CS1002 W03-Piratical 170025298 Tutor:Alice Toniolo Date: Nov 3rd

Overview:

This program is the first game I have ever programmed and this is such hard for my logical mind to be honest. I use 2-D array for finishing the practical and have used a lot of if command and for loop for finish the game.

In the program, I have 3 classes and an EasyIn class that used to input the number and get number from users.

It's quite hard and I have tried my best!

Design:

For this practical, I have tried twice and the first one is totally in a mess.

```
55              gameBoard.setBoard(board);
56              board = gameBoard.getBoard();
57              //boolean move judge 1 step
58              boolean move;
59              //if(board[x1+1|x1-1][y1+1] == board[x2][y2])
60              if((x1+1==x2||x1-1==x2||x1==x2)&&(y1+1==y2||y1-1==y2||y1==y2)){
61                  move = true;
62              }
63              else{
64                  move = false;
65              }
66              //boolean eat
67              boolean eat;
68              if((x1+2==x2||x1-2==x2||x1==x2)&&(y1+2==y2||y1-2==y2||y1==y2)){
69                  for(int i=-1;i<1;i++){
70                      for(int o=-1;o<1;o++){
71                          if(board[x1+i][y1+o] == 'o'){
72                              eat = true;
73                          }
74                          else{
75                              eat = false;
76                          }
77                      }
78                      //choose goose
79               if(eat = true){
80                   this.x3 = (x1 + x2) / 2;
81                   this.y3 = (y1 + y2) / 2;
82              }
83                     /* if(board[x1+i][y1+i] == 'o'){
84                          eat = true;
85                      }
                        else {
```

I put everything together and found that I cannot find the logic mistake so that I coded twice:

```
            }

            //get int from reader
            x1 = reader.getInt();
            y1 = reader.getInt();
            x2 = reader.getInt();
            y2 = reader.getInt();

            //judge the chosen object
            if(turn == 1){
                if((board[x1][y1] == FOX)||(board[x1][y1] == FREE)||(board[x1][y1]==INVALID)){
                    System.out.println(ILLEGALMOVE_MSG);
                }
                else{
                    NUM = NUM + 1;
                }
            }
            else{
                if((board[x1][y1] == GOOSE)||(board[x1][y1] == FREE)||(board[x1][y1]==INVALID)){
                    System.out.println(ILLEGALMOVE_MSG);
                }
                else{
                    NUM = NUM + 1;
                }
            }
            //move and eat
            if((x1+1==x2||x1-1==x2||x1==x2)&&(y1+1==y2||y1-1==y2||y1==y2)){
                move = true;
            }
            else{
                move = false;
            }
```

It’s much more clear.
However I have a huge problem about my second coding: However it looks like less messy and is easier to look, but I cannot ran it

```
Exception in thread "main" java.lang.NullPointerException
        at Board.printBoard(Board.java:60)
        at Game.play(Game.java:44)
        at W09Practical.main(W09Practical.java:7)
```

And I cannot find the mistake of my program.
To solve the problem, I went to the lab for help and tried to find where is the mistake.
//wait for tomorrow

//the process
I have 3 classes in total:
The first one is main class for sure, the second one is Game to make the rule of the game; the third one is about board and to print the board and take details (eg.the amount of goose).
I have nothing to say about main class because there is only one code here:

```
public class W09Practical {

    public static void main(String[] args) {

        Game game = new Game();

        game.play();
    }
}
```

In the game class, I thought like following:
What is necessary to a game?
Rules.
So I make the rule as four parts:
How to move?
How to eat?

When to win?
Whose turn?
Let me start with whose turn is it.
To judge the turn, I take a number of turn and make it %2: if the result is 0, then it should be geese's turn; else, it should be fox's turn.
How to eat and move:
Because generally I use the condition command to judge whether user has moved the right character, so I set a Boolean to judge whether they can move or not: eat and move.
Eat for judge whether the fox can eat goose and the move means whether goose and fox can move. If the fox cannot move or eat and the winner is geese. If there is no more goose and the winner is fox.
As to move, I make the previous board into FREE and the new board into previous, such as:

```
if(turn == 1){
        if(move = true){
            board[x2][y2] = board[x1][y1];
            board[x1][y1] = FREE;
        }
        if(eat = true){
            board[x2][y2] = board[x1][y1];
            board[(x1+x2)/2][(y1+y2)/2] = FREE;
            board[x1][y1] = FREE;
        }
    }
else{
    if(move = true){
        board[x2][y2] = board[x1][y1];
        board[x1][y1] = FREE;
    }
}
```

However the comparative equal should be "==" so that with the help of teachers I changed it into "=="

```
if (turn == 1) {

    if ((move == true) && (board[x1][y1] == GOOSE) && (board[x2][y2] == FREE)) {
        board[x2][y2] = board[x1][y1];
        board[x1][y1] = FREE;
        NUM = NUM + 1;
    } else {
        System.out.println(ILLEGALMOVE_MSG);
    }
} else {
    if (((move == true) || (eat == true)) && (board[x1][y1] == FOX) && (board[x2][y2] == FREE)) {
        if (move == true) {
            board[x2][y2] = board[x1][y1];
            board[x1][y1] = FREE;
            NUM = NUM + 1;
        }
        if (eat == true) {
            board[x2][y2] = board[x1][y1];
            board[x1][y1] = FREE;
            NUM = NUM + 1;
            board[(x1 + x2) / 2][(y1 + y2) / 2] = FREE;
        }
    } else {
        System.out.println(ILLEGALMOVE_MSG);
    }
}
```

