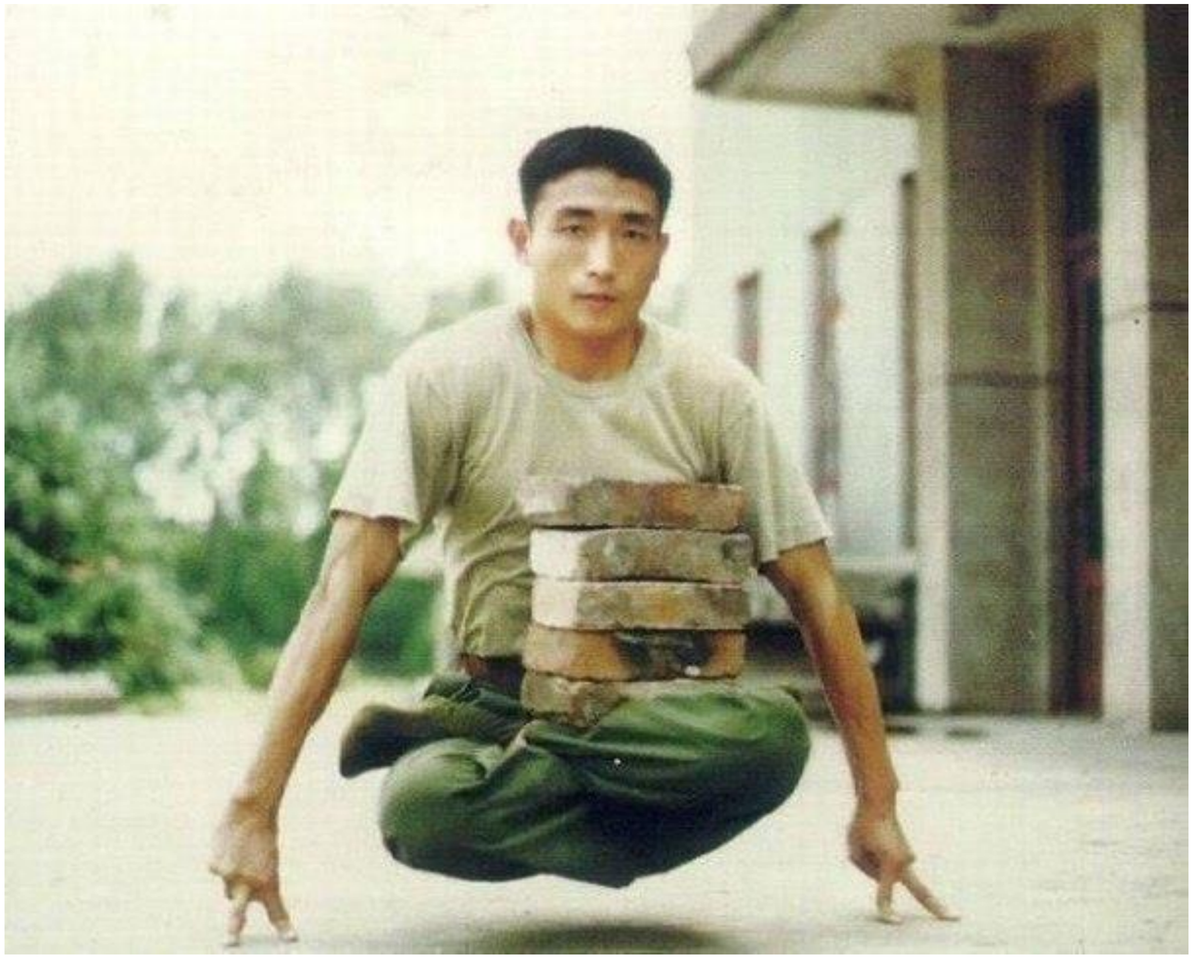
Good coding practice in real life

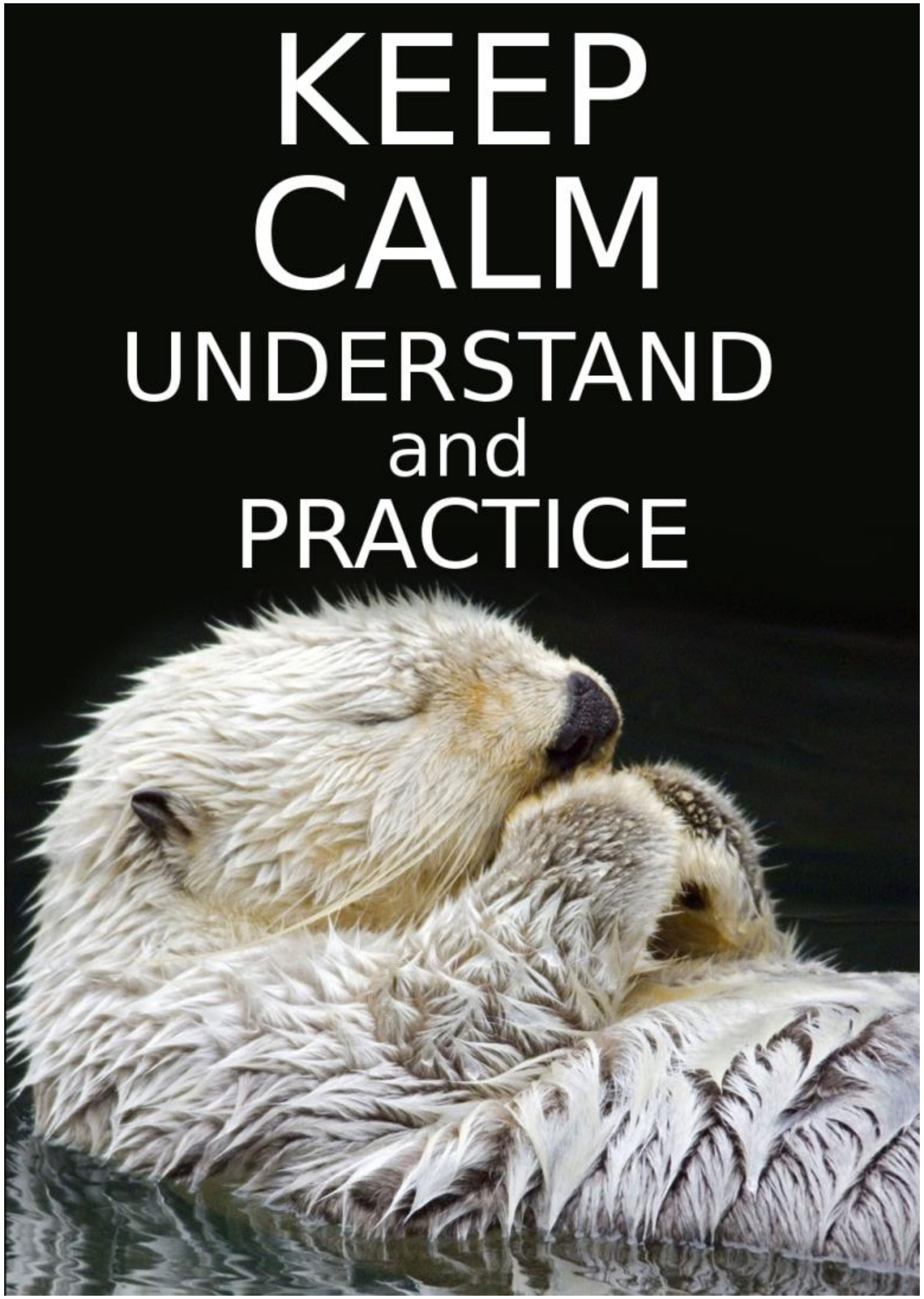
(with a focus on OCP)

Vladimir Alekseichenko

Good practices make life easier

Good practices are not easy




KEEP
CALM
UNDERSTAND
and
PRACTICE

Agenda

Open-Closed Principle

- Introduction to OCP (with related subjects).
- The definition of OCP.
- How should mean in practice?
- Design patterns that can help (Strategy, Template Method).
- Some examples.

Ivar Jacobson

http://seafood.inf.ethz.ch/2010/speakers.html

Axiom

All systems change during their life cycles.

This must be borne in mind when developing systems expected to last longer than the first version.

Alistair Cockburn

http://en.wikipedia.org/wiki/Alistair_Cockburn

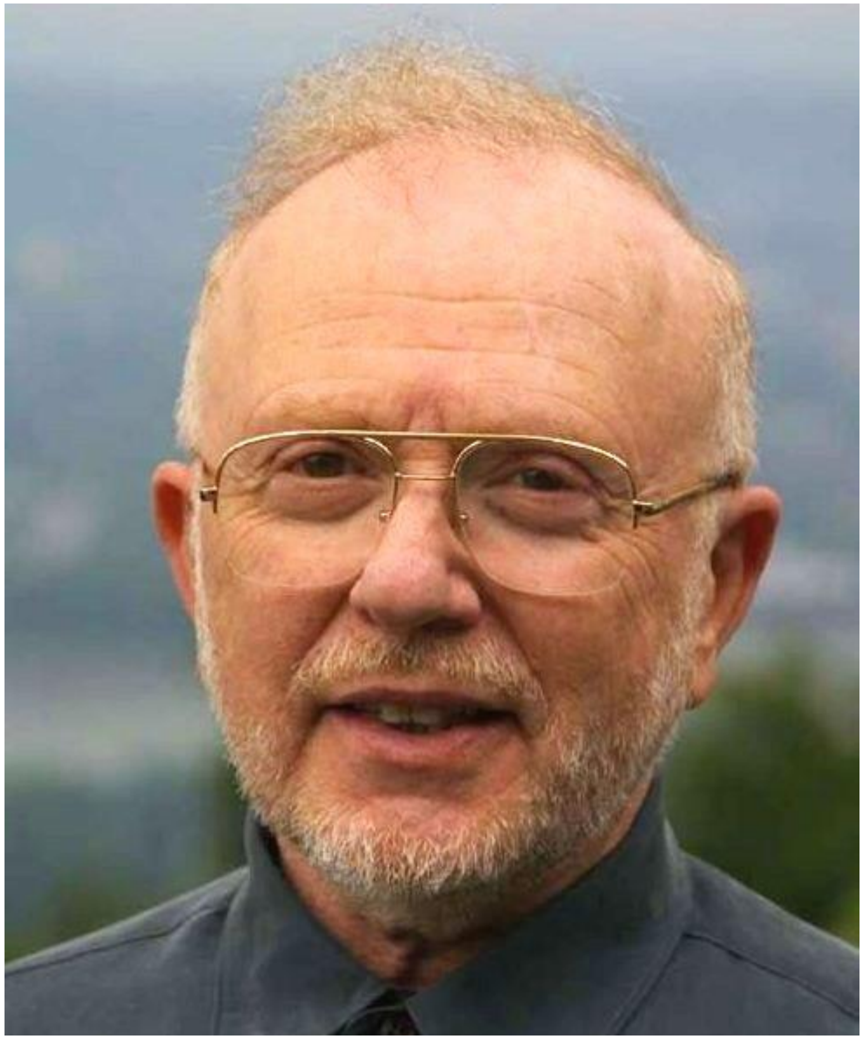
Protected variation

Identify points of predicted variation and create a stable interface around them.

OCP is essentially equivalent to the *protected variation pattern.*

http://martinfowler.com/ieeeSoftware/protectedVariation.pdf

David Parnas



http://www.techcn.com.cn/index.php?doc-view-131774.html

Hide information

… each module is then designed to hide such a decision from the others.

http://www.cs.umd.edu/class/spring2003/cmsc838p/Design/criteria.pdf

Hide information

It’s not simply data encapsulation, which is but one of many techniques **to hide design information**.

It’s the same principle expressed in PV or OCP.

http://martinfowler.com/ieeeSoftware/protectedVariation.pdf

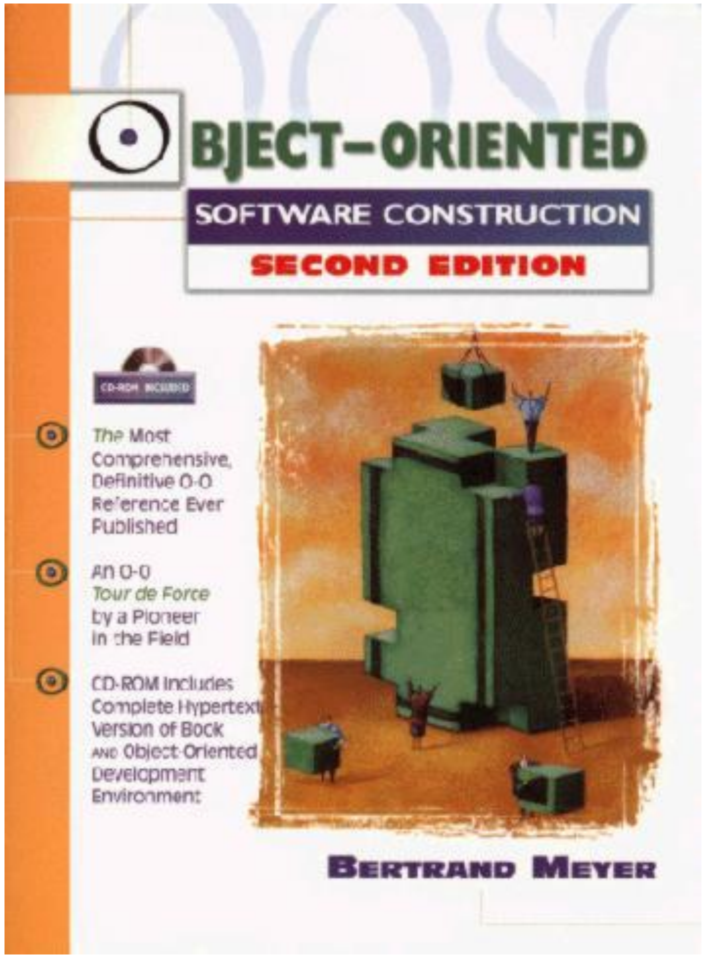
Hide information

http://theverybesttop10.com/2013/02/16/images-of-cats-in-boxes/

Bertrand Meyer

http://en.wikipedia.org/wiki/Bertrand_Meyer

Object-Oriented Software Construction


OBJECT-ORIENTED
SOFTWARE CONSTRUCTION
SECOND EDITION
CD-ROM INCLUDED
The Most Comprehensive, Definitive O-O Reference Ever Published
An O-O Tour de Force by a Pioneer in the Field
CD-ROM Includes Complete Hypertext Version of Book AND Object-Oriented Development Environment
BERTRAND MEYER

http://en.wikipedia.org/wiki/Object-Oriented_Software_Construction

Open closed principle

Modules should be both **open** and **closed**

Object Oriented Software Construction by *Bertrand Meyer*

Open closed principle

A module is said:

- to be open if it is still available for extension.
- to be closed if it is available for use by other modules.

