CSE 110 Discussion 10/26

Daniel Pan - Object Oriented Design By Example

Goals of Object Oriented Design

- Robustness, adaptability, reusability, abstraction, encapsulation, modularity
- The design is driven from user stories
- Design should be DRY and SRP
- Code should be easily understood by the project team

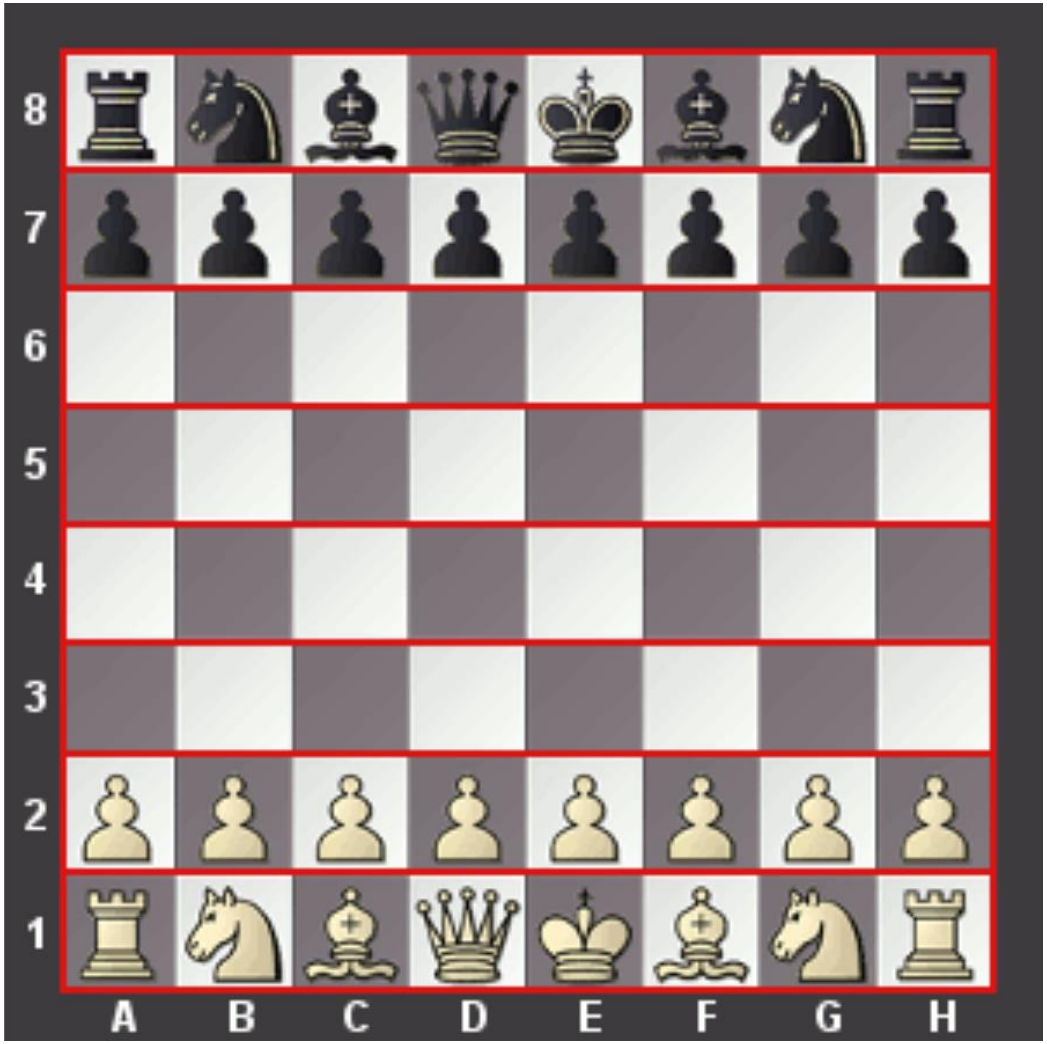
Definitions

- **Don't Repeat Yourself (DRY)** is a principle of software development, aimed at reducing repetition. The DRY principle is stated as "Every piece of knowledge must have a single, unambiguous, authoritative representation within a system".
- **Single Responsibility Principle (SRP)** states that every module or class should have responsibility over a single part of the functionality provided by the software, and that responsibility should be entirely encapsulated by the class. All its services should be narrowly aligned within that responsibility.

