Assignment Due Date: 3 January, 11:00am

Instructions to Students

1. This assignment should be done by each individual student.
2. You are required to apply Object-Oriented Programming Techniques in this assignment.
3. Plagiarism will be treated seriously. All assignments that have been found involved wholly or partly in plagiarism (no matter these assignments are from the original authors or from the plagiarists) will score **Zero** marks. You will be asked a random question on your program to verify that your submission is your own work.
4. Your program must use JDK 1.8 or above to develop.
5. Your program must be structured and well documented. The first few lines in the source file must be a comment stating the name of the source file, and the name, course, class, student number as well as the main methods of the program. Marks will be deducted if such comments are not included.
6. You are required to submit your program code (java files), game file(s), documentation and checklist to moodle before deadline. Late submission is **NOT** accepted.
7. Weight of this Assignment: **50% of module (100% of EA)**

Set All to Zero

Problem Specification

In this assignment, you are required to implement a board game. The game board is a 2-D matrix of integer loaded from file. The goal of the game is to set all the integers in the matrix to zero by applying different items to the game board, different items have different properties and cost (step count). There is no limit in steps taken, but if the item set any value in the matrix to negative number, the game will over.

Game Flow

A game file is required to specify in the command line to start the game, following is an example:

```
> java Game game01.txt
```

The file described the height, width, content and target step of the game, following is an example:

game01.txt	remark
5	// height of the matrix
5	// width of the matrix
1 1 1 1 0	// line 0 of the matrix
1 0 0 1 0	// line 1 of the matrix
1 0 2 4 2	// line 2 of the matrix
1 1 4 1 2	// line 3 of the matrix
0 0 2 2 2	// line 4 of the matrix
4	// target step

After the game is initialized, it starts to ask user to apply different items to the game board. Item can be described by a 2-D matrix of integer, the *default setting* of an Item will print the number stored on screen, except 0 will be replaced by a space. For example,

data stored in item	***the way to display item***
1 1 1 1	1 1 1 1
1 0 0 1	1 1
1 0 0 1	1 1
1 1 1 1	1 1 1 1

but it can be overridden according to the needs of different item type (will be explained in detail later). There are two types of item, they are Bomb and SuperBomb,

Bomb shared the same display property of Item, it costs **1 step** to apply Bomb on game board. When a Bomb is applied to the game board, the value in the game board will be reduced by the corresponding value in the Bomb. For example,

```
If             is applied to 2,1 of the game board
    1 1 1                                        2 2 2 2 2
    1 2 1                                        2 2 2 2 2
    1 1 1                                        2 2 2 2 2
                                                 2 2 2 2 2
                                                 2 2 2 2 2

i.e. row is 2 and col is 1, the game board will become

2 2 2 2 2
2 2 2 2 2
2 1 1 1 2
2 1 0 1 2
2 1 1 1 2
```

SuperBomb behave differently, it costs **2 steps** to apply SuperBomb on game board. When a SuperBomb is applied to the game board, the value in the game board will be divided by the corresponding value in the SuperBomb. For example,

```
If             is applied to 2,1 of the game board
    2 2 2                                        3 2 3 4 3
    2 1 2                                        3 2 3 4 3
    2 2 2                                        3 1 2 3 3
                                                 3 3 4 4 3
                                                 3 5 6 7 3

i.e. row is 2 and col is 1, the game board will become

3 2 3 4 3
3 2 3 4 3
3 0 1 1 3
3 1 4 2 3
3 2 3 3 3
```

In this version, SuperBomb only stored 2 values, either 2 or 1. The display property of SuperBomb is different from the Item's default. 2 will be displayed as * and 1 will be displayed as space. For example, the SuperBomb above will be displayed as follow:

data stored in item	***the way to display item***

`2 2 2`
`2 1 2`
`2 2 2`
	`* * *`
	`*   *`
	`* * *`

Player is asked to apply an item to the game board repeatedly until the player win the game or the game is over. Player wins the game when all the values in the game board are zero and the game is over when there is a negative number in the game board.

There are two conditions when player wins:

Condition 1: Step count <= target step count
Condition 2: Step count > target step count

For Condition 1, it displays **`"Congratulation, well done!"`** and the game will be ended. For Condition 2, it displays **`"Congratulation, you have finished the game...but you can do even better!"`** and player will be asked if he/she want to play again.

When the game is over (player lose), it displays **`"Game Over!"`** and also the game board. Note that, negative value will be replaced as "X" and player will be asked if he/she want to play again too.

