CS 535 Object-Oriented Programming & Design
Fall Semester, 2001
Doc 9 Assignment 3 Comments
Contents

A Solution

Problem 1

```
SimpleCircle>>printOn: aStream
   aStream
      nextPutAll: 'SimpleCircle(';
      print: origin;
      nextPutAll: ', ';
      print: radius;
      nextPutAll: ')'
```

Problem 2

```
'From VisualWorks® NonCommercial, Release 5i.4 of August 9,
2001 on October 9, 2001 at 5:33:29 pm'!

Smalltalk.CS535 defineClass: #TestSimpleCircle
   superclass: #{XProgramming.SUnit.TestCase}
   indexedType: #none
   private: false
   instanceVariableNames: ''
   classInstanceVariableNames: ''
   imports: ''
   category: 'Course-Examples'!

!CS535.TestSimpleCircle methodsFor: 'basic tests'!

testArea
   | circle area |
   circle := SimpleCircle
      origin: 0@1
      radius: 1.
   area := circle area.
   self assert:  (area - Float pi) < 0.00001 .

   circle := SimpleCircle
      origin: 0@1
      radius: 2.
   area := circle area.
   self assert:  ( Float pi * 2 * 2 - area) < 0.00001 .!
```

```
testCreation
   | circle |
   circle := SimpleCircle
      origin: 1@1
      radius: 5.
   self
      assert: circle notNil;
      assert: circle radius = 5;
      assert: circle origin = (1 @ 1).!

testEquality
   | a b c |
   a := SimpleCircle
      origin: -1@1
      radius: 2.
   b := SimpleCircle
      origin: -1@1
      radius: 2.
   c := SimpleCircle
      origin: -1@1
      radius: 1.
   self
      assert:  (a = b);
      deny: (a = c)!
```

```
testIncludes
   | circle |
   circle := SimpleCircle
      origin: -1@1
      radius: 2.
   self
      assert:  (circle includes: -1@1);
      assert: (circle includes: 1@1);
      assert: (circle includes: 0@0);
      deny: (circle includes: 1.002 @ 1)! !
```

Problem 3

```
'From VisualWorks® NonCommercial, Release 5i.4 of August 9,
2001 on October 9, 2001 at 5:39:37 pm'!

Smalltalk.CS535 defineClass: #Assignment2Test
   superclass: #{XProgramming.SUnit.TestCase}
   indexedType: #none
   private: false
   instanceVariableNames: ''
   classInstanceVariableNames: ''
   imports: ''
   category: 'Course-Examples'!

!CS535.Assignment2Test methodsFor: 'prime'!

testPrime
   self
      deny: -12 isPrime;
      deny: 1 isPrime;
      assert: 2 isPrime;
      assert: 3 isPrime;
      deny: 4 isPrime;
      deny: 9 isPrime;
      deny: (13*13) isPrime;
      assert: 7 isPrime;
      assert: 11 isPrime;
      assert: 19 isPrime!
```

testPrimeWithArray

```
  #( 2 3 5 7 11 13 179 ) do: [:each | self assert: each isPrime].

  "13*13 = 169"
  #( -12 1 4 169) do: [:each | self deny: each isPrime]! !
```

!CS535.Assignment2Test methodsFor: 'asLetterGrade'!

testLettergradeFloat

```
  self
    assert: 101.0 asLetterGrade = $A;
    assert: 90.0 asLetterGrade = $A;
    assert: (45.0 + 40.0 + 5.0) asLetterGrade = $A;
    assert: 89.0 asLetterGrade = $B;
    assert: 70.0 asLetterGrade = $C;
    assert: 60.0 asLetterGrade = $D;
    assert: -12.0 asLetterGrade = $F!
```

```
testLettergradeInteger
   self
      assert: 101 asLetterGrade = $A;
      assert: 90 asLetterGrade = $A;
      assert: 89 asLetterGrade = $B;
      assert: 70 asLetterGrade = $C;
      assert: 60 asLetterGrade = $D;
      assert: -12 asLetterGrade = $F! !

testLettergrade

   | scores grades |
   scores := #( -12  0  55  60  61  69 70  80  81 89  90 91 201).
   grades :=#( $F  $F $F $D $D  $D $C $B $B $B $A $A $A).

   scores
      with: grades
      do: [:score :grade | self assert: score asLetterGrade = grade]!

!CS535.Assignment2Test methodsFor: 'divides'!

testDivides
   self
      assert: (2 divides: 4);
      assert: (3 divides: 9);
      deny: (2 divides: 9);
      deny: ( 0 divides: 2)! !
```

Assignment 3 Comments
Issues
Indentation

Indent to show the structure of your code.

Bad

```
printOn: aStream
   aSteam
      nextPutAll: ‘SimpleCircle(‘;
         print: radius;
      nextPutAll: ‘, ‘;
         print: origin;
      nextPut: $)
```

```
printOn: aStream
aSteam
   nextPutAll: ‘SimpleCircle(‘;
   print: radius;
   nextPutAll: ‘, ‘;
   print: origin;
   nextPut: $)
```

Transcript and printString

```
printString
   Transcript
      clear;
      show: ‘SimpleCircle(‘;
      blah.
```

Treat the Transcript as a debugging window

Don’t use ii in non-debugging code

The standard is to override printOn: not printString

Testing Formats

```
testIncludes
   | aCircle aPoint |
   aCircle := SimpeCircle radius: 5 origin: 0 @ 0.
   aPoint := 10 @ 10.
   self assert: (aCircle includes: aPoint) = false.
```

Or

```
   self assert: (aCircle includes: 10 @ 10) = false.
```

Or

```
   self deny: (aCircle includes: 10 @ 10 ).
```

Extra Code

Why the extra code?

It does not do anything useful

```
testIncludes
   | aCircle aPoint |
   aCircle := SimpeCircle radius: 5 origin: 0 @ 0.
   aPoint := 10 @ 10.
   aCircle includes: aPoint.
   self deny: (aCircle includes: aPoint).
```

Weak Tests

```
testArea
   | aCircle |
   aCircle := SimpeCircle radius: 5 origin: 0 @ 0.
   self assert: (aCircle area > 0).

testArea
   | aCircle |
   aCircle := SimpeCircle radius: 0 origin: 0 @ 0.
   self assert: (aCircle area = 0).

testArea
   | aCircle |
   aCircle := SimpeCircle radius: 1 origin: 0 @ 0.
   self assert: (aCircle area = Float pi).

testIncludes
   | aCircle aPoint |
   aCircle := SimpeCircle radius: 5 origin: 0 @ 0.
   aPoint := 1 @ 1.
   self assert: (aCircle includes: aPoint).
```