



CS120 - INTRODUCTION TO  
OBJECT ORIENTED PROGRAMMING

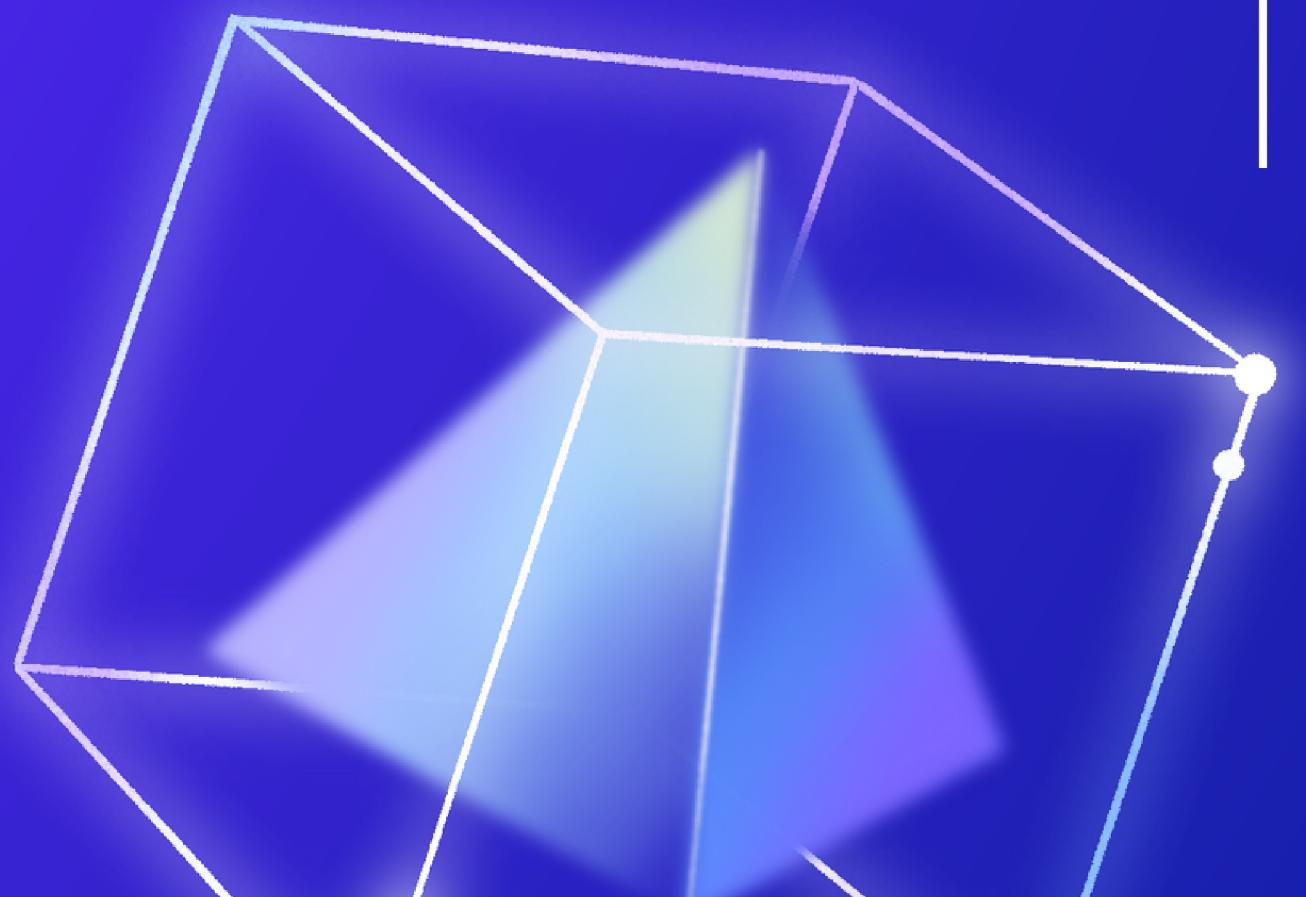
