CS 240: Phase 1 Overview Transcript

[00:00:00] This video is going to provide an overview of phase one of the chess project.

[00:00:05] In phase zero, you implemented the basic peace movement algorithms, and you also implemented your chess board and the logic for initializing the chess board.

Start visual description. The professor demonstrates the basic peace movement algorithms and the logic for initializing the chess board. The screen shows a basic board class, a chess piece class, and the capability for calculating legal moves. End visual description.

[00:00:15] So you've got a basic board class, you've got um some, a chess piece class and then some capability for calculating legal moves in phase one.

[00:00:23] What we're going to do is focus on implementing the rest of the rules for the chess game.

[00:00:29] So we're going to implement things like uh check, checkmate, stalemate.

Start visual description. The professor demonstrates the implementation of chess game rules such as check, checkmate, and stalemate. The screen shows a class diagram of the chess game class with methods like get team turn and valid moves. End visual description.

[00:00:35] We also need to do things like keep track of whose turn it is and, and, and things like that.

[00:00:39] And so for this purpose, the starter code that we provided um contains a class named chess game.

[00:00:48] And so in this diagram here, we can, we can look at um a, a class diagram and you can see here the chess game class, it has uh the following methods um get team turn.

[00:01:01] So the chess game, one of its responsibilities is to keep track of whose turn it is and um valid moves. So anytime that the game wants to calculate the valid moves for a chess piece. Um This, this method will be called.

[00:01:16] Now this method, what it will do is it will delegate to the chess piece classes and ask them what their moves are.

Start visual description. The professor demonstrates the valid moves method, which filters out illegal moves due to check or checkmate issues. The screen shows the chess game class delegating to the chess piece classes and removing invalid moves. End visual description.

[00:01:24] So most of the movement algorithms and calculations are not in the chess game class.

[00:01:30] But one thing the chess game will do in the valid moves method is it will actually remove any movements that the pieces thought they could make but actually aren't legal because of check or checkmate issues.

[00:01:42] And so this valid moves method does some filtering of the moves that are returned by the pieces.

[00:01:47] Uh Then we have a make move method which um causes a, a move to be made that would up update the state of the board, uh move a piece from one place to the other and, and um deal with any, any things like pawn promotion or, or things that need to be dealt with there.

[00:02:03] Um We also provide methods on the, the chess game class for determining if the, the player whose turn it currently is in check.

[00:02:11] So is in check is um one of the methods you pass in the team color.

[00:02:16] And uh this method will tell you if that player is currently in check.

[00:02:20] We also have is in check mate.

[00:02:22] Of course, tells us when the game is over. We also have is in stalemate.

[00:02:28] Now, checkmate is when it's my turn to move and my king is in check, and I don't have any moves that would get him out of check.

Start visual description. The professor demonstrates the methods for detecting checkmate and stalemate. The screen shows the is in check, is in checkmate, and is in stalemate methods, explaining the conditions for each. End visual description.

[00:02:42] Now, when we say the king is in check, that means that pieces on the opposing side can currently attack the king and could capture it.

[00:02:50] And so it's, it's now our responsibility to get out of check by moving our pieces in such a way that the king can no longer be attacked or captured.

[00:02:58] But if I don't have any moves that can save my king, then, then it's over, that's checkmate.

[00:03:03] Now, there is in stalemate uh method.

[00:03:06] It's kind of similar actually to checkmate except it's, it's the situation where it's my turn to move, my king is not in check.

[00:03:15] So the opposing pieces cannot attack or capture my king.

[00:03:19] But I don’t have any moves that would not put my king in check.

[00:03:23] So So I can’t make any moves because that would result in putting my king in danger, but he’s not currently in danger. So, it’s basically a draw or a tie.

[00:03:32] And so that’s what we call a stalemate. So, you need to be able to detect that as well.

[00:03:37] And then we have some other methods for setting and getting the board on the chess game.

[00:03:41] So um as you can see the some, some number of the chess rules are, are implemented in, in this class.

[00:03:51] Now, it’s important that you don’t uh modify the public interface of this chess game class as in phase zero, the starter classes have method interfaces that are depended on by the auto grader.

[00:04:03] So make sure you don’t change the signatures of these methods. You can add additional variables and methods but don’t change these ones.

[00:04:13] OK. So that’s the focus of uh phase one.

[00:04:21] Now, one, something that you do have available in phase one is there is some extra credit available.

[00:04:26] If you want to implement some advanced peace moves, then you can do that specifically we’re talking about on pass and castling.

Start visual description. The professor demonstrates the extra credit opportunities for implementing advanced moves like en passant and castling. The screen shows the starter code for phase one and the additional test cases for these advanced moves. End visual description.

[00:04:35] So these are more advanced uh moves on Passan deals with, with ponds and then Castling deals with rooks and kings.

[00:04:45] And in phase zero, we didn't even introduce this uh possibility. But now that we're in phase one, we do offer you the opportunity if you want to implement on Passan and Castling and if you do so correctly, then uh we'll give you 10 extra points.

[00:05:01] So this whole phase is worth 100 and 25 points currently that could change over time.

[00:05:05] So when you watch this video, it might have changed. But currently the uh the phase is worth 100 and 25 points and we give you um an extra 10 points.

[00:05:15] Um If, if you do this extra credit, five points per move, so do that. If you wish the starter code for phase one provides some pass off test cases just like we did in uh phase zero. We have some additional pass-off test cases for the chess game class.

[00:05:36] So they, they're in a file named chess game test dot Java.

[00:05:39] So you're going to need to copy that file into your project.

[00:05:43] If you implement on Passat and casting, there's some additional extra credit test cases that you can uh copy in a project to demonstrate that, that you've implemented those, those movements correctly.

[00:05:55] So those are in the extra credit package.

[00:06:00] And again, we're going to use the auto grater to pass off this phase.

[00:06:07] And so at that point, I think you have everything you need to succeed on, on phase one. So good luck.