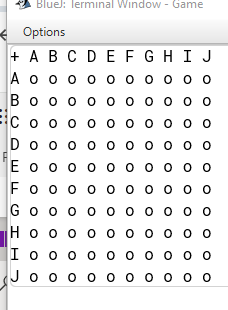
Trialing Document

|  |  |
| --- | --- |
| Date | What is the thing I am trialing |
| 14/05/24 | GUI of the game. |

*Copy and paste the table above before filling it out*

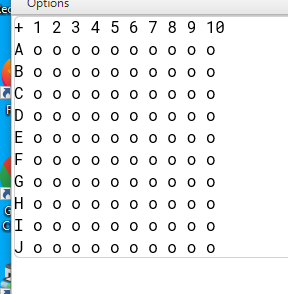
### Version A

### Coordinate x and y have the same label which is hard to understand.



### Version B

X and y coordinates have different types of label (alphabet and number).



Person: Mr. Fairhall (my teacher)

Date: 14/05/24

Comment: From version A if the user puts the coordinate of e.g. [D][C] the user might not know which grid they interacted with (because there are two possibilities DC or CD)

Person:

Date:

Comment

### Evaluation and next steps

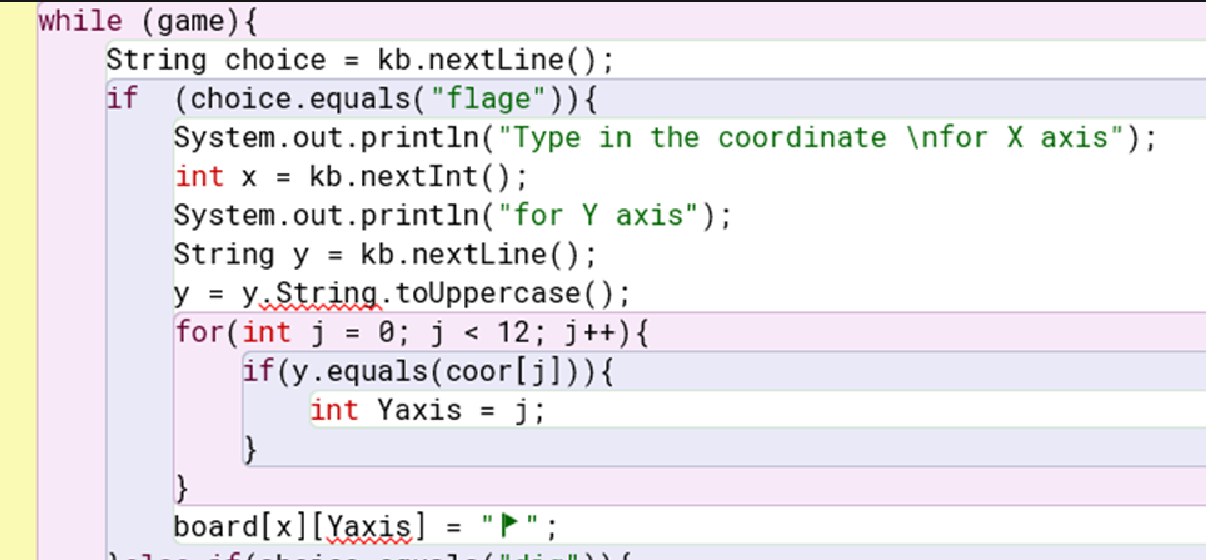
Based on this trialing and the feedback, I have decided to use version B because its easier to understand and use.

|  |  |
| --- | --- |
| Date | What is the thing I am trialing |
| 21/05/24 | Find out the Y coordinate as an integer for the arrays (because our y coordinate is alphabet) |

