CS 535 Object-Oriented Programming & Design
Fall Semester, 2011
Doc 8 Assignment 2 Comments
Sept 20 2011

Problem 8 - Character>>previous

```
previous
    | temp |
    temp = self.
    temp == 97
        ifTrue: [^(temp asInteger + 25) asCharacter]
        ifFalse: [
                temp == 65
                    ifTrue:  [^(temp asInteger + 25) asCharacter]
                    ifFalse: [^(temp asInteger -1) asCharacter]]
```


Issues: name (temp), magic numbers (97, 65, 25), nesting of if statements

Problem 8 - Character>>previous

previous

```
self = $a
    ifTrue: [^$z]
    ifFalse: [
            self = $A
                ifTrue:  [^$Z]
                ifFalse: [^(self asInteger -1) asCharacter]
```

Problem 8 - Character>>previous

```
previous
	self = $a ifTrue: [^$z].
	self = $A ifTrue:  [^$Z].
	^(self asInteger -1) asCharacter]
```

Boundary Conditions

```
previous
    self = $a ifTrue: [^$z].
    self = $A ifTrue:  [^$Z].
    ^(self asInteger -1) asCharacter]
```

Error prone
Make code complex

Handled boundary conditions first
Then handled main case

Formatting

```
previous
(self = $a) ifTrue: [
    ^(self asInteger + 25) asCharacter
] ifFalse: [
(self = $a) ifTrue: [
    ^(self asInteger + 25) asCharacter
] ifFalse: [
    ^(self asInteger -1) asCharacter
]]
```

Problem 9 - String>>rotate: n

```
rotate: n
    | input_string i n1 temp flag p |
    input_string := Array new: self size.
    input_string size = 0 ifTrue: [^'String is empty'].
    i := 1.
    temp := flag := 0.
    n1 := n.
    n1 := n1\\26.
    (n1 < 0) ifTrue: [flag := 1];
    [i <= self size ] whileTrue:
        [p := 0.
        input_string at: i put: (self at: i).
        temp := (input_string at: i) asInteger;
```

(continued next page)

Issues: names, names, names, unneeded statements

Problem 9 - String>>rotate: n

```
rotate: n
    | rotatedString currentIndex reducedShiftAmount currentValue doLeftShift p |
    rotatedString := Array new: self size.
    currentIndex := 1.



    reducedShiftAmount := n\\26.
    doLeftShift := reducedShiftAmount < 0;
    [currentIndex <= self size ] whileTrue:
        [p := 0.

        currentValue := (self at: currentIndex) asInteger;
        (continued next page)
```


Issues: names, names, names, unneeded statements

Continued

```
flag = 1
    ifTrue: [
        (temp <= 90)
            ifTrue: [
                (temp + n1 < 65) ifTrue: [p := 1].
                temp := temp + n1 + (26*p)]
            ifFalse: [
                (temp + n1 < 97) ifTrue: [p := 1].
                temp := temp + n1 + (26*p)]]
    ifFalse: [
        (temp <= 90)
            ifTrue: [
                (temp + n1 > 90) ifTrue: [p := 1].
                temp := temp + n1 - (26*p)]
            ifFalse: [
                (temp + n1 > 122) ifTrue: [p := 1].
                temp := temp + n1 - (26*p)]].
    input_string at: i put: temp asCharacter.
    i := i +1.
].
^input_string
```


Issues: name, nested if statement, magic numbers

```
doShiftLeft
    ifTrue: [
        (currentValue <= $Z asInteger)
            ifTrue: [
                (currentValue + reducedShiftAmount < $A asInteger) ifTrue: [p := 1].
                currentValue := currentValue + reducedShiftAmount + (26*p)]
            ifFalse: [
                (currentValue + reducedShiftAmount < $a asInteger) ifTrue: [p := 1].
                currentValue := currentValue + reducedShiftAmount + (26*p)]]
    ifFalse: [
        (currentValue <= $Z asInteger)
            ifTrue: [
                (currentValue + reducedShiftAmount > $Z asInteger) ifTrue: [p := 1].
                currentValue := currentValue + reducedShiftAmount - (26*p)]
            ifFalse: [
                (currentValue + reducedShiftAmount > $z asInteger) ifTrue: [p := 1].
                currentValue := currentValue + reducedShiftAmount - (26*p)]].
    rotatedString at: currentIndex put: currentValue asCharacter.
    currentIndex := currentIndex +1.
].
^rotatedString
```