Definitions

- **"Is A"**
  - **One object is a specialized example of another**
  - **Inheritance. Implemented in Java with extends keyword**
  - **Example: Knight is a job class**
- **"Has A"**
  - **One object is a component of another**
  - **Composition. Implemented by one object having another as a field**
  - **Example: Motorcycle has an Engine**

Chess Game



Chess Game – User Stories

- As an user, I can start a Chess game so that users can play Chess.

- QUESTIONS:
  - 1. How would I create a scenario for this User Story?
    - Given ___ When ___ Then ___
  - 2. What steps should I take to design the scenario?

  (TAKE A FEW MINUTES TO THINK ABOUT AND WRITE DOWN A SCENARIO)

Creating a Scenario

- As an user, I can start a Chess game so that users can play Chess.

- GIVEN the Chess app is open, WHEN the user selects 'New Game', THEN a board is created with 8 rows and 8 columns AND 32 pieces (16 light and 16 dark) in their starting positions including 8 pawns, 2 rooks, 2 knights, 2 bishops, 1 queen, 1 king for each color.

Designing the Scenario

- GIVEN the Chess app is open, WHEN the user selects 'New Game', THEN a board is created with 8 rows and 8 columns AND 32 pieces (16 light and 16 dark) in their starting positions including 8 pawns, 2 rooks, 2 knights, 2 bishops, 1 queen, 1 king for each color.

- Identify the objects

(TAKE A FEW MINUTES TO THINK ABOUT AND WRITE DOWN A SCENARIO)

Designing the Scenario

- GIVEN the Chess app is open, WHEN the user selects 'New Game', THEN a board is created with 8 rows and 8 columns AND 32 pieces (16 light and 16 dark) in their starting positions including 8 pawns, 2 rooks, 2 knights, 2 bishops, 1 queen, 1 king for each color.

- Identify the objects
  - Board
  - PieceSet, Pieces, Knight, Pawn, etc
  - Game
  - Player

Designing the Scenario

- GIVEN the Chess app is open, WHEN the user selects 'New Game', THEN a board is created with 8 rows and 8 columns AND 32 pieces (16 light and 16 dark) in their starting positions including 8 pawns, 2 rooks, 2 knights, 2 bishops, 1 queen, 1 king for each color.

- Identify the messages

(TAKE A FEW MINUTES TO THINK ABOUT AND WRITE DOWN MESSAGES)

Designing the Scenario

- Identify the messages
  - Board.Squares
  - Board.PieceSet
  - PieceSet.List_RemainingPieces
  - PieceSet.Piece
  - Game.Players
  - Game.Turn
  - Game.Result
  - Player.PieceColor
  - Player.Engine

Designing the Scenario

- GIVEN the Chess app is open, WHEN the user selects 'New Game', THEN a board is created with 8 rows and 8 columns AND 32 pieces (16 light and 16 dark) in their starting positions including 8 pawns, 2 rooks, 2 knights, 2 bishops, 1 queen, 1 king for each color.

- Assemble into classes

(TAKE A FEW MINUTES TO THINK ABOUT AND ASSEMBLE THE OBJECTS AND MESSAGES INTO CLASSES)

EX. Board has a Pieceset. King extends Piece.

Designing the Scenario

```
Class Board {
   PieceSet[ ] pieceSets;
   Square[ ][ ] squares;
}

Class PieceSet {
   List<Piece> pieces;
   PieceColor color;
}
```

Designing the Scenario

```
Class Piece {
   Square placeAt;
   Square[ ] validMoves();
   Square[ ] attackSquares();
}

Class Pawn extends Piece {
   bool promoted;
}
```

Designing the Scenario

- GIVEN the Chess app is open, WHEN the user selects 'New Game', THEN a board is created with 8 rows and 8 columns AND 32 pieces (16 light and 16 dark) in their starting positions including 8 pawns, 2 rooks, 2 knights, 2 bishops, 1 queen, 1 king for each color.

