Informatics 1: Object Oriented Programming

Tutorial 02

Week 3: 28/01 - 01/02

Volker Seeker (`volker.seeker@ed.ac.uk`)
Naums Mogers (`naums.mogers@ed.ac.uk`)

1 Introduction

Most programming tasks you will face after graduating will be related to bigger projects rather than small exercises aimed at practicing a single aspect. Therefore, the INF1OP tutorial exercises will lead you through a bigger project where you will build a Sudoku[1]game application from a simple command line interface up to a proper graphical user interface (GUI). During this process you will make use of the object oriented programming techniques you will learn throughout the course by restructuring your application and adding more and more features to it.

This second tutorial sheet of the INF1OP course will give you a gentle introduction into the tutorial project you will develop during this semester. Most techniques you will need for these initial exercises you should already know from the previous semester INF1A branch or learn about during this week's lectures and Lab exercises.

Techniques You will need to apply your knowledge of: **types**, **variables**, **loops**, **conditionals**, simple **static functions** and **arrays**. Additionally, you will have to work with **console input** and **output** to read data from the user and present results.

Eclipse IDE and JUnit Unlike the INF1A branch, the INF1OP course works with the ECLIPSE IDE which you will start using this week. In the first tutorial you saw examples on how to use JUnit to test semantic correctness of your program. For the lab exercises full test coverage is provided and you can check your solutions for correctness. For the exam, this will not be the case. You will only get some basic tests to check the function headers of your implementation. Hence, basic JUnit tests will be provided for every tutorial to give you a chance to practice this.

After a short introduction into the given template, you will be presented with three exercise tasks:

1. **Printing the Game Grid** Print all numbers of the Sudoku in a grid shape to the console output.
2. **Game Loop** Present a menu within the game's main loop showing a print and an exit option.
3. **Changing Game Fields** Add two menu entries for setting and deleting fields in the Sudoku.

[1]If you are unfamiliar with Sudoku, please have a look at this Wikipedia article `https://en.wikipedia.org/wiki/Sudoku` and play a few games at `https://sudoku.game/`.

You can work on this in groups but make sure you do your own programming work and not just watch others do it for you. After this course you will likely be on your own for programming tasks and will need the cognitive muscle you will gain by practicing now.

2 Sudoku Template

More often than not you will work on an already existing code base rather than starting from zero. You usually do not need to fully understand all of what is given to you. It is important to develop an eye for the information you need and the code you have to understand in order to accomplish your task. Object oriented programming helps you with this by hiding away unnecessary complexity and presenting clean and (hopefully) well documented interfaces. You will learn more about that in later lectures.

The template file you are given for this task `Sudoku01.java` has a `main` entry function which you need to extend and three helper functions for you to be used.

The main function As introduced during the lecture and in the lab exercises, the `main` function is the entry point into your application. Your application does not yet read command line arguments and simply has a *hard-coded* version of a single Sudoku game grid, called `grid`.

The Sudoku number data is held in a variable of the type `int`. But it is not just a simple integer number, in fact, it is a two-dimensional array or matrix of integers. You will learn all about arrays in week two. To give you some support for the initial tutorial, here are the basics you will need for the coming exercises:

Reading You can read the contents of an array (or multidimensional array) by indexing the field you want to read. For that purpose use the instruction `grid[y][x]` where `y` and `x` are integer numbers of the required x and y coordinates. This will return the integer value at position `(x,y)` of the Sudoku field. The top left corner of the Sudoku grid has the coordinates `(0,0)`.

If you put the following two lines into your main function after the grid has been declared and initialised, the number 2 will be printed to the command line:

```java
int result = grid[3][1];
System.out.println(result);
```

Writing You can write a new integer into the grid by using the same instruction but assigning a number rather than reading one. The following line will change the number at position `(0,0)` to 5:

```java
grid[0][0] = 5;
```

Helper Functions The three helper functions `printMenu`, `parseInput` and `requestInt` are documented using JAVADOC style comments. The comments should contain all the information you need for you to understand how they work. If not, simply try them out in the main function and see what happens.

Basic JUnit Tests While implementing your solution, make sure you are regularly running and passing the provided basic unit tests. In this initial exercise, the provided test checks if you have the correct function header for the `printGrid` helper function you need to implement later. Initially, this test will likely fail since you might not have written the actual code yet.

3 Exercises

Task 1 - Printing the Sudoku Grid ◁ Task

In this task you need to write another helper function which prints the current state of the grid, i.e. all numbers in the grid to the command line. Specifically, write a `public static` function `printGrid`, taking an `int[][] grid` argument and returning nothing, i.e. `void`. This function should iterate through all indices of the game grid and print the corresponding integer values to the command line in the following format:

```
9 4 0  1 0 2  0 5 8
6 0 0  0 5 0  0 0 4
0 0 2  4 0 3  1 0 0

0 2 0  0 0 0  0 6 0
5 0 8  0 2 0  4 0 1
0 6 0  0 0 0  0 8 0

0 0 1  6 0 8  7 0 0
7 0 0  0 4 0  0 0 3
4 3 0  5 0 9  0 1 2
```

Once complete, call this function in your `main` function for the initialised grid variable.

HINT Consider how you can make use of `System.out.print` in contrast to `System.out.println` which does not automatically add a newline after it printed a message.

Task 2 - Game Loop ◁ Task

Nearly every game, and many other applications for that matter, have a main loop which runs continuously checking user or other input until a condition for exiting the application is met. They then leave this loop and exit the `main` function.

In this task you need to implement a main loop which presents the user with a menu for selecting the next action to be taken. Use the `printMenu` function to display a menu and the `parseInput` function to request a menu entry to be executed next.

Once you have received user input, you need to handle the following three cases:

1. Menu entry one has been selected. In this case you should print the game grid again.
2. Menu entry two has been selected. In this case you should exit the program.
3. Invalid user input has been provided. In this case you should display an error message to the user.

After a selection has been handled, the menu should be printed again for the next action to be taken (unless the user chose to quit the program).

HINT If you are stuck, peek at the `requestInt` function to get an idea of how this could be done.

Task 3 - Changing Sudoku Fields ◁ Task

In this last task you need to extend the menu by two entries: One for setting a Sudoku field to a new number and one for clearing a field. The resulting menu should look somewhat like this:

```
1. Set field
2. Clear field
3. Print game
4. Exit
```

For both actions you can make use of the `requestInt` function to ask the user to input an `x` and a `y`-coordinate. Make sure the proper minimum and maximum values are considered. For inserting a new number, you can use this method as well.

After you have altered the Sudoku field successfully, you should print the grid automatically to display the change to the player using the functions you created above.

Clearing a field means setting its current value to zero.

4 Optional Extra Tasks

If you have finished all of the above tasks and look to do more, please consider the following suggestions or come up with your own ideas:

- Have a look at JAVA's `switch` statement and use it for handling user input in the menu section. You can find a description here: https://www.tutorialspoint.com/java/switch_statement_in_java.htm
- Modify your `printGrid` function to display line and row numbers along the side of the printed grid. Also, add functionality to print borders around numbers.