Software Engineering HomeWork 5:

Aufg1: Improvements of Code Review

- In the 1st Point: Politeness and constructive/don't show off (– e.g.: l. 5 "Basic Programming 101 knowledge ;)")
- In the 2nd point: Try running the code (-e.g.: l.6 "**seem** inefficient" -> not tested and 4th point)
- In the 3rd point: Do get an understanding about the code's context (- e.g.: l.8 "it's not clear -> was the Context really looked at?)
- In the 4th point: again, testing and context of code
- In Final Comment: Don't discuss style if there are no guidelines (unclear whether there are guidelines, but does not seem to be the case)
- Do highlight the good parts (unclear whether there are good parts, but likely the case), only in 5th point ("...which is good,...")

Aufg2: Black-Box Testing

Test Cases	TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8	TC9
totalStud > 0	X	X		X	X	X	X	X	
totalStud ≤ 0			X						X
groupSize > 0	X	X	X		X			X	
groupSize <= 0				X		X	X		X
availableGroups >0	X	X	X			X	X		
availableGroups <= 0				X	X			X	X
Output = 0			X						
Output > 0	X	X			X	X			
Output exception				X			X	X	X
Input totalStud	100	Max Int	0	100	Max Int	100	100	100	0
Input groupSize	20	5	20	-10	20	0	-20	20	-10
Input availableGroups	5	1	10	-4	0	5	5	-4	-5
Expected Output	0	Max-(5*1)	0	Exception	Max Int	100	Exception	Exception	Exception
Result									

Aufgabe 3: White Box Testing

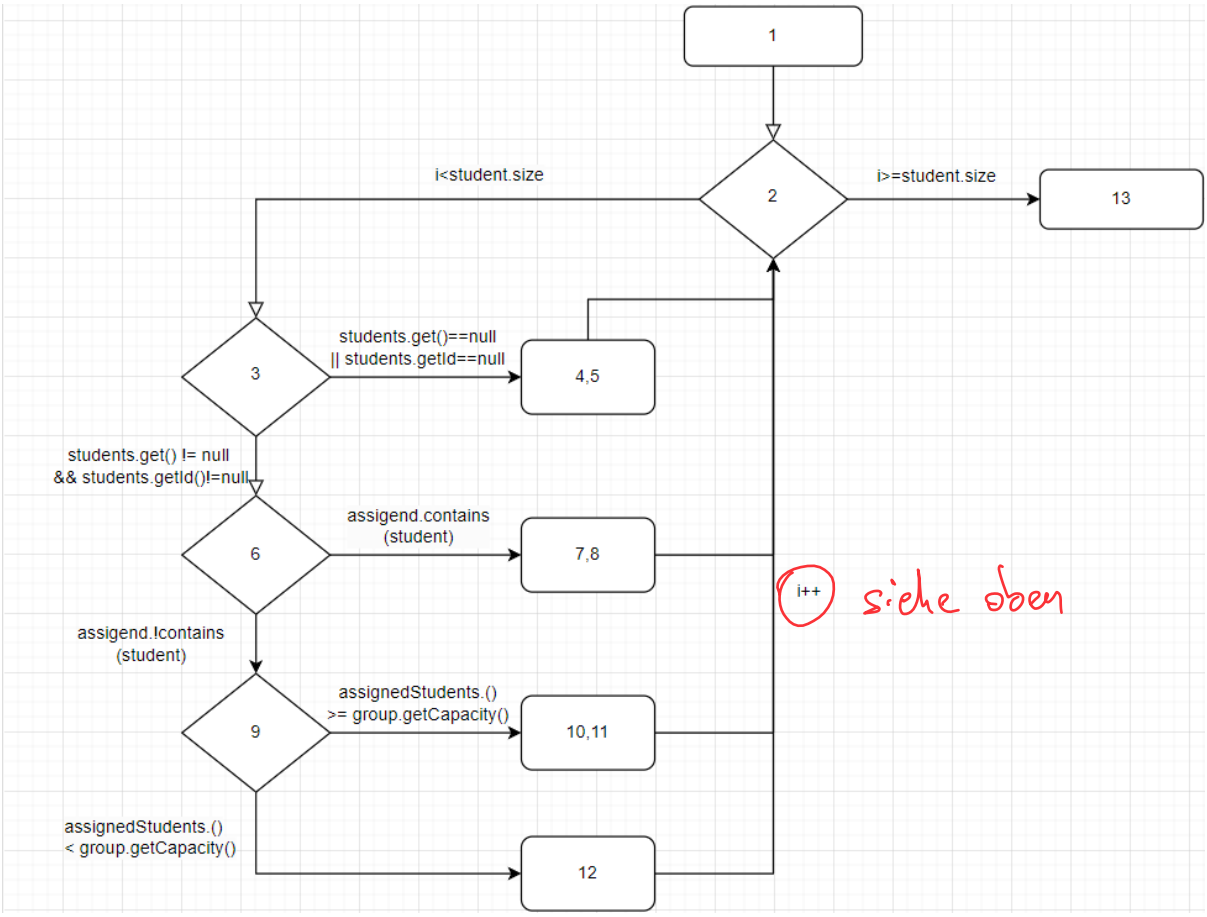
Zeilen:

```
1  List<Student> assignedStudents = new ArrayList<>();

2  for(int i = 0; i < students.size(); i++) {
3      if(students.get(i) == null
               || student.get(i).getID() == null) {
4          System.out.println("Invalid student or student ID");
5          continue;
       }
6      if(assignedStudents.contains(student)) {
7          System.out.println("Student already assigned");
8          continue;
       }
9      if(assignedStudents.size() >= group.getCapacity()) {
10         System.out.println("Group is full");
11         continue;
       }

       // All checks passed, add student to group
12     assignedStudents.add(student);
   }

13 return assignedStudents;
}
```

Das sind 3 einzelne Statements, die zu unterschiedlichen Zeitpunkten an die Reihe kommen. -0,5p

Dadurch stimmt auch der Nenner bei der Statement Coverage nicht.

Control Flow Graph:



Coverage:

TestInvalidStudentID:

Statement Coverage: 6/13 = 0,4615 ==> 46,15%

Branch Coverage: 1/8 = 0,125 ==> 12,5%

Condition Coverage: 1/10 = 0,10 ==> 10%

Path Coverage: 1/5 = 0,2 ==> 20%

TestSuccessfulAssignment:

Statement Coverage: 7/13 = 0,5385 ==> 53,85%

Branch Coverage: 4/8 = 0,5 ==> 50%

Condition Coverage: 4/10 = 0,40 ==> 40%

Path Coverage: 1/5 = 0,20 ==> 20%

Overall:

Statement Coverage: 11/13 = 84,6%

Branch Coverage: 5/8 = 62,5% -1p

Condition Coverage: 5/10 = 50% -1,5p

Path Coverage: 2/5 = 40% ✓

Hier fehlt ein kurzes Statement, dass die Path Coverage aufgrund der Loops eigentlich nicht berechnet werden kann, wir aber alternativ z.B das Loop Adequacy Criterion verwenden könnten -1p