A reason

The need for modules **to be closed**, and the need for them **to remain open**, arise for different reasons.

An openness

- **Openness** is a natural concern for **software developers**, as they know that it is almost **impossible to foresee all the elements** — data, operations — that a module will need in its lifetime.

A closure

- In a system comprising many modules, most will **depend** on some others.
- If we never closed a module until we were sure it includes all the needed features, no multi-module software **would ever reach completion**.

Robert Cecil Martin aka "Uncle Bob"

http://www.butunclebob.com/CleanCoder.UncleBob

Open closed principle

You should be **able to extend** a classes behavior, **without modifying it**.

http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod

Open closed principle

- It says that you should design modules that **never change**.
- When **requirements change**, you **extend** the

behavior of such modules by **adding new code**, **not** by **changing** old code that already works.

http://www.objectmentor.com/resources/articles/ocp.pdf

Open closed principle

Modules that conform to the open-closed principle have **two primary attributes**:

- Open for extension.
- Closed for modification.

http://www.objectmentor.com/resources/articles/ocp.pdf

Open closed principle

Open for extension

- This means that the behavior of the module **can be extended**.
- That we can make the module behave in new and different ways as the requirements of the application change, or to meet the needs of new applications.

http://www.objectmentor.com/resources/articles/ocp.pdf

Open closed principle

Closed for modification

- The source code of such a module is

inviolate.

- No one is allowed to make source code changes to it.

http://www.objectmentor.com/resources/articles/ocp.pdf

On the other hand…

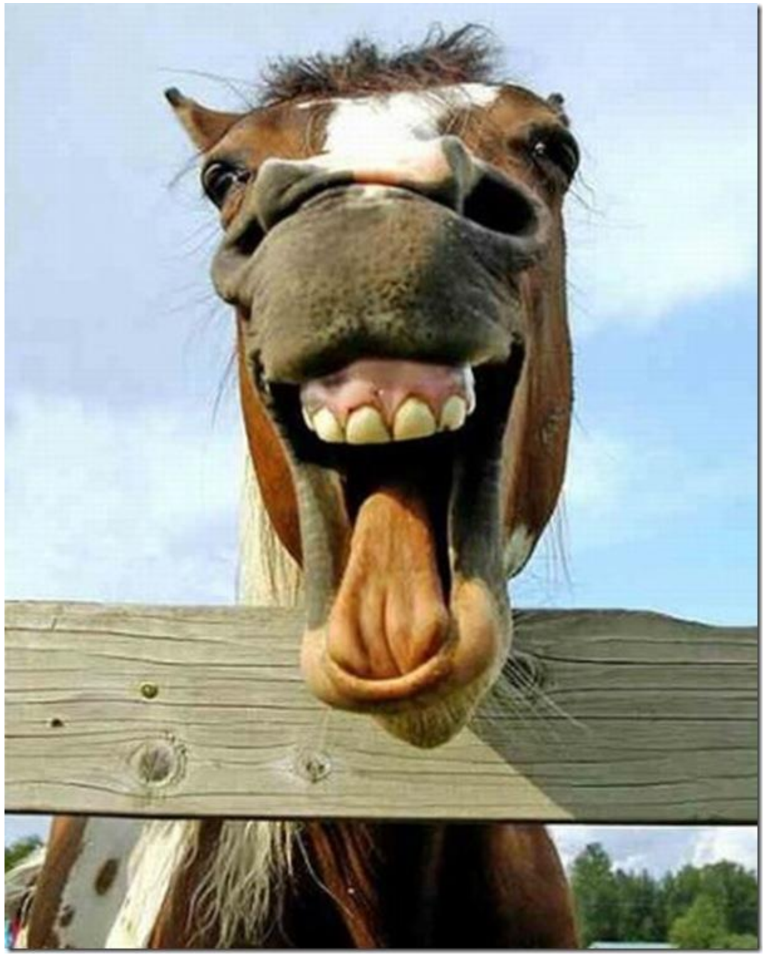
Presumably true OCP fans barely use version control, btw. Only reason to change a source file is for bug fixes, right?

http://blog.8thlight.com/uncle-bob/2013/03/08/AnOpenAndClosedCase.html

Open closed principle

You should be **able to change** the environment surrounding a module **without changing** the module itself.

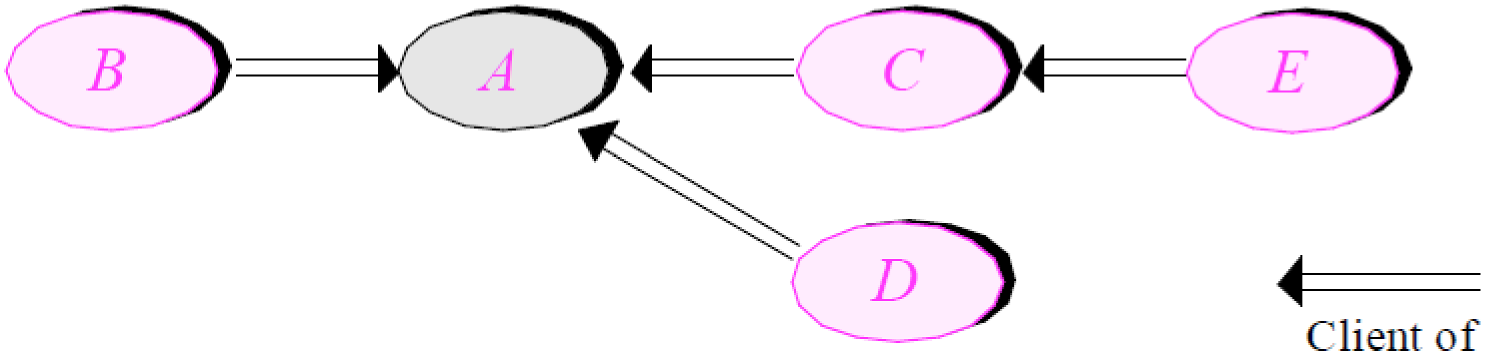
It's an impossible dream



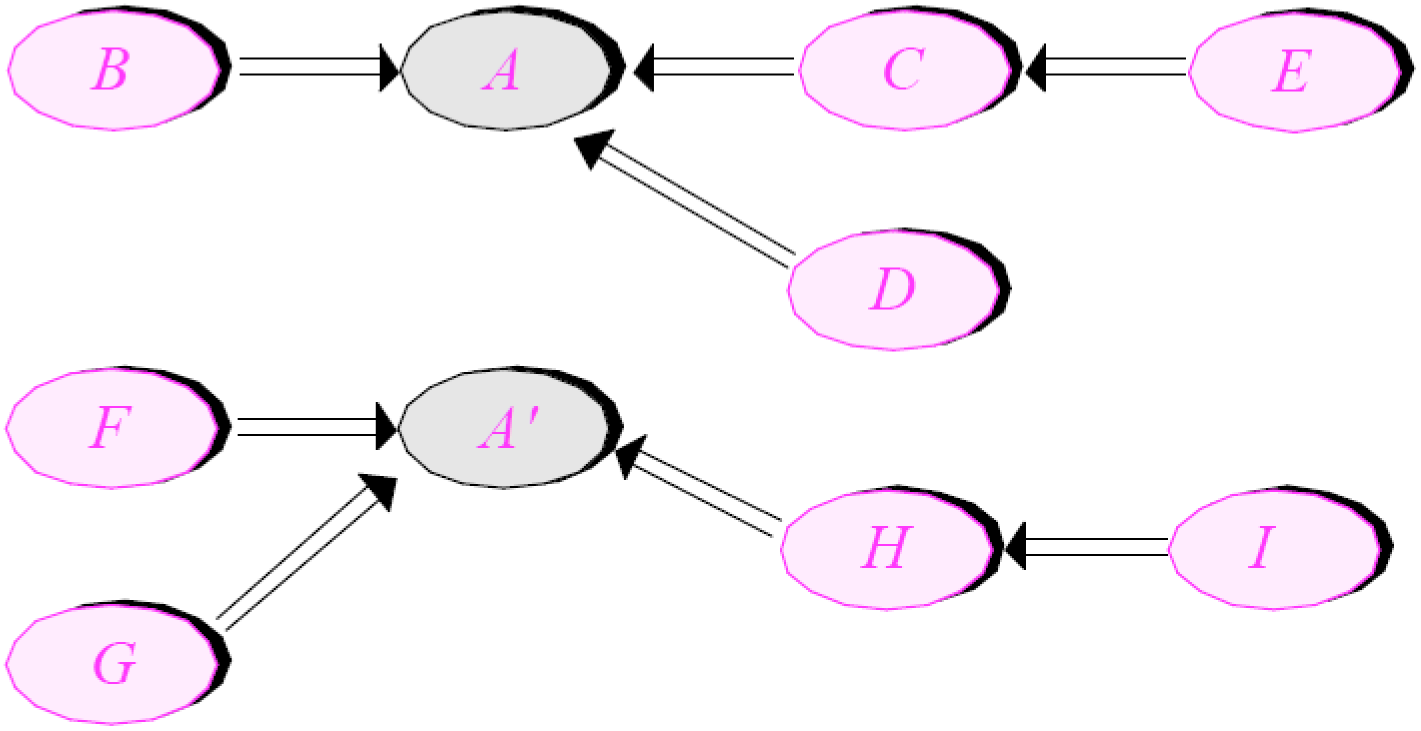
With **traditional** techniques, the two goals are **incompatible**.

- you keep a **module open**,
  and others cannot use it yet.
- you **close it**,
  and any change or extension can trigger a painful chain reaction of changes in many other modules.

