CS 683 Emerging Technologies: Embracing Change
Spring Semester, 2001
Doc 16 Assignment 2 & 3 Comments
Contents

Assignment 2 & 3 Comments ................................................................. 2
Testing and Readability ........................................................................ 2
Assignment 3 Solution ......................................................................... 5
   Problem 1 .......................................................................................... 5
   Problem 2 .......................................................................................... 5
   Problem 3 .......................................................................................... 6
Assignment 3 Tests .............................................................................. 8
Formatting ............................................................................................ 10
   Spaces & Tabs .................................................................................. 10
   Line Wrap ......................................................................................... 11
Formatting Patterns ............................................................................. 12
   Indented Control Flow ..................................................................... 12
   Rectangular Block .......................................................................... 13
   Guard Clause .................................................................................. 14
   Cascade .......................................................................................... 15
Accessors ............................................................................................. 16

References

Smalltalk Best Practice Patterns, Kent Beck, Prentice Hall, 1997

Student papers

Copyright©, All rights reserved. 2001 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/pl.shtml) license defines the copyright on this document.
Assignment 2 & 3 Comments
Testing and Readability

Which is easier to read?
Which shows the best shows the intent?

TestCase subclass: #StringTests

StringTests>>testPalindrome
| result |
result := 'mom' isPalindrome.
self assert: result = true

StringTests>>testPalindrome
| result |
result := 'mom' isPalindrome.
self assert: result

StringTests>>testPalindrome
self assert: 'mom' isPalindrome
Booleans, If statements, Readability

boolean result;
result = someMethod();
if (result = true )
    blah
else
    more blah

or

boolean result;
result = someMethod();
if (result)
    blah
else
    more blah

Style texts recommend the latter
deny

StringTests>>testPalindrome
| result |
result := 'cat' isPalindrome.
self assert: result = false

StringTests>>testPalindrome
self deny: 'cat' isPalindrome.
Assignment 3 Solution
Problem 1

String>>evaluate
| result |
result := Compiler evaluate: self.
^result isNil
  ifTrue:["]
  ifFalse:[result]

FileStream>>evaluate
  ^self contentsOfEntireFile evaluate

Problem 2

evaluateASP
| result |
result := WriteStream on: (String new).
[self atEnd]
  whileFalse:
    [result
      nextPutAll: (self upToAll: '<%');
      nextPutAll: (self upToAll: '%>') evaluate stringRepresentation].
self close.
^result contents
Problem 3

Object subclass: #SampleWebApplication
  instanceVariableNames: ''
  classVariableNames: ''
  poolDictionaries: ''
  category: 'Whitney-ASP'

Class Methods

aspDirectory
  '^Other/asp'
  copyReplaceAll: '/'
  with: FileDirectory slash

startServerAction: aString port: anInteger
  PWS stopServer.
  PWS
    link: aString
    to: SampleWebApplication new.
  PWS
    serveOnPort: anInteger
    loggingTo: 'log'.

Instance Methods

aspAt: aFileURL
  | aspFile |
  ^aspFile evaluateASP
fullPathFor: aFileURL
   | partialPath |
   partialPath := aFileURL
      copyReplaceAll: '/'
         with: FileDirectory slash.
   ^self class aspDirectory , partialPath

isFile: aFileURL
   ^FileStream isAFileNamed: (self fullPathFor: aFileURL)

process: aRequest
   (self isFile: aRequest url)
      ifTrue:[self processSuccess: aRequest]
      ifFalse:[self processFileNotFound: aRequest]

processFileNotFound: aRequest
   aRequest
      reply: PWS notFound;
      reply: PWS contentHTML, PWS crlf;
      reply: ('File: ', aRequest url , ' not found')

processSuccess: aRequest
   aRequest
      reply: PWS success;
      reply: PWS contentHTML, PWS crlf;
      reply: (self aspAt: aRequest url)
Assignment 3 Tests

TestCase subclass: #ASPTests
  instanceVariableNames: ''
  classVariableNames: ''
  poolDictionaries: ''
  category: 'Whitney-ASP'!

