CS 535 Object-Oriented Programming & Design
Fall Semester, 2001
Doc 23 Two Questions

References

Refactoring: Improving the Design of Existing Code, Fowler, Addison-Wesley, 1999

Question 11

Q11. In my employee class there are many instance variable like ID, name address phone number...? What is the best way to organize it? should I divide it into subclass?

```
Smalltalk.CS535 defineClass: #Customer
    superclass: #{UI.ApplicationModel}
    indexedType: #none
    private: false
    instanceVariableNames: 'firstName lastName middleName homePhoneNumber
workPhoneNumber cellPhoneNumber streetAddress city state zipCode id '
    classInstanceVariableNames: ''
    imports: ''
    category: 'Course-GUI-Examples'
```

Class Methods

```
firstName: aFirstNameString middleName: aMiddleNameString lastName:
aLastNameString homePhoneNumber: aHomePhoneNumberString
workPhoneNumber: aWorkPhoneNumberString cellPhoneNumber:
aCellPhoneNumberString streetAddress: aStreetAddressString city: aCityString
state: aStateString zipCode: aZipCodeString id: aNumber
    ^super new
        firstName: aFirstNameString;
        middleName: aMiddleNameString;
        lastName: aLastNameString;
        homePhoneNumber: aHomePhoneNumberString;
        workPhoneNumber: aWorkPhoneNumberString;
        cellPhoneNumber: aCellPhoneNumberString;
        streetAddress: aStreetAddressString;
        city: aCityString;
        state: aStateString;
        zipCode: aZipCodeString;
        id: aNumber
```

Instance Methods

```
city
   ^city

city: aString
   city:= aString

firstName
   ^firstName

firstName: aString
   firstName:= aString

id
   ^id

id: anInteger
   id := anInteger.

lastName
   ^lastName

lastName: aString
   lastName := aString

middleName
   ^middleName

middleName: aString
   middleName := aString

streetAddress
   ^streetAddress

streetAddress: aString
   streetAddress:= aString
```

etc.

So what are the Problems?

So what are the Solutions?

Code Smells

If it stinks, change it

-- Grandma Beck on child-rearing

Some Smells

Long Parameter List

Data Class

A class with instance variable, setter and getter methods and nothing else

Some Classes

Address
Name
PhoneNumber

It is clear what the instance variable would be

What are the responsibilities of each class?

Name

Knowing when two names are equal
Knowing the full name
Knowing the short name
Knowing when two names might be equal
- One name mispelled
- One name with an abbreviation

  Roger Whitney
  Roger E Whitney
  Roger Earl Whitney

Displaying self in a Window

Serialize/deserialize it to/from an ASCII string

Phone Number

Know the number
Know type of phone number
- Cell phone
- Work
- Home
- Beeper

Format the number

Know if it is local or long distance

Serialize/deserialize it to/from an ASCII string

Start of a Name Class

```
Smalltalk.CS535 defineClass: #Name
   superclass: #{Core.Object}
   indexedType: #none
   private: false
   instanceVariableNames: 'title first middle last '
   classInstanceVariableNames: ''
   imports: ''
   category: 'Course-GUI-Examples'
```

```
CS535.Name class methodsFor: 'Instance Creation'
```

```
title: titleString first: firstNameString middle: middleNameString
last: lastNameString
   ^super new
      setTitle: titleString
      setFirst: firstNameString
      setMiddle: middleNameString
      setLast: lastNameString
```

```
CS535.Name methodsFor: 'accessing'
```

```
fullName
   | name |
   name := String new writeStream.
   title isNil ifFalse: [name nextPutAll: title , ' '].
   name nextPutAll: first , ' '.
   middle isNil ifFalse: [name nextPutAll: middle , ' '].
   name nextPutAll: last.
   ^name contents
```

```
shortName
   ^title isNil
      ifTrue:[first , ' ' , last]
      ifFalse:[title , ' ' , last]
```

CS535.Name methodsFor: 'initialize'

```
setTitle: titleString
   titleString isNil ifTrue:[^nil].
   titleString isEmpty ifTrue:[^nil].
   titleString size > 3 ifTrue:[^title := titleString].
   title := titleString last = $.
      ifTrue:[titleString]
      ifFalse:[titleString , (String with: $.)]

setTitle: titleString setFirst: firstString setMiddle: middleString
setLast: lastString
   self setTitle: titleString.
   first := firstString.
   middle := middleString.
   last := lastString
```

CS535.Name methodsFor: 'comparing'

```
= aName
   ^self fullName = aName fullName
```

Customer Class Becomes

```
Smalltalk.CS535 defineClass: #Customer
   superclass: #{UI.ApplicationModel}
   indexedType: #none
   private: false
   instanceVariableNames: 'name phoneNumber address id '
   classInstanceVariableNames: ''
   imports: ''
   category: 'Course-GUI-Examples'
```

Class Methods

```
name: aName address: anAddress id: id
   ^super new
      setName: aName
      setAddress: aNumber
      setId: id
```

Question 12

If one of my methods maybe in my video class is simply '^ videoID', what is the simplest way for the GUI to use that?