# QUARTO

by Hrachya Hovakimyan  
and Khachik Zakaryan

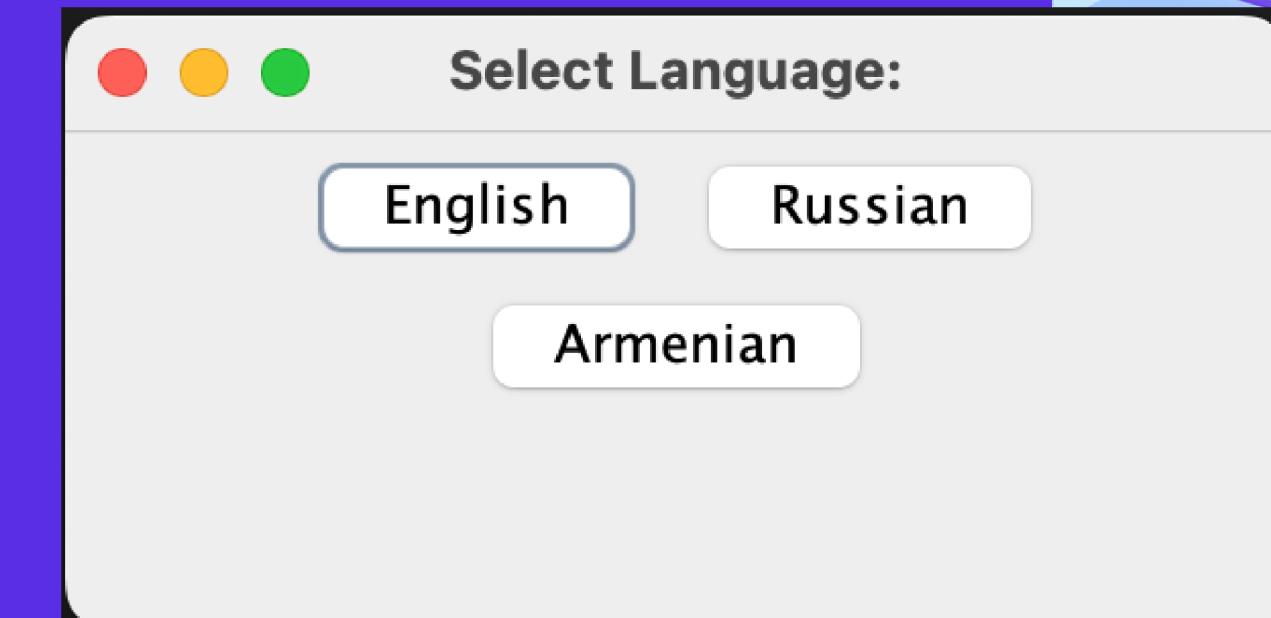
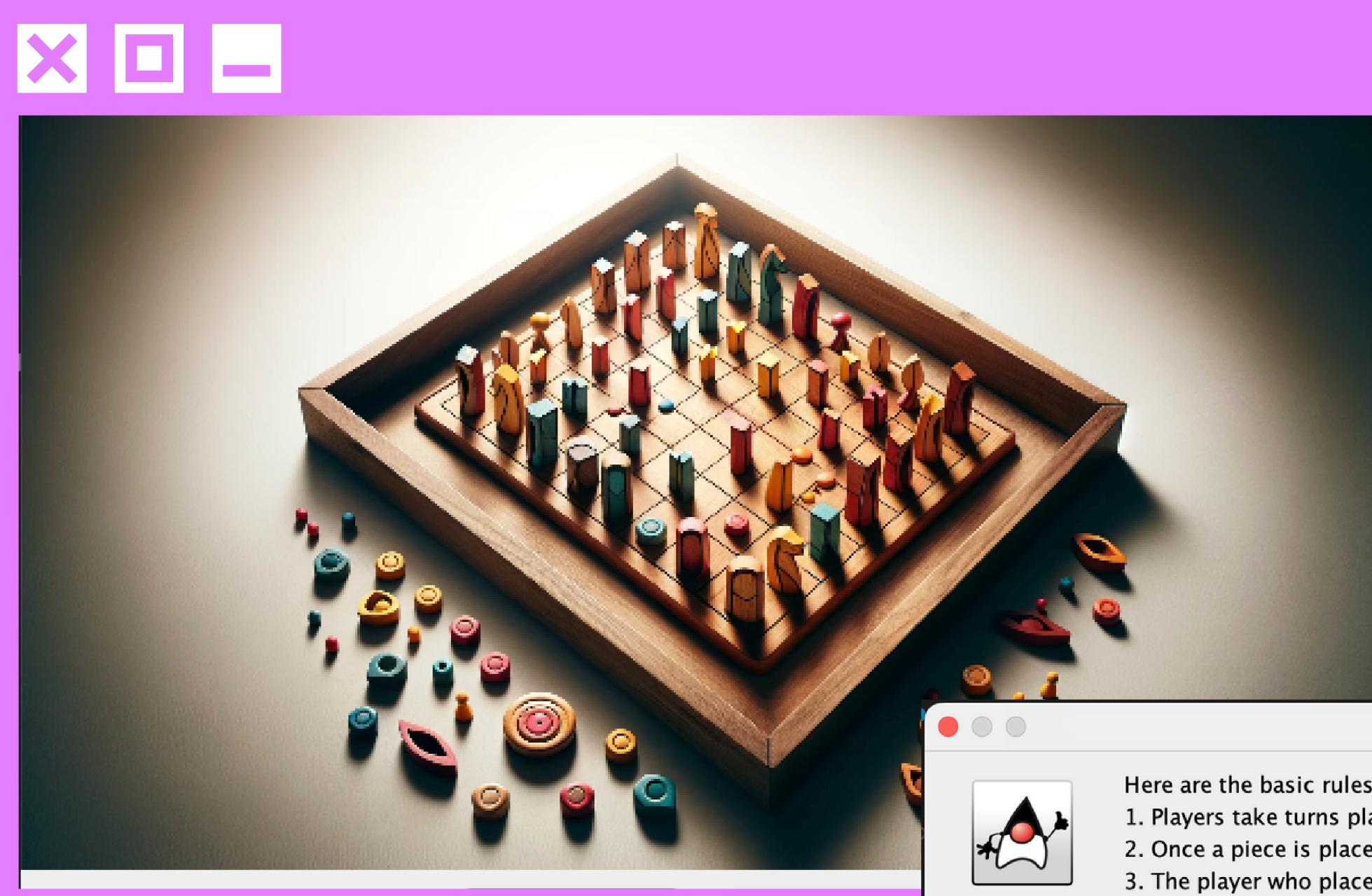


