

# Assignment 3

## 2DV600 -Assignment 3: Graphs and Algorithms

This assignment consists of three parts.

- In the first part, you should implement a directed graph according to a given set of interfaces.
- In the second part, you should implement a few simple algorithms.
- Finally, in the third part, you should present (and motivate) a time-complexity estimate for a few of your implemented algorithms.

**Submission:** Following two files are required for submission:

1. A **zip file** archiving all your .java (source) files.
2. A **report** document in doc, docx or pdf format. The report must contain the Java source code for all your solutions and other details required for the solution. Images are not allowed to show the code. The report must be submitted as a **stand-alone file, i.e., it must not be compressed as zip or any other file format.**
3. Separate submission of both the **report** and the **zip file** is mandatory to pass the assignment, **otherwise, a submission will get a grade "F".**
4. Submissions through Email or any way other than Moodle are not accepted.

**Your solutions should be your own!** You are not allowed to copy from other students, books articles, blogs, wikis, or any other source! Each submission will pass through a plagiarism/clone detection system before correction. If plagiarism is detected, the assignment is failed and a formal investigation will be initiated.

**Grading:** Your submission will receive a grade from A to F where F is Failed. You are allowed to improve your work after the initial submission before the deadline. Don't forget to submit! The grade is final, i.e., you will not get an opportunity to improve after grading.

### Part 1: Graph Implementations

You should provide one implementation of a graph that implements interface [graphs.DirectedGraph](#) that uses the node class [graphs.Node](#). The Java code for these and other interfaces and classes used in this assignment can be found [here](#). Read the instructions in the javadoc files carefully before you start.

**Important 1:** To simplify the testing, later on, you must follow a given name schema.

Assume that your LNU user-name is X (e.g. na222fx) then:

- All your code should be in a package named X.
- The graph class (one that you implement) should be named X.MyGraph and
- the node class should be named X.MyNode

Furthermore, the graph class must contain a default constructor `MyGraph()` generating an empty graph.

**Important 2:** We have put together a JUnit test case to help you verify the correctness of your graph implementation.

- The graph JUnit test case `TestDirectedGraph` can be found in the package **graphs.test**
- Passing this JUnit test is a minimum criterion to pass this assignment. (That is if your graph implementation not passes this test you will receive a grade F and be asked to hand in a new improved version after the course has been completed.)

## Part 2: Simple Algorithms

You should provide a set of algorithm implementations. These algorithms are specified by a set of interfaces and abstract classes. Your task is to implement:

- A class `X.MyGML` that extends the abstract class [graphs.GML](#)
  - A GML mark-up example: [GML in Wikipedia](#)
  - More information about [GML](#)
  - More information about [Yed](#)

Yed is a tool to download and install. Yed is also available online as a web application. We recommend you download the tool since the webapp doesn't handle the GML format that well. (Alternatively, generate GraphML and use the webapp.)

- A class `X.MyDFS` that implements [graphs.DFS](#)
- A class `X.MyBFS` that implements [graphs.BFS](#)
- A class `X.MyTransitiveClosure` that implements [graphs.TransitiveClosure](#)
- A class `X.MyConnectedComponents` that implements [graphs.ConnectedComponents](#)

These algorithms must not depend on any special features in your graph implementation (`MyGraph`, `MyNode`). They should work on any implementation of the `DirectedGraph` and `Node` interfaces.

Your implementation shouldn't be too slow. You can find a benchmark for testing the performance of your graph in the **graphs.benchmark** package. It should take a reasonable time for the benchmark to run on your implementation. More information about these algorithms is presented during the graph lecture.

**Important:** We have put together a JUnit test case to help you verify the correctness of your algorithms.

- The algorithm JUnit test cases TestDFS and TestAlgorithms can be found in the package graphs.test
- Once again, passing these JUnit tests is a minimum criterion to pass this assignment.

## Part 3: Time complexity Estimates

This is a theoretical exercise. You should calculate the time complexity  $O(\dots)$  for *your* implementations of the following algorithms:

- Depth-First Search as implemented in X.MyDFS.dfs(DirectedGraph<E> graph)
- Breadth-First Search as implemented in X.MyBFS.bfs(DirectedGraph<E> graph)
- Transitive Closure as implemented in X.MyTransitiveClosure.computeClosure(DirectedGraph<E> graph)
- Connected Components as implemented in X.MyConnectedComponents.computeComponents(DirectedGraph<E> graph)


Each of the calculations should come with a brief description/motivation.

**Notice:** Make sure that you give a proper reference in your report if you have taken your ideas from a book, article, or some resource found on the Internet. This holds for both the time complexity estimates and the algorithm implementations.

## Inlämningsstatus

Försök nummer	Detta är försök 1.
Inlämningsstatus	Inlämnad för betygsättning
Betygsättningsstatus	Betygsatt
Stoppdatum/tid	måndag, 25 oktober 2021, 23:59
Återstående tid	Uppgift lämnades in 4 dagar 6 timmar tidigt
Senast ändrad	torsdag, 21 oktober 2021, 17:31
Filinlämningar	<div>  2dv600_assignment_3.pdf Opt-out Ouriginal21 oktober 2021, 17:30            rk222rd_assign3.zip ⚠️ 17 oktober 2021, 14:18         </div>
Inlämningskommentarer	<div>  Kommentarer (0)         </div>

# Återkoppling

Betyg	A		
Betygsatt den	torsdag, 18 november 2021, 08:55		
Betygsatt av	Charilaos Skandylas		
Återkopplingsfiler	 <a href="#">Feedback.pdf</a>	18 november 2021, 08:55	



FÖREGÅENDE AKTIVITET  
Assignment 2

NÄSTA AKTIVITET  
Assignment 4



## Lnu

[Lnu.se/Student](https://lnu.se/Student)

[Hantering av personuppgifter](#)

[Moodle App](#)

## Hjälp

[Hjälp](#)

[IT-support](#)

[Moodle tillgänglighet](#)

## Länkar

[LnuPlay](#)

[Zoom login](#)

[Moodle Academy](#)

## Universitetsbiblioteket

[Introduktion till UB](#)

[UB:s handledningsyta](#)

[Artiklar och databaser](#)

 Moodle Docs för den här sidan.