Making Promises with Interfaces and Abstract Classes



Jesse Liberty

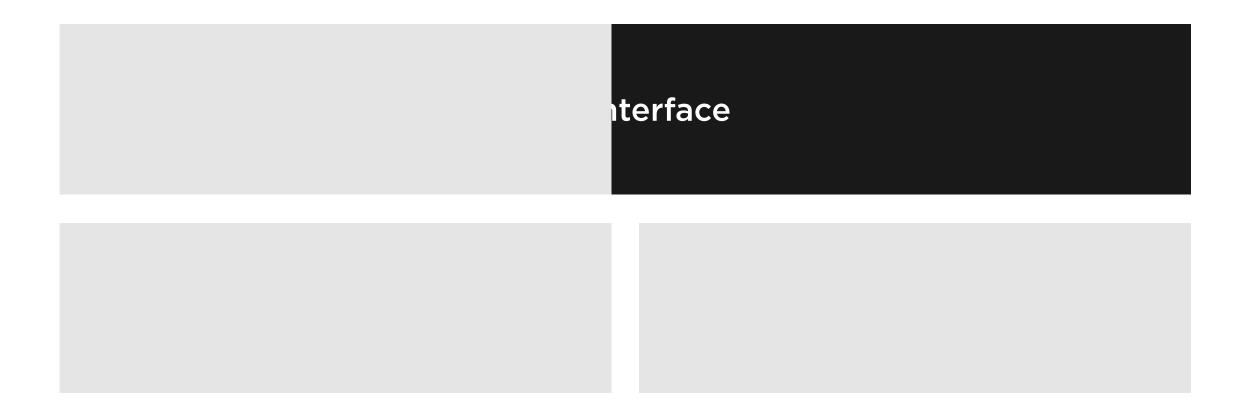
@jesseliberty http://jesseliberty.com



An Interface Is a Contract



Keeping Our Terms Straight





The Interface

```
public interface IStorable {
   void Read(string fileName);
   void Write(string fileName);
}
```



The Implementing Class

```
public class Document : IStorable {
public void Read(string fileName) {
 Console.WriteLine($"Reading {fileName} into this document");
public void Write(string fileName) {
 Console.WriteLine($"Writing {fileName} to disk...");
```



Client Class Calls Interface Methods

```
public class SomeOtherClass {
  public void SomeMethod(){
    IStorable document = new Document();
    document.Read("myFile");
    // work
    document.Write("myNewFile");
```



You Can't Instantiate an Interface

```
IStorable storable = new IStorable(); // No
```

```
IStorable storable = new Document(); // OK
```



Caution! If you do this, you can only access the methods in the Interface



Demo



Interfaces



Abstract Classes



Exist to provide a base class, but are never instantiated



Abstract Methods

Methods that must be overridden to create a concrete class.



Abstract vs. Concrete Classes

Concrete Class | Abstract Class

Acts as a base class to other classes

Can be instantiated

Cannot have abstract methods

Child classes may override methods

Acts as a base class to other classes

Can not be instantiated

Has at least one abstract method

Concrete child classes *must* override all abstract methods



Abstract classes can have abstract children, until all abstract methods have been overridden



Demo



Abstract Classes



Summary



An interface is a contract

An interface is fulfilled by a class

An abstract class has at least one abstract method

An abstract class cannot be instantiated