# TABLE OF CONTENTS

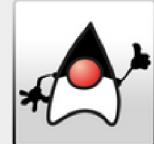
- Introduction
- Project Structure
- Results and Demo game
- QA



# INTRODUCTION



Game Rules



Here are the basic rules:

1. Players take turns placing pieces on the board.
2. Once a piece is placed on the board, it cannot be moved or removed.
3. The player who places a piece chooses which piece the opponent will play next.
4. The game continues until a player forms a line of four pieces with a common attribute (e.g., four tall pieces, four dark pieces, etc.).
5. If the board is filled without any player achieving a winning line, the game ends in a draw.

OK



last login: Thu May 9 16:36:51 on ttys060  
achzak@Khachiks-MacBook-Pro ~ % █

█

# PROJECT STRUCTURE



CS120-IntroToOOP-Project Public

master · 1 Branch · 0 Tags

zakaryan2107 remove gitignore rule · 32ee2a0 · 19 hours ago · 62 Commits

src · improving ui · yesterday

.gitignore · remove gitignore rule · 19 hours ago

README.md · Adding multilanguage rules selection · 2 days ago

[README](#)

## Quarto Game

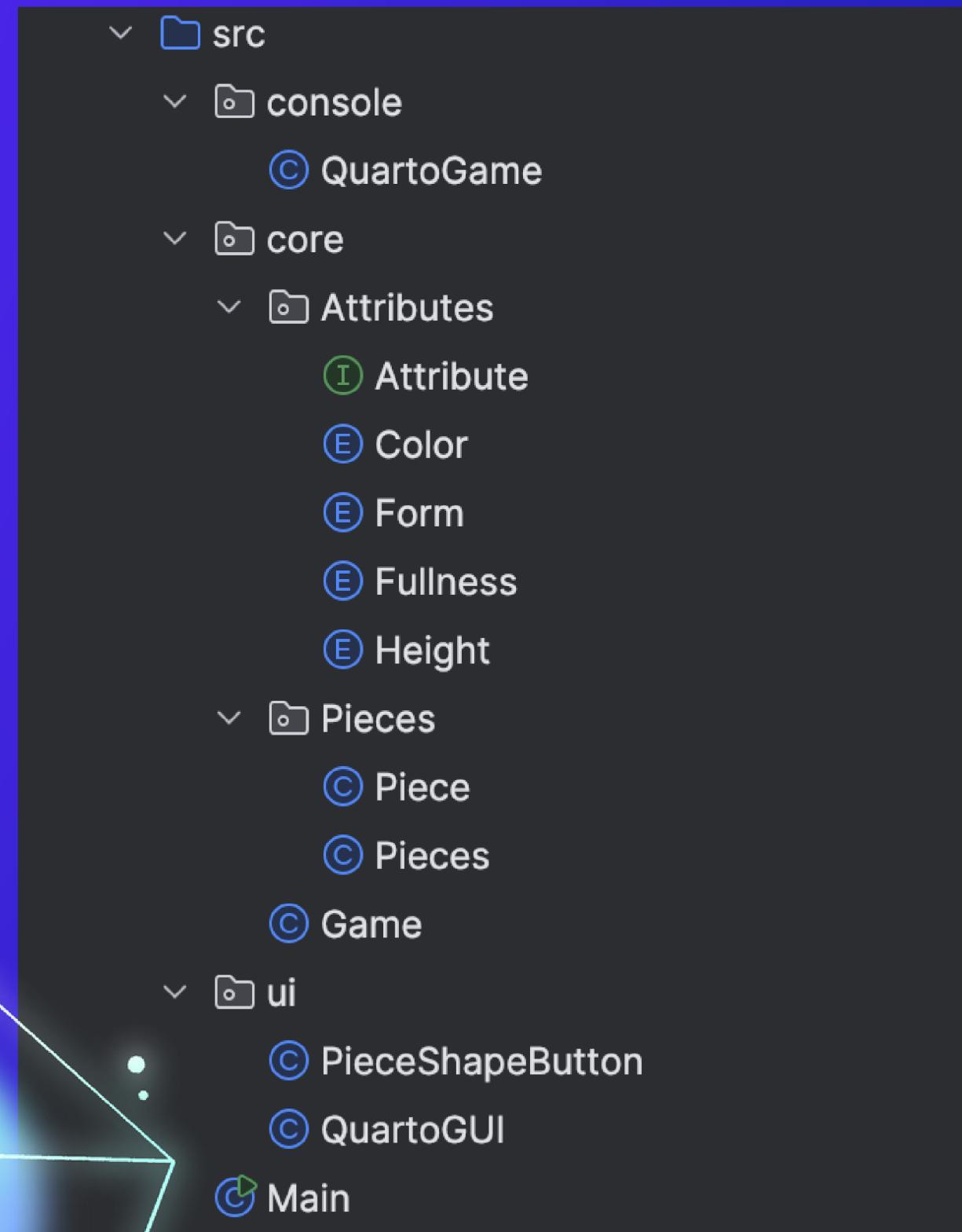
Welcome to Quarto Game, a Java implementation of the classic board game Quarto.