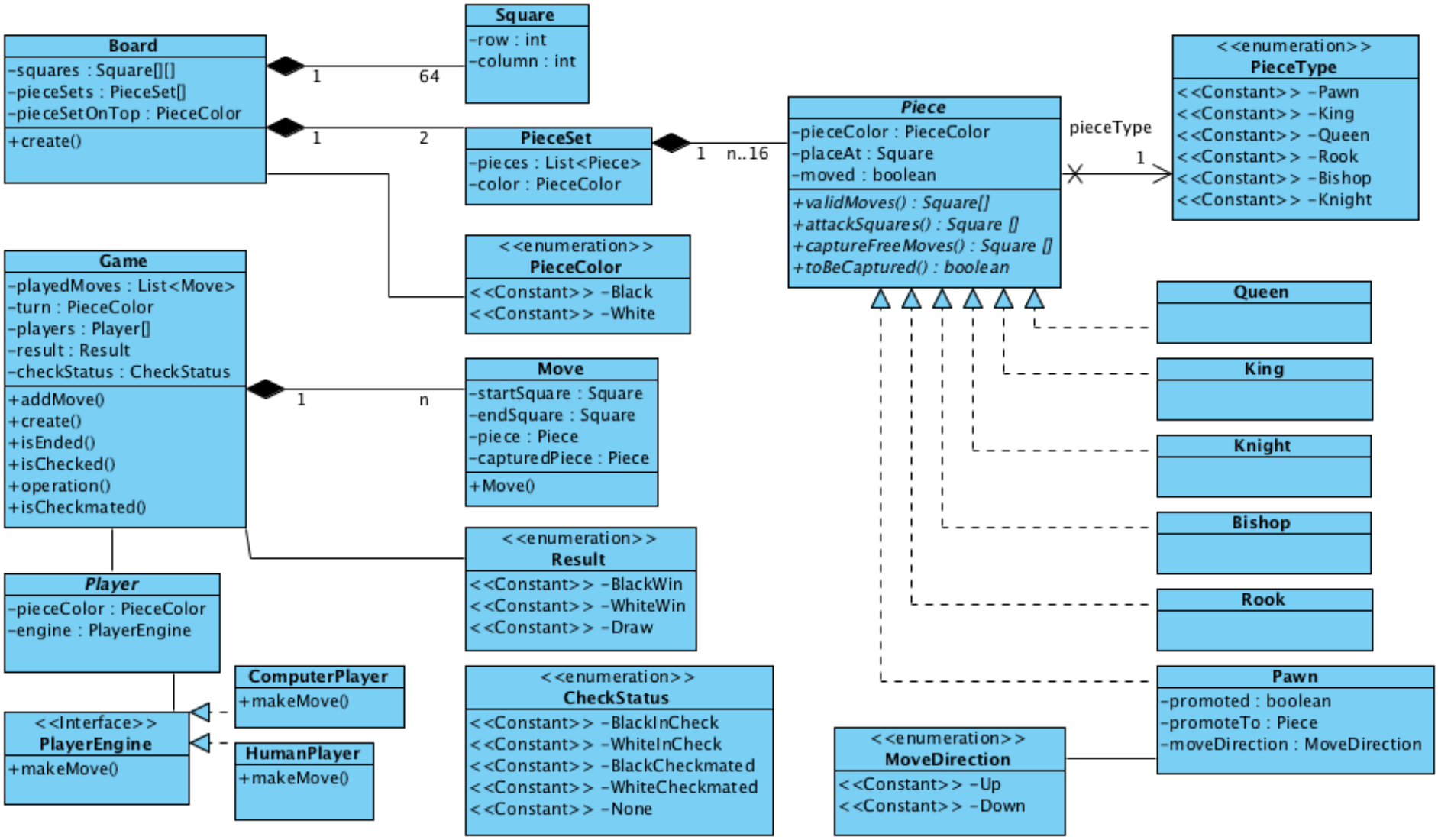
- I leave it as an exercise for you to think about how you would assemble the rest of the classes

- Remember to verify that right objects call right messages

UML Diagram

- Unified Modeling Language (UML) is a general developmental modeling language used to visualize the design of a system
- Originally intended for object oriented design documentation, but has since been extended to a larger set of design documentation and used in many contexts
- Visualizes a system's architectural blueprints in a diagram including system components, activities, an external user interface, how components and interfaces interact, and how the overall system runs

