Exam CS2115
Software Engineering Methods

Delft University Of Technology
April 8, 2021 - 13:30-16:30

Please, read these instructions carefully. Failure to comply with any of the instructions means invalidating your exam.

The exam consists of six open questions that cover the context of the course. Be aware that the exam questions are designed to prevent plagiarism and student collaborations using variants and by introducing subtle differences across the variants. We will also use plagiarism detection tools to identify possible plagiarized text and potential frauds.

Potential plagiarism and frauds will be reported to the Board of Examiners.

Question 1. Social media plays a critical role in our daily life, allowing people to learn new information, share ideas, interact with new people independently of their geographical distance. Social media are web-based online platforms and are very challenging to develop due to the number of users who use them daily.

For this first question, we ask you to choose one social media platform among the following ones: Facebook, Instagram, Twitter, and Youtube.

QUESTION: Given the social media you have chosen, please answer the following questions:

1. Describe 15 functional requirements (FR) the chosen platform must have
2. Describe 15 non-functional requirements (NFR) the platform must-have. For each non-functional requirement, please specify to which category it belongs.

Question 2. Consider once again the social media you have chosen for Question 1. Answer the following questions:

1. Which software architecture would you choose to implement the social media you have chosen? Motivate your answer.
2. Describe the main components you would consider for implementing the software architecture you have chosen.
3. Which quality attributes does your architecture satisfy? In answering this question, keep in mind the NFRs you have suggested in Question 1.

Question 3. Consider the following three **design patterns**:

1. Chain of Responsibility
2. Composite
3. Proxy

Choose (only) one of these design patterns and explain how this can be used for the assigned social media platform. Motivate your answer by:

A. Describe **which** problem the chosen design pattern solves for the specific platform.
B. Describe **how** the chosen design pattern solves the problem mentioned in subpoint A.

Question 4.

Consider the program in Listing 1. Complete the following exercises:

1. Create a mutant by applying one small syntactic change in the “contains” method. Please specify the line you have mutated and the exact change you have made.
2. Write a test case that kills the mutant you have created.

Listing 1

```
1.  public class Search {
2.     public int contains(int v[], int element) {
3.         if (element<0)
4.             throw new IllegalArgumentException("Element must be
5.  positive");
6.
7.         if (v.length <= 2){
8.             return this.simpleSearch(v, element);
9.         }
10.
11.        int begin = 0;
12.        int end = v.length - 1;
13.        int middle;
14.        while (begin <= end) {
15.            middle = (begin + end) / 2;
16.            if (v[middle] < element) {
17.                begin = middle + 1;
18.            } else if (v[middle] > element) {
19.                end = middle - 1;
20.            } else {
21.                return middle;
22.            }
23.        }
24.        return -1;
25.    }
26.
27.    public int simpleSearch(int v[], int element){
28.        for (int i=0; i<v.length; i++) {
29.            if (v[i] == element) {
30.                return element;
31.            }
32.        }
33.        return -1;
34.    }
35. }
```

Question 5.

Consider the program in Listing 2. Complete the following exercises:

1. Create a mutant by applying one small syntactic change to the code. Please specify the line you have mutated and the exact change you have made.
2. Write a test case that kills the mutant you have created.