Issues: name, nested if statement, magic numbers

Repeated actions

Converting characters to integers to compare them

Adding character integer values to get new character integer value

Converting character integer values back to characters

Send message to objects

```
Character>> < aCharacter
    ^self asInteger < aCharacter asInteger
```

```
Character>> <= aCharacter
    ^self asInteger <= aCharacter asInteger
```

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

Hiding details simplefies code

```
String>>rotate: n
    blah
    character := self at: k.
    character asInteger + n > $z asInteger
        ifFalse: [rotatedCharacter := (character asInteger + n) asCharacter]
```

```
String>>rotate: n
    blah
    character := self at: k.
    character + n > $z
        ifFalse: [rotatedCharacter := character + n]
```

```
doShiftLeft
    ifTrue: [
        (currentChar <= $Z)
            ifTrue: [
                (currentChar + reducedShiftAmount < $A) ifTrue: [p := 1].
                shiftedChar := currentChar + reducedShiftAmount + (26*p)]
            ifFalse: [
                (currentValue + reducedShiftAmount < $a) ifTrue: [p := 1].
                shiftedChar := currentChar + reducedShiftAmount + (26*p)]]
    ifFalse: [
        (currentValue <= $Z)
            ifTrue: [
                (currentValue + reducedShiftAmount > $Z) ifTrue: [p := 1].
                shiftedChar := currentChar + reducedShiftAmount - (26*p)]
            ifFalse: [
                (currentChar + reducedShiftAmount > $z) ifTrue: [p := 1].
                shiftedChar := currentChar + reducedShiftAmount - (26*p)]].
    rotatedString at: currentIndex put: shiftedChar.
    currentIndex := currentIndex +1.
].
^rotatedString
```


Issues: name, nested if statement, magic numbers

Keep data and operations together

```
doShiftLeft
    ifTrue: [
        (currentChar <= $Z)
            ifTrue: [
                (currentChar + reducedShiftAmount < $A) ifTrue: [p := 1].
                shiftedChar := currentChar + reducedShiftAmount + (26*p)]
            ifFalse: [
                (currentValue + reducedShiftAmount < $a) ifTrue: [p := 1].
                shiftedChar := currentChar + reducedShiftAmount + (26*p)]]
```

What are we doing here?

Extract shifting characters to Character Class

```
Character>>shiftLeft: anInteger
    (self <= $Z)
        ifTrue: [
            (self + anInteger < $A)
                ifTrue: [^self + anInteger + 26]
                ifFalse: [^self + anInteger]].
    (self + anInteger < $a)
        ifTrue: [^self + anInteger + 26]
        ifFalse: [^self + anInteger]].
```

We get

```
        shiftedChar := doShiftLeft
            ifTrue: [currentChar shiftLeft: reducedShiftAmount]
            ifFalse: [currentChar shiftRight: reducedShiftAmount].
        rotatedString at: currentIndex put: shiftedChar.
        currentIndex := currentIndex +1.
    ].
    ^rotatedString
```

The entire method

```
rotate: n
    | rotatedString currentIndex reducedShiftAmount currentChar doLeftShift |
    rotatedString := Array new: self size.
    currentIndex := 1.
    reducedShiftAmount := n\\26.
    doLeftShift := reducedShiftAmount < 0;
    [currentIndex <= self size ] whileTrue:
        [currentChar := (self at: currentIndex);
        shiftedChar := doShiftLeft
            ifTrue: [currentChar shiftLeft: reducedShiftAmount]
            ifFalse: [currentChar shiftRight: reducedShiftAmount].
        rotatedString at: currentIndex put: shiftedChar.
        currentIndex := currentIndex +1.
].
^rotatedString
```

Using Collect we get

```
rotate: n
    | reducedShiftAmount |
    reducedShiftAmount := n\\26.

    ^self collect: [:currentChar |
        (reducedShiftAmount < 0)
            ifTrue: [currentChar shiftLeft: reducedShiftAmount]
            ifFalse: [currentChar shiftRight: reducedShiftAmount]].
```

Problem 9 Another Solution

```
Character>>next

	self = $z ifTrue: [^$a].
	self = $Z ifTrue: [^$A].
	^(self asInteger + 1) asCharacter
```

```
Character>>previous

	self = $a ifTrue: [^$z].
	self = $A ifTrue: [^$Z].
	^(self asInteger - 1) asCharacter
```

```
Character>>rotate: anInteger

	| result |
	result := self.
	anInteger abs timesRepeat:
		[result := anInteger < 0
			ifTrue: [result previous]
			ifFalse: [result next]].
	^result
```

```
String>>rotate: anInteger

	^self collect: [:each | each rotate: anInteger]
```