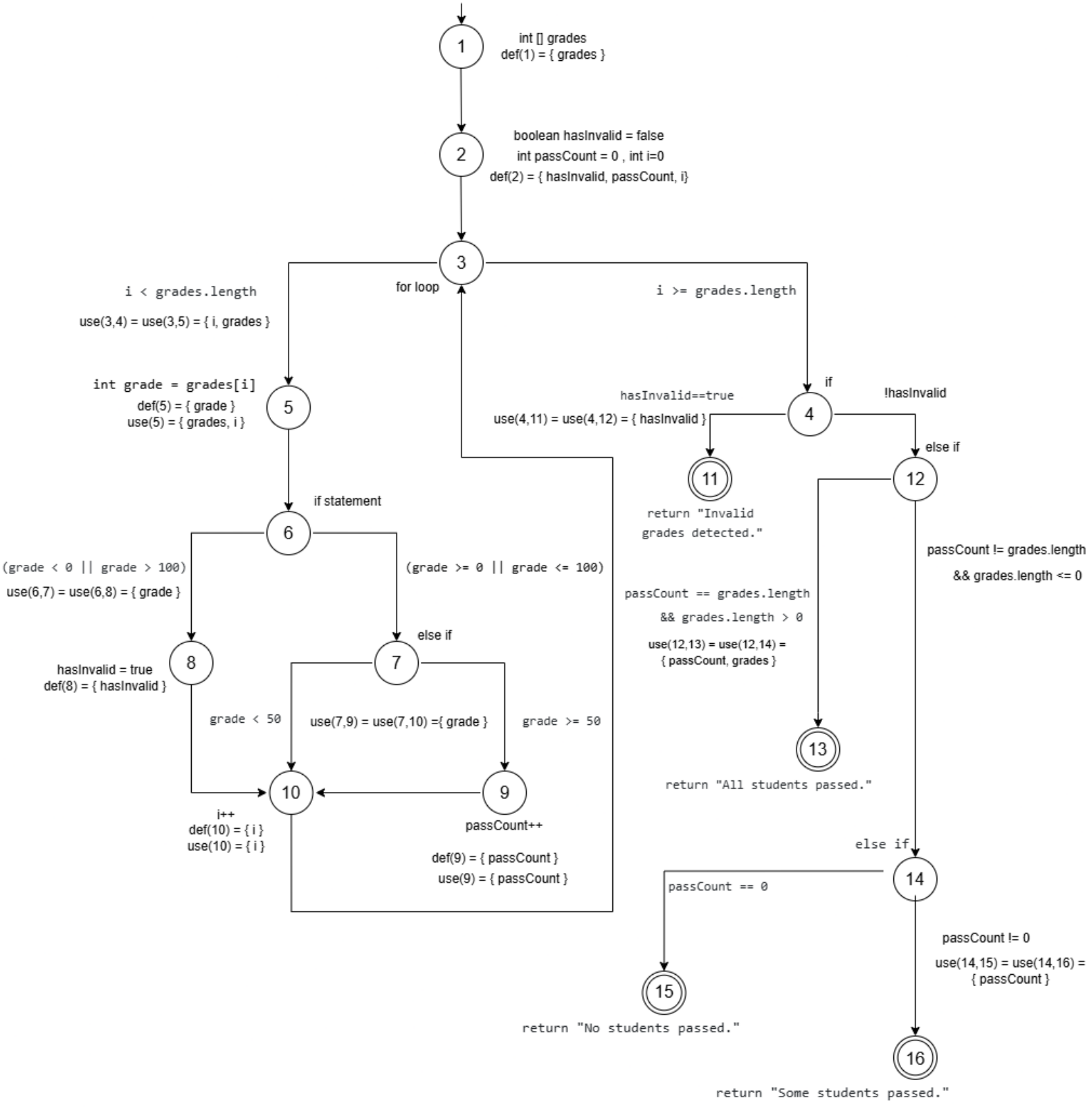
СКИТ-лаб.2

Стефани Вуксанова [221218]

Graph Coverage for Source Code

```
/**

 * Analyzes an array of student grades.
 * Returns a summary message based on:
 * - Number of passing grades (>= 50)
 * - If any grade is invalid (< 0 or > 100)
 * - If all students passed
 *
 * @param grades an array of integers representing student grades
 * @return summary message
 */
public static String analyzeGrades(int[] grades) {
    boolean hasInvalid = false;
    int passCount = 0;

    for (int i = 0; i < grades.length; i++) {
        int grade = grades[i];

        if (grade < 0 || grade > 100) {
            hasInvalid = true;
        } else if (grade >= 50) {
            passCount++;
        }
    }

    if (hasInvalid) {
        return "Invalid grades detected.";
    } else if (passCount == grades.length && grades.length > 0) {
        return "All students passed.";
    } else if (passCount == 0) {
        return "No students passed.";
    } else {
        return "Some students passed.";
    }
}
```

