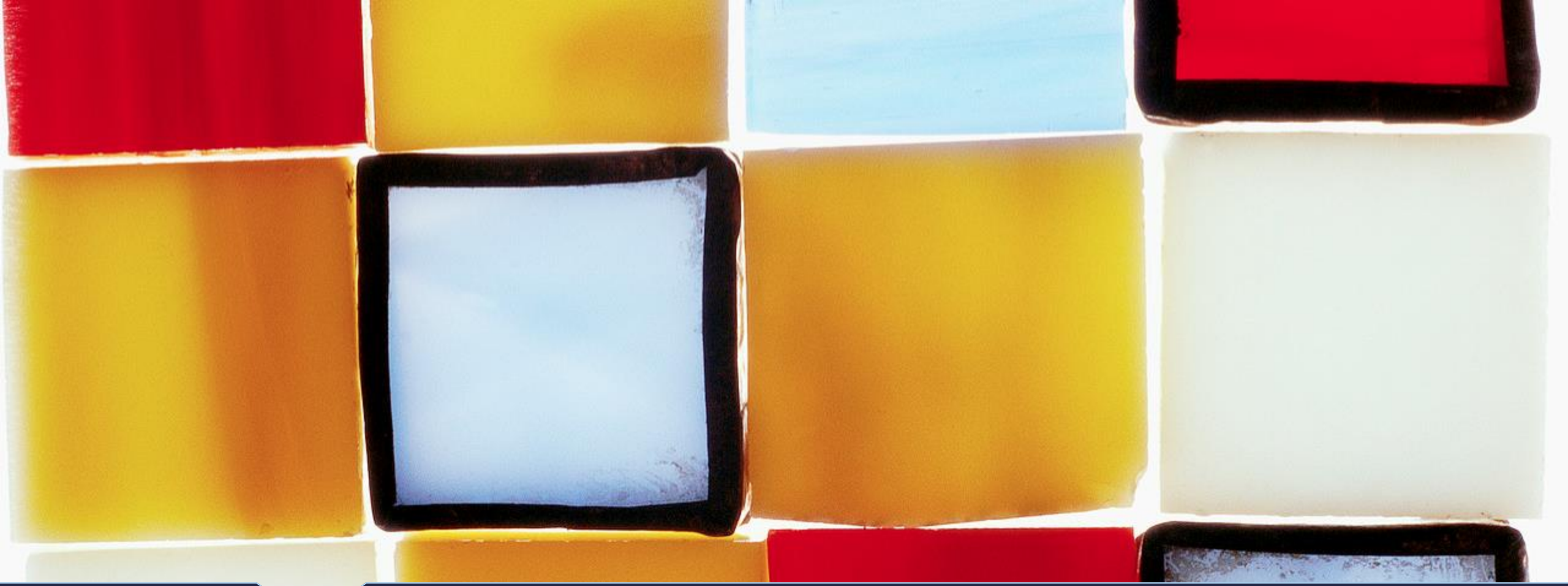
Using Abstraction for Good

(and not evil)



Program to an abstraction, not a concrete type.

Programming Best Practice

Just a Starting Point

Be sure to check your surroundings before following any recommendations.

Abstraction is Awesome!

- Maintain
- Extend
- Test

Abstraction is Awful!

Complexity

Confusion

Difficulty Debugging

Frustration

Abstract Class vs Interface

Abstract Class

- May contain implementation code
- A class may only descend from a single base class
- Members contain access modifiers
- May contain fields, properties, constructors, destructors, methods, events and indexers

Interface

- May not contain implementation code
- A class may implement any number of interfaces
- Members are automatically public
- May contain properties, methods, events, and indexers (not fields, constructors or destructors)

Abstraction Can…

- Add flexibility
  - Code that bends in the face of change
- Add separation
  - Code that doesn't have to deal with implementation details
- Add extensibility
  - Code that adapts to new functionality

a href="…"

- http://www.jeremybytes.com/Demos.aspx#INT
  - IEnumerable, ISaveable, IDontGetIt: Interfaces in .NET
  - Full walkthrough of interfaces
- http://www.jeremybytes.com/Demos.aspx#AA
  - Abstract Art: Getting Things "Just Right"
  - Collection of articles talking about the pros and cons of abstraction
- http://www.jeremybytes.com/Demos.aspx#PR
  - Practical Reflect in .NET
  - A more detailed description of the Rules Engine example

Using Abstraction for Good

Next Up: Dependency Injection