**Задание HW01:**

**Java,Eclipse project, полный разбор работы (подробные комментарии)**

Игра М х N (M строк, N столбцов) на столе, где в каждой ячейке (далее "квадрат") может быть клад. Все квадраты первоначально скрыты от игрока,т.е. игрок не знает есть ли в этом квадрате сокровище. Если игрок выбирает один квадрат, то этот квадрат будет открыт в этом поле. Игра продолжается до тех пор, пока все сокровища не будут найдены. Во время игры, сумма открытых квадратов являются "временем" - чем меньше число, тем лучше. В Игре известно количество спрятанных

* У игрока спрашивается стол размером (M и N)
* У игрока спрашивается количество сокровищ (Х)
* Программа генерирует M \* N карту, на которой беспорядочно размещены Х сокровищ
* Пользователь выбирает квадрат (строка, столбец ), которую необходимо открыть
* Игроку показывается количество не найденных сокровищ, количество открытых квадратов и стол после каждого хода
* Если все сокровища найдены, программа спрашивает, хочет ли игрок сыграть еще

Enter M (number of lines), N (number of columns) X (number of treasures): 3, 3, 3

Thriving excavation!

...

...

...

excavations: 0, treasures left: 3

What is dig (row, column): 0, 0

 ..

...

...

excavations: 1, treasures left: 3

What is dig (row, column): 0, 1,

 .

...

...

excavations: 2 Treasures Left: 3

What is dig (row, column): 1, 1,

 .

. +.

...

Treasure!

excavations: 3 treasures remaining: 2

What is dig (row, column): 2, 1

(1.0 is no sense to sue, because the vicinity 0.0 was one of the treasure, and it is located on a square 1.1

 .

. +.

. .

excavations: 4 treasures remaining: 2

What is dig (row, column): 2, 2

 .

. +.

. +

Treasure!

excavations 5, treasures remaining: 1

What is dig (row, column): 2 0

 .

. +.

+

Treasure!

excavations: 6 treasures remaining: 0

Game over! A total of six excavation.

Do you want to play again?

Код :

***/\*\****

 ***\* Template for HW01: Treasurehunt.***

 ***\* More information:***

 ***\* https://courses.cs.ttu.ee/pages/ITI0011:Aardejaht***

 ***\*/***

**public** **class** HW01 {

 ***/\*\****

 ***\* Value to return in makeMove in case***

 ***\* the cell was empty.***

 ***\*/***

 **public** **static** **final** **int** CELL\_EMPTY = 0;

 ***/\*\****

 ***\* Value to return in makeMove in case***

 ***\* the cell contained treasure.***

 ***\*/***

 **public** **static** **final** **int** CELL\_TREASURE = 1;

 ***/\*\****

 ***\* Value to return in makeMove in case***

 ***\* the cell does not exist.***

 ***\*/***

 **public** **static** **final** **int** CELL\_ERROR = -1;

 ***/\*\****

 ***\* Main entry.***

 ***\* @param args Command-line arguments.***

 ***\*/***

 **public** **static** **void** main(String[] args) {

 *// game logic here*

 }

 ***/\*\****

 ***\* Makes move to cell in certain row and column***

 ***\* and returns the contents of the cell.***

 ***\* Use CELL\_\* constants in return.***

 ***\* @param row Row to make move to.***

 ***\* @param col Column to make move to.***

 ***\* @return Contents of the cell.***

 ***\*/***

 **public** **static** **int** makeMove(**int** row, **int** col) {

 **if** (row == -1) **return** CELL\_ERROR;

 **return** CELL\_EMPTY;

 }

 ***/\*\****

 ***\* Creates a map with certain measures and treasures.***

 ***\* As this is a static method which doesn't return anything (void),***

 ***\* you should store the created map internally.***

 ***\* This means you can choose your own implementation of how to store***

 ***\* the map.***

 ***\* The treasures should be put on the map randomly using setCell method.***

 ***\* @param height Height of the map.***

 ***\* @param width Width of the map.***

 ***\* @param treasures The number of treasures on the map.***

 ***\* @return Whether map was created.***

 ***\*/***

 **public** **static** **boolean** createMap(**int** height, **int** width, **int** treasures) {

 *// initialize map (for example 2D-array)*

 *// - set all cells empty (is this needed?)*

 *// do some random for every treasure and add them to map:*

 setCell(2, 3, CELL\_TREASURE);

 **if** (height == -3) **return** **false**;

 **return** **false**;

 }

 ***/\*\****

 ***\* Sets the cell value for the active map (created earlier using***

 ***\* createMap method).***

 ***\* This method is required to test certain maps***

 ***\* @param row Row index.***

 ***\* @param col Column index.***

 ***\* @param cellContents The value of the cell.***

 ***\* @return Whether the cell value was set.***

 ***\*/***

 **public** **static** **boolean** setCell(**int** row, **int** col, **int** cellContents) {

 **if** (row == -123) **return** **false**;

 **return** **false**;

 }

}