Lab 1, Artificial Intelligence, spring 2017

Table of Contents

Introduction

This is the first lab of a series of labs that will result to a program in which you can play and also automatic solve an n-puzzle, where n can be 3, 8 or 15. You have an example of such a puzzle at http://mypuzzle.org/sliding. This is a classical problem used in courses about AI. In your textbook an 8-puzzle is used as an example. Use the textbook for support and help in your work.

3x3 8 Puzzle
8x8 15 Puzzle
15x15 24 Puzzle

You have to do your coding in Java and with help of a project in NetBeans IDE. You are not allowed to use any third-party software libraries; you are limited to use the Java SE.

Deadline last Lab 09.03.17 Code + Report

You shall not hand in the parts of your solution that will be the result from each separate lab. You only have to hand in your final solution that you should have after the last lab. The hand in shall consist of the project directory that is created in NetBeans plus a short report where you explain your solution.

Although there are many solutions on the Internet you have to construct your own program that consists solely of your own written code. You may not copy someone else's solution or parts of it and you may not allow someone else to copy your solution or parts of it. The fact that you send in a solution also confirms that you have read and understand the rules that apply to cheating and plagiarism, see http://www.du.se/en/Library/Academic-Writing/Copyright-and-Plagiarism/ and its sub sites. Your solution will be checked against other solutions with help of MOSS. MOSS is a system specially designed for checking program code for plagiarism. You can read more about MOSS on the following website, http://theory.stanford.edu/~aiken/moss/ .

The application

In this lab, you should start by designing and building the first parts of your program. When you finish this lab you should have a program in which you can manually solve an n-puzzle. In the following labs you will add artificial intelligence so that the program itself can work out a solution to the n-puzzle.

When you finish this lab you shall have a program that

- Execute in the command prompt
- Have a menu-system with at least following options:
  - Quit the program.
  - Create a new n-puzzle of a selectable size of n.
  - Mix the tiles by automatic move around them.
  - Manually solve the puzzle by move tile after tile.
- The program shall consist of at least two classes, one class that is responsibility for the menu system and one class for the puzzle.

You must be aware that when the program mixes the tiles, it must be done in the same way as when you manually move tile after tile, otherwise the puzzle may fall into a state that is unsolvable. Don't put lot of work in to viewing the puzzle in an all too neat way. In the coming labs you will print out the puzzle repeatedly in order to follow the changes the program does when it searches a solution.

First aid!

If you have troubles to find a nice design of your application, maybe this section can be helpful.

Figure 1 shows a class-diagram over simple solution of the application you are going to construct.

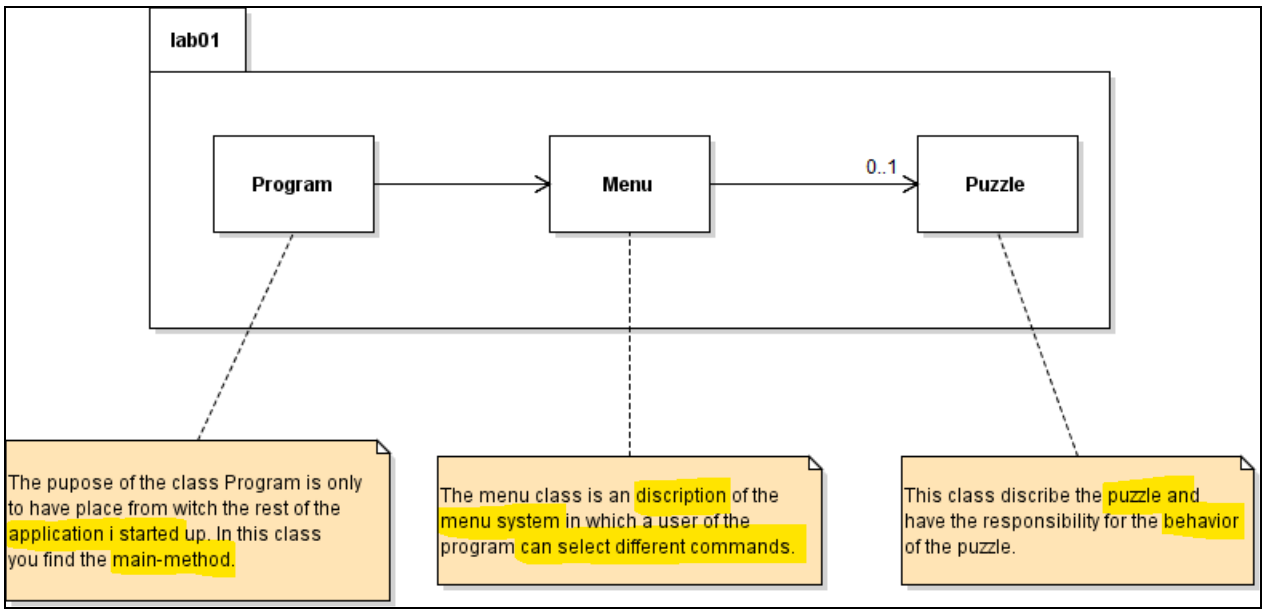

Figure 1: A class diagram that show a simple design of the application.

As you can see in figure 1 the menu has knowledge about the puzzle. That is important because the menu has to evaluate the commands that are selected with help of the puzzle. Be careful to not mix up the responsibilities of the two classes. The class Menu has the responsibility of the menus functionality and the class Puzzle has the responsibility of the things that should be possible to do with a puzzle.

You will find a proposal of a code skeleton for the above-described design in the file archive lab01.zip. If you want, you can download this code skeleton and continue to develop it to a solution for lab 1. The content of the file lab01.zip is a project directory that you can open up and continue to work with in NetBeans.