    }
   // Save() deleted
   // Other members not shown
 }
```

Concrete Type

```
Catalog catalog = new Catalog();
catalog.Save(); // **COMPILER ERROR**
```

Interface Variable

```
ISaveable saveable = new Catalog();
saveable.Save(); // “ISaveable Save"
```

Cast to Interface

```
((ISaveable)catalog).Save();
//  “ISaveable Save"
```

Mandatory Explicit Implementation

Declaration A

```
public interface ISaveable
{
  string Save();
}
```

Declaration B

```
public interface IVoidSaveable
{
  void Save();
}
```

Implementation

```
public class Catalog :
  ISaveable, IVoidSaveable
{
  public string Save()
  {
    return "Catalog Save";
  }
   void IVoidSaveable.Save()

  {
    // no return value
  }
   // Other members not shown
}
```

Mandatory Explicit Implementation

Declaration A

```
public interface ISaveable
{
  string Save();
}
```

Declaration B

```
public interface IVoidSaveable
{
  void Save();
}
```

Implementation

```
public class Catalog :
  ISaveable, IVoidSaveable
{
  string ISaveable.Save()
  {
    return "ISaveable Save";
  }
  public void Save()

  {
    // no return value
  }
   // Other members not shown
}
```

Mandatory Explicit Implementation

Declaration A

```
public interface ISaveable
{
  string Save();
}
```

Declaration B

```
public interface IVoidSaveable
{
  void Save();
}
```

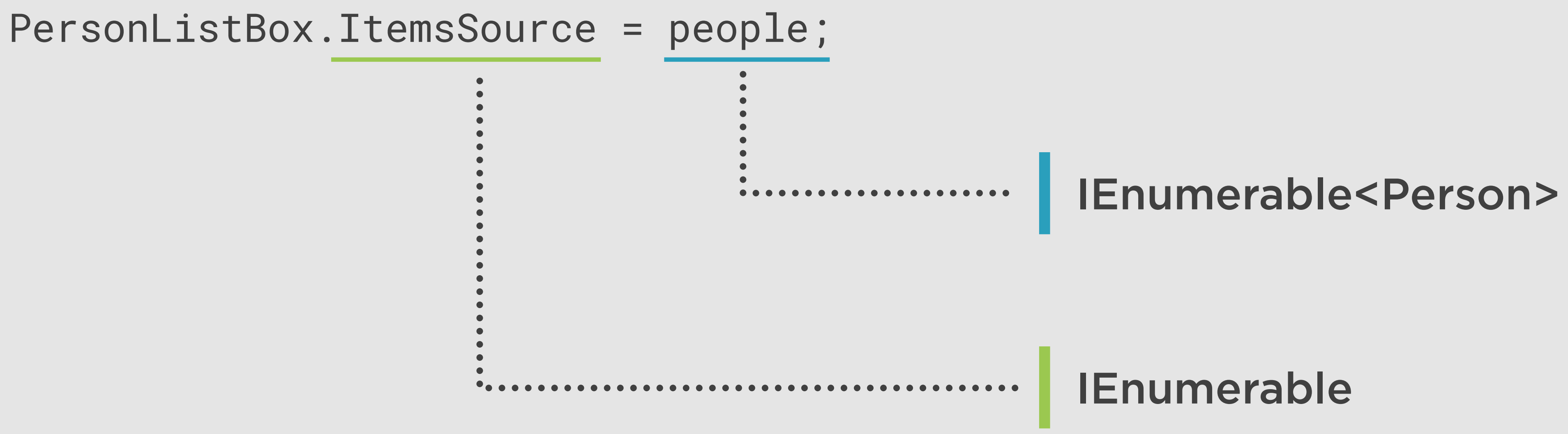
Implementation

```
public class Catalog :
  ISaveable, IVoidSaveable
{
  string ISaveable.Save()
  {
    return "ISaveable Save";
  }
  void IVoidSaveable.Save()

  {
    // no return value
  }
   // Other members not shown
}
```

Type Mismatch?

```
PersonListBox.ItemsSource = people;
```



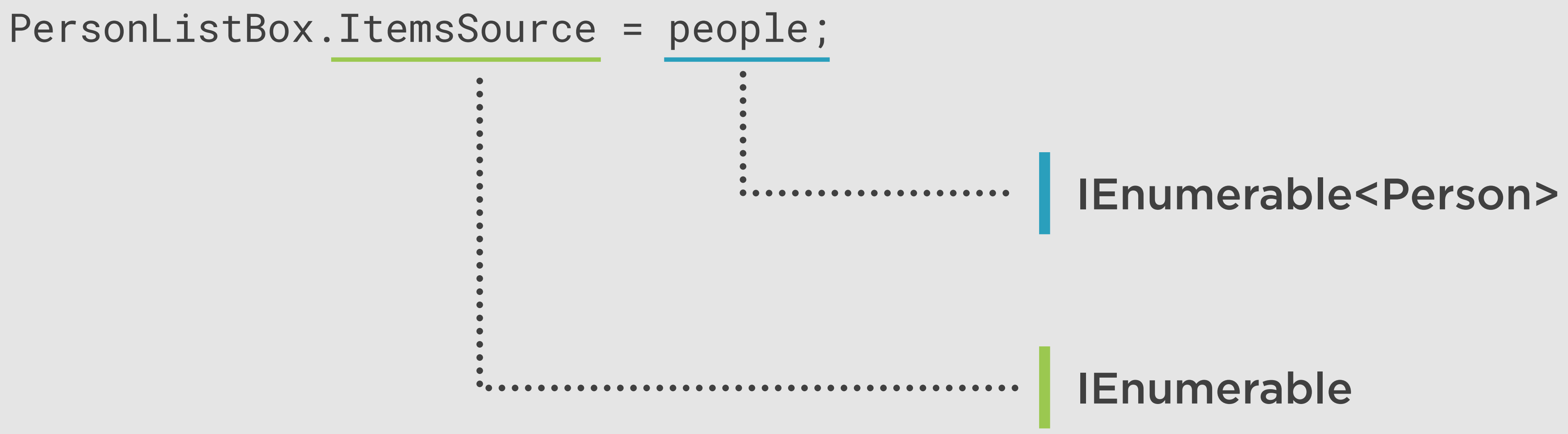
```
public interface IEnumerable<T> : IEnumerable
```

Interface Inheritance

IEnumerable<T> inherits IEnumerable

When a class implements IEnumerable<T>, it must also implement IEnumerable

Type Mismatch?



```
PersonListBox.ItemsSource = people;
```

Type Mismatch?

```
PersonListBox.ItemsSource = people;
```

IEnumerable<Person>
+ IEnumerable

IEnumerable

Interface Members

IEnumerable<T> Members

```
public interface IEnumerable<T>: IEnumerable
  {
    IEnumerator<T> GetEnumerator();
  }
```

IEnumerable Members

```
public interface IEnumerable
  {
    IEnumerator GetEnumerator();
  }
```

Summary

Standard Implementation

Explicit Implementation

- Save method for class
- Save method for interface

Mandatory Explicit Implementation

- Methods with Different Return Types

Interface Inheritance

- IEnumerable<T> and IEnumerable

UP NEXT:

Interfaces and Dynamic Loading