There are a number of ways depending on:

- Does your class need to change the value when the GUI is showing?
- Does the GUI need to change the value?
- Is the GUI in the video class or not?

GUI & Program read/writes values at the same time

There are two options when

- GUI needs to read and write the value
- Program may change the value while the GUI is showing

Option 1 - Use Value Holder

We can

- Make the instance variable a value holder
- Provide the GUI access to the value holder
- Provide the program access to the actual value
- Only provide access program needs

```
Smalltalk.CS535 defineClass: #Video
   superclass: #{UI.ApplicationModel}
   indexedType: #none
   private: false
   instanceVariableNames: 'id '
   classInstanceVariableNames: ''
   imports: ''
   category: 'Course-GUI-Examples'

id
   ^self idHolder value

idHolder
   ^id isNil
      ifTrue:
         [id := String new asValue]
      ifFalse:
         [id]
```

Option 2 - Use Adapter

- Make sure when value is changed call changed: #id
- Provide different program/GUI access to the value

```
Smalltalk.CS535 defineClass: #Video
   superclass: #{UI.ApplicationModel}
   indexedType: #none
   private: false
   instanceVariableNames: 'id '
   classInstanceVariableNames: ''
   imports: ''
   category: 'Course-GUI-Examples'

id: aNumber
        id := aNumber.
        self changed: #id

id
   ^id

idHolder
        | adaptor |
        adaptor := AspectAdaptor forAspect: #id.
        adaptor
                subject: self;
                subjectSendsUpdates: true.
        ^adaptor
```

Program doesn't change the value while GUI is displayed

In this case option 1 & 2 still work, but we another option

Option 3 - Use Adapter no broadcast

- Provide different program/GUI access to the value
- No need to change existing get/set values

```
Smalltalk.CS535 defineClass: #Video
   superclass: #{UI.ApplicationModel}
   indexedType: #none
   private: false
   instanceVariableNames: 'id '
   classInstanceVariableNames: ''
   imports: ''
   category: 'Course-GUI-Examples'

id: aNumber
        id := aNumber.

id
   ^id

idHolder
        | adaptor |
        adaptor := AspectAdaptor forAspect: #id.
        adaptor subject: self;
        ^adaptor
```

GUI is read-only

Program doesn't change value when GUI displayed

Options 1-3 work but we have option 4 & 5

Option 4 - No setter method with adaptor

```
Smalltalk.CS535 defineClass: #Video
   superclass: #{UI.ApplicationModel}
   indexedType: #none
   private: false
   instanceVariableNames: 'id '
   classInstanceVariableNames: ''
   imports: ''
   category: 'Course-GUI-Examples'

id
   ^id

idHolder
        | adaptor |
        adaptor := AspectAdaptor forAspect: #id.
        adaptor subject: self;
        ^adaptor
```

Option 5 - No setter method with ValueHolder

```
Smalltalk.CS535 defineClass: #Video
   superclass: #{UI.ApplicationModel}
   indexedType: #none
   private: false
   instanceVariableNames: 'id '
   classInstanceVariableNames: ''
   imports: ''
   category: 'Course-GUI-Examples'

id
   ^id

idHolder
   ^self id asValue
```

Comments on Option 4 & 5

The original questioner
- Had one getter method
- Wanted the simplest way to add a GUI

Option 4 & 5 require
- Only adding one method
- No changes to other methods/variable
- The idHolder method does not have to be in the same class

For those Who like to live Dangerously

No accessors needed

```
Smalltalk.CS535 defineClass: #Video
   superclass: #{UI.ApplicationModel}
   indexedType: #none
   private: false
   instanceVariableNames: 'id '
   classInstanceVariableNames: ''
   imports: ''
   category: 'Course-GUI-Examples'

idHolder
   |adaptor idIndex |
   idIndex := self class allInstVarNames indexOf: 'id'.
   adaptor := SlotAdaptor forIndex: idIndex.
   adaptor subject: self.
   ^adaptor
```

SlotAdaptor will set/get values of instance variable without needing accessor methods

Code above breaks if change the name of the instance variable