CS 535 Object-Oriented Programming & Design
Fall Semester, 2013
Doc 7 Assignment 2 Comments
Sept 17 2013

Issues

Names
Magic Numbers
Formatting
Complex Code
    Which mask errors

```
rotate: t1

	| t2 t3 t4 |
	t3 := (t2 := self asString asArray) size.
	t4 := 0.
	[(t4 := t4 + 1) < t3 whileTrue.
	^self collect: [:t5 | t5 next2: t1]
```

```
rotate: number
| y len z a strcon Flag charr |
strcon := ''.
len := (self size).
1 to: len do: [:i |
charr := (self at: i) asInteger
charr<=90 ifTrue: [Flag := 0].
charr>=97 ifTrue: [Flag := 1].
Flag = 0
ifTrue: [y := (self at: i) asInteger + number.
y>90 ifTrue: [y := y -26].
].
etc.
```

Flag

Not informative

Indicates that we are testing for some condition

What condition?

```
charr<=90 ifTrue: [Flag := 0].
```

```
charr<=90 ifTrue: [isLowercase := false].
```

```
charr<= $Z asInteger ifTrue: [isLowercase := false].
```

```
rotate: anInteger
^self collect: [:each | (each asInteger + anInteger) > ($z asInteger)
ifTrue: [(each asInteger + anInteger - 26) asCharacter]
ifFalse: [(each asInteger + anInteger) < ($a asInteger)
ifTrue: [(each asInteger + anInteger + 26) asCharacter]
ifTrue: [(each asInteger + anInteger) asCharacter]]]
```

After Using Control-o

```
rotate: anInteger

	^self collect:
		[:each |
		each asInteger + anInteger > $z asInteger
			ifTrue: [(each asInteger + anInteger - 26) asCharacter]
			ifFalse:
				[each asInteger + anInteger < $a asInteger
			ifTrue: [(each asInteger + anInteger + 26) asCharacter]
			ifTrue: [(each asInteger + anInteger) asCharacter]]]
```

Using local variable

```
rotate: anInteger

        ^self collect:
                [:each |
                | shifted |
                shifted := each asInteger + anInteger.
                shifted > $z asInteger
                        ifTrue: [(shifted - 26) asCharacter]
                        ifFalse:
                                [shifted < $a asInteger
                        ifTrue: [(shifted + 26) asCharacter]
                                ifFalse: [shifted asCharacter]]]
```


Is this easer to follow? Can you see the error now?

```
rotate: n
    ^(self asByteArray collect: [:ch | ch + n > 122
                ifTrue: [ch+n -122+96]
                ifFalse: [ch+n>90 & ch < 90
                        ifTrue: [ ch+n-90+64]
                        ifFalse:[ch+n<65
                            ifTrue: [90-64+(ch+n]
                            ifFalse: [ch+n<97
                                ifTrue: [122-96+(ch+n)]
                                ifFalse:[ch+n]]]]]) asString
```

```
Character>>getNextCharacter: nextIndex
    | character characterShift finalCharacter |

    "Find the ASCII value of the character"
    character := self asInteger.

    "if the character is uppercase"
    (91 > character) & (character>64) ifTrue: [
        "Shift character by the argument number"
        characterShift := character+nextIndex.
        "if the new character value exceeds the ASCII value of the Z"
        (characterShift > 90 ) ifTrue: [
            finalCharacter := characterShift - 90 + 64.
            ^finalCharacter asCharacter].
        "more code, but not shown ..."
```

"Find the ASCII value of the character"
character := self asInteger.

asciiValue := self asInteger

"if the character is uppercase"
(91 > character) & (character>64) ifTrue:

self isUppercase ifTrue:

"if the new character value exceeds the ASCII value of the Z"

(characterShift > 90) ifTrue:

(characterShift > $Z asInteger) ifTrue:

```
Character>>getNextCharacter: nextIndex
    | asciiValue characterShift finalCharacter |

    asciiValue := self asInteger.

    self isUppercase ifTrue: [
        "Shift character by the argument number"
        characterShift := asciiValue +nextIndex.

    (characterShift > $Z asInteger ) ifTrue: [
        finalCharacter := characterShift - 90 + 64.
        ^finalCharacter asCharacter].
    "more code, but not shown ..."
```

```
Character>>+ anIntegerOrCharacter
    ^(self asInteger + anIntegerOrCharacter asInteger) asCharacter
```

```
Character>>- anInteger
    ^(self asInteger - anIntegerOrCharacter asInteger) asCharacter
```

```
Character>>getNextCharacter: nextIndex
    |  characterShift |

    characterShift := self + nextIndex.

    self isUppercase ifTrue: [
        (characterShift > $Z) ifTrue: [
            ^characterShift - $Z + $A - 1].
        "more code, but not shown ..."
```

```
Character>>+ anIntegerOrCharacter
    ^(self asInteger + anIntegerOrCharacter asInteger) asCharacter
```

```
Character>>- anInteger
    ^(self asInteger - anIntegerOrCharacter asInteger) asCharacter
```

```
Character>>uppercaseMod
     ^$A + ((self asInteger - $A asInteger) \\ 26)
```

```
Character>>getNextCharacter: nextIndex
    |  characterShift |

    characterShift := self + nextIndex.

    self isUppercase ifTrue: [
        (characterShift > $Z) ifTrue: [
            ^characterShift uppercaseMod].
        "more code, but not shown ..."
```

```
Character>>+ anIntegerOrCharacter
    ^(self asInteger + anIntegerOrCharacter asInteger) asCharacter
```

```
Character>>- anInteger
    ^(self asInteger - anIntegerOrCharacter asInteger) asCharacter
```

```
Character>>lowercaseMod
     ^$a + ((self asInteger - $a asInteger) \\ 26)
```

```
Character>>uppercaseMod
     ^$A + ((self asInteger - $A asInteger) \\ 26)
```

```
Character>>circularShiftBy: anInteger


     ^self isUppercase
          ifTrue: [(self + anInteger) uppercaseMod]
          ifFalse: [(self + anInteger) lowercaseMod]
```

```
Character class>>englishAlphabetSize
	^26
```

```
Character>>+ anIntegerOrCharacter
	^(self asInteger + anIntegerOrCharacter asInteger) asCharacter
```

```
Character>>lowercaseMod
	^$a + ((self asInteger - $a asInteger) \\ self class englishAlphabetSize)
```

```
Character>>uppercaseMod
	^$A + ((self asInteger - $A asInteger) \\ self class englishAlphabetSize)
```

```
Character>>circularShiftBy: anInteger
	^self isUppercase
		ifTrue: [(self + anInteger) uppercaseMod]
		ifFalse: [(self + anInteger) lowercaseMod]
```

```
String>>rotate: anInteger
	^self collect: [:each | each circularShiftBy: anInteger]
```

```
Character class>>englishAlphabetSize
	^26
```

```
Character>>+ anIntegerOrCharacter
	^(self asInteger + anIntegerOrCharacter asInteger) asCharacter
```

```
Character>>alphabetModAt: aCharacter
	^aCharacter + ((self asInteger - aCharacter asInteger) \\
		self class englishAlphabetSize)
```

```
Character>>circularShiftBy: anInteger
		^self + anInteger
			alphabetModAt: (self isUppercase ifTrue: [$A] ifFalse: [$a])
```

```
String>>rotate: anInteger
	^self collect: [:each | each circularShiftBy: anInteger]
```