Team 01: Doug Russo, Katy Martinson, Phil Morgan, Kiera Egan

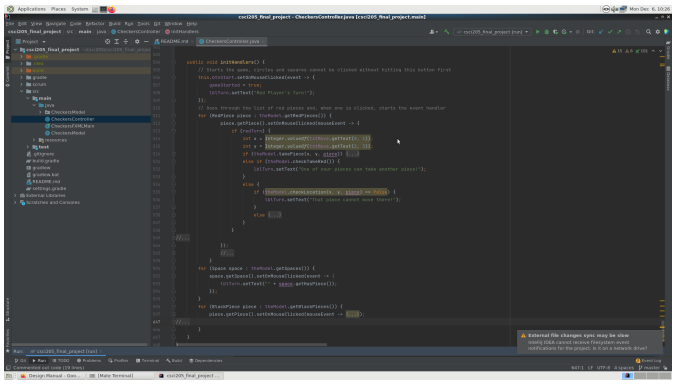
CSCI 205

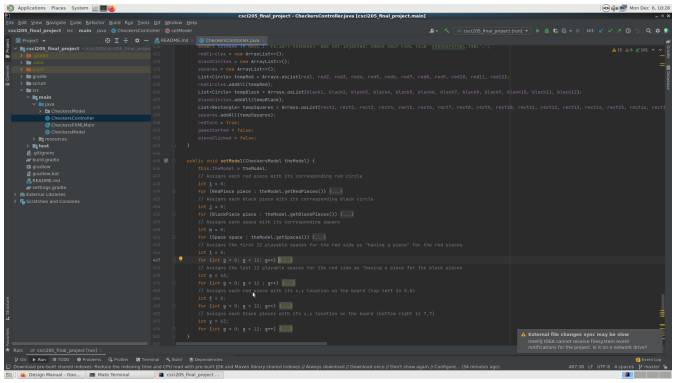
5 December 2021

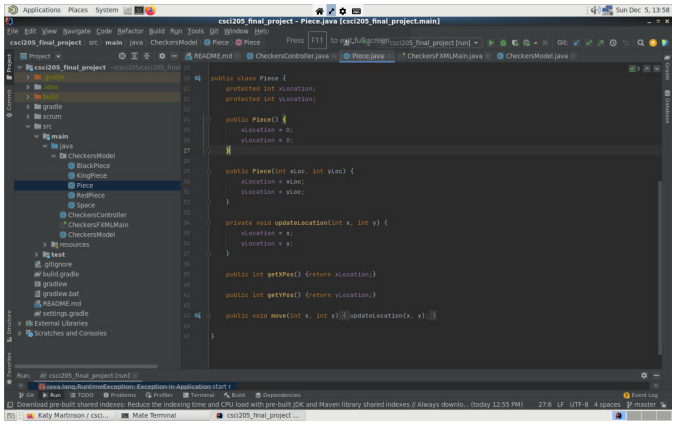
Design Manual

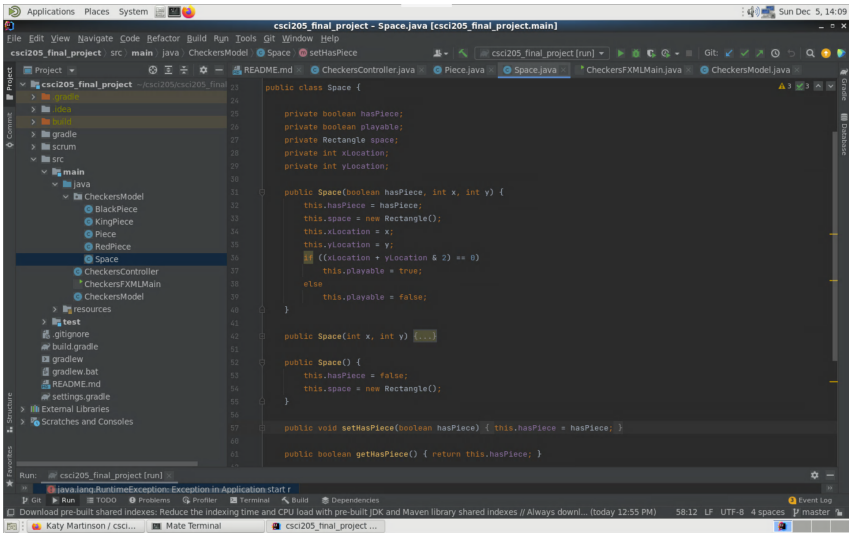
Our system is designed to try and create checkers using a GridPane as the board, Circles for the checkers, and a start button to start the game, all of which was created in SceneBuilder.