Listing 1

```
1.  public class Complex {
2.     public double real, imag;
3.
4.     public Complex(double real, double imag) {
5.         this.real = real;
6.         this.imag = imag;
7.     }
8.
9.     public Complex pow(Complex p) {
10.        double a = real, b = imag, c = p.real, d = p.imag;
11.        double ns = a * a + b * b;
12.        double dArg = d / 2;
13.        double cArg = c * arg();
14.        double dDenom = Math.pow(Math.E, d * arg());
15.
16.        double newReal = Math.pow(ns, c / 2) / dDenom * (Math.cos(dArg) *
       Math.cos(cArg) * Math.log(ns) - Math.sin(dArg) * Math.sin(cArg) *
       Math.log(ns));
17.        double newImag = Math.pow(ns, c / 2) / dDenom * (Math.cos(dArg) *
       Math.sin(cArg) * Math.log(ns) + Math.sin(dArg) * Math.cos(cArg) *
       Math.log(ns));
18.        return new Complex(newReal, newImag);
       }

19.    public Complex multiply(Complex c) {
20.        double real = 0.0, imag = 0.0;
21.
22.        real += this.real * c.real;
23.        real -= this.imag * c.imag;
24.        imag += this.real * c.imag;
25.        imag += this.imag * c.real;
26.        return new Complex(real, imag);
27.    }
28.
29.    private double abs() {
30.        return Math.sqrt(real * real + imag * imag);
31.    }
32.
33.    private Complex conjugate() {
34.        return new Complex(real, -imag);
35.    }
36.
37.    public Complex divide(Complex c) {
38.        Complex result = multiply(c.conjugate());
39.        double divisor = Math.pow(c.abs(), 2);
40.        return new Complex(result.real / divisor, result.imag / divisor);
       }
41.
42.    private double angle() {
43.        return Math.atan2(imag, real);
44.    }
45.
46.    private double arg() {
47.        return angle();
48.    }
49.
```

```
50.     /**
51.      * Returns a string representation of this object with double
52.  truncation.
         */
53.     public String toString(int doubleLength) {
54.         StringBuffer temp = new StringBuffer();
55.         temp.append(trim(real, doubleLength));
56.         if(imag < 0.0) {
57.             temp.append(" - ");
58.             temp.append(trim(-imag, doubleLength));
59.             temp.append(" i");
60.         } else {
61.             temp.append(" + ");
62.             temp.append(trim(imag, doubleLength));
63.             temp.append(" i");
64.         }
65.         return temp.toString();
66.     }
67.
68.     /**
69.      * This method returns the passed double only specified to a certain
70.  number
         * of digits.
71.      */
72.     public static double trim(double d, int digits) {
73.         double scalar = Math.pow(10, digits);
74.         return ((int) (d * scalar)) / scalar;
75.     }
76. }
77.
```

Question 6. Consider the coverage matrix (rows are test cases and columns are branches) depicted in Table 1.

IMPORTANT: To avoid fraud and student cooperation during the exam, you have to adapt the coverage matrix based on your student number. Complete the coverage matrix based on the instructions below:

Mark with "x" the entries in the grey area of the table whose test case index corresponds to the last three digits of your student number. For example, if your student number is XXX521, you have to mark with "x" the entries [t5, b1], [t2, b1], and [t1, b1].

If one of the last three digits of your student number is zero you ignore that digit. For example, if your student number is XXX520, you have to mark with "x" the entries [t5, b1], and [t2, b1].

If the last three digits of your student number contain duplicates, you ignore the duplicates. For example, if your student number is XXX522, you have to mark with "x" the entries [t5, b1], and [t2, b1].

QUESTION: Please answer the following questions:

a. Which test cases are selected when using the additional greedy algorithm to reach 100% branch coverage?
b. Which test cases are selected when using the greedy algorithm to reach 100% branch coverage?

Report the test case(s) in the exact order they are selected by the additional greedy algorithm (point a) and the greedy algorithm (point b).

Table 1

	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12	b13	b14	b15	b16	b17	b18	b19	b20
t1		x	x	x						x	x									
t2					x			x							x	x				x
t3					x	x				x	x			x						
t4		x		x		x	x	x												
t5									x	x				x					x	x
t6		x	x									x	x					x		
t7											x				x	x	x			x
t8				x	x				x										x	x
t9			x	x						x	x	x								
t10					x	x								x	x			x	x	
t11							x	x			x	x								x
t12			x	x	x								x		x	x				
t13																	x	x	x	x
t14		x	x						x	x				x	x					
t15	x																x	x	x	x