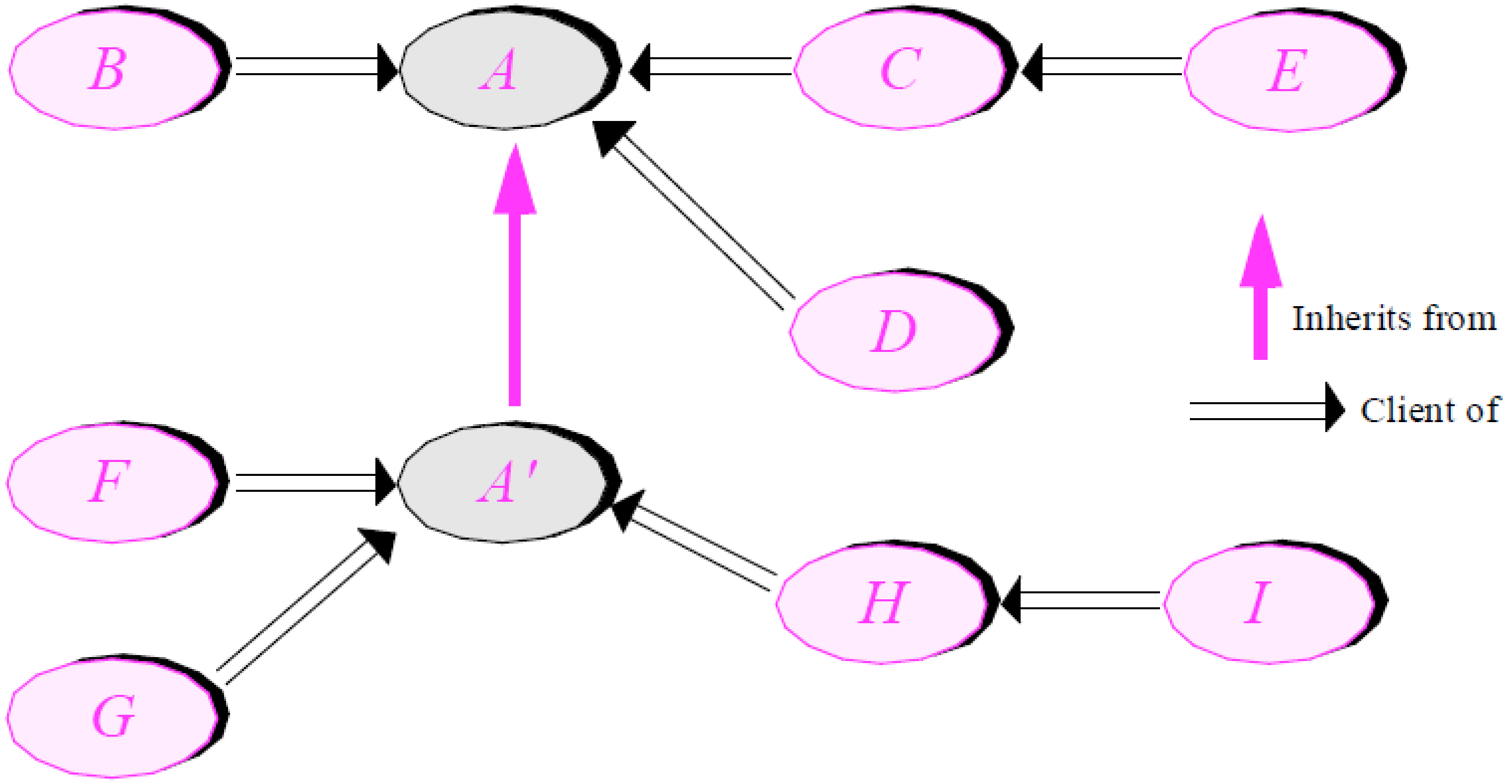
A module and its clients



Old and new clients



Adapting a module to new clients



Organized hacking

http://stroimbanu.ru/kak_legko_kolot_drova/

Organized hacking

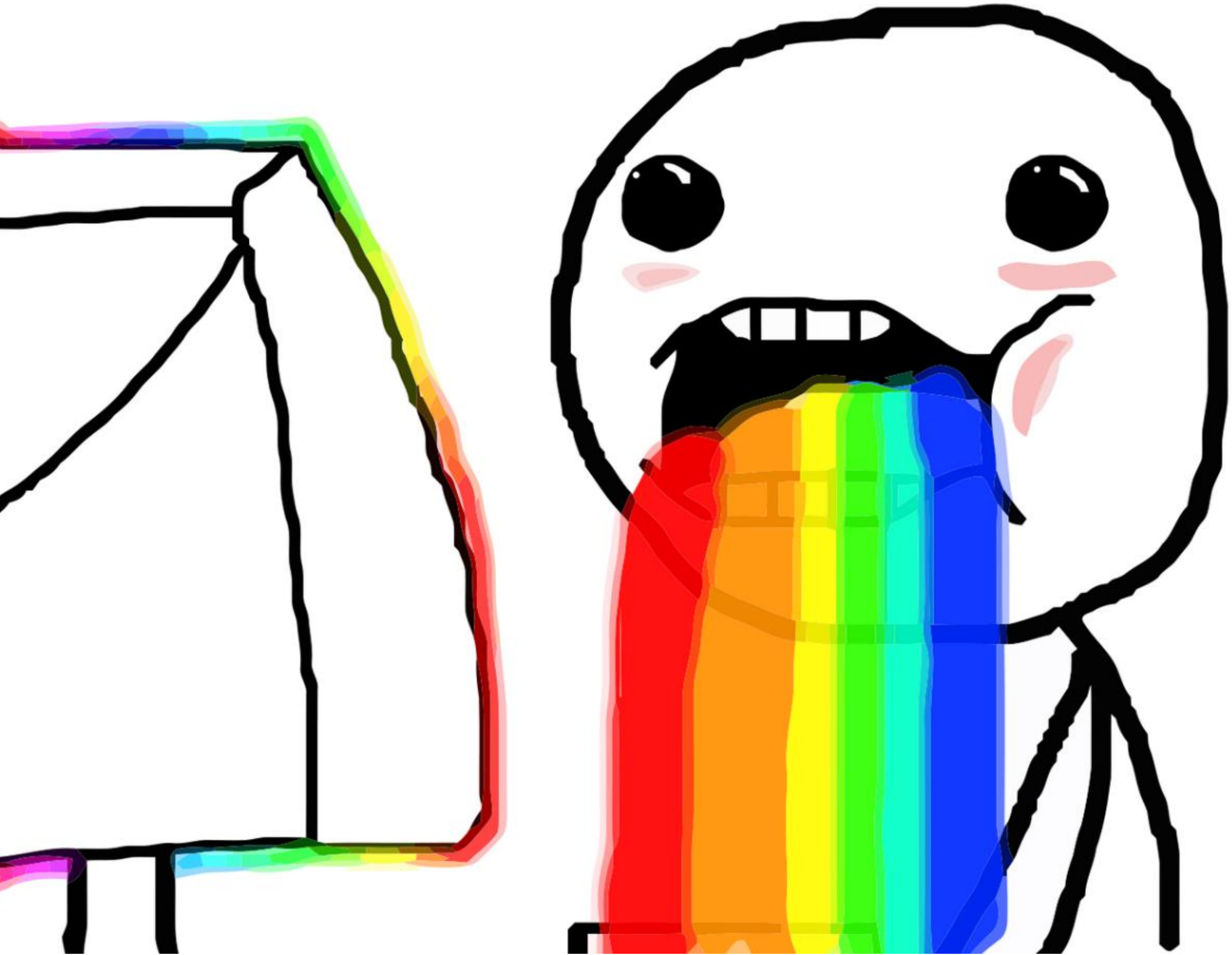
One way to describe the **OCP** and the consequent **OO** techniques is to think of them as a ***organized hacking***.

Organized hacking

- "**Hacking**" is understood here as a slipshod approach to building and modifying code *(not in the more recent sense of breaking into computer networks, which, organized or not, no one should condone)*.
- The hacker may seem bad but often his heart is pure.

Organized hacking

He sees a useful piece of software, which is ***almost*** able to address the needs of the moment, more general than the software’s original purpose.



. . .

WHAT IF

PROGRAMMERS WERE BUILDING HOUSES...

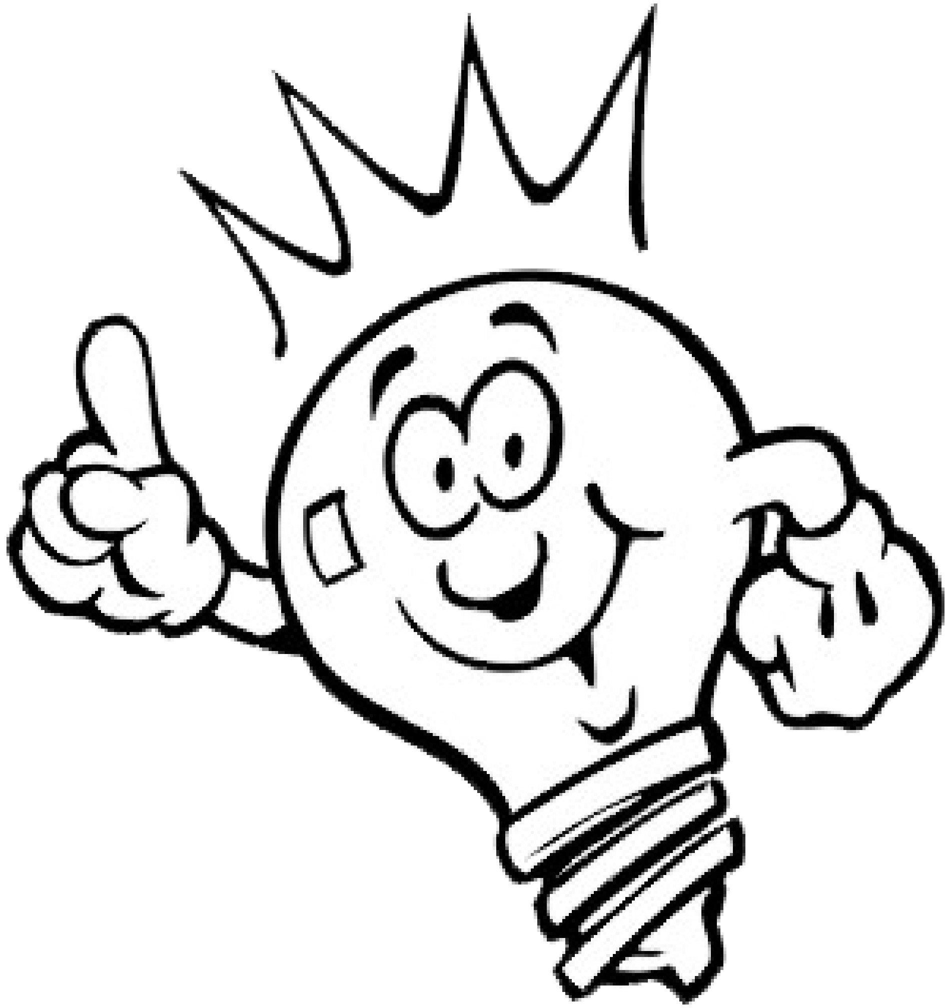
Uh-Ohhh!

Strategic closure

- It should be clear that no significant program can be 100% closed.
- Since closure **cannot be complete**, it must be **strategic**.
- This takes a certain amount of prescience derived from **experience**.

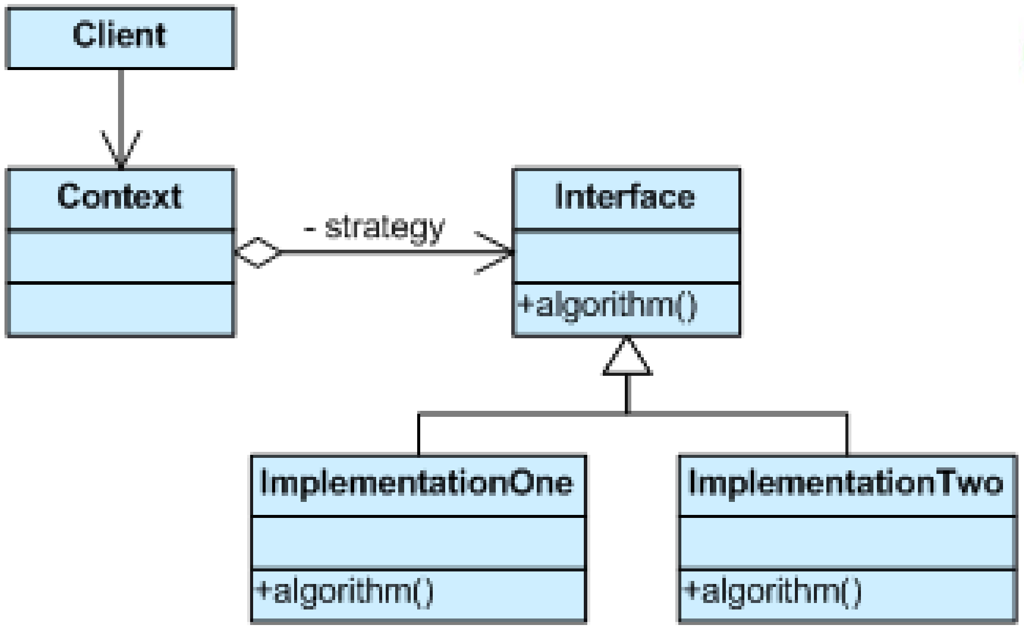
http://www.objectmentor.com/resources/articles/ocp.pdf

