Code testing

Contents

[1. User input 2](#_Toc8730704)

[User input name: 2](#_Toc8730705)

[User input surname: 2](#_Toc8730706)

[User input class: 2](#_Toc8730707)

[User input answer: 2](#_Toc8730708)

[2. Testing strategy: 3](#_Toc8730709)

[3. Testing name input: 3](#_Toc8730710)

[4. Testing answer input 4](#_Toc8730711)

# User input

## User input name:

When the user inputs a name, it will look at table to see if any of the characters are included in this table. If this is not the case it will allow the user to input this name. If not, it will tell the user the name contains possible illegal characters or the name is too long and the user has to re input the name.

The table it looks at includes these characters: '[@!#$%^&\*()<>?/\\\|}{~:0-9`=+;]'

## User input surname:

Surname follows the same table as name. The only difference is that surname allows longer names than name. 20 instead of 15.

## User input class:

Class has a different table because it might need characters which name and surname don’t allow. For example numbers. It works the same as name and surname just a different table

Table in question: '[@!#$%^&\*()<>?/\\\|}{~:=+`;]'

## User input answer:

The user input works different from name, surname and class. Instead of looking at a table, it checks if it can convert it to an integer. If this is possible it will allow the user to input the value. If not, it will re ask the question to the user and tell that the answer before was not allowed.

# Testing strategy:

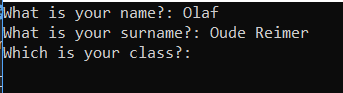
The strategy for testing will be to just input as many inputs as I can. So I’ll try Olaf() and Olaf= if any of these are allowed which they shouldn’t I’ll fix it by adding that character to the table. If have also asked friends to play the game and tell me if they managed to break any of it. This is so if I forgot to check something. Someone else might try it. There isn’t much else the user can do to try and break it Because it can only input data. (Unless the user goes into the code and chances something)

# Testing name input:

**Names, surname and class input:**

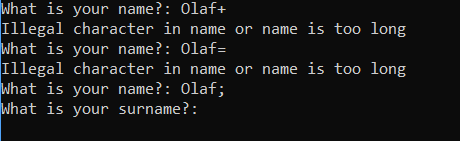
**Works:**

* Olaf= (Not intended fixed now)
* Olaf; (Not intended fixed now
* Olaf+ (Not intended fixed now)
* Olaf-
* Olaf
* Olaf or
* Olaf’s
* Oléf
* Neß
* Olaf1 (Works for class only)



**Doesn’t work:**

* Olaf()
* Olaf1
* Olaf/
* Thisisareallylongnamethatkeepsongoing (Too long)
* Olaf&
* Olaf^
* Olaf$
* Olaf%
* Olaf#
* Olaf@
* Olaf!
* Olaf\
* Int(1)
* Str(a)
* “Blank space”
* “Enter”



# Testing answer input

**Works:**

* 1
* -2

**Doesn’t work:**

* A
* a
* int(1)
* ()
* @
* 1.0 (Decimal point is not needed

