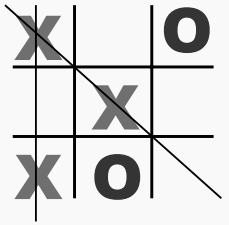
Noughts and Crosses (Tick-tack-toe)

- A simple game project in Java
- Illustrates:
  - User Interface
  - Model-View-Controller design
  - automated game strategy
- Systematic commenting

Rules of the game

- Two players, X and O.
- X goes first, writes an X on the board.
- Then they take it in turns.
- Aim: get 3 in a row, vertically, horizontally or diagonally.
- Otherwise it's a draw.



O goes next, but can't prevent X winning on one of two lines.

Features of program

In demonstration, computer and human take turns to move.

Human moves are made by mouse click.

- If one player wins, then no more moves.
- Illegal moves are prevented.
- Interface has buttons for:
  - a new game
  - letting the computer go first (letting computer be X)
  - exit
- Computer calculates its next move – how?

Overall project

Need classes and methods to:

- Maintain board
- Check state (Whose move? Has anybody won? Is it a draw?)
- Administer game – take it in turns, no illegal moves.
- Implement GUI – using MVC.
- Implement strategy for computer moves.

Graphical User Interface

Model-View-Controller

Exercise 1b

Deadline: 11.00 Friday 23rd Jan.

Show your implementation to a lab demonstrator and get a mark (0, 1 or 2).

Exercise is to implement Java class to maintain board.

Board

grid: 3x3 array of char

clear()

putCross(i: int, j: int)

putNought(i:int, j:int)

getSymbol(i: int, j:int): char

isBlank(i: int, j:int): boolean

```
char[][] grid;
grid = new char[3][3];
```

[0][0]	[0][1]	[0][2]
[1][0]	[1][1]	[1][2]
[2][0]	[2][1]	[2][2]

Use characters 'X', 'O' and '.'

Other classes

Use simple text interface.

BoardDisplay

theBoard: Board

display()

BoardInput

theBoard: Board

getMove(crossMove: boolean)

MVC:

Model class was Board.

View

Controller

Test

…

main()

Comments: *Invariants*

Some techniques for systematic commenting.

You'll see them in my solutions, so you need to understand what they mean. Try using them yourself.

invariant: A property that you intend always to be true.

e.g. char[][] grid;

//invariant: grid is 3x3

Constructor's task is to ensure the invariant property starts off true.

Other methods must be written so that invariant remains true after they've been called.

Otherwise you've got a bug!

Comments: *requires* and *ensures*

Used to be precise about how *methods* behave.

requires: A property that has to be true on entry for the method to work correctly.

ensures: A property that you intend should be made true on return by what the method does (*provided* the "requires" condition was true on entry).

If the method fails to do that then you've got a bug!

e.g. void putCross(int i, int j)

//requires: i and j between 0 and 2

//ensures: grid[i][j]=='X', and rest of grid unchanged

Summary

We have now looked at –

- The Noughts and Crosses game.
- A version of the overall project, with computer playing human.
- Exercise 1b, with a basic Board class.
- 2-dimensional arrays.
- Systematic comments using *invariant*, *requires* and *ensures*.