Some design patterns



http://www.mainewomensnetwork.com/Default.aspx?pageId=761712

Strategy
design pattern



http://sourcemaking.com/design_patterns/strategy

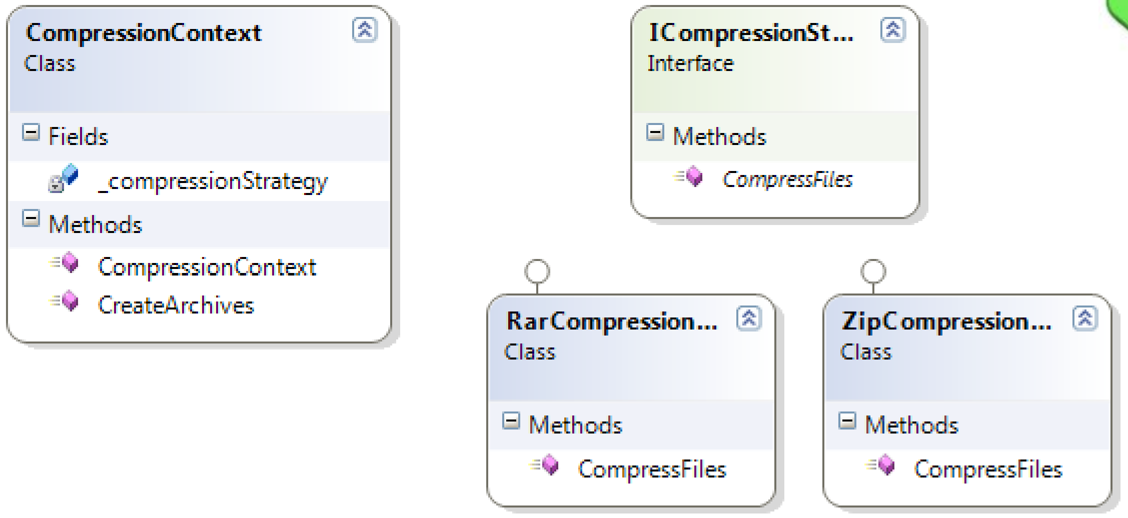
Strategy

design pattern

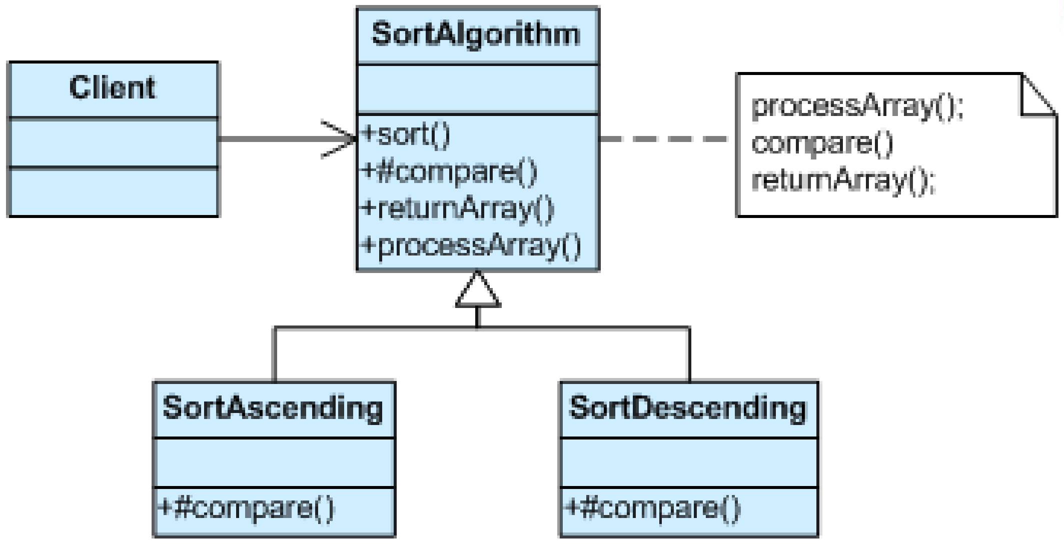
```
public interface ICompressionStrategy
{
   void CompressFiles(IList<IFile> files);
}

public class ZipCompressionStrategy : ICompressionStrategy
{
   public void CompressFiles(IList<IFile> files)
   {
      //...
   }
}

public class RarCompressionStrategy : ICompressionStrategy
{
   public void CompressFiles(IList<IFile> files)
   {
      //...
   }
}

public class CompressionContext
{
   private readonly ICompressionStrategy _compressionStrategy;

   public CompressionContext(ICompressionStrategy compressionStrategy)
   {
      _compressionStrategy = compressionStrategy;
   }

   public void CreateArchives(IList<IFile> files)
   {
      _compressionStrategy.CompressFiles(files);
   }
}

//var context = new CompressionContext(new ZipCompressionStrategy());
//...get file lists
//context.CreateArchives(files);
```

http://java.dzone.com/articles/design-patterns-strategy

Strategy
design pattern



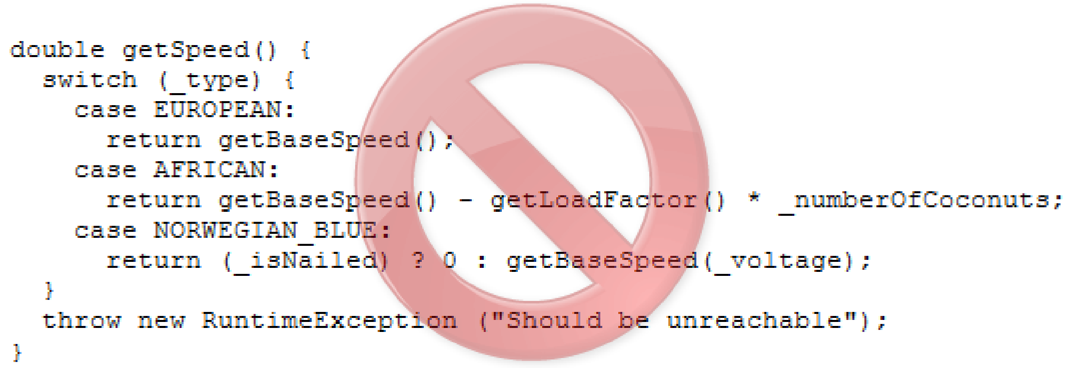
Template method
design pattern



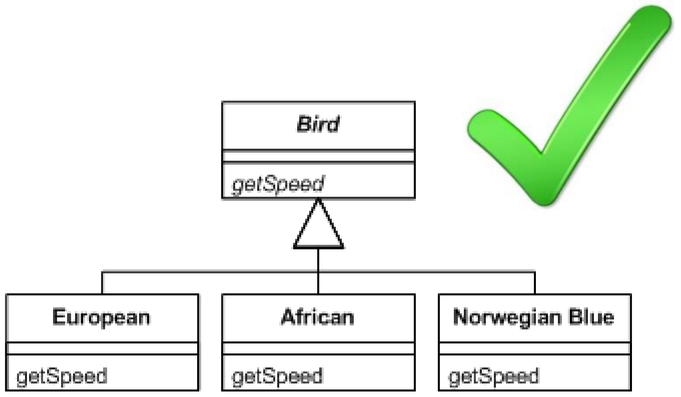
http://sourcemaking.com/design_patterns/template_method

Replace conditional with polymorphism

Move each leg of the conditional to an overriding method in a subclass. Make the original method abstract.



```
double getSpeed() {
  switch (_type) {
    case EUROPEAN:
      return getBaseSpeed();
    case AFRICAN:
      return getBaseSpeed() - getLoadFactor() * _numberOfCoconuts;
    case NORWEGIAN_BLUE:
      return (_isNailed) ? 0 : getBaseSpeed(_voltage);
  }
  throw new RuntimeException ("Should be unreachable");
}
```



http://refactoring.com/catalog/replaceConditionalWithPolymorphism.html

Some examples

The first example

Class isn’t closed

Opportunities to expand

GreaterOrEqeal() „>=”

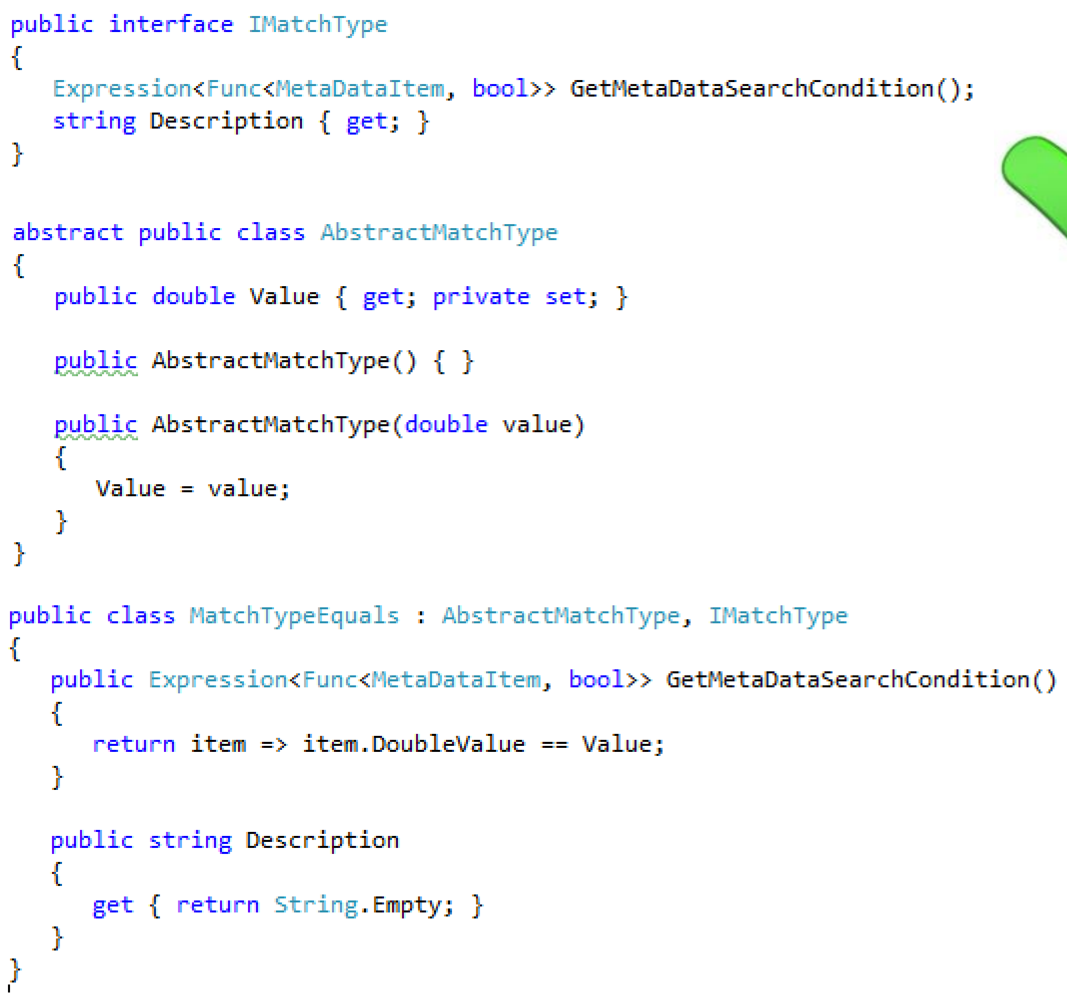
```
public enum MatchType
{
    Equals,
    LowerThan,
    GreaterThan
}
```

```
protected override Expression<Func<MetaDataItem, bool>> GetMetaDataSearchCondition()
{
    switch (Type)
    {
        case MatchType.Equals:
            return item => item.DoubleValue == Value;
        case MatchType.GreaterThan:
            return item => item.DoubleValue > Value;
        case MatchType.LowerThan:
            return item => item.DoubleValue < Value;
    }
    throw new NotSupportedException("Nie wspierany rodzaj porównania: '{0}'.".AsFormat(Type));
}

private String MatchTypeOperationDescription()
{
    switch (Type)
    {
        case MatchType.Equals: return String.Empty;
        case MatchType.GreaterThan: return Language.GreaterThan;
        case MatchType.LowerThan: return Language.LessThan;
    }
    throw new NotSupportedException("Nie wspierany typ operacji: '{0}'.".AsFormat(Type));
}
```

Is there a better solution?

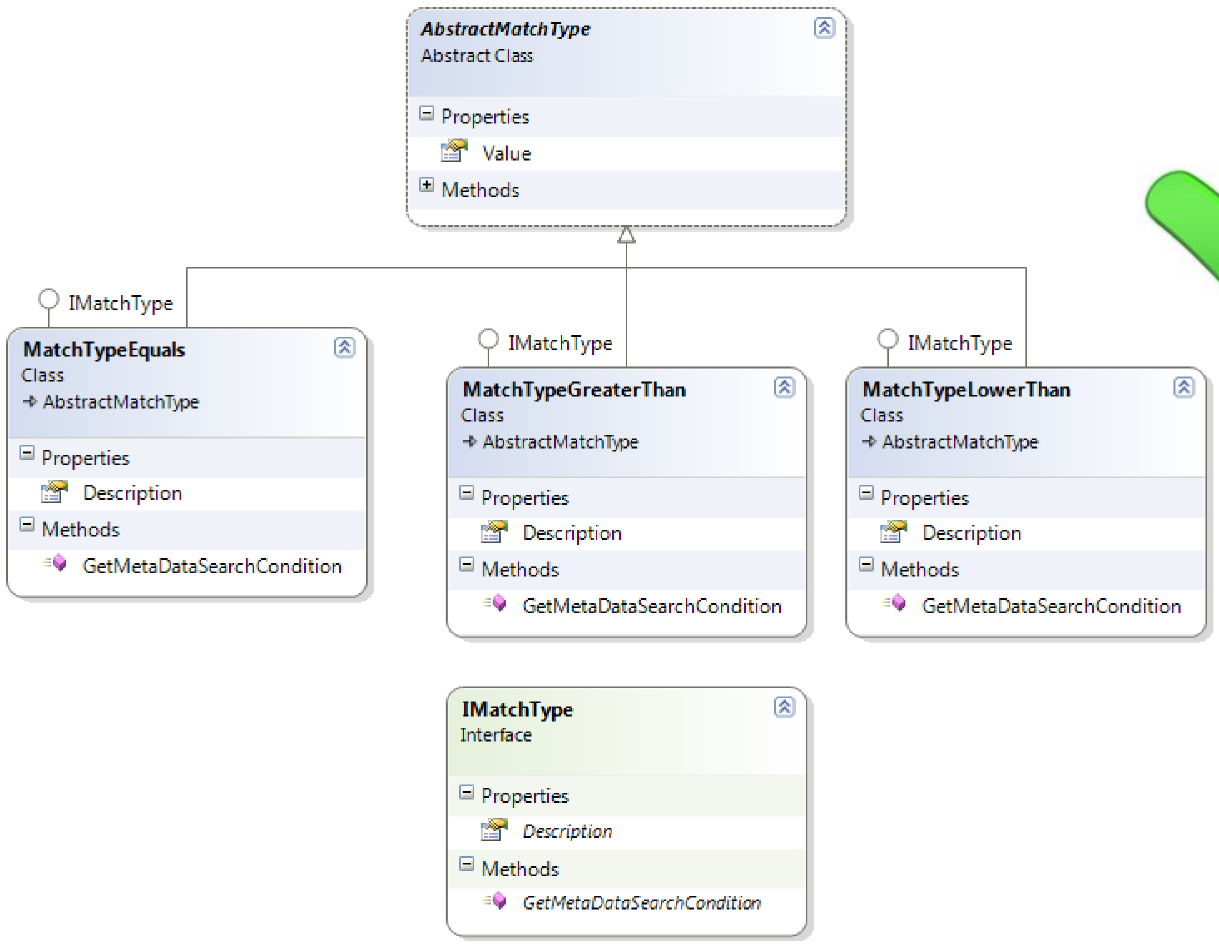
http://siriusbuzz.com/what-happens-with-that-sirius-lifetime-subscription.php/big-question-mark

