Explicit Interface Implementation

Jeremy Clark
www.jeremybytes.com
jeremy@jeremybytes.com

Class with No Interface

Declaration

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

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

```
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

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

```
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

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

```
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

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

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

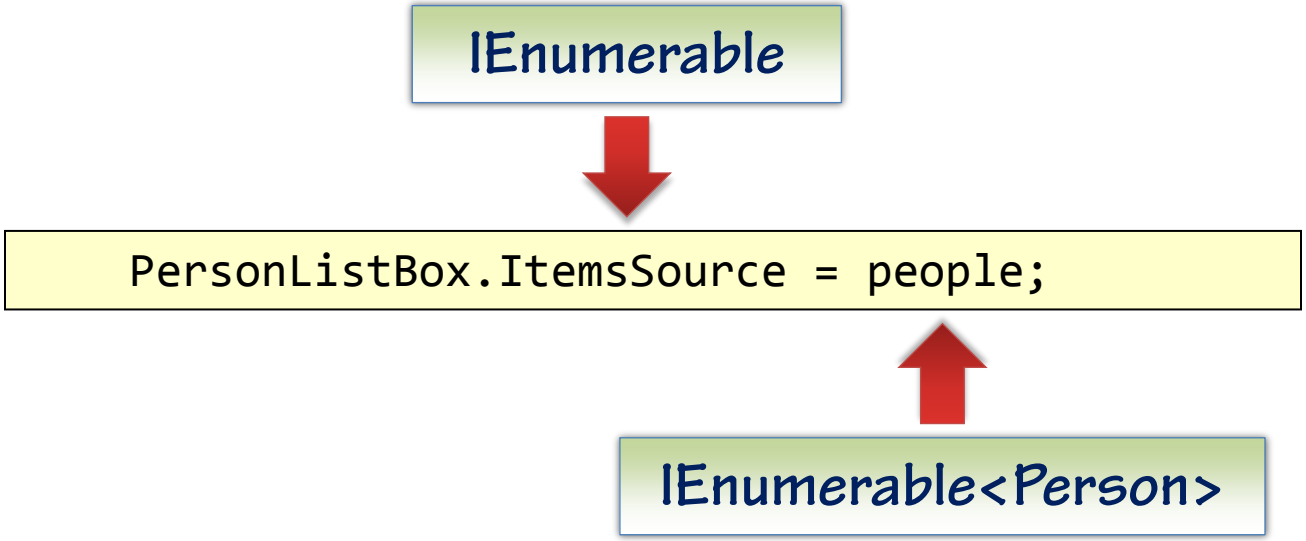
Implementation

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

  void IVoidSaveable.Save()
  {
    // no return value
  }

  // Other members not shown
}
```

Type Mismatch?

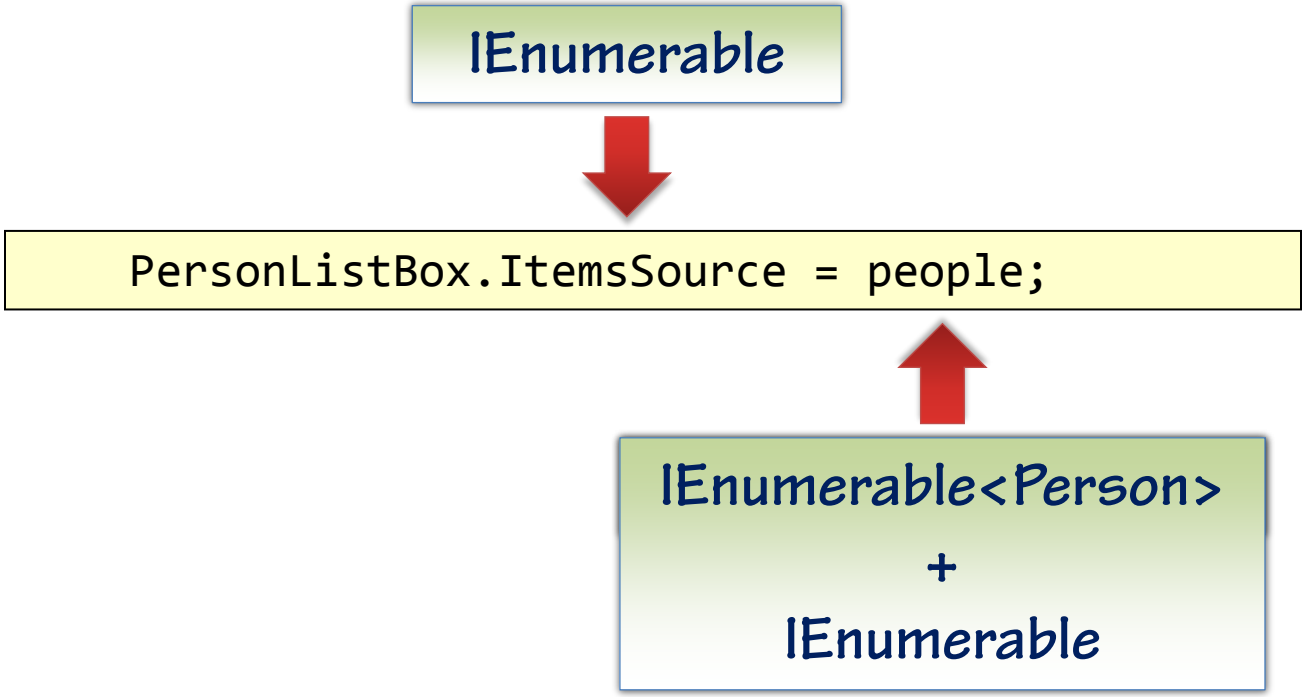


Interface Inheritance

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

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

Type Mismatch?



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**

```
string ISaveable.Save()
{
  return "ISaveable Save";
}
```

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

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

- **Mandatory Explicit Implementation**
  - Methods with Different Return Types
- **Interface Inheritance**
  - IEnumerable<T> and IEnumerable

- **Next up: Interfaces and Dynamic Loading**