Professorship of Embedded Systems and Internet of Things
Department of Electrical and Computer Engineering
Technical University of Munich

Exercise Sheet 2 – Solution

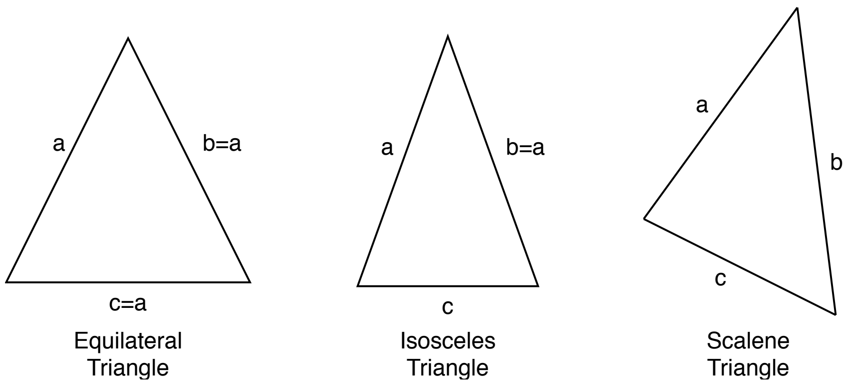
Software Architecture for Distributed Embedded Systems, WS 2021/22

Prof. Steinhorst, M. Sc. Emanuel Regnath, M. Sc. Jens Ernstberger

Exercise 2.1: Just a Triangle

After desperately trying to find your first customer as a small team of developers, you stumble upon a customer with an ambitious software project in mind. The customer gives you this requirement:

> I want a software that can detect different types of triangles. Whenever I provide it three numbers, let's say `a`,`b` and `c`, it should output equilateral, isosceles or scalene triangle. Like in the following figure!



Software development requires planning and execution of this planning in order to build large applications, generally referred as software development cycles. We will do a very quick software development using the V cycle (modified waterfall cycle) (even if seems like an overkill *in the beginning*), starting from these customer requirements. *There is one rule: You are not allowed to write code unless it is explicitly allowed!*

1. Starting from the customer requirement, what are the next steps before the software is delivered to the customer?

Solution:

- Requirement Analysis
- Specifying Software Requirements
- Implementation
- Testing and Verification

2. What are the analyzed software requirements? Generally the customer requirements can be vague where you need to find the actual requirements to start implementing. These analyzed requirements can be considered as the design documents.

Solution:

- The software should obtain inputs from a human agent
- The software should provide an output to a human agent
- On an input of three equal numbers, the software should output equilateral
- On an input of exactly two equal numbers, the software should output isosceles
- On an input where no number is equal to the other one, the software should output scalene.

3. Write a Python script or pseudo code, or even logic, according to these requirements. You can either assume that the numbers are passed as arguments or the script prompts the user to input numbers. *You are allowed to write code here.*

Solution: The following code is also available in Moodle as a python file. This software has problems that will be revealed in question 5

```python
'''
This function asks the user 3 input values and returns them in a list
The list items have the same order as the input
For debugging, print line can be enabled
'''
def receive_inputs():
    input_a = input('Enter the first edge length a ==> ')
    input_b = input('Enter the second edge length b ==> ')
    input_c = input('Enter the third edge length c ==> ')
    #print('a=', input_a, 'b=', input_b, 'c=', input_c)
    return [input_a,input_b,input_c]

'''
This function gets a list of three variables and compares them with each other
If exactly two of them are equal the return is isosceles
If all three are equal the return is equilateral
If non are equal to another, the return is scalene
'''
def select_triangle(edge_list):
    if(edge_list[0] == edge_list[1]):
        if(edge_list[1] == edge_list[2]):
            return "equilateral"
        else:
            return "isosceles"
    elif edge_list[1] == edge_list[2]:
        return "isosceles"
    elif edge_list[0] == edge_list[2]:
        return "isosceles"
    else:
        return "scalene"

'''
```

```
The main function simply runs the receive_inputs function
Then it uses these values as an input for the select_triangle function
The results of select_triangle function is displayed to the user
'''
def main():
    print("Waiting␣user␣to␣input␣edge␣values␣.....")
    edges=receive_inputs()
    triangle_type=select_triangle(edges)
    print(triangle_type)


main()
```

4. How can you rewrite the code if you had the requirement that the code has to be object oriented?

Solution: Three classes, each for one triangle. Their constructor checks whether it is that type of triangle.

Code is also available in Moodle as a python file (ending with OOP)

5. Write the tests that will test the code based on your analyzed requirements. *Do not* look at your code as these should be based on the analyzed requirements.

Solution:

- Test whether inputs are received by the software
- Test whether the software outputs
- Test whether three equal numbers output equilateral
- Test whether only two equal numbers output isosceles
- Test whether three unequal numbers output scalene

6. Once your tests are passed, you ship the software the customer. What happens when the customer puts the following values into the code (will be revealed during the session)? Do you have any unexpected behavior? If yes, what needs to change **first**?

Solution:

1. a=3,b=3,c=3 $\Rightarrow$ equilateral
2. a=2,b=2,c=3 $\Rightarrow$ isosceles
3. a=4,b=2,c=3 $\Rightarrow$ scalene

4. a=0,b=0,c=0 ⇒ equilateral
5. a=0,b=1,c=2 ⇒ scalene
6. a=1,b=1,c=0 ⇒ isosceles
7. a="esi",b="esi",c="esi" ⇒ equilateral
8. a=true,b=true,c=false ⇒ isosceles
9. a=[1,2,3] b=null, c=null ⇒ isosceles
10. a=-1,b=-1,c=-1 ⇒ equilateral
11. a=1,b=1,c=3 ⇒ isosceles

According to the requirements we developed a program that satisfies these requirements but of course the customer doesn't do what he said by inputting values other than numbers. So the requirement has to change by mentioning that we have to check for the input type and also range The last input values actually do not even make a triangle so we have to also add the requirement that we have to check whether the values even make a triangle at all.

Since we are using the Waterfall approach, we have to go back to the requirement phase, change our requirements and go through the whole process once again.

7. Identify the source of each unexpected behavior? (E.g. customer requirements, analyzed requirements, user, code)

Solution:

- Not checking for type ⇒ Analyzed Requirements
- Not converting into integer ⇒ Code
- Saying number instead of positive integer ⇒ Customer Requirements
- Not checking for a triangle first ⇒ Analyzed Requirements

8. What would be the new analyzed requirements?

Solution:

- The software should obtain inputs from a human agent
- The software should provide an output to a human agent
- On an input of three equal numbers, the software should output equilateral
- On an input of exactly two equal numbers, the software should output isosceles

- On an input where no number is equal to the other one, the software should output scalene.
- Do not accept numbers smaller than or equal to 0
- Do not accept non number inputs
- Check whether the input values constitutes a triangle

9. How would this development change in an agile approach?

Solution:

- Instead of testing the whole product at the end, splitting the product into different sprints. One for detecting a triangle, one for detecting an equilateral, one for detecting an isosceles.
- Each of them would have tests made for them during the sprint.
- Tests can be written before the design phase (see slide 43 of the first lecture). This would also allow the requirements to be analyzed.