Strategy Design Pattern



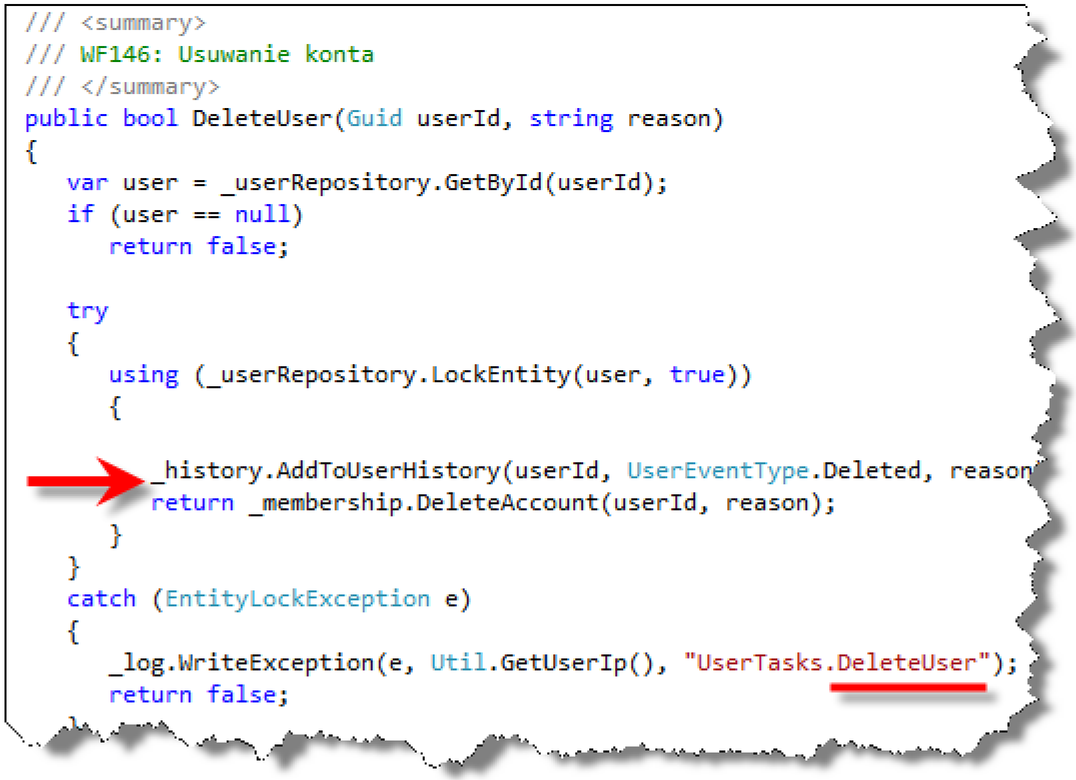
```
public interface IMatchType
{
    Expression<Func<MetaDataItem, bool>> GetMetaDataSearchCondition();
    string Description { get; }
}

abstract public class AbstractMatchType
{
    public double Value { get; private set; }

    public AbstractMatchType() { }

    public AbstractMatchType(double value)
    {
        Value = value;
    }
}

public class MatchTypeEquals : AbstractMatchType, IMatchType
{
    public Expression<Func<MetaDataItem, bool>> GetMetaDataSearchCondition()
    {
        return item => item.DoubleValue == Value;
    }

    public string Description
    {
        get { return String.Empty; }
    }
}
```

Strategy Design Pattern



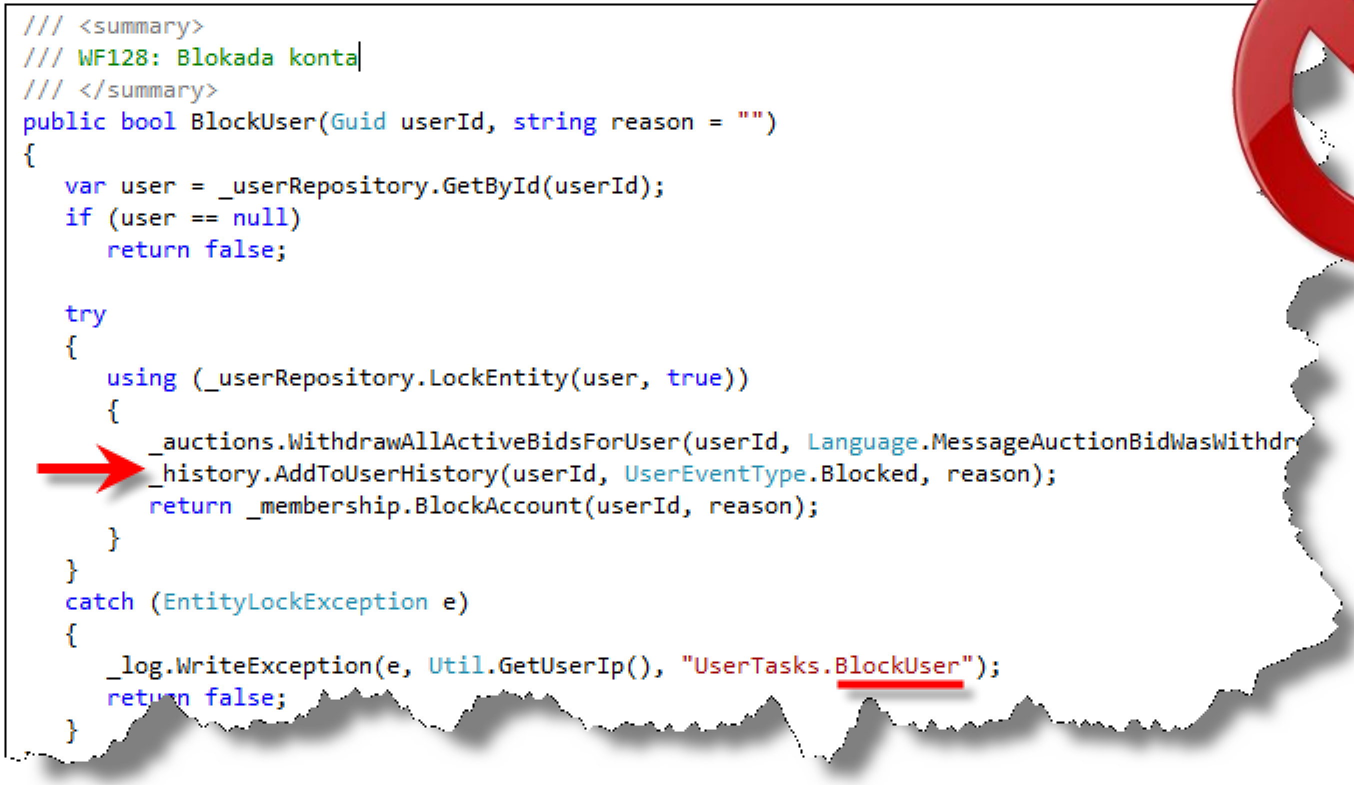
The second example

Class isn’t closed



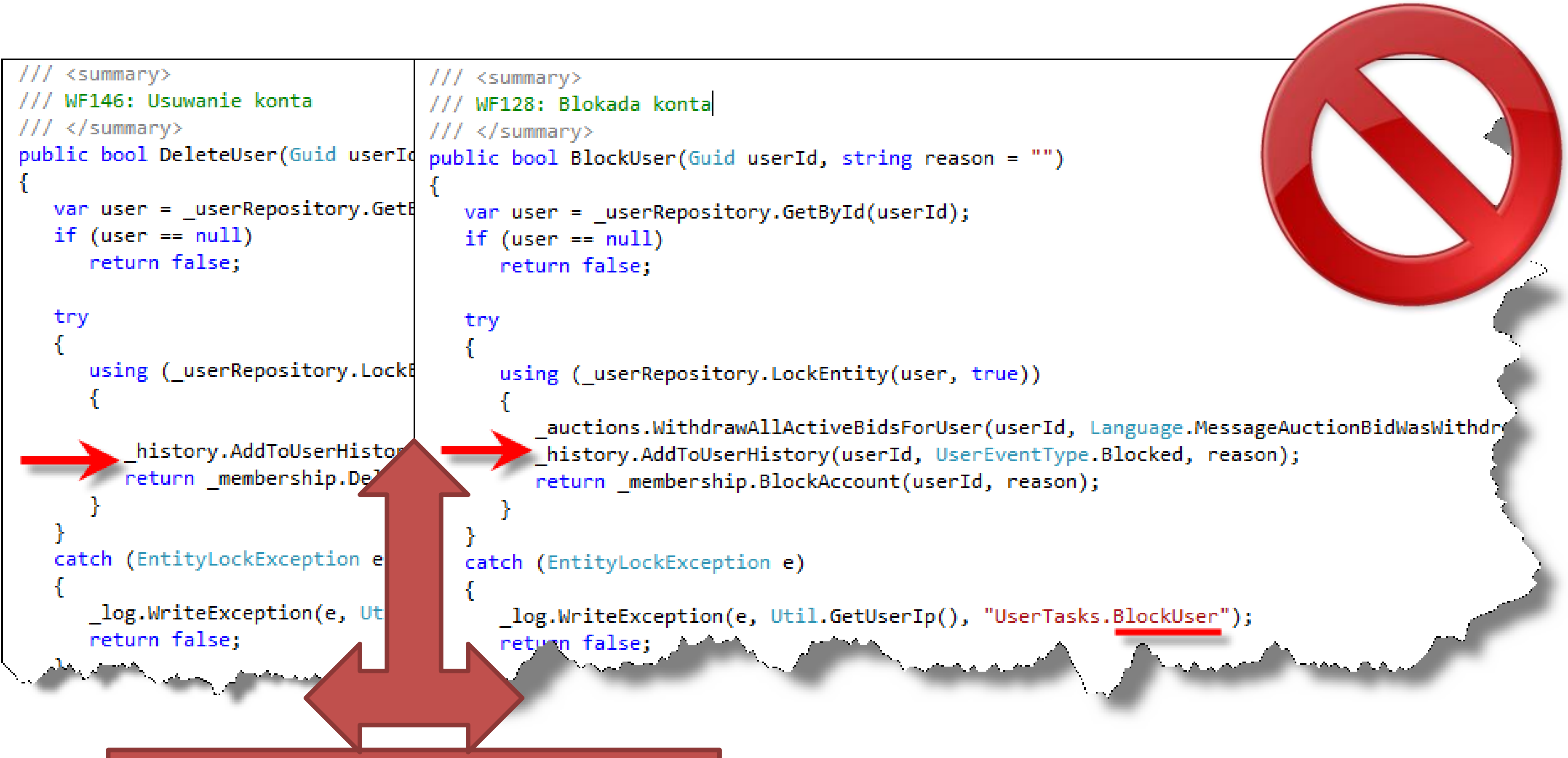
```
/// <summary>
/// WF146: Usuwanie konta
/// </summary>
public bool DeleteUser(Guid userId, string reason)
{
   var user = _userRepository.GetById(userId);
   if (user == null)
      return false;

   try
   {
      using (_userRepository.LockEntity(user, true))
      {

         _history.AddToUserHistory(userId, UserEventType.Deleted, reason
         return _membership.DeleteAccount(userId, reason);
      }
   }
   catch (EntityLockException e)
   {
      _log.WriteException(e, Util.GetUserIp(), "UserTasks.DeleteUser");
      return false;
```

Class isn’t closed



```
/// <summary>
/// WF128: Blokada konta
/// </summary>
public bool BlockUser(Guid userId, string reason = "")
{
   var user = _userRepository.GetById(userId);
   if (user == null)
      return false;

   try
   {
      using (_userRepository.LockEntity(user, true))
      {
         _auctions.WithdrawAllActiveBidsForUser(userId, Language.MessageAuctionBidWasWithdr
         _history.AddToUserHistory(userId, UserEventType.Blocked, reason);
         return _membership.BlockAccount(userId, reason);
      }
   }
   catch (EntityLockException e)
   {
      _log.WriteException(e, Util.GetUserIp(), "UserTasks.BlockUser");
      return false;
   }
```

Class isn’t closed



```
/// <summary>
/// WF146: Usuwanie konta
/// </summary>
public bool DeleteUser(Guid userId
{
   var user = _userRepository.GetE
   if (user == null)
      return false;

   try
   {
      using (_userRepository.LockE
      {

         _history.AddToUserHistor
         return _membership.De
      }
   }
   catch (EntityLockException e
   {
      _log.WriteException(e, Ut
      return false;
```