### About

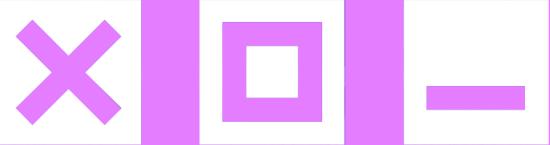
Quarto is a two-player abstract strategy game where players take turns placing pieces on a 4x4 board. The goal is to be the first to create a line of four pieces with a common attribute (color, shape, height, or fullness).

This project provides a graphical user interface (GUI) for playing Quarto against another human player. It features intuitive controls, visual representation of game pieces, and win/draw detection.

### Features



# enum TYPES



## Color

• `m ↳ Color()`  
• `f ↳ DARK`  
• `f ↳ LIGHT`  
• `◊ ↳ valueOf(String) Color`  
• `◊ ↳ values() Color[]`

## Attribute

## Form

• `m ↳ Form()`  
• `f ↳ ROUND`  
• `f ↳ SQUARE`  
• `◊ ↳ valueOf(String) Form`  
• `◊ ↳ values() Form[]`

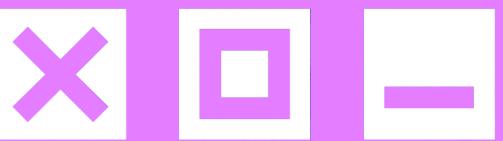
## Height

• `m ↳ Height()`  
• `f ↳ TALL`  
• `f ↳ SHORT`  
• `◊ ↳ valueOf(String) Height`  
• `◊ ↳ values() Height[]`

## Fullness

• `m ↳ Fullness()`  
• `f ↳ SOLID`  
• `f ↳ HOLLOW`  
• `◊ ↳ values() Fullness[]`  
• `◊ ↳ valueOf(String) Fullness`

# INTERFACE



```
/*
 * The Attribute interface represents a generic attribute for Quarto game pieces.
 * Implementing classes define specific attributes such as color, size, shape, etc.
 */
public interface Attribute {} 6 usages 4 implementations · zakaryan2107
```