*Copy and paste the table above before filling it out*

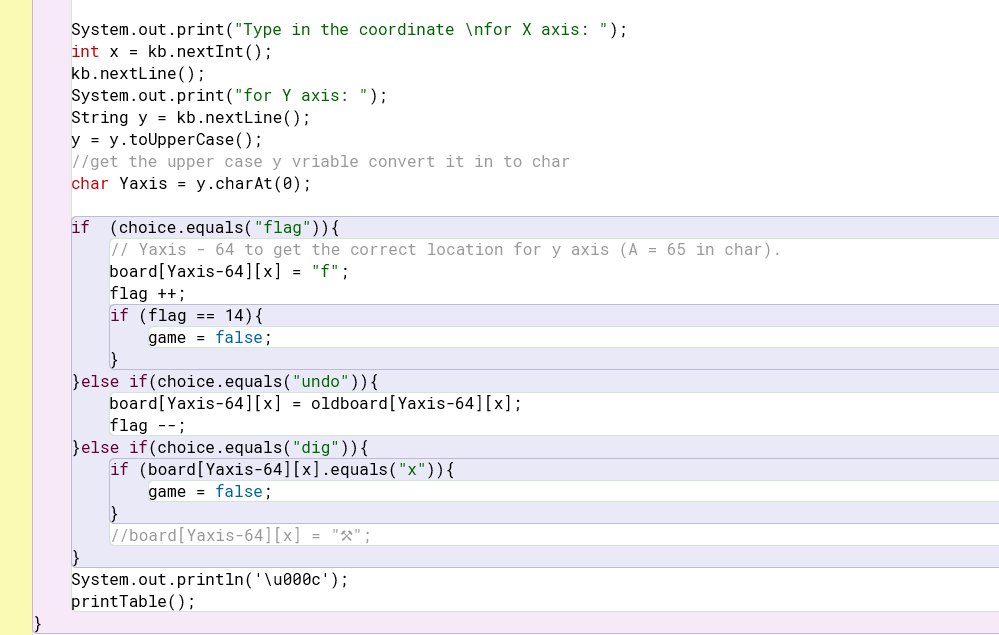
### Version A

To find the y coordinate I use a for loop to loop through the arrays of y coordinate [a, b, c, d…] and keep track of how many times we loop through [ j ] then set it to equal to the y axis coordinate which over complicate it. The better way to do it is to use char.



### Version B

Because char is an ASCII it has a value in int as well (A = 65) so I converted a string of y axis to char then minus it with 64 to get the y coordinate. Now it works fine with minimal code (compared with loop through the coordinate as string).



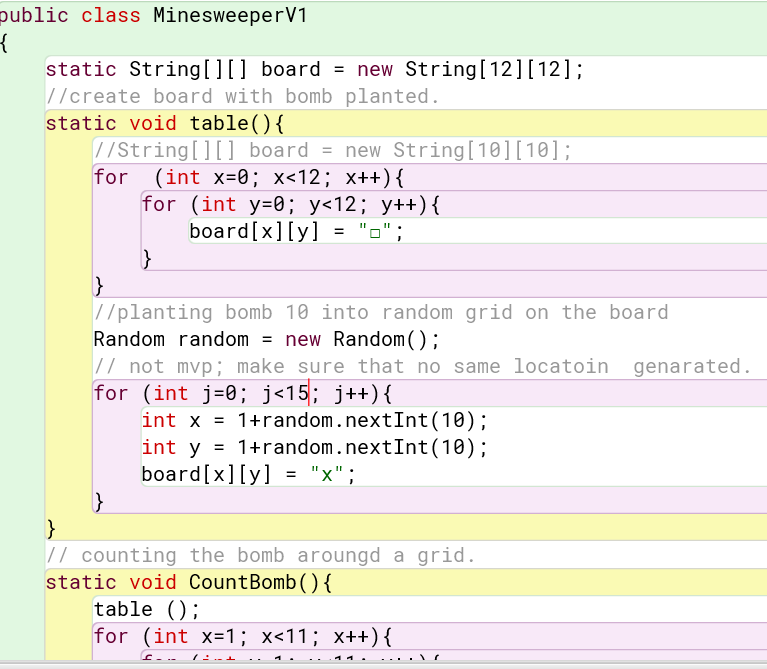
Evaluation and next steps

Based on this trial I decided to use version B because version B is faster, more efficient, and requires less code for the same output.

|  |  |
| --- | --- |
| Date | What is the thing I am trialing |
| 5/06/24 | Different ways of counting a bomb around each cell. |

### Version A

To count how many bombs are around a cell I created a method that takes the input of x and y coordinates then I ran a check on the cell around it ( [x-1, y-1] [x, y-1] [x+1, y+1]… ) if it contains a bomb or not but this will not work if the cell that we are checking is on the corner or at the rim of the board so what I do is create a bigger board (12x12) and only use the middle (10x10) part of it so if the cell that is being checked is at the edge of 10x10 board it will still works fine because the board is 12x12 so there will be no index out of range error.