If the player choose to play again, the game including the step count will be reset and start again. Otherwise the game will be ended. Note that, **`"Bye Bye!"`** will be printed when the game ends.

To simplify the game, the item list is fixed in this version, there are 6 Bombs and 3 SuperBombs, their matrix and id are stated as follow:

```
Items
0        1      2      3      4    5  6      7      8
-----------------------------------------------------------
1 1 1 1  1 1 1  1   1    1    1 1  1  * * *  * * *    *
1     1  1   1    1    1 2 1  1 1     *   *  * * *  * * *
1     1  1 1 1  1   1    1            * * *  * * *    *
1 1 1 1
```

Player can use the items as many time as they wanted.

Following are some sample output without error and exception handling.

Sample Output 1:

```
E:\ITP4713_IOOP\Assignment\v2>java Game game01.txt


Items
0        1      2      3      4    5  6      7      8
-----------------------------------------------------------
1 1 1 1  1 1 1  1   1    1    1 1  1  * * *  * * *    *
1     1  1   1    1    1 2 1  1 1     *   *  * * *  * * *
1     1  1 1 1  1   1    1            * * *  * * *    *
1 1 1 1

Current Count / Target Step Count: 0/4

  | 0 1 2 3 4
--+-----------
0 | 1 1 1 1 0
1 | 1 0 0 1 0
2 | 1 0 2 4 2
3 | 1 1 4 1 2
4 | 0 0 2 2 2

Please choose an item: 6
You have chosen:
* * *
*   *
* * *

Where do you want to put the item (row no)? 2
Where do you want to put the item (col no)? 2


Items
0        1      2      3      4    5  6      7      8
-----------------------------------------------------------
1 1 1 1  1 1 1  1   1    1    1 1  1  * * *  * * *    *
1     1  1   1    1    1 2 1  1 1     *   *  * * *  * * *
1     1  1 1 1  1   1    1            * * *  * * *    *
1 1 1 1

Current Count / Target Step Count: 2/4

  | 0 1 2 3 4
--+-----------
0 | 1 1 1 1 0
1 | 1 0 0 1 0
2 | 1 0 1 2 1
3 | 1 1 2 1 1
4 | 0 0 1 1 1

Please choose an item: 1
You have chosen:
1 1 1
1   1
1 1 1
```

```
Where do you want to put the item (row no)? 2
Where do you want to put the item (col no)? 2


Items
0        1      2      3      4    5  6      7      8
-----------------------------------------------------------
1 1 1 1  1 1 1  1   1    1    1 1  1  * * *  * * *    *
1     1  1   1    1    1 2 1  1 1     *   *  * * *  * * *
1     1  1 1 1  1   1    1            * * *  * * *    *
1 1 1 1

Current Count / Target Step Count: 3/4

  | 0 1 2 3 4
--+-----------
0 | 1 1 1 1 0
1 | 1 0 0 1 0
2 | 1 0 0 1 0
3 | 1 1 1 1 0
4 | 0 0 0 0 0

Please choose an item: 0
You have chosen:
1 1 1 1
1     1
1     1
1 1 1 1

Where do you want to put the item (row no)? 0
Where do you want to put the item (col no)? 0

Current Count / Target Step Count: 4/4

  | 0 1 2 3 4
--+-----------
0 | 0 0 0 0 0
1 | 0 0 0 0 0
2 | 0 0 0 0 0
3 | 0 0 0 0 0
4 | 0 0 0 0 0

Congratulation, well done!
Bye Bye!

E:\ITP4713_IOOP\Assignment\v2>
```

