# OXO Exercise Continues

## (and informed use of AI)

### COMSM0086

### Dr Simon Lock  &  Dr Sion Hannuna

# Extra OXO Features

This week we will continue with the OXO exercise

We will add in some error handling mechanisms

Plus extensions to make game more "interesting"

Let's look at each of these in turn...

# Error Handling

It's likely users will make mistakes during gameplay
Entering 'invalid' cell identifiers into the GUI:

- Invalid Identifier Length: Command is not 2 chars
- Invalid Identifier Character: Row character is not
    a letter or column character not a numerical digit
- Outside Range: Valid characters, but identifier
    values are out of range (i.e. too big or too small)
- Already Taken: Cell has previously been claimed

In Java we handle run-time errors using 'Exceptions'

# Exceptions Hierarchy

We've provided you with a hierarchy of exceptions

One to represent each of the previous user errors

Commonalities are factored out into a superclass

See workbook for examples of how to use these
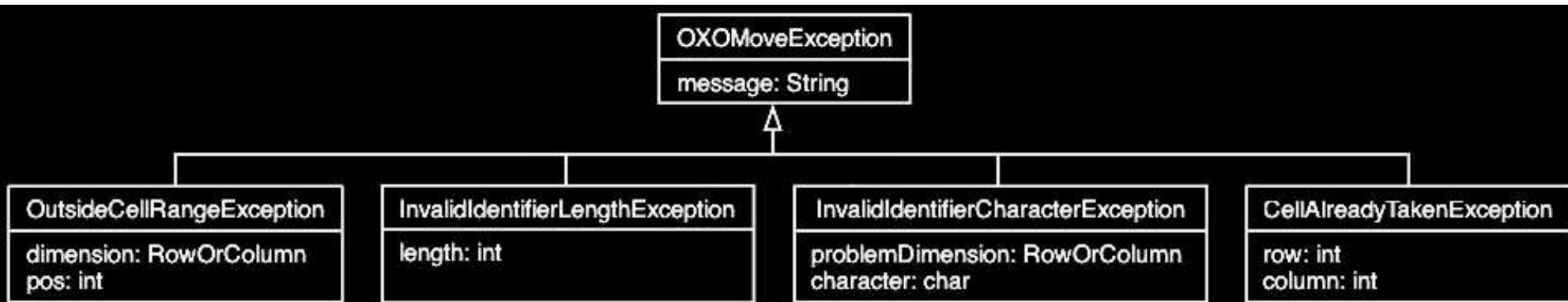
# Adjustable Win Threshold

"Win Threshold" is number of cells required to win
More interesting if we can alter this threshold !

OXOGame allows users to set the win threshold
Altered by pressing the `+` and `-` keys
(actually the `=` and `-` keys for convenience)

Controller is then notified through two methods:
    `increaseWinThreshold()`   `decreaseWinThreshold()`

You should update the threshold held by OXOModel
Then use this value when performing win detection

# Greater Number of Players

Traditional number of players in an OXO game is 2
Additional players makes game more interesting !

Add features to support ANY number of players
(data structures, turn taking, win detection etc.)

Number of players can't be changed using GUI
This number can only be changed programatically
(A good opportunity for automated testing !)

Let's spend some time talking about AI

# Various Types of AI Tools Exist

1. Suggestion/completion tools embedded in IDE
   Such as Copilot, which acts as a "collaborator"

2. Problem solving and error fixing search tools…
   Tools like ChatGPT as frontend to StackOverflow

3. Web-based code "generation" tools…
   Tools like Gemini or DeepSeek to synthesise code

   The first two scenarios are fairly uncontentious
   We need to be concerned about "generative" AI

# Appropriate Use of AI

AI tools can often provide really good prompts
Integrated code completion tools can be helpful
(avoid following advice blindly - try to understand !)

Also useful for understanding errors/fixing problems
Paste obscure compiler errors into chat for insight
Asking AI what is wrong with specific lines of code

Avoid dropping in large fragments of your code…
Remember that AIs are always learning !
(You don't want to be "giving away" your code !!!)