UML Diagram


Square
-row : int
-column : int
Board
-squares : Square[][]
-pieceSets : PieceSet[]
-pieceSetOnTop : PieceColor
+create()
1
64
1
2
PieceSet
-pieces : List<Piece>
-color : PieceColor
1
n..16
Piece
-pieceColor : PieceColor
-placeAt : Square
-moved : boolean
+validMoves() : Square[]
+attackSquares() : Square []
+captureFreeMoves() : Square []
+toBeCaptured() : boolean
pieceType
1
<<enumeration>>
PieceType
<<Constant>> -Pawn
<<Constant>> -King
<<Constant>> -Queen
<<Constant>> -Rook
<<Constant>> -Bishop
<<Constant>> -Knight
<<enumeration>>
PieceColor
<<Constant>> -Black
<<Constant>> -White
Game
-playedMoves : List<Move>
-turn : PieceColor
-players : Player[]
-result : Result
-checkStatus : CheckStatus
+addMove()
+create()
+isEnded()
+isChecked()
+operation()
+isCheckmated()
1
n
Move
-startSquare : Square
-endSquare : Square
-piece : Piece
-capturedPiece : Piece
+Move()
<<enumeration>>
Result
<<Constant>> -BlackWin
<<Constant>> -WhiteWin
<<Constant>> -Draw
Queen
King
Knight
Bishop
Rook
Pawn
-promoted : boolean
-promoteTo : Piece
-moveDirection : MoveDirection
Player
-pieceColor : PieceColor
-engine : PlayerEngine
ComputerPlayer
+makeMove()
<<Interface>>
PlayerEngine
+makeMove()
HumanPlayer
+makeMove()
<<enumeration>>
CheckStatus
<<Constant>> -BlackInCheck
<<Constant>> -WhiteInCheck
<<Constant>> -BlackCheckmated
<<Constant>> -WhiteCheckmated
<<Constant>> -None
<<enumeration>>
MoveDirection
<<Constant>> -Up
<<Constant>> -Down

Designing the Scenario

- GIVEN the Chess app is open, WHEN the user selects 'New Game', THEN a board is created with 8 rows and 8 columns AND 32 pieces (16 light and 16 dark) in their starting positions including 8 pawns, 2 rooks, 2 knights, 2 bishops, 1 queen, 1 king for each color.

- Identify the objects
- Identify the messages
- Assemble into classes
- Verify that right objects call right messages

Tips

- Everything is based off of the user story. Write your code so that it sounds like your scenario which is based off of the user story.
- Try to break things down into objects and messages and use them to write SRP classes. Don't violate DRY.
- Always look out for how you can improve your design both before and after you begin coding.

Additional User Stories for Practice

- As an User, I want to move a Chess Piece so that I can play my Turn.
- As a Map App user, I want to see nearby restaurants on the map so that I know my options on where to eat.
- As an UCSD Free Food App user, I want to see where I can find free food on campus.
- I will get you started on the first one. The second and third one I leave as exercises for you to work out.

Additional User Stories for Practice

- As an User, I want to move a Chess Piece so that I can play my Turn.
  - Piece is at position rank (1-8) and file (A-H). Ex. Pawn at A2

Additional User Stories for Practice

- As an User, I want to move a Chess Piece so that I can play my Turn.
  - Piece is at position rank (1-8) and file (A-H). Ex: Pawn at A2
  - Piece is moved to new rank and file. Ex: Pawn to A3

Additional User Stories for Practice

- As an User, I want to move a Chess Piece so that I can play my Turn.
  - Piece is at position rank (1-8) and file (A-H). Ex: Pawn at A2
  - Piece is moved to new rank and file. Ex: Pawn to A3
  - What if I try to move a piece to a spot occupied by an enemy?

Additional User Stories for Practice

- As an User, I want to move a Chess Piece so that I can play my Turn.
  - Piece is at position rank (1-8) and file (A-H). Ex: Pawn at A2
  - Piece is moved to new rank and file. Ex: Pawn to A3
  - What if I try to move a piece to a spot occupied by an enemy?
  - After my turn ends, it becomes the other player's turn

Additional User Stories for Practice

- As an User, I want to move a Chess Piece so that I can play my Turn.
  - Piece is at position rank (1-8) and file (A-H). Ex: Pawn at A2
  - Piece is moved to new rank and file. Ex: Pawn to A3
  - What if I try to move a piece to a spot occupied by an enemy?
  - After my turn ends, it becomes the other player's turn

I'll leave you to think about how you would design this. Remember to begin by designing a scenario. For the solution, feel free to go back to the UML Diagram.