We created a boolean statement so that once the start button is pressed, the game would begin, prompting the red checker player to go first, and alternate after each turn, as depicted in the image to the right. This decides where the checker can be moved. The Checker pieces are split up into two classes, with a parents class. Those classes are RedPiece, BlackPiece, and the Piece parents class. There is another KingPiece class that extends the Piece parents class as well that is not currently implemented. The intention of these classes is to store a checker's location, as well its color. For a player to be created, we decided to create a CheckersModel class that would store an array of pieces for each side. The CheckersModel class also has many helper methods that allow it to check the location,

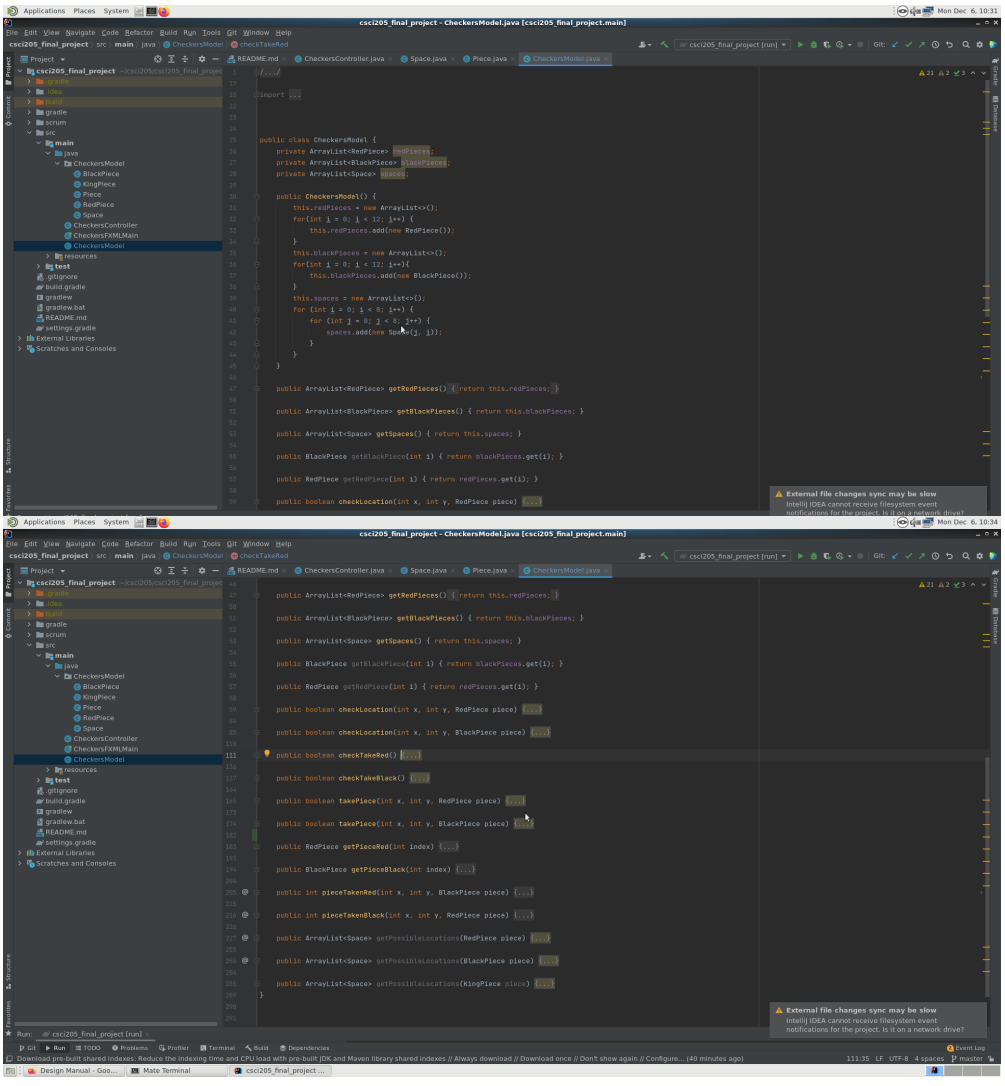








check if there is an available spot to go, and take pieces.A Space class was also created that stores a certain space's location, does it have a piece, and it updates whether that space is playable or not. In the controller class, we declare all of the circles, spaces, and create the checkers model, that way everything works together when the program is run.