Private instance methods

createFile: aLocalFileName contents: aString
  | file |
  file nextPutAll: aString.
  file close

deleteFile: aFilePath
  FileDirectory deleteFilePath: aFilePath

simpleAsp
  ^'<HTML><BODY>hi <% 1 + 2 %></BODY></HTML>'

startServerAction: aString port: anInteger
  SampleWebApplication
    startServerAction: aString
    port: anInteger

stopServer
  PWS stopServer
Test Instance Methods

testASPFileEvaluate

| asp result |
self
    createFile: 'testAsp'
    contents: self simpleAsp.
asp := FileStream fileNamed: 'testAsp'.
result := asp evaluateASP.
self assert: result = '<HTML><BODY>hi 3</BODY></HTML>'.
self deleteFile: 'testAsp'

testStringEvaluate

self
    assert: '1 + 1' evaluate = 2;
    assert: ' ' evaluate isEmpty

testWebApplications

| result |
self startServerAction: 'test' port: 8080.
self
    createFile: SampleWebApplication aspDirectory , ':test:sample.asp'
    contents: self simpleAsp.
result := (HTTPSocket httpGet: '127.0.0.1:8080/test/sample.asp') contents.
self assert: result = '<HTML><BODY>hi 3</BODY></HTML>'.
self deleteFile: SampleWebApplication aspDirectory , ':test:sample.asp'.

testWebApplicationsBadFile

| result |
self startServerAction: 'test' port: 8080.
result := (HTTPSocket httpGet: '127.0.0.1:8080/test/crap.asp') contents.
self assert: (result includesSubString: 'not found').
Formatting
Spaces & Tabs

The following is:
• Not readable
• Not professional
• Not acceptable
• I will no longer grade papers where the formatting does not show the structure of the code

at: anInteger put: anObject
(smallKey ~= largeKey)
ifTrue:
[(anInteger < smallKey)
ifTrue: [self atLeftTree: anInteger put: anObject]
ifFalse: [(smallKey = anInteger)
ifTrue: [smallValue := anObject]
ifFalse: [(anInteger < largeKey)
ifTrue: [self atMiddleTree: anInteger put: anObject]
ifFalse: [(largeKey = anInteger)
ifTrue: [largeValue := anObject]
ifFalse: [(largeKey < anInteger)
ifTrue: [self atRightTree: anInteger put: anObject]]])]
ifFalse:
[self addNewKey: anInteger with: anObject].

Squeak uses tabs to indent lines

Do not use spaces at the beginning of a line to indent!
Line Wrap

Don't ever do this to anyone under any circumstance

Find the problem and fix it

I will no longer grade papers with line wrap

at: anInteger put: anObject
   (smallKey <= largeKey)
      ifTrue:
         [(anInteger < smallKey)
            ifTrue: [self atLeftTree: anInteger put:
            anObject]
            ifFalse: [(smallKey = anInteger)
                ifTrue: [smallValue := anObject]
                ifFalse: [(anInteger < largeKey)
                    ifTrue: [self atMiddleTree:
                    anInteger put: anObject]
                    ifFalse: [(largeKey = anInteger)
                        ifTrue: [largeValue :=
                        anObject]
                        ifFalse: [(largeKey <
                        anInteger)
                            ifTrue: [self atRightTree:
                            anInteger put: anObject]]]]]]
            ifFalse: [self addNewKey: anInteger with:
            anObject].
Formatting Patterns
Indented Control Flow

How do you indent messages?

- Put zero or one argument messages on same line as receiver
- With two or more keywords put each keyword/argument pair on its own line, indented uses tab

Examples

2 + 3
a < b ifTrue: [code ]

a < b
    ifTrue: [ code ]
    ifFalse: [ more code]

tree
    at: 5
    put: 'cat'

---

1 Beck 1997 pp. 175-177
Rectangular Block

Make blocks rectangular

Use the square brackets as the upper left and bottom right corners of the rectangle

If the block contains a simple statement, the block can stay on one line

If the block contains a compound statement bring the block onto its own line and indent

angle isNil ifTrue: [self computeAngle]

self isDegrees ifTrue: [^angle * 90 + 270 degreesToRadians]

self isDirty
  ifTrue:
    [self clearCaches.
     self recomputeAngles]

self isNil
  ifTrue:
    [self
     at: each
     put: 0]

---

2 Beck 1997 pp. 177-178
Guard Clause

How do you format code that should not execute if a condition holds?

connect
  self isConnected
    ifFalse: [self connectConnection]

Format the one-branch conditional with an explicit return

connect
  self isConnected ifTrue: [^self]
  self connectConnection

\(^3\) Beck 1997 pp. 178-179
Cascade

How do you format multiple messages to the same receiver?

Use a cascade

Put each message on its own line and indent one tab

Only use cascades for messages with zero or one arguments

OrderedCollection new
  add: 5;
  add: 10

---

4 Beck 1997 pp. 183-185
Accessors

Smalltalk convention does not use `getX` `setX` for naming accessor methods