And this is more brief and more convenient and brief for eating and moving.
//set winner:

To make my program finished, I think the hardest part should be set winner.
It's easy for coding the fox like this:

```
int amount;
amount = gameBoard.amount();
if(amount == 0){
    gameBoard.printBoard();
    System.out.println(FOXWINS_MSG);
    done = true;
}
```

I set a amount of goose in the Board and if the goose unavailable the fox is winner.
It is the same as goose. However we should make sure that the number is less than 7. So I use try and catch and decide the winner with the amount surrounded by fox is 16.

```
    for(int i = 0;i<3;i++) {
        for(int j = 0;j<3;j++) {
            //try catch to calculate how many  place it can move
            try {
                if (board[x + o[i]][y + f[j]] != FREE) {
                    unfree = unfree + 1;
                }
            } catch (Exception e) {
                unfree = unfree + 1;
            }
            try {
                if (board[x + u[i]][y + p[j]] != FREE) {
                    unmove = unmove + 1;
                }
            } catch (Exception e) {
                unmove = unmove + 1;
            }
        }
    }//for end
    //there have to calculate goose itself so is 19
    if(unfree + unmove == 18){
        gooseWin = true;
    }
}
```

Because we have to think about both I and j is 0, so I set unfree and fox unable to move number in total is 18 and then return the winner is goose.

```
if(gooseWin == true){
    gameBoard.printBoard();
    System.out.println(GEESEWIN_MSG);
    done = true;
}
```

//board class:
In the board class I have to set the position of goose and fox and set the play station in to fixed shape, so I code like this:

```
// Put a single fox in the middle
board[boardsize/2][boardsize-2] = FOX;
//put goose on the top
for(int i=0;i<boardsize;i++){
    for(int j=0;j<4;j++){
        board[i][j] = GOOSE;
    }
}
for(int i=2;i<5;i++){
    board[i][3] = FREE;
}
//delete invalid area
int boardsizeC = boardsize - 1;
for(int x=0;x<2;x++){
    for(int y=0;y<2;y++){
        board[x][y] = INVALID;
        board[boardsizeC-x][boardsizeC-y] = INVALID;
        board[boardsizeC-x][y] = INVALID;
        board[x][boardsizeC-y] = INVALID;
    }
```

And in the board I also take the amount of goose via for loop:

```
public int amount(){
    //get the rest of goose
    int amount = 0;
    for(int i=0;i<boardsize;i++){
        for(int j=0;j<boardsize;j++){
            if(board[i][j] == GOOSE){
                amount = amount + 1;
            }
        }
    }
    return amount;
}
```

To get the amount of goose and judge the win of goose.

To be honest, try and catch is quite a good way to calculate the amount out of boundary. Without this and there would throw an error and made the program exit.

Testing:

```
klovia:~/Documents/cs1002/W09-Practical/source hl74$ stacscheck /cs/studres/CS1
002/Practicals/W09/Tests
Testing CS1002 W09 Practical
- Looking for submission in a directory called 'source': Already in it!
* BUILD TEST - build-all : pass
* COMPARISON TEST - basic/prog-run-0_initialBoard.out : pass
* COMPARISON TEST - basic/prog-run-1_veryBasicMoves.out : pass
* COMPARISON TEST - basic/prog-run-2_legalMoves.out : pass
* COMPARISON TEST - basic/prog-run-3_illegalMoves.out : pass
* COMPARISON TEST - basic/prog-run-4_boundaryChecks.out : pass
* COMPARISON TEST - basic/prog-run-5_incompleteGame.out : pass
* COMPARISON TEST - basic/prog-run-z_geeseWin.out : pass
8 out of 8 tests passed
```

I tested in the putty and obviously I gain a success.
I have been stuck in the final test by ignoring the trouble of how to make goose win. Luckily I finished the code and solve the problem.

Evaluate:
Actually I have programmed this game twice and the first time is an obvious mess. I cannot even find where is the mistake. This makes me realise how important is brief and comment to the program.
As a result, I think this could be the second-best program I have ever programmed.
Conclusion:
This program have spent me about 7 hours without extension. The use of try and catch have opened a new door and made me find a new method of programming. However, the most valuable thing I have learned is about to be make the program as brief as it could be. This is quite a tiny game, but I have used all of my knowledge I have learned from both lecture and tutorial.
The use of 2-d array and array is certainly an absolutely new way of doing the program and try and catch can prevent the error and count the number when the error exist.
It is also very important for program a new code with making through the execute logic and be more creative.

Ps: I still feel exhausted with my last program. However, it is not brief, but I hope you can use it at least once and that have spent me quite a long time even though it is not count. I will finish extension next and the details will in the following.

There are 3 Extensions in the folder:
The first is AI
Second about random board
Third is about increased goose;