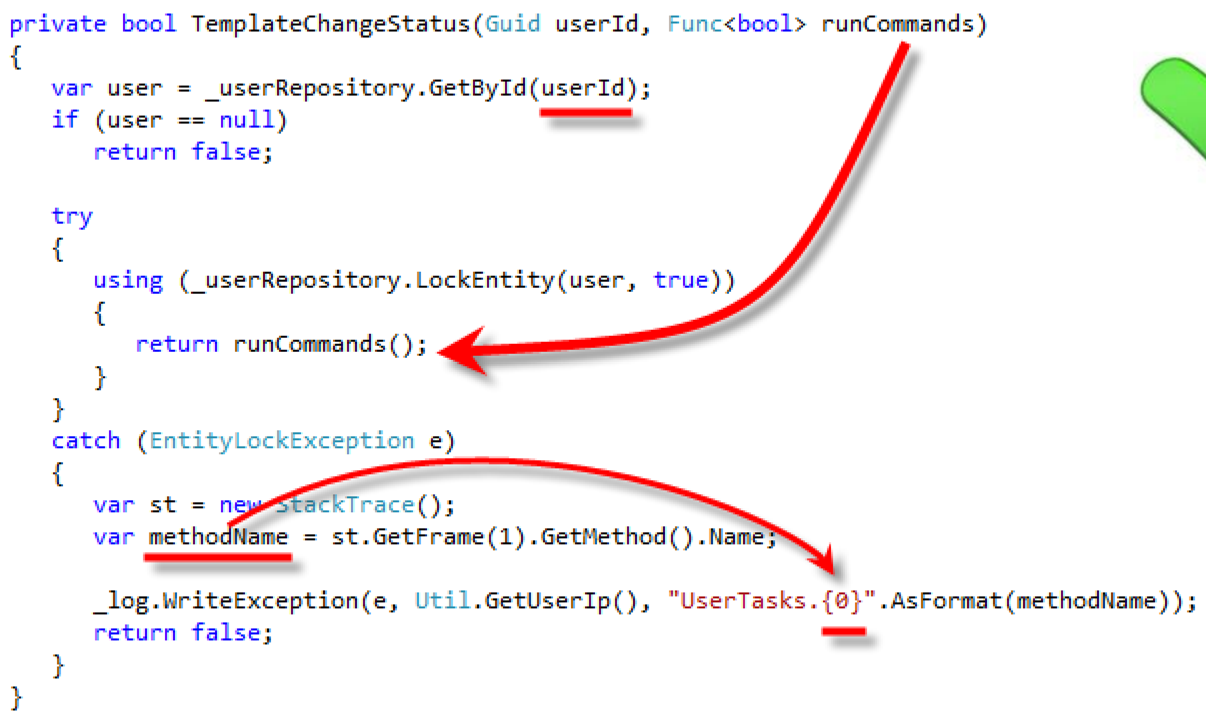
```
/// <summary>
/// WF128: Blokada konta
/// </summary>
public bool BlockUser(Guid userId, string reason = "")
{
   var user = _userRepository.GetById(userId);
   if (user == null)
      return false;

   try
   {
      using (_userRepository.LockEntity(user, true))
      {
         _auctions.WithdrawAllActiveBidsForUser(userId, Language.MessageAuctionBidWasWithdr
         _history.AddToUserHistory(userId, UserEventType.Blocked, reason);
         return _membership.BlockAccount(userId, reason);
      }
   }
   catch (EntityLockException e)
   {
      _log.WriteException(e, Util.GetUserIp(), "UserTasks.BlockUser");
      return false;
```

Duplicate Code

Is there a better solution?

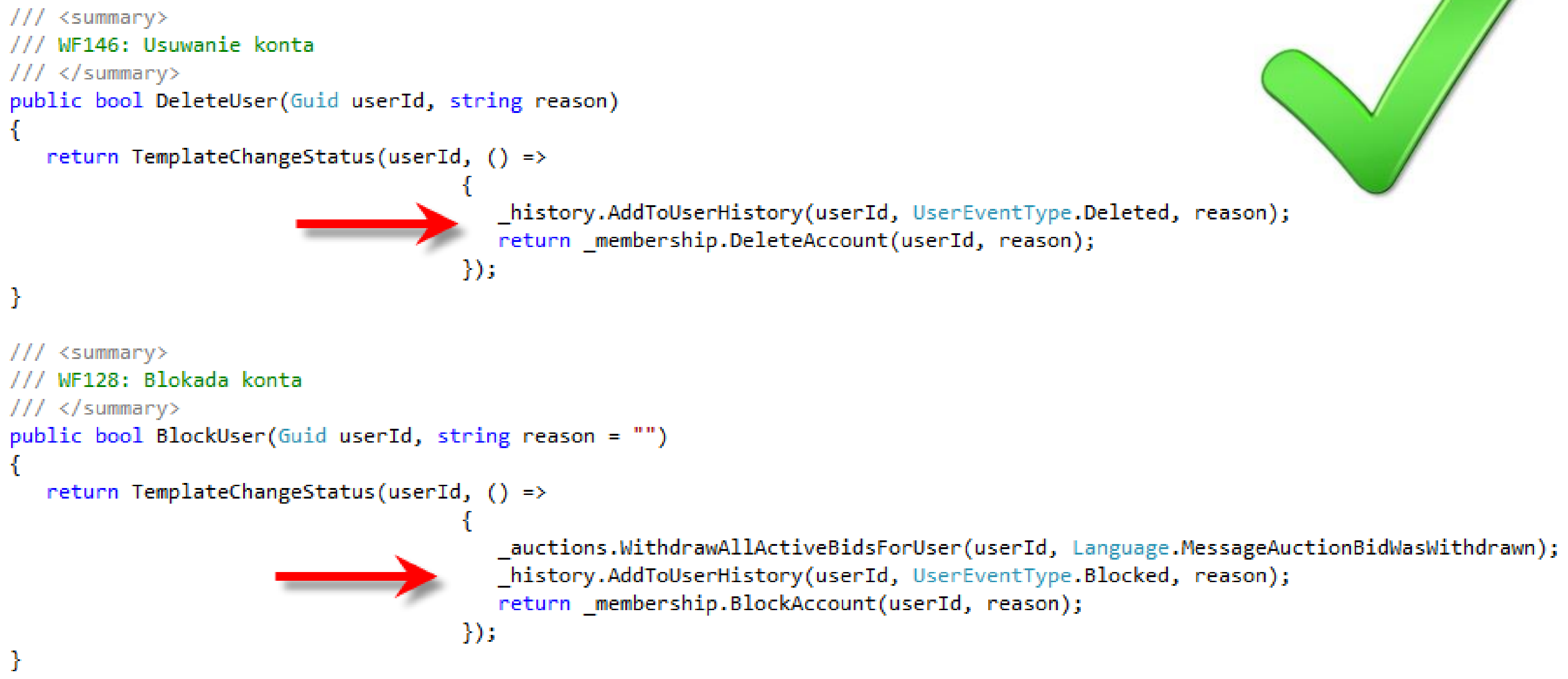
http://siriusbuzz.com/what-happens-with-that-sirius-lifetime-subscription.php/big-question-mark

Template Method Design Pattern



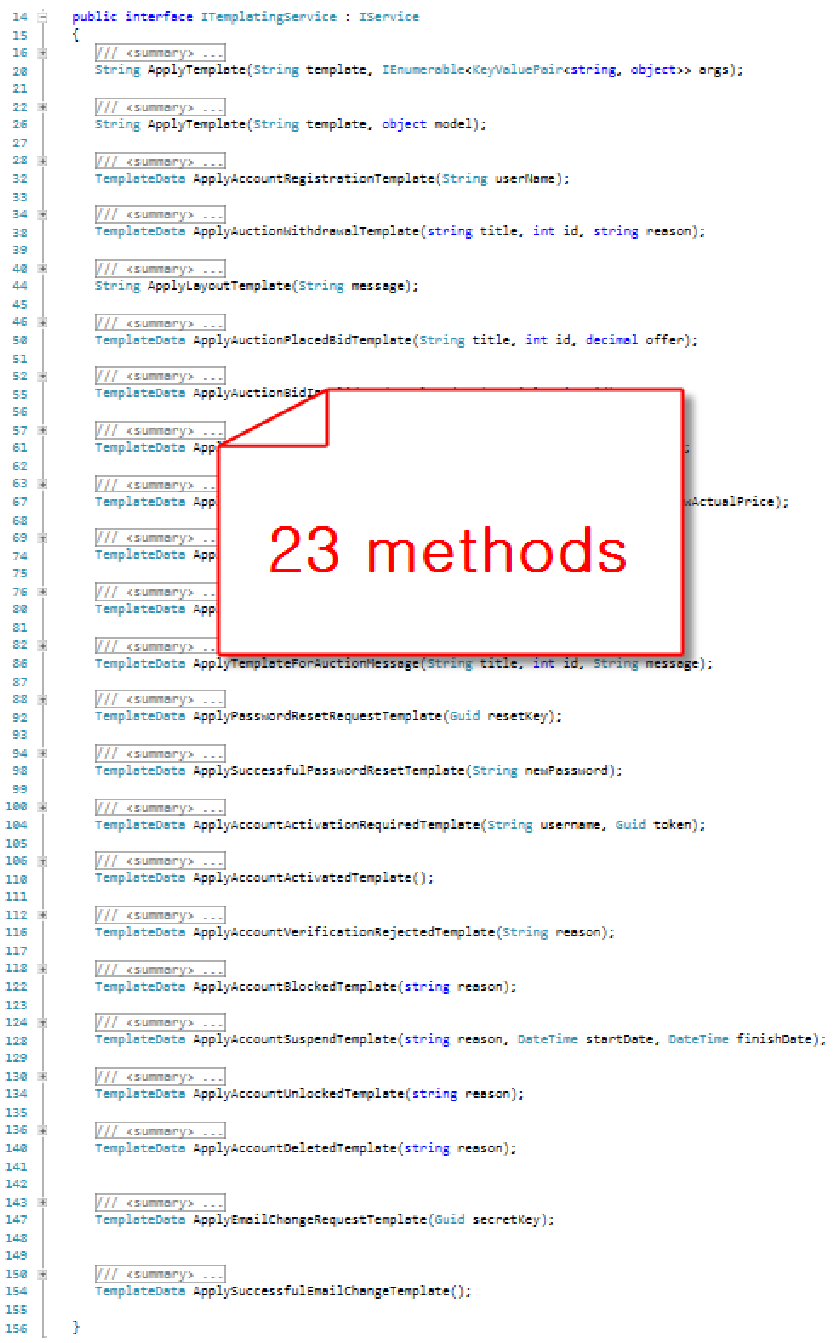
```
private bool TemplateChangeStatus(Guid userId, Func<bool> runCommands)
{
    var user = _userRepository.GetById(userId);
    if (user == null)
        return false;

    try
    {
        using (_userRepository.LockEntity(user, true))
        {
            return runCommands();
        }
    }
    catch (EntityLockException e)
    {
        var st = new StackTrace();
        var methodName = st.GetFrame(1).GetMethod().Name;

        _log.WriteException(e, Util.GetUserIp(), "UserTasks.{0}".AsFormat(methodName));
        return false;
    }
}
```

Template Method Design Pattern



```
/// <summary>
/// WF146: Usuwanie konta
/// </summary>
public bool DeleteUser(Guid userId, string reason)
{
    return TemplateChangeStatus(userId, () =>
                                    {
                                        _history.AddToUserHistory(userId, UserEventType.Deleted, reason);
                                        return _membership.DeleteAccount(userId, reason);
                                    });
}

/// <summary>
/// WF128: Blokada konta
/// </summary>
public bool BlockUser(Guid userId, string reason = "")
{
    return TemplateChangeStatus(userId, () =>
                                    {
                                        _auctions.WithdrawAllActiveBidsForUser(userId, Language.MessageAuctionBidWasWithdrawn);
                                        _history.AddToUserHistory(userId, UserEventType.Blocked, reason);
                                        return _membership.BlockAccount(userId, reason);
                                    });
}
```

The third example



```
public interface ITemplatingService : IService
{
    /// <summary> ...
    String ApplyTemplate(String template, IEnumerable<KeyValuePair<string, object>> args);

    /// <summary> ...
    String ApplyTemplate(String template, object model);

    /// <summary> ...
    TemplateData ApplyAccountRegistrationTemplate(String userName);

    /// <summary> ...
    TemplateData ApplyAuctionWithdrawalTemplate(string title, int id, string reason);

    /// <summary> ...
    String ApplyLayoutTemplate(String message);

    /// <summary> ...
    TemplateData ApplyAuctionPlacedBidTemplate(String title, int id, decimal offer);

    /// <summary> ...
    TemplateData ApplyAuctionBidI

    /// <summary> ...
    TemplateData App                                                      ;

    /// <summary> ...
    TemplateData App                                                      wActualPrice);

    /// <summary> ...
    TemplateData App

    /// <summary> ...
    TemplateData App

    /// <summary> ...
    TemplateData ApplyTemplateForAuctionMessage(String title, int id, String message);

    /// <summary> ...
    TemplateData ApplyPasswordResetRequestTemplate(Guid resetKey);

    /// <summary> ...
    TemplateData ApplySuccessfulPasswordResetTemplate(String newPassword);

    /// <summary> ...
    TemplateData ApplyAccountActivationRequiredTemplate(String username, Guid token);

    /// <summary> ...
    TemplateData ApplyAccountActivatedTemplate();

    /// <summary> ...
    TemplateData ApplyAccountVerificationRejectedTemplate(String reason);

    /// <summary> ...
    TemplateData ApplyAccountBlockedTemplate(string reason);

    /// <summary> ...
    TemplateData ApplyAccountSuspendTemplate(string reason, DateTime startDate, DateTime finishDate);

    /// <summary> ...
    TemplateData ApplyAccountUnlockedTemplate(string reason);

    /// <summary> ...
    TemplateData ApplyAccountDeletedTemplate(string reason);


    /// <summary> ...
    TemplateData ApplyEmailChangeRequestTemplate(Guid secretKey);


    /// <summary> ...
    TemplateData ApplySuccessfulEmailChangeTemplate();

}
```

23 methods

```csharp
public TemplateData ApplyAccountRegistrationTemplate(string userName)
{
    return BuildTemplatedMessageBody(NotificationType.RegisterAccount, new { Username = userName });
}

public TemplateData ApplyAuctionWithdrawalTemplate(string title, int id, string reason)
{
    return BuildTemplatedMessageBody(NotificationType.WithdrawAuction, new { Title = title, Id = id, Reason = reason });
}

public TemplateData ApplyAuctionPlacedBidTemplate(string title, int id, decimal offer)
{
    return BuildTemplatedMessageBody(NotificationType.PlacedBid, new { Title = title, Id = id, Offer = offer });
}
```

