	Carátula para entrega de prácticas
Facultad de Ingeniería	Ingeniería en Computación

Profesor(a):	
Asignatura:	Programación Orientada a Objetos
Grupo:	1
No de Práctica(s):	1
Integrante(s):	11700202-9
No. de Equipo de cómputo empleado:	
Semestre:	2024-1
Fecha de entrega:	1 de septiembre de 2023
Observaciones:	

CALIFICACIÓN: ___________________

Index.

Practice 1 - Environment and programming language.

Numerical bases converter.

1.- Introduction.

This report has the objective of explaining the first practice of the Object oriented programming subject, being our introduction to the paradigm and the Java language.

We must build an algorithm for converting numerical bases, particularly the standard ones (decimal, hexadecimal, binary and octal), displaying a menu for every possible conversion between bases.

1.1.- Hypothesis

My teammate and I discussed some ways to make the converter, thinking first about recursive methods using modulo and then converting to string. But this would be a little tedious in terms of implementation. After a brainstorm of approximately ten minutes we've decided to generalize part of the modulo idea and attack the problem in pieces.

2.- Development

The steps to follow were:

- Create a method for converting from any base, to decimal.
- Create a method for converting any base to decimal.
- Use these two methods for creating the global base converter method
- Create a method for converting hexadecimal to decimal (because of the letters needed for represent the numbers)

"convertToDecimal"

We need this formula:

$$\sum_{n}^{i=0} = (a_1\, mod\, 10) * b^0 + (a_2\, mod\, 10) * b^1 + \dots + (a_n mod\, 10) * b^{n-1}$$

a_i = number (divided by 10 in each iteration for obtain the digit)

b = source base

n = amount of digits

Pseudocode:

```
function convertToDecimal(number, base):
    decimalValue = 0
    power = 0

    while number > 0:
        digit = number % 10
        decimalValue += digit * base^power
        power++
        number = number / 10

    return decimalValue
```

“convertFromDecimal”

The principle we need to follow is to obtain the modulo of the number with the base and insert it at the front of a string (in this case StringBuilder for better string manipulation), then divide the number by the base until 0.

Adding to the process the recognition of the digits greater than 9 in case of Hexadecimal numbers.

Pseudocode:

```
function convertFromDecimal(decimalValue, base):
    result = StringBuilder()

    while decimalValue > 0:
        remainder = decimalValue % base
        if remainder == 10:
            result.insert(0, "A")
        else if remainder == 11:
            result.insert(0, "B")
        else if remainder == 12:
            result.insert(0, "C")
        else if remainder == 13:
            result.insert(0, "D")
        else if remainder == 14:
            result.insert(0, "E")
        else if remainder == 15:
            result.insert(0, "F")
        else:
            result.insert(0, remainder)
        decimalValue = decimalValue / base

    return result.toString()
```

“hexToDecimal”

We need to follow this process because we are treating the number as a string:

$$\sum_{n}^{i=0} = S_i(16) + d_i$$

S = current sum

d = digit

n = total digits

Pseudocode:

```
function hexToDecimal(hex):
    decimal = 0
    digits = "0123456789ABCDEF"
    hex = toUpperCase(hex)

    for i = 0 to length(hex) - 1:
        c = charAt(hex, i)
        digitValue = indexOf(digits, c)
        decimal = decimal * 16 + digitValue

    return decimal
```

Validations (on strings) pseudocode:

Hexadecimal:

```
function ValHexa(hex):
    hexChars = "0123456789ABCDEF"
    hex = toUpperCase(hex)

    for i = 0 to length(hex) - 1:
        tmp = charAt(hex, i)
        if indexOf(hexChars, tmp) == -1:
            return false

    return not isEmpty(hex)
```

Binary and Octal.

```
validate = toString(number)
binChars = "01"
octChars = "01234567"

if sourceBase == 2:
    for i = 0 to length(validate) - 1:
        tmp = charAt(validate, i)
        if indexOf(binChars, tmp) == -1:
            print("Numero no permitido en base 2")
            exit(0)

if sourceBase == 8:
    for i = 0 to length(validate) - 1:
```

```
tmp2 = charAt(validate, i)
if indexOf(octChars, tmp2) == -1:
    print("Numero no permitido en base 8")
    exit(0)
```

Conclusion: The hypothesis was correct, because we only need four methods to convert all bases instead of twelve. We could also convert non-standard bases with the formula we've used, so the solution is kind of global for this conversion problem.

I enjoyed this first practice. I've tried Java before, but this program made me remember the syntax, classes and some other things essential to the language. I'm also excited about the next practices and the new knowledge that is coming for the rest of the course.

References:

- [1] Oracle. (2023, July). Java API, Class String [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/lang/String.html
- [2] Oracle. (2023, July). Java API, Class StringBuilder [Online] https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html