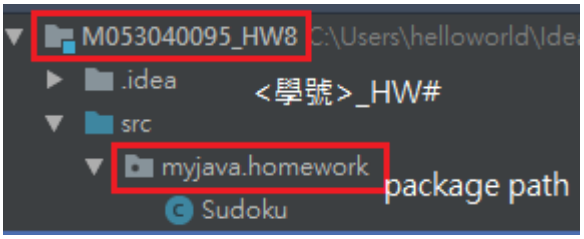
【2018 JAVA 物件導向程式設計 Homework 8】

- 注意事項

1. 請使用 JAVA 語言，配合 IntelliJ IDEA 寫本次作業並進行測試，並安裝、使用 JAVA SE Development Kit (JDK) 8 函式庫。
2. 請依據作業規定設定 IntelliJ IDEA 專案名稱與 package name，若未依照規定將根據狀況扣分。
3. 嚴禁抄襲其他同學作業，參與者 (抄襲與被抄襲) 該次作業均以零分計算。
4. 請對你的程式碼有深入瞭解，demo 時助教會問。
5. 對題目有問題可以寄信問助教群(java_ta@net.nsysu.edu.tw)或是到實驗室(EC5018)詢問，但不幫忙 debug。
6. 逾期以零分計算，不接受補交，有任何因素導致無法如期繳交，請事先告知；Demo 時間會另外通知。

- 作業規定與上傳

1. IntelliJ IDEA 專案名稱：<學號>_HW8
2. Package path：myjava.homework
3. 作業請繳交專案之 tar 或 zip archive 並上傳至網路大學。請於 2018 年 5 月 9 日 (週三) 23:59 前上傳完畢，逾期以零分計算，不接受補交，再次強調，有任何因素導致無法如期繳交，請事先告知，Demo 時間會另外通知。
4. Example of package explorer (請根據作業規定修改)：



- Homework

Sudoku is a popular logic puzzle that use a 9 by 9 array of squares that are organized into 3 by 3 subarray. The puzzle solver must fill in the squares with the digits 1 to 9 such that no digit is repeated in any row, any column, or any of the nine 3 by 3 subgroups of squares. Initially, some squares are filled in already and cannot be changed. For example, the following might be a starting configuration for Sudoku puzzle:

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

Create a class `SudokuPuzzle` that has the attributes

- `board` – a 9 by 9 array of integers that represents the current state of the puzzle, where 0 indicates a blank square
- `start` – a 9 by 9 array of boolean values that indicates which squares in `board` are given values that cannot be change

and the following methods:

- `SudokuPuzzle` - a constructor that creates an empty puzzle
- `toString` – return a string representation of the puzzle that can be printed
- `addInitial (row, col, value)` – sets the given square to the given value as an initial value that cannot be changed by the puzzle solver
- `addGuess (row, col, value)` - sets the given square to the given value; the value can be changed later by another call to `addGuess`
- `checkPuzzle` – returns true if the value in the puzzle do not violate the restrictions
- `getValueIn (row, col)` – returns the value in the given square
- `getAllowedValue (row, col)` – returns a one-dimensional array of nine booleans, each of which corresponds to a digit and is true if the digit can be placed in the given square without violating the restrictions
- `isFull` – returns true if every square has a value
- `reset` – changes all of the nonpermanent square back to blanks (0s)

Write a `main` method in the class `Sudoku` that creates a `SudokuPuzzle` object and sets its initial configuration. Then use a loop to allow someone to play Sudoku. Display the current configuration and ask for row, column, and value. Update the game board and display it again. If the configuration does not satisfy the restrictions, let the user know. Indicate when the puzzle has been solved currently. In that case, both `checkPuzzle` and `isFull` would return true. You should also allow options for resetting the puzzle and displaying the values that can be placed in a given square.