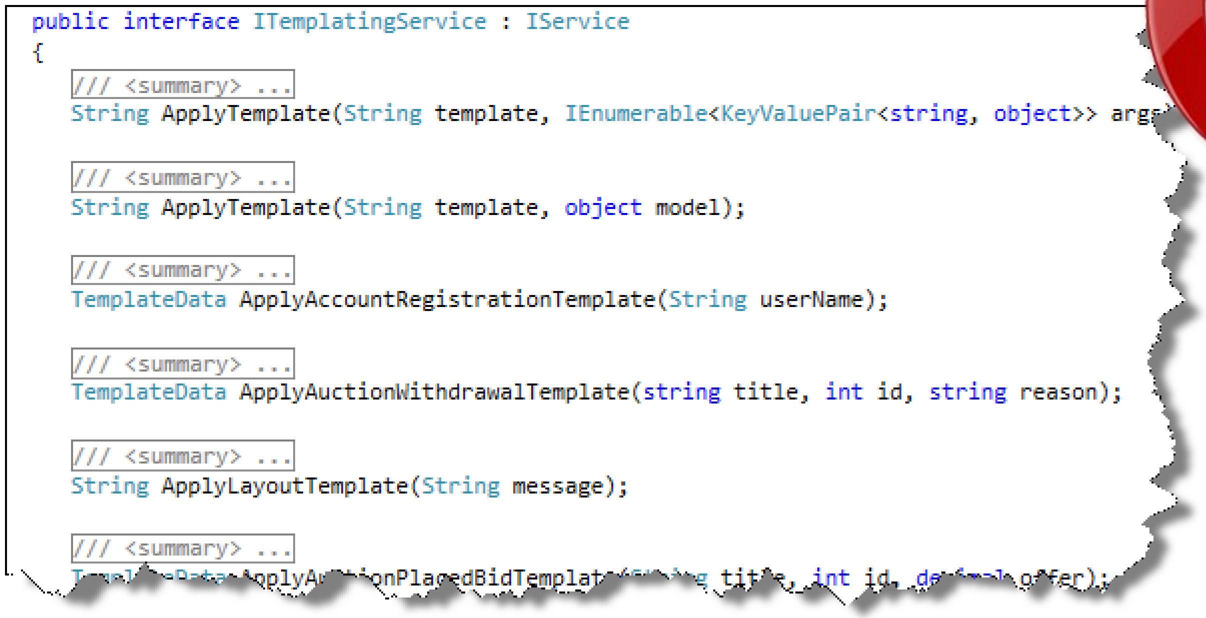
How to add new functionality in this solution?

http://siriusbuzz.com/what-happens-with-that-sirius-lifetime-subscription.php/big-question-mark

First step: change the enum

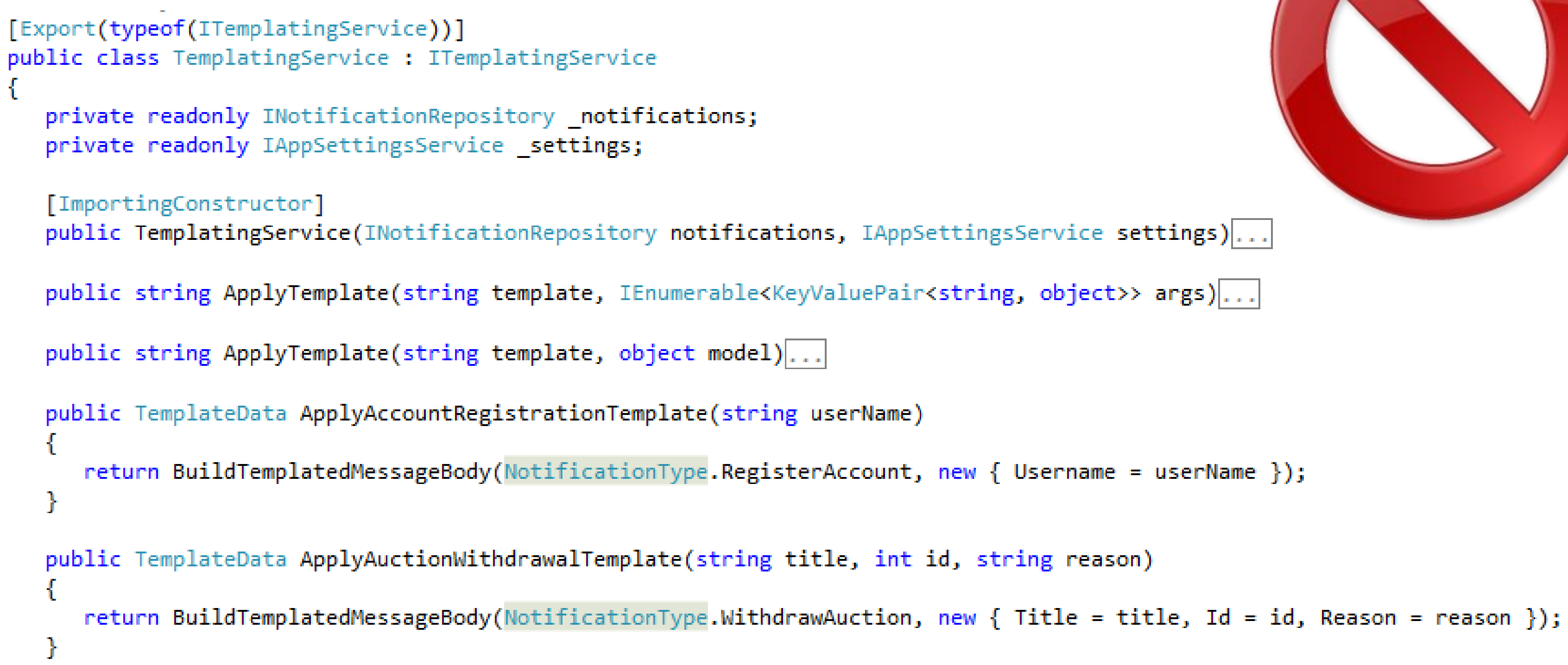
```
public enum NotificationType
{
    /// <summary>
    /// Template specjalny, trzymający layout całego maila.
    /// Tempalte ten jest używany do każdego maila i do jego wnętrza wstawiane są pozostał[illegible]mpal
    /// dla innych wiadomości.
    /// </summary>
    Layout = 1,
    RegisterAccount = 2,
    WithdrawAuction = 3,
    PlacedBid = 4,
    BidInvalidated = 5,
    AuctionWon = 6,
    BidSurpassed = 7,
    BidIsNowBest = 8,
    AuctionEnd = 9,
    AuctionMessage = 10,
    PasswordResetRequest = 11,
    SuccessfulPasswordReset = 12,
    AccountActivationRequired = 13,
    AccountActivated = 14,
    AccountVerificationRejected = 15,
    BlockedAccount = 16,
    SuspendAccount = 17,
    UnlockedAccount = 18,
    DeletedAccount = 19,
    EmailChangeRequest = 20,
    SuccessfulEmailChange = 21
}
```

Second step: change the interface



```
public interface ITemplatingService : IService
{
    /// <summary> ...
    String ApplyTemplate(String template, IEnumerable<KeyValuePair<string, object>> arg

    /// <summary> ...
    String ApplyTemplate(String template, object model);

    /// <summary> ...
    TemplateData ApplyAccountRegistrationTemplate(String userName);

    /// <summary> ...
    TemplateData ApplyAuctionWithdrawalTemplate(string title, int id, string reason);

    /// <summary> ...
    String ApplyLayoutTemplate(String message);

    /// <summary> ...
    TemplateData ApplyAuctionPlacedBidTemplate(String title, int id, decimal offer);
```

Third step: change the class

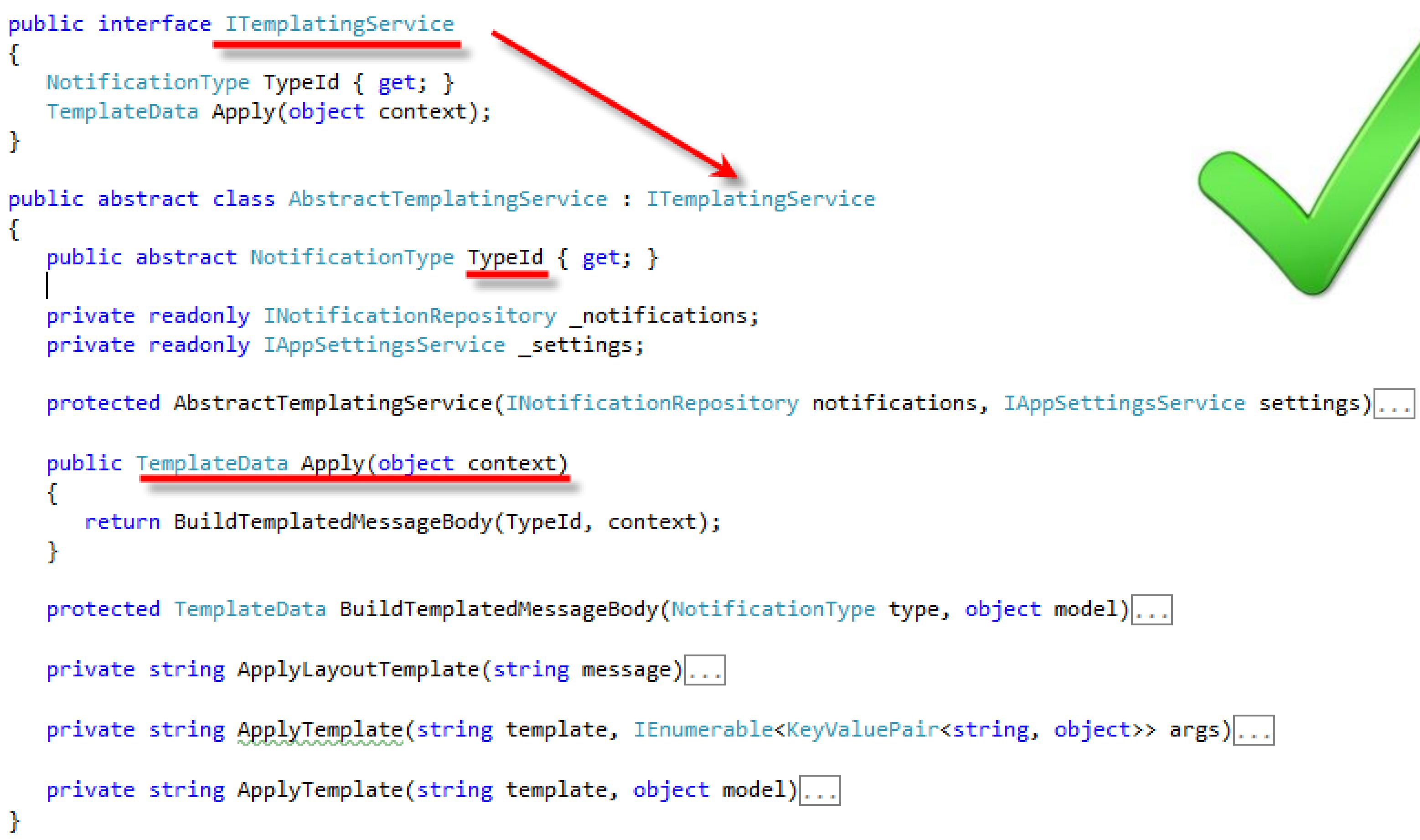


```
[Export(typeof(ITemplatingService))]
public class TemplatingService : ITemplatingService
{
    private readonly INotificationRepository _notifications;
    private readonly IAppSettingsService _settings;

    [ImportingConstructor]
    public TemplatingService(INotificationRepository notifications, IAppSettingsService settings)...

    public string ApplyTemplate(string template, IEnumerable<KeyValuePair<string, object>> args)...

    public string ApplyTemplate(string template, object model)...

    public TemplateData ApplyAccountRegistrationTemplate(string userName)
    {
        return BuildTemplatedMessageBody(NotificationType.RegisterAccount, new { Username = userName });
    }

    public TemplateData ApplyAuctionWithdrawalTemplate(string title, int id, string reason)
    {
        return BuildTemplatedMessageBody(NotificationType.WithdrawAuction, new { Title = title, Id = id, Reason = reason });
    }
```

Is there a better solution?