# "Generative" AI

There is a reason why I put "Generative" in quotes
It's not really generating new material, but rather...
just recycling/reusing/recombining existing work
More like a "mashup" than creating something new

Just consider recent upset of Artists and Musicians

...and publishing companies:
https://www.bbc.co.uk/news/articles/ckrrr8yelzvo

# Example: "Write me an OXO game in Java"

ChatGPT

Sure! Here's an example of an OXO game (Tic-Tac-Toe) implemented in Java:

```java
import java.util.Scanner;

public class OXOGame {
    private char[][] board;
    private char currentPlayer;

    public OXOGame() {
        board = new char[3][3];
        currentPlayer = 'X';
        initializeBoard();
    }

    private void initializeBoard() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
```

# General Warnings about AI Tools

AI tools are getting pretty good...
But they still have their limitations

Don't just use them blindly

It's important to be critical about their use

(as with any piece of technology you use !)

Let's take a look at some deficiencies...

# AI shifts problem from coding to debugging

# "Generative" AI Tools

"Generative" AI is good at performing small tasks
e.g. creating a first-cut attempt at a specific function
Clearly defined jobs, expressed in a short query:

    write me a Java method to parse an CSV file

They're not well suited to creating complex systems
Code that has multiple interwoven features
Where expressing exactly what YOU want is hard

Effectiveness depends on "asking the right question"

# "extend OXO to support more than 2 players"

Extending the OXO game code to include an arbitrary number of players becomes complex due to the way winning conditions are typically defined in Tic Tac Toe. Here's why:

- **Winning Conditions:** Traditionally, Tic Tac Toe is played with two players ('X' and 'O') who take turns marking the board. The goal is to get a line (row, column, or diagonal) of your symbol.

With more players, defining winning conditions becomes ambiguous. Should a player win by getting a line of their symbol, or should it be the last player to mark a valid cell on a full board?

The AI says "no"

15/25

# Limitations of current AI

Even if you are able to formulate suitable questions

There are still limitations in the abilities of AI tools

Good at assimilating material already "out there"

(summarising large amounts of complex material)

Not so good at making creative custom adaptations...

# "rewrite OXO so that it doesn't use arrays"

**ChatGPT**

Certainly! Here's an example of an OXO game (Tic-Tac-Toe) implemented in Java without using arrays:

```java
java

import java.util.Scanner;

public class OXOGameWithoutArrays {
    private static final int BOARD_SIZE = 3;
    private static char[][] board = new char[BOARD_SIZE * BOARD_SIZE];
    private static char currentPlayer = 'X';
```

# Chat-Driven Software Engineering ?

It may seem like chat UI is ideal for working with AI
This suits some tasks, but not Software Engineering !

Chat AIs require you to "keep conversation going"
Hard to manage large codebase with a chat interface

Must keep AI up-to-date with current version of code
(e.g. if you have manually added some functionality)

Big challenges for long-term maintenance/evolution

Use of AI in assessments...

# But before we do...

We are almost halfway through the teaching block
Useful to get your feedback on how things are going

Please visit the "Blue" website:
https://evaluation.bristol.ac.uk

Complete the questionnaire
for the "OOP with Java" unit

Monitor

# Mashup Coursework Submissions

The notion of a "mashup" codebase is nothing new

It does not rely on availability of effective AI tools

Students have been creating mashup work for years

<span style="color:magenta">borrow a bit of code from here, borrow a bit from there</span>

Modern AI tools just make process quicker & easier

Students often don't even realise they are doing it !

But that's what you get when you Gen AI some code

# Addressing this Problem

Aim of unit is to assess YOUR programming skills
We aren't going to be able to do that if you are:
- "Reusing" bits and pieces from online solutions
- "Generating" fragments of code using AI tools

If we could *detect* code that has been "derived"…
We could factor it out during the marking process:

you don't get marks for code you didn't write

This would allow us to give credit where credit's due

# Implementing this Approach

We have a library of 500 solutions to the assignment
ALL submissions of ALL students from previous years

Many are available online (as part of code portfolios)
These will be part of the training set used by AI tools

These 500 solutions also borrow from online examples
(which provides us with even broader sampling)

All we do is compare YOUR code with this training set
The more similarities that are found...
The more "derivative" your submission has become

# I know what you are thinking…

There is BOUND to be some level of similarly
Some standard boilerplate code that everyone has
Or some convergent algorithm that everyone uses

That's fine, we expect this kind of thing to happen
We will be using a variety of filters and thresholds
The aim being to factor out "natural" commonality

Also human-in-the-loop to catch any false-positives

# REMEMBER

You will only be rewarded for code that YOU write

Any "derived" code ("found" items or IA generated)

Will be discounted during the marking process

(resulting in a reduced final mark)