Interface core.Attributes.Attribute of core.Attributes 6 usages

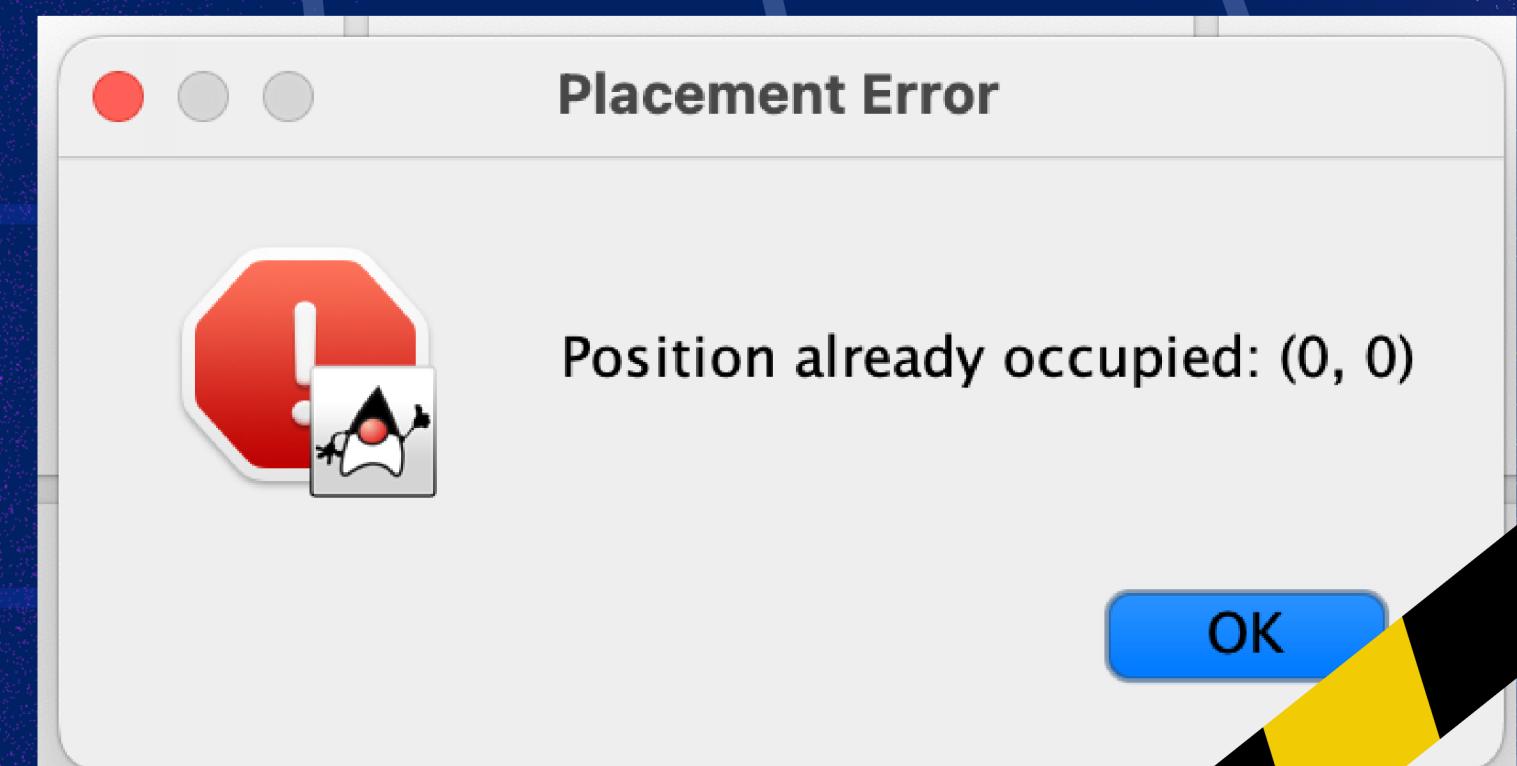
Color.java 7 ↗ public enum Color implements Attribute {  
Form.java 7 ↗ public enum Form implements Attribute {  
Fullness.java 7 ↗ public enum Fullness implements Attribute {  
Height.java 7 ↗ public enum Height implements Attribute {  
Game.java 183 ↗ private <TAttribute extends Attribute> boolean all  
Game.java 192 ↗ private <TAttribute extends Attribute> boolean isE

# EXCEPTION HANDLING

```
package core.exceptions;

public class InvalidPiecePlacementException extends Exception {

    public InvalidPiecePlacementException(String message) {
        super(message);
    }
}
```



# Piece Class



## © `Piece`

<code>m ⌂ Piece(Height, Fullness, Form, Color)</code>	
<code>o f ⌂ fullness</code>	Fullness
<code>o f ⌂ height</code>	Height
<code>o f ⌂ form</code>	Form
<code>o f ⌂ color</code>	Color
<code>m ⌂ getFullness()</code>	Fullness
<code>m ⌂ matchesAttribute(TAttribute)</code>	boolean
<code>m ⌂ allPiecesMatch(Piece[], TAttribute)</code>	boolean
<code>m ⌂ getHeight()</code>	Height
<code>m ⌂ getForm()</code>	Form
<code>m ⌂ getColor()</code>	Color
<code>m ⌂ toString()</code>	String

## © `Pieces`

<code>m ⌂ Pieces()</code>	
<code>o f ⌂ pieces</code>	List<Piece>
<code>m ⌂ getAllPieces()</code>	List<Piece>
<code>m ⌂ initializePieces()</code>	List<Piece>