Sample Output 2:

```
E:\Assignment\v2>java Game game01.txt


Items
0        1      2      3      4    5  6      7      8
-----------------------------------------------------------
1 1 1 1  1 1 1  1   1    1    1 1  1  * * *  * * *    *
1     1  1   1    1    1 2 1  1 1     *   *  * * *  * * *
1     1  1 1 1  1   1    1            * * *  * * *    *
1 1 1 1

Current Count / Target Step Count: 0/4

  | 0 1 2 3 4
--+-----------
0 | 1 1 1 1 0
1 | 1 0 0 1 0
2 | 1 0 2 4 2
3 | 1 1 4 1 2
4 | 0 0 2 2 2

Please choose an item: 3
You have chosen:
  1
1 2 1
  1

Where do you want to put the item (row no)? 2
Where do you want to put the item (col no)? 1


Items
0        1      2      3      4    5  6      7      8
-----------------------------------------------------------
1 1 1 1  1 1 1  1   1    1    1 1  1  * * *  * * *    *
1     1  1   1    1    1 2 1  1 1     *   *  * * *  * * *
1     1  1 1 1  1   1    1            * * *  * * *    *
1 1 1 1

Current Count / Target Step Count: 1/4

  | 0 1 2 3 4
--+-----------
0 | 1 1 1 1 0
1 | 1 0 0 1 0
2 | 1 0 1 4 2
3 | 1 0 2 0 2
4 | 0 0 1 2 2

Please choose an item: 0
You have chosen:
1 1 1 1
1     1
1     1
```

```
1 1 1 1

Where do you want to put the item (row no)? 0
Where do you want to put the item (col no)? 0

Current Count / Target Step Count: 2/4

  | 0 1 2 3 4
--+-----------
0 | 0 0 0 0 0
1 | 0 0 0 0 0
2 | 0 0 1 3 2
3 | 0 X 1 X 2
4 | 0 0 1 2 2


Game Over!
Do you want to play again (0 for No, 1 for Yes)? 1


Items
0        1      2      3      4    5  6      7      8
-----------------------------------------------------------
1 1 1 1  1 1 1  1   1    1    1 1  1  * * *  * * *    *
1     1  1   1    1    1 2 1  1 1     *   *  * * *  * * *
1     1  1 1 1  1   1    1            * * *  * * *    *
1 1 1 1

Current Count / Target Step Count: 0/4

  | 0 1 2 3 4
--+-----------
0 | 1 1 1 1 0
1 | 1 0 0 1 0
2 | 1 0 2 4 2
3 | 1 1 4 1 2
4 | 0 0 2 2 2

Please choose an item: 3
You have chosen:
  1
1 2 1
  1

Where do you want to put the item (row no)? 2
Where do you want to put the item (col no)? 1


Items
0        1      2      3      4    5  6      7      8
-----------------------------------------------------------
1 1 1 1  1 1 1  1   1    1    1 1  1  * * *  * * *    *
1     1  1   1    1    1 2 1  1 1     *   *  * * *  * * *
1     1  1 1 1  1   1    1            * * *  * * *    *
1 1 1 1
```

```
Current Count / Target Step Count: 1/4

  | 0 1 2 3 4
--+-----------
0 | 1 1 1 1 0
1 | 1 0 0 1 0
2 | 1 0 1 4 2
3 | 1 0 2 0 2
4 | 0 0 1 2 2

Please choose an item: 6
You have chosen:
* * *
*   *
* * *

Where do you want to put the item (row no)? 2
Where do you want to put the item (col no)? 2


Items
0        1      2      3      4    5  6      7      8
-----------------------------------------------------------
1 1 1 1  1 1 1  1   1    1    1 1  1  * * *  * * *    *
1     1  1   1    1    1 2 1  1 1     *   *  * * *  * * *
1     1  1 1 1  1   1    1            * * *  * * *    *
1 1 1 1

Current Count / Target Step Count: 3/4

  | 0 1 2 3 4
--+-----------
0 | 1 1 1 1 0
1 | 1 0 0 1 0
2 | 1 0 0 2 1
3 | 1 0 1 0 1
4 | 0 0 0 1 1

Please choose an item: 6
You have chosen:
* * *
*   *
* * *

Where do you want to put the item (row no)? 2
Where do you want to put the item (col no)? 2


Items
0        1      2      3      4    5  6      7      8
-----------------------------------------------------------
1 1 1 1  1 1 1  1   1    1    1 1  1  * * *  * * *    *
1     1  1   1    1    1 2 1  1 1     *   *  * * *  * * *
1     1  1 1 1  1   1    1            * * *  * * *    *
```

```
1 1 1 1

Current Count / Target Step Count: 5/4

  | 0 1 2 3 4
--+-----------
0 | 1 1 1 1 0
1 | 1 0 0 1 0
2 | 1 0 0 1 0
3 | 1 0 0 0 0
4 | 0 0 0 0 0

Please choose an item: 7
You have chosen:
* * *
* * *
* * *

Where do you want to put the item (row no)? 0
Where do you want to put the item (col no)? 0


Items
0        1      2      3      4    5  6      7      8
-----------------------------------------------------------
1 1 1 1  1 1 1  1   1    1    1 1  1  * * *  * * *    *
1     1  1   1    1    1 2 1  1 1     *   *  * * *  * * *
1     1  1 1 1  1   1    1            * * *  * * *    *
1 1 1 1

Current Count / Target Step Count: 7/4

  | 0 1 2 3 4
--+-----------
0 | 0 0 0 1 0
1 | 0 0 0 1 0
2 | 0 0 0 1 0
3 | 1 0 0 0 0
4 | 0 0 0 0 0

Please choose an item: 7
You have chosen:
* * *
* * *
* * *

Where do you want to put the item (row no)? 0
Where do you want to put the item (col no)? 1


Items
0        1      2      3      4    5  6      7      8
-----------------------------------------------------------
1 1 1 1  1 1 1  1   1    1    1 1  1  * * *  * * *    *
1     1  1   1    1    1 2 1  1 1     *   *  * * *  * * *
```

```
1     1  1 1 1  1   1    1            * * *  * * *    *
1 1 1 1

Current Count / Target Step Count: 9/4

  | 0 1 2 3 4
--+------------
0 | 0 0 0 0 0
1 | 0 0 0 0 0
2 | 0 0 0 0 0
3 | 1 0 0 0 0
4 | 0 0 0 0 0

Please choose an item: 5
You have chosen:
1

Where do you want to put the item (row no)? 3
Where do you want to put the item (col no)? 0

Current Count / Target Step Count: 10/4

  | 0 1 2 3 4
--+------------
0 | 0 0 0 0 0
1 | 0 0 0 0 0
2 | 0 0 0 0 0
3 | 0 0 0 0 0
4 | 0 0 0 0 0

Congratulation, you have finished the game...
   but you can do even better!
Do you want to play again (0 for No, 1 for Yes)? 0
Bye Bye!

E:\Assignment\v2>
```