CRC CARDS

Space	
Responsibilities Determines if a piece can move to each space (square); aka if a space is playable based on several factors	Collaborators Model Piece

RedPiece	
Responsibilities Extends Piece Creates pieces that can move diagonally upwards across the board	Collaborators Piece Board

BlackPiece	
Responsibilities Extends Piece Creates pieces that can move diagonally downwards across the board	Collaborators Piece Board

KingPiece	
Responsibilities Extends Piece Creates pieces that have made it to the other side of the board These pieces can move diagonally both directions	Collaborators Piece Board

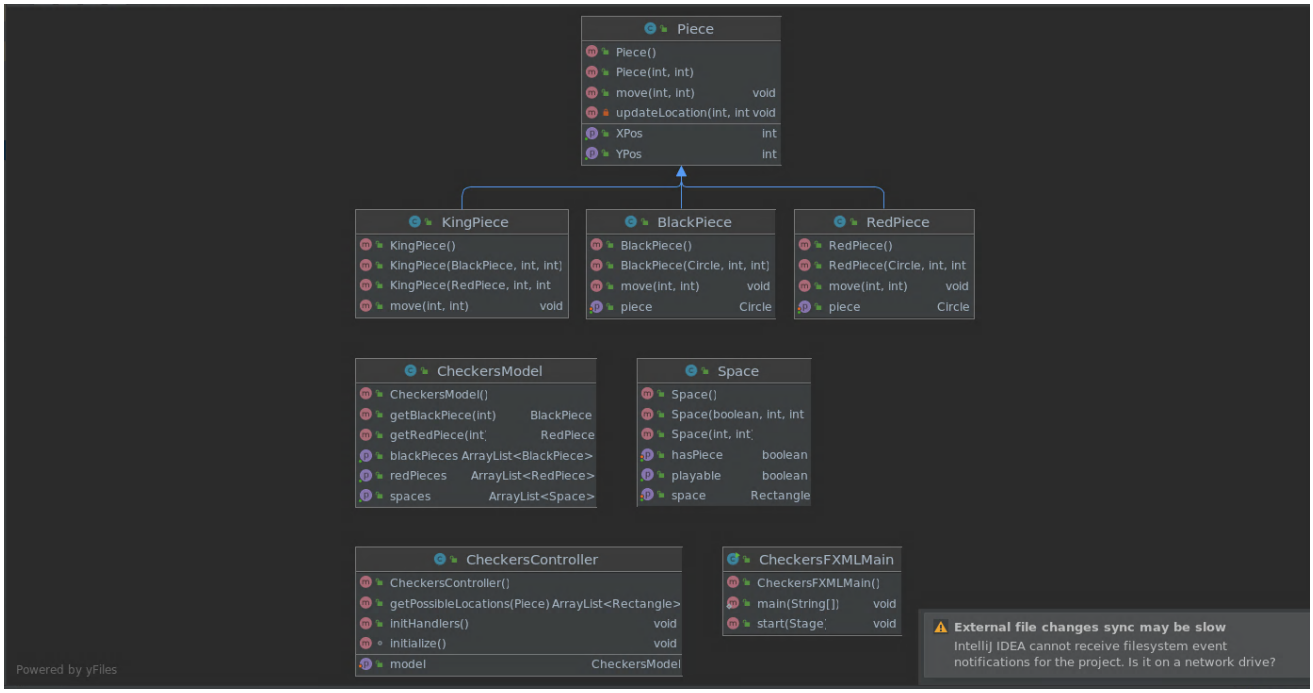
Piece	
Responsibilities	Collaborators

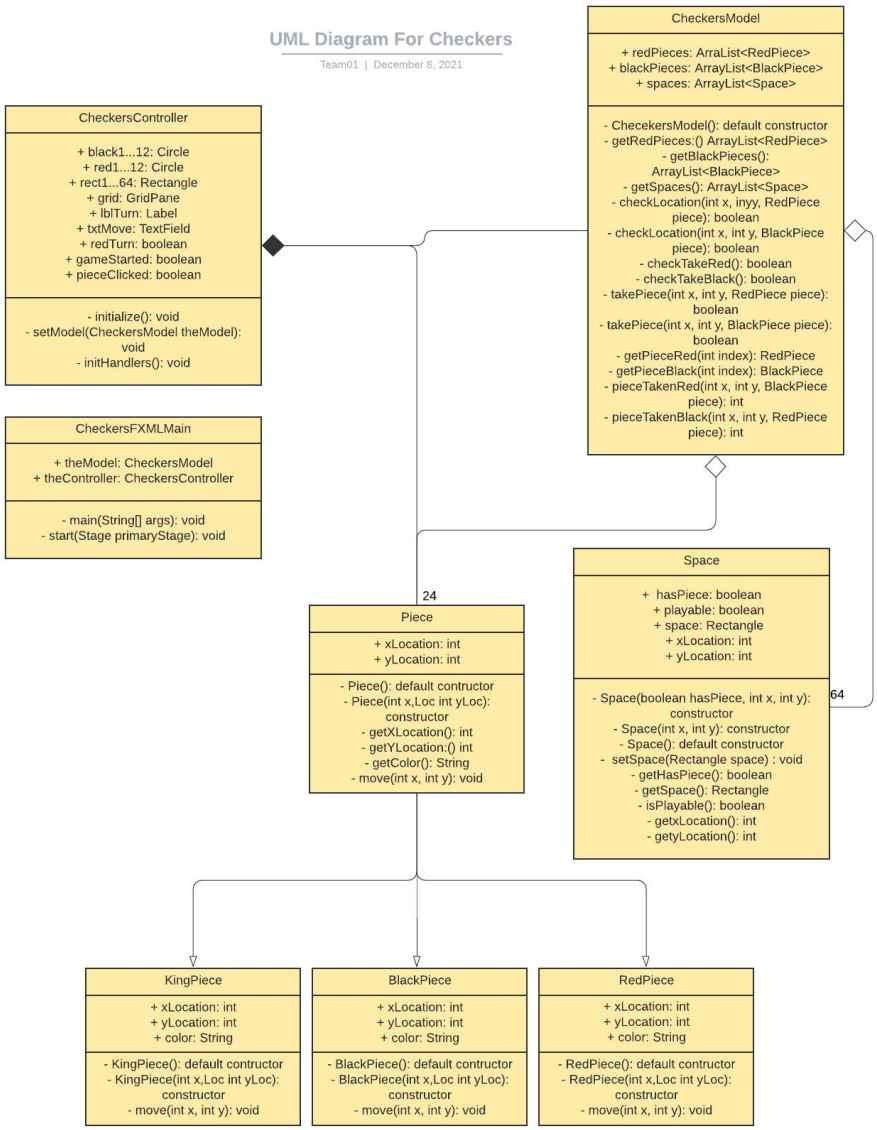
Parent Piece class Creates a general piece Has location of the piece	CheckersPiece KingPiece Board

CheckersController	
Responsibilities Event handlers for pieces and buttons Conditions for pieces (Do they take an enemy piece?) Conditions for movement (Where can a piece move depending on its type?)	Collaborators Model

CheckersFXMLMain	
Responsibilities A test game?	Collaborators

checkersfxml.fxml	
Responsibilities Visually creates the board with squares	Collaborators


Piece
Piece()
Piece(int, int)
move(int, int) void
updateLocation(int, int void
XPos int
YPos int
KingPiece
KingPiece()
KingPiece(BlackPiece, int, int)
KingPiece(RedPiece, int, int
move(int, int) void