# Many Pieces

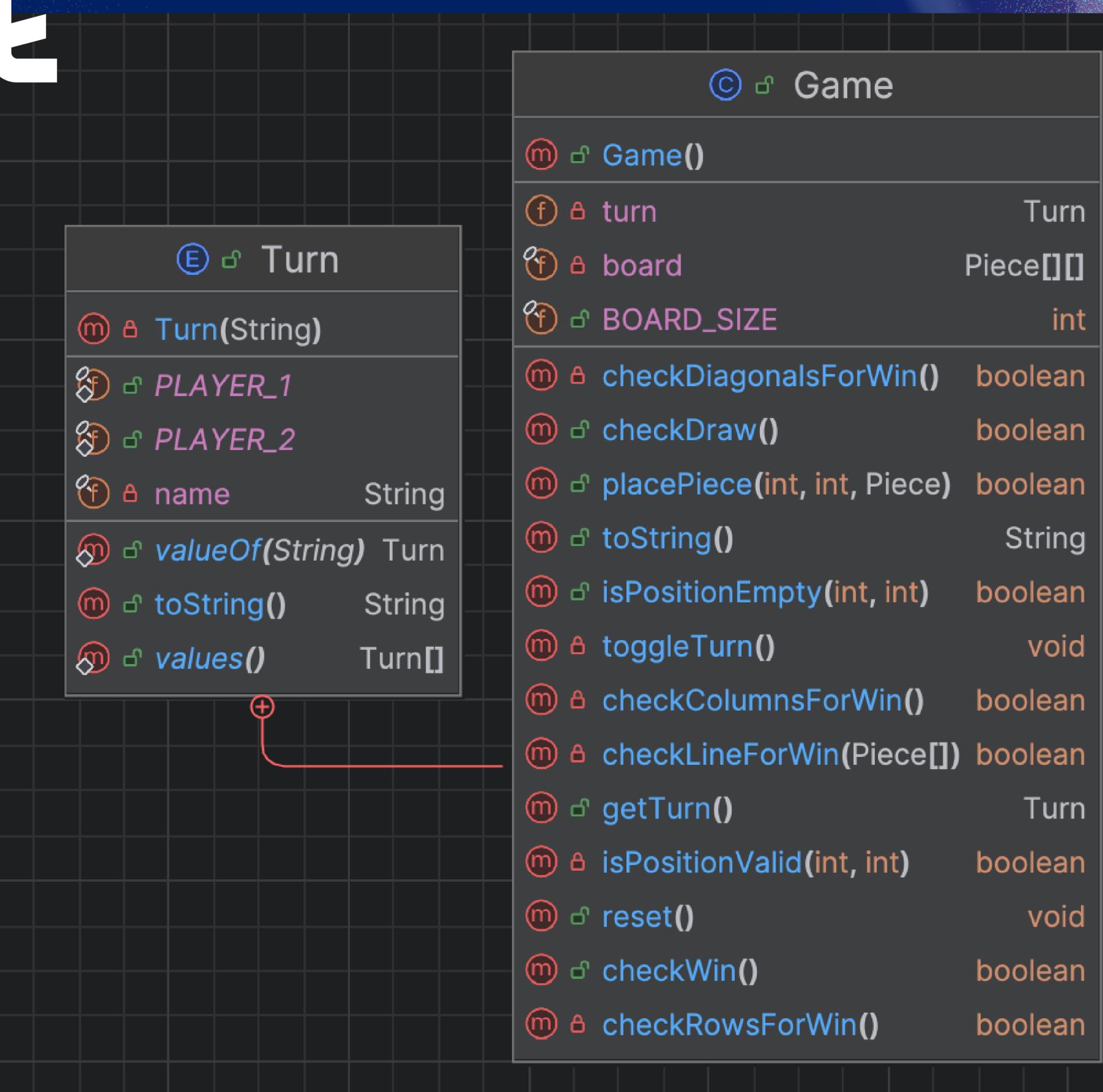
```
/**  
 * List to store all possible game pieces  
 */  
private static final List<Piece> pieces = initializePieces(); 1 usage  
  
/**  
 * Initializes the list of game pieces  
 * @return The list of generated game pieces  
 */  
private static List<Piece> initializePieces() { 1 usage • Hrachya Hovakimyan  
    List<Piece> pieces = new ArrayList<>();  
    for (Height height : Height.values()) {  
        for (Fullness fullness : Fullness.values()) {  
            for (Color color : Color.values()) {  
                for (Form form : Form.values()) {  
                    pieces.add(new Piece(height, fullness, form, color)); 1 usage • Hrachya Hovakimyan  
                }  
            }  
        }  
    }  
    return pieces;  
}
```