### Version

I used a normal-size board (10x10) and passed the surrounding coordinates ( [x-1, y-1] [x, y-1] [x+1, y+1]… ) to another method called countbomb() which takes x and y coordinates as an argument. If the x and y coordinates are out of range the method will return 0 but if that cell is not out of range and contains a bomb the method will return 1 to the main Checkbomb method which will keep track of the amount of the bomb.



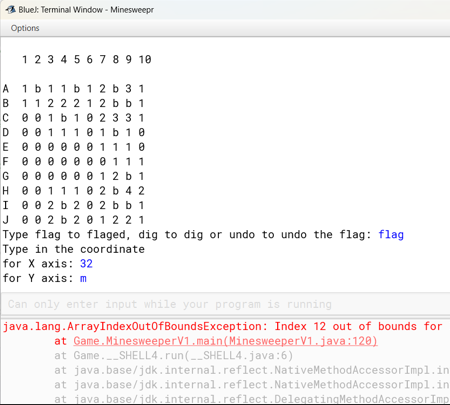
Evaluation and next steps

Based on this trial, I have decided to use version B because the board size of the board is being used throughout the entire game. If I use a size 12x12 board when I print the board out or generate a bomb inside the board the coordinates have to be in a range of 1 – 10 and the number of board lengths is not consistent throughout the game which made the game quite hard to deal with. Also, I think version B is easier to understand for everyone in case anyone wants to develop this project.

|  |  |
| --- | --- |
| Date | What is the thing I am trialing |
| 11/06/24 | Error checking the input from the user. |