Error checking and Exception Handling

Try to implement your program as robust as possible by considering errors that may occurs in different input. For example:

game01.txt	***Possible error***
5	int value, range: 4 - 9
5	int value, range: 4 - 9
1 1 1 1 0	**matrix:**
1 0 0 1 0	correct size
1 0 2 4 2	correct range, i.e.
1 1 4 1 2	int value, range: 0 - 9
0 0 2 2 2	
4	int value, >= 1
lines below are ignored	
may put suggested answer here	
6 2 2	when error occurs while
1 2 2	reading file, the program may
0 0 0	terminate with error message

During the game, you may check the range and type of different input, try your best to prevent exceptions like ArrayIndexOutOfBoundsException display to player, as they may not know what it is. You may display meaningful error message to player and ask again until correct answer is input.

Documentation

The documentation should include the following:

- Evidence of Testing (output capture of all the features that you have implemented), remember to provide few lines of description of each capture
- Explanation of where your program used the concept of polymorphism
- Check List (see next page)

Checklist

Game Logic			
Load Game File	❑	Check Win and	❑
Display Items and GameBoard	❑	Game Over	❑
Ask and display selected item	❑	Correct Game Flow	❑
Applying selected item to GameBoard	❑		
Error and Exception Handling			
Game file: check file existence	❑	Input (select item): check data type	❑
Game file: check data type	❑	Input (select item): check range	❑
Game file: check range	❑	Input (row/column): check data type	❑
Game file: check matrix size	❑	Input (row/column): check range	❑
More Error and Exception Handling (if any)			
Write down the error that you can handle:		How you handle the errors stated on left hand side:	

Put ☑ if you have implemented successfully.

Implementation Guidelines

You are required to use Object-Oriented Programming technique in this assignment.

You may have, but not limited to the following classes: `Item`, `Bomb`, `SuperBomb`, `GameBoard` and `Game`. You need to model the description to classes, attributes and methods. **For example**:

When *playing* the `Game`, `Item` *is applied* to the `GameBoard` and the *cost* of the `Item` used will be accumulated to the *step count* of the `Game`.

In the statement above, we know that Game has a step count instance variable and also play method. In order to apply Items to GameBoard, Game may store one GameBoard object and 9 items (can be model by Item[]), besides there are also two methods of Item mentioned, but their implementation are unknown in Item yet.

Mark Distribution

Game Logic and Class Modelling (55%)

- Load Game File, Display Items and GameBoard (15%)
- Ask player for input and display selected item (5%)
- Applying the selected item to the GameBoard (10%)
- Check Win and Game Over (10%)
- Game Flow, such as asking re-play correctly (5%)
- Class modelling (10%)

Programming Style (5%)

- Adequate number of comments,
- Good indentation and following Java Naming Convention

Exception Handling (25%)

- File Format (10%)
- Correct Input type and Range (5%)
- Prevent Array Index Out Of Bounds Exception (10%)

Documentation (15%)