# WHAT ABOUT THE GAME LOGIC?

Our vision



```
/**  
 * Resets the game by clearing the game board and setting the turn to Player 1.  
 */  
  
public void reset() { 1 usage  ↳ zakaryan2107  
    for (int i = 0; i < BOARD_SIZE; i++) {  
        for (int j = 0; j < BOARD_SIZE; j++) {  
            board[i][j] = null;  
        }  
    }  
    // Reset turn to PLAYER_1  
    turn = Turn.PLAYER_1;  
}
```



# "WHITE AND BLACK" GAME REPRESENTATION



```
public void startGame() throws InvalidPiecePlacementException{ no usages }

    System.out.println("Welcome to Quarto!");
    System.out.println("Let's start the game.");

    while (!game.checkWin() && !pieces.isEmpty()) {
        // Player 1's turn
        System.out.println(game); // Print the current game board
        System.out.println("Player 1's turn:");
        Piece pieceForOpponent = selectPieceForOpponent(1);
        int[] position = selectPosition();
        game.placePiece(position[0], position[1], pieceForOpponent);
        if (game.checkWin()) {
            System.out.println("Player 1 wins!");
            break;
        }
        if (pieces.isEmpty()) {
            System.out.println("No more pieces to select. It's a draw!");
            break;
        }

        // Player 2's turn
        System.out.println(game); // Print the current game board
        System.out.println("Player 2's turn:");
        pieceForOpponent = selectPieceForOpponent(2);
        position = selectPosition();
        game.placePiece(position[0], position[1], pieceForOpponent);
        if (game.checkWin()) {
            System.out.println("Player 2 wins!");
            break;
        }

        // Check for draw after the loop ends
        if (!game.checkWin() && pieces.isEmpty()) {
            System.out.println("No more pieces to select. It's a draw!");
        }
    }
}
```

# GUI REPRESENTATION

© QuartoGUI	
m d QuartoGUI()	
f o infoPanel	JPanel
d SIDE_BUTTON_SIZE	int
f o game	Game
f o winCountLabel	JLabel
f o selectedPieceShapeButton	PieceShapeButton
f o boardPanel	JPanel
f o player1Wins	int
f o infoLabel	JLabel
f o label	String
f o boardButtons	PieceShapeButton[] []
f o pieceWrapperPanel	JPanel
f o selectedPiece	Piece
d BOARD_BUTTON_SIZE	int
d WIDTH	int
f o player2Wins	int
f o controlPanel	JPanel
d HEIGHT	int
m o constructPiecesPanel()	JPanel
m o showRulesArmenian()	void
m o showRules()	void
m o constructInfoPanel()	JPanel
m o getButton(PieceShapeButton[] [], int, int)	PieceShapeButton
m o getPieceShapeButton(Piece)	PieceShapeButton
m o updateWinCountLabel()	void
m o showRulesEnglish()	void
m o restartGame()	void
m o updateLabel(String)	void
m o resetGame()	void
m o showWinDialog()	void
m o constructControlPanel()	JPanel
m o resetBoardButtons()	void
m o showDrawDialog()	void
m o resetLeftSidePiecePanel()	void
m o showRulesRussian()	void
m o showInitialPopup()	void
m o constructBoardButtonsPanel()	JPanel



# THE “main” PART....

```
1 import javax.swing.SwingUtilities;  
2 import ui.QuartoGUI;  
3 /**  
4 * Main Class for execution  
5 */  
6 ▷ public class Main { ↳ zakaryan2107  
7     /**  
8      * Execute the QuartoGUI constructor within the event dispatch thread  
9      * @param args Argument (Not used)  
10     */  
11     public static void main(String[] args) { ↳ zakaryan2107  
12         SwingUtilities.invokeLater(QuartoGUI::new);  
13     }  
14 }  
15 }
```

```
public class Main {  
    public static void main (String[] args) {  
        System.out.print("Thank you!");  
    }  
}
```