BlackPiece
BlackPiece()
BlackPiece(Circle, int, int)
move(int, int) void
piece Circle
RedPiece
RedPiece()
RedPiece(Circle, int, int
move(int, int) void
piece Circle
CheckersModel
CheckersModel()
getBlackPiece(int) BlackPiece
getRedPiece(int) RedPiece
blackPieces ArrayList<BlackPiece>
redPieces ArrayList<RedPiece>
spaces ArrayList<Space>
Space
Space()
Space(boolean, int, int
Space(int, int)
hasPiece boolean
playable boolean
space Rectangle
CheckersController
CheckersController()
getPossibleLocations(Piece) ArrayList<Rectangle>
initHandlers() void
initialize() void
model CheckersModel
CheckersFXMLMain
CheckersFXMLMain()
main(String[]) void
start(Stage) void
External file changes sync may be slow
IntelliJ IDEA cannot receive filesystem event notifications for the project. Is it on a network drive?
Powered by yFiles



User Stories
A checker space should be able to be clicked once in order to select the piece and then the user should be able to click the direction that the piece should go out of a couple of selection options for movement.
Checkers should be able to move to diagonal squares in their given direction. Checkers should also be able to jump over opposing checkers, given that they are not blocked.
Once a Checker gets to the opposite side of the board it will be able to travel backwards (King Piece)
The game will start when the button is clicked and end at the termination of the game