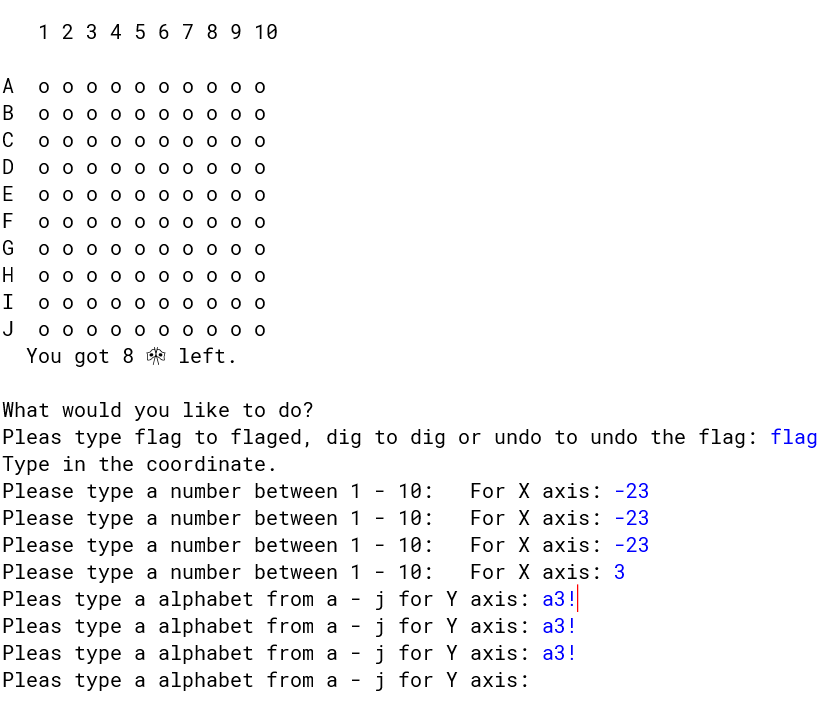
### Version A

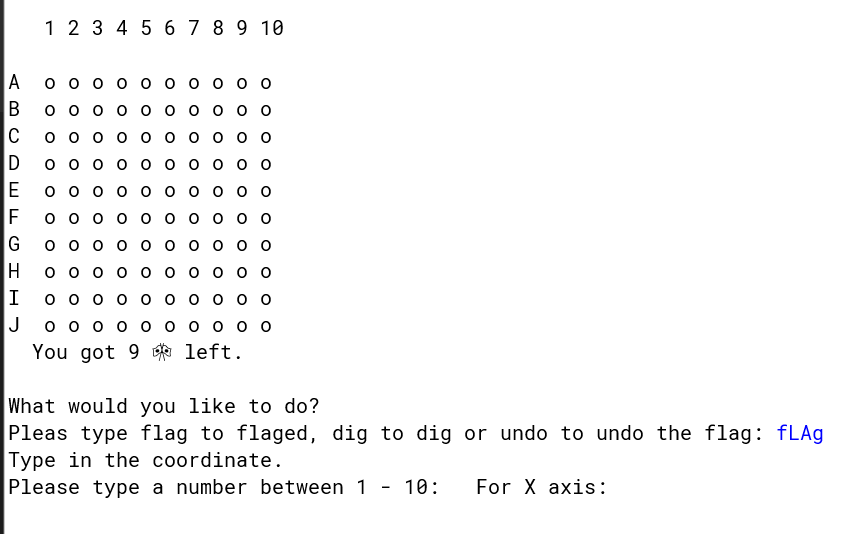
This version has no error checking at all, if the user misses typing something or spells some word with a mix of lower- or upper-case letters the game will end.



### Version B

This version now has error checking if the player types something wrong or spells with the wrong case the program will loop through until it gets the valid input. I did it by passing every input into a method and will loop through until get the valid input.





### Feedback

Person: Huy (my classmate)

Date: 12/06/24

Comment:” Version B is better because if I type something wrong the game will just loop through until it gets a valid input”

Person:

Date:

Comment

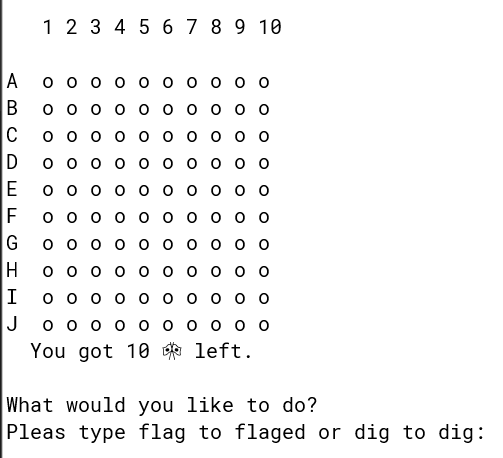
### Evaluation and next steps

Based on this trialing and the feedback, I have decided to use version B because it is better compared to version A the player doesn’t have to start the whole thing over again if they type something wrong, this will increase user experience on the game.

|  |  |
| --- | --- |
| Date | What is the thing I am trialing |
| 5/06/24 | Undo option |

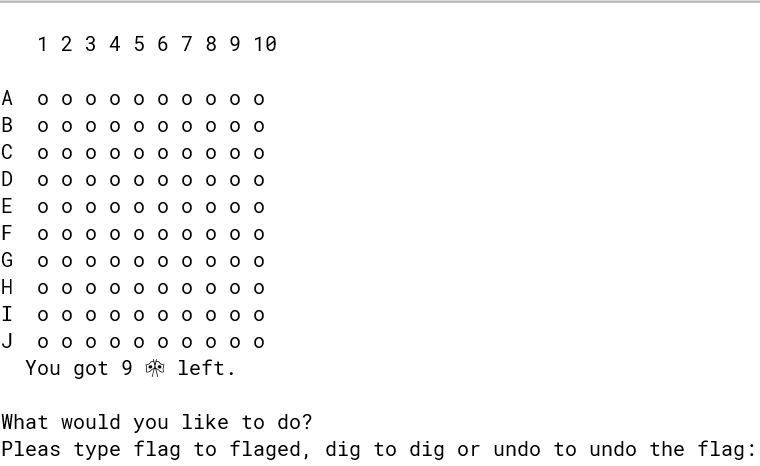
### Version A

If the player is placed at the wrong place by accident or finds out afterward, they cannot undo it



### Version B

I add an undo option to the game.



Person: Aksel (my friend that try the game)

Date: 3/05/24

Comment According to Aksel he cannot undo the flag after accidentally placing it in the wrong spot.

### Evaluation and next steps

Based on this trialing and the feedback, I have decided to use version B because now the player can undo the flag if they make any mistake.