1. create a graph (full drawing) →



2. find all du-paths that satisfy the criteria for All-Du-Paths Coverage →

Node	def	use
1	{ grades}	
2	{ hasInvalid, passCount, i }	
3		
4		
5	{ grade }	{ grades, i }
6		
7		
8	{ hasInvalid }	
9	{ passCount }	{ passCount }
10	{ i }	{ i }
11		
12		
13		
14		
15		
16		

Edge	use
(1, 2)	
(2, 3)	
(3, 4)	{ i, grades }
(3, 5)	{ i, grades }
(4, 11)	{ hasInvalid }
(4, 12)	{ hasInvalid }
(5, 6)	
(6, 7)	{ grade }
(6, 8)	{ grade }
(7, 9)	{ grade }
(7, 10)	{ grade }
(8, 10)	
(9, 10)	
(10, 3)	
(12, 13)	{ passCount, grades }
(12, 14)	{ passCount, grades }
(14, 15)	{ passCount }
(14, 16)	{ passCount }

Variable	du-path set	du-paths
grades	du(1, grades)	[1,2,3,5] [1,2,3,4] [1,2,3,4,12,13] [1,2,3,4,12,14]
hasInvalid	du(2, hasInvalid)	[2,3,4,11] [2,3,4,12]
	du(8, hasInvalid)	[8,10,3,4,11] [8,10,3,4,12]
passCount	du(2, passCount)	[2,3,5,6,7,9] [2,3,4,12,13] [2,3,4,12,14] [2,3,4,12,14,15] [2,3,4,12,14,16]
	du(9, passCount)	[9,10,3,5,6,7,9] [9,10,3,4,12,13] [9,10,3,4,12,14] [9,10,3,4,12,14,15] [9,10,3,4,12,14,16]
i	du(2, i)	[2,3,5] [2,3,5,6,7,10] [2,3,5,6,7,9,10] [2,3,5,6,8,10] [2,3,4] [2,3,5,6,7,10,3,4] [2,3,5,6,7,9,10,3,4] [2,3,5,6,8,10,3,4]
	du(10, i)	[10,3,5] [10,3,5,6,8,10] [10,3,5,6,7,10] [10,3,5,6,9,10] [10,3,4]
grade	du(5, grade)	[5,6,7] [5,6,8] [5,6,7,9] [5,6,7,10] [5,6,7,10,3,5,6,7] [5,6,7,9,10,3,5,6,7] [5,6,7,9,10,3,5,6,8] [5,6,7,9,10,3,5,6,7,9] [5,6,7,9,10,3,5,6,7,10]

Test-paths:

- [1,2,3,5,6,7,9,10,3,5,6,7,9,10,3,4,12,13]
- [1,2,3,5,6,8,10,3,5,6,7,9,10,3,4,11]
- [1,2,3,5,6,7,9,10,3,4,12,14,16]
- [1,2,3,5,6,7,9,10,3,4,12,14,15]
- [1,2,3,5,6,7,10,3,4,12,14,16]
- [1,2,3,5,6,7,9,10,3,4,12,13]
- [1,2,3,5,6,7,10,3,4,12,13]
- [1,2,3,5,6,8,10,3,4,12,13]
- [1,2,3,5,6,7,9,10,3,4,11]
- [1,2,3,5,6,7,10,3,4,11]
- [1,2,3,5,6,8,10,3,4,11]
- [1,2,3,4,12,14,16]
- [1,2,3,4,12,14,15]
- [1,2,3,4,12,13]
- [1,2,3,4,11]

3. find the minimal test set that achieves Prime Path Coverage and create real Junit tests →

Test-paths:

- [1, 2, 3, 4, 11]
- [1, 2, 3, 4, 12, 13]
- [1, 2, 3, 4, 12, 14, 15]
- [1, 2, 3, 4, 12, 14, 16]
- [1, 2, 3, 5, 6, 7, 10, 3, 4, 11]
- [1, 2, 3, 5, 6, 7, 10, 3, 4, 12, 13]
- [1, 2, 3, 5, 6, 7, 10, 3, 4, 12, 14, 15]
- [1, 2, 3, 5, 6, 7, 10, 3, 4, 12, 14, 16]
- [1, 2, 3, 5, 6, 7, 9, 10, 3, 4, 11]
- [1, 2, 3, 5, 6, 7, 9, 10, 3, 4, 12, 13]
- [1, 2, 3, 5, 6, 7, 9, 10, 3, 4, 12, 14, 15]
- [1, 2, 3, 5, 6, 7, 9, 10, 3, 4, 12, 14, 16]
- [1, 2, 3, 5, 6, 8, 10, 3, 4, 11]
- [1, 2, 3, 5, 6, 8, 10, 3, 4, 12, 13]
- [1, 2, 3, 5, 6, 8, 10, 3, 4, 12, 14, 15]
- [1, 2, 3, 5, 6, 8, 10, 3, 4, 12, 14, 16]