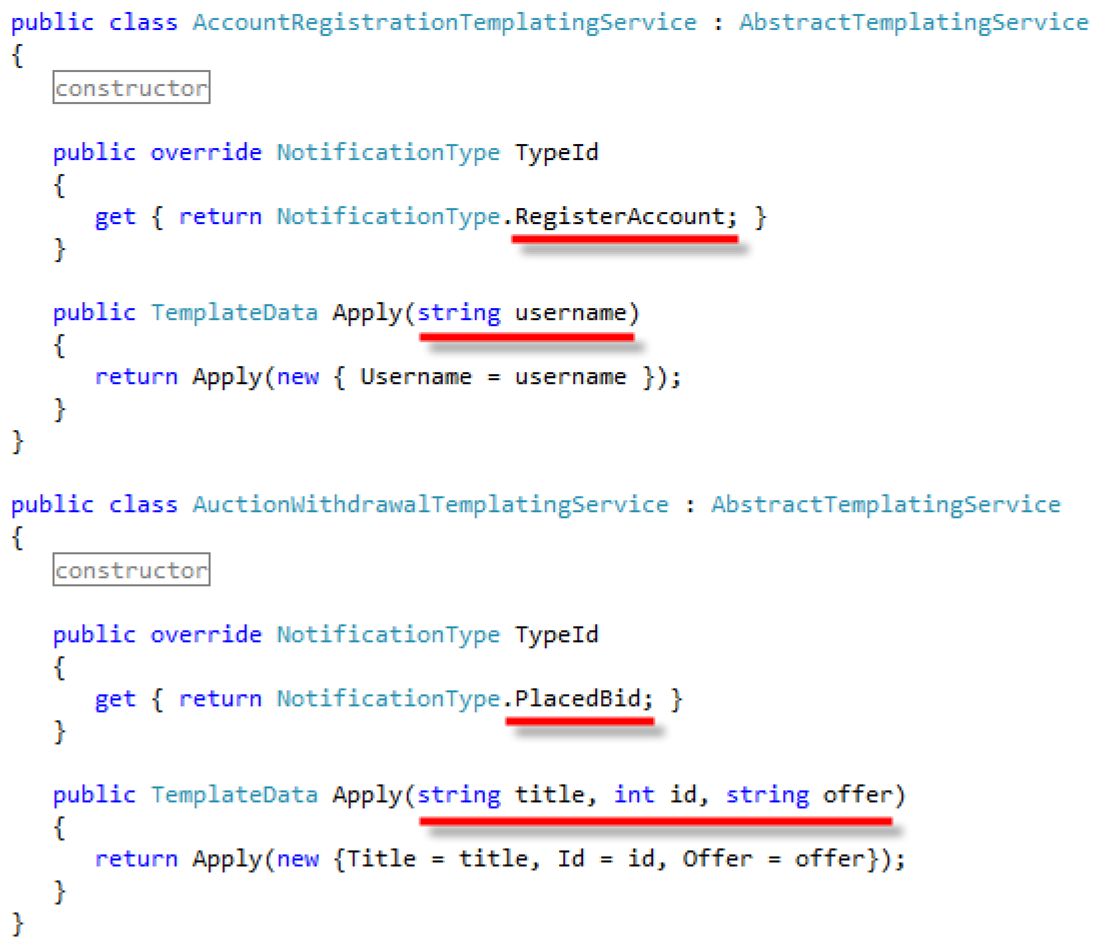
http://siriusbuzz.com/what-happens-with-that-sirius-lifetime-subscription.php/big-question-mark

Template method design pattern



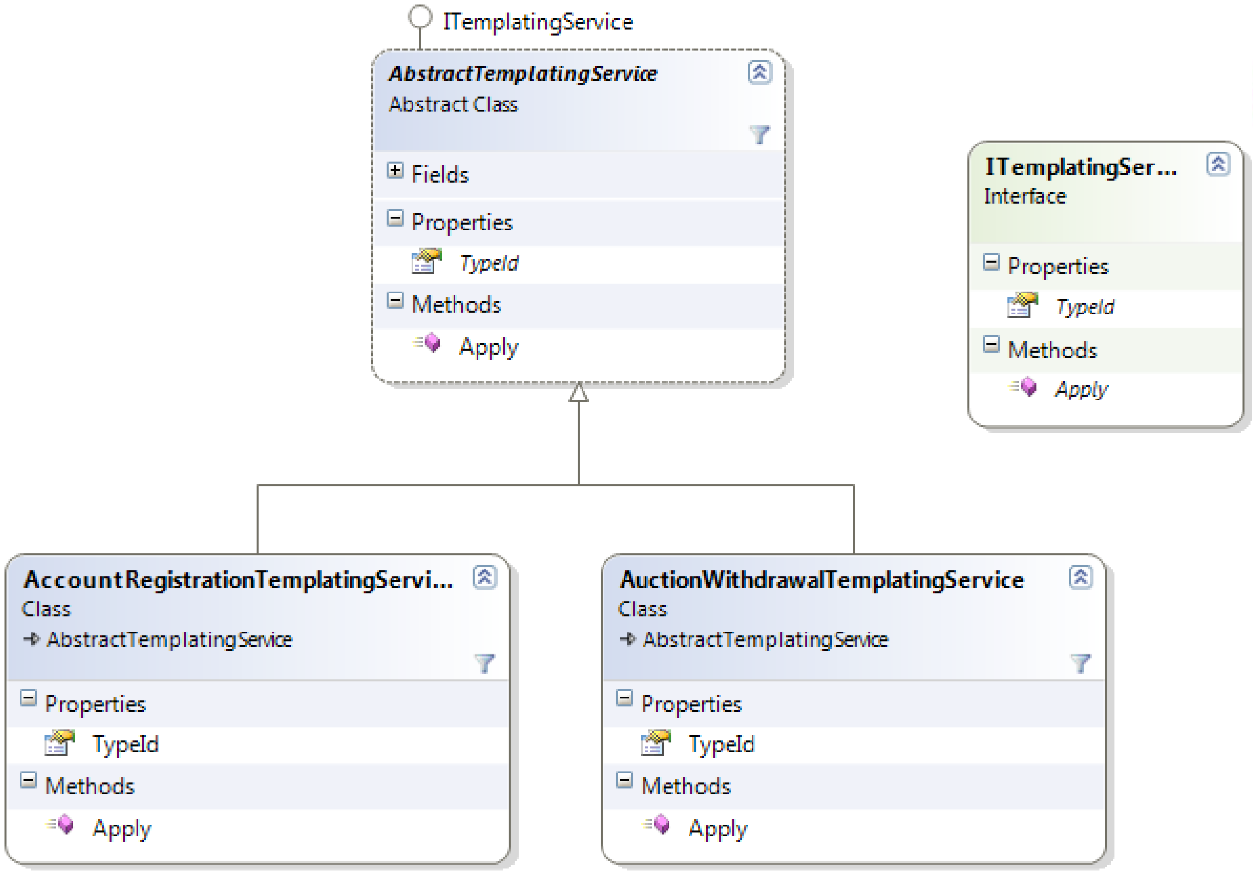
```
public interface ITemplatingService
{
    NotificationType TypeId { get; }
    TemplateData Apply(object context);
}

public abstract class AbstractTemplatingService : ITemplatingService
{
    public abstract NotificationType TypeId { get; }

    private readonly INotificationRepository _notifications;
    private readonly IAppSettingsService _settings;

    protected AbstractTemplatingService(INotificationRepository notifications, IAppSettingsService settings)...

    public TemplateData Apply(object context)
    {
        return BuildTemplatedMessageBody(TypeId, context);
    }

    protected TemplateData BuildTemplatedMessageBody(NotificationType type, object model)...

    private string ApplyLayoutTemplate(string message)...

    private string ApplyTemplate(string template, IEnumerable<KeyValuePair<string, object>> args)...

    private string ApplyTemplate(string template, object model)...
}
```

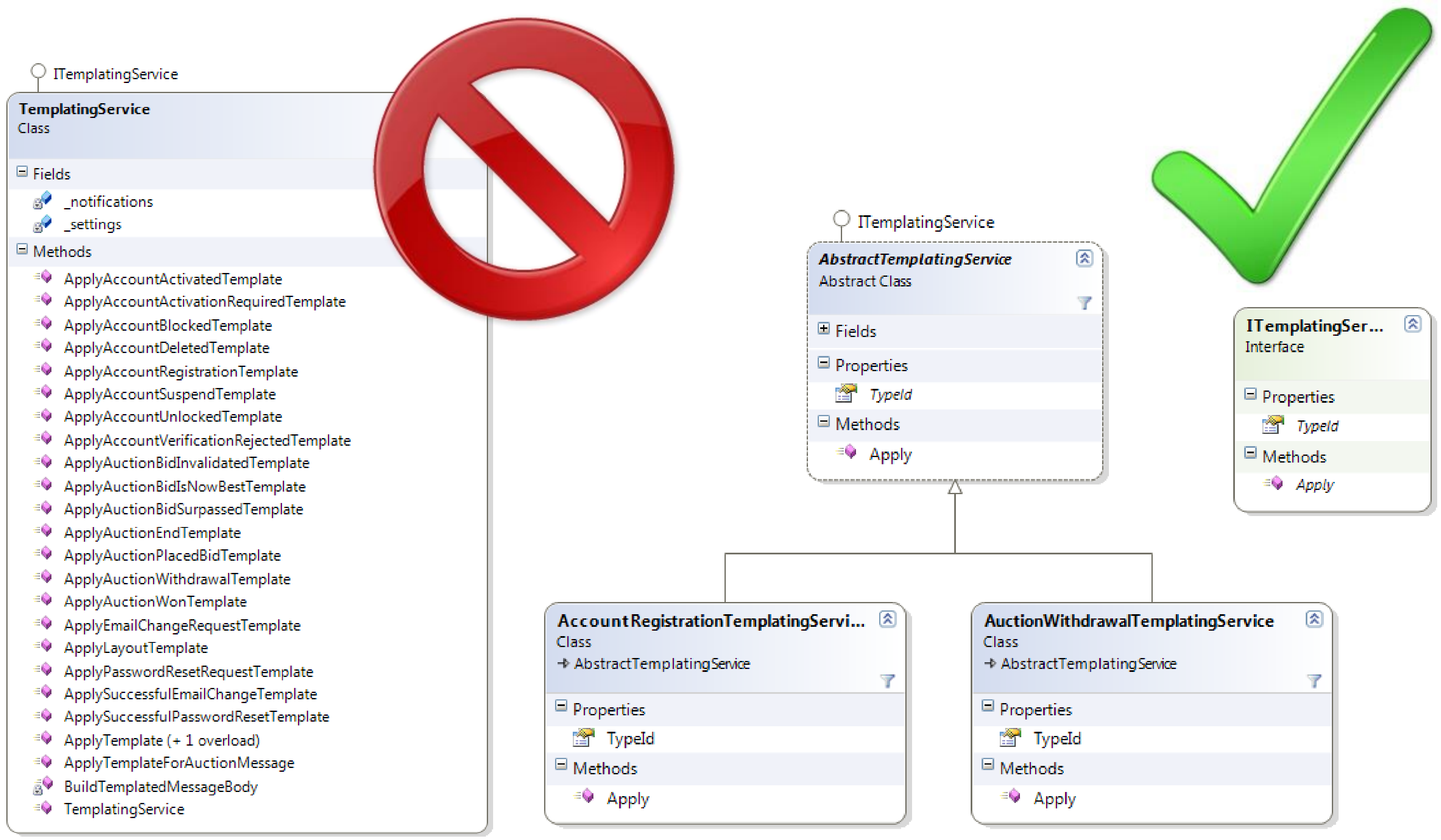
Template method design pattern



```
public class AccountRegistrationTemplatingService : AbstractTemplatingService
{
    constructor

    public override NotificationType TypeId
    {
        get { return NotificationType.RegisterAccount; }
    }

    public TemplateData Apply(string username)
    {
        return Apply(new { Username = username });
    }
}

public class AuctionWithdrawalTemplatingService : AbstractTemplatingService
{
    constructor

    public override NotificationType TypeId
    {
        get { return NotificationType.PlacedBid; }
    }

    public TemplateData Apply(string title, int id, string offer)
    {
        return Apply(new {Title = title, Id = id, Offer = offer});
    }
}
```

Template method design pattern




ITemplatingService
TemplatingService
Class
Fields
_notifications
_settings
Methods
ApplyAccountActivatedTemplate
ApplyAccountActivationRequiredTemplate
ApplyAccountBlockedTemplate
ApplyAccountDeletedTemplate
ApplyAccountRegistrationTemplate
ApplyAccountSuspendTemplate
ApplyAccountUnlockedTemplate
ApplyAccountVerificationRejectedTemplate
ApplyAuctionBidInvalidatedTemplate
ApplyAuctionBidIsNowBestTemplate
ApplyAuctionBidSurpassedTemplate
ApplyAuctionEndTemplate
ApplyAuctionPlacedBidTemplate
ApplyAuctionWithdrawalTemplate
ApplyAuctionWonTemplate
ApplyEmailChangeRequestTemplate
ApplyLayoutTemplate
ApplyPasswordResetRequestTemplate
ApplySuccessfulEmailChangeTemplate
ApplySuccessfulPasswordResetTemplate
ApplyTemplate (+ 1 overload)
ApplyTemplateForAuctionMessage
BuildTemplatedMessageBody
TemplatingService
ITemplatingService
AbstractTemplatingService
Abstract Class
Fields
Properties
TypeId
Methods
Apply
ITemplatingSer...
Interface
Properties
TypeId
Methods
Apply
AccountRegistrationTemplatingServi...
Class
AbstractTemplatingService
Properties
TypeId
Methods
Apply
AuctionWithdrawalTemplatingService
Class
AbstractTemplatingService
Properties
TypeId
Methods
Apply

In summary

- Conformance to OCP yields the greatest benefits claimed for object oriented technology (flexibility, reusability, maintainability)
- It involves additional time and effort to create the appropriate abstractions.
- Abstractions increase the complexity of the software design.

Resources

Books and papers

- **Ivar Jacobson** – „Object Oriented Software Engineering: A Use Case Driven Approach".
- **Bertrand Meyer** – „Object Oriented Software Construction".
- **Craig Larman – „**Protected Variation: The Importance of Being Closed".
- **John Vlissides, James O. Coplien, Norman L. Kerth, Pattern „**Languages of Program Design 2".
- **Jon Skeet** - „The Open-Closed Principle, in review".
- **Robert C. Martin** – „Clean Code: A Handbook of Agile Software Craftsmanship".
- **Robert C. Martin** – „An Open and Closed Case".
- **Robert C. Martin** – „Agile Software Development, Principles, Patterns, and Practices".
- **Robert C. Martin** – „The Open-Closed Principle".