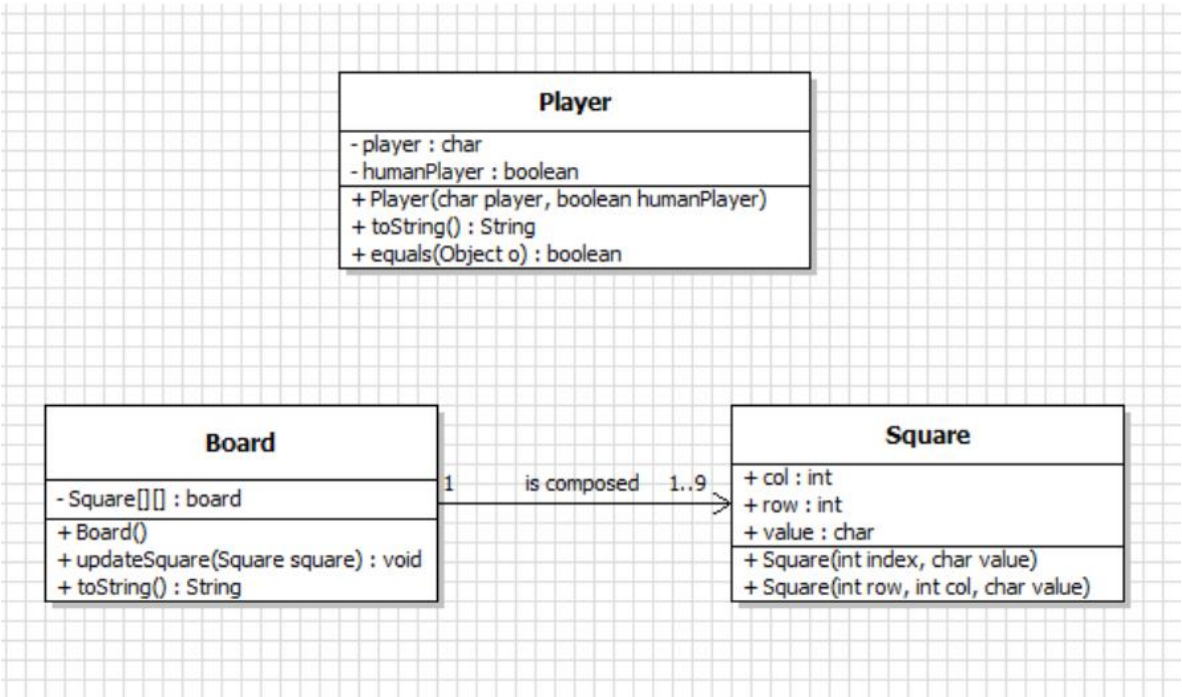
DATA LAYER



BOARD CLASS

Overriding default constructor

```
public class Board {

    private Square[][] board;

    public Board() {

        char value = '0';
        board = new Square[3][3];

        for (int row = 0; row < board.length; row++) {
            for (int col = 0; col < board.length; col++) {
                Square square = new Square(row, col, (char) (++value));
                board[row][col] = square;
            }
        }
    }

    public Square[][] getBoard() {...}

    public void updateSquare(Square square) {...}

    @Override
    public String toString() {...}
}
```

PRINTING USER DEFINED METHOD

```
public static void printBoard(Board board) {
    System.out.println(board);
}
```

```
tictactoe.data.Board@530daa
```

TO STRING METHOD

Important, do not forget it

```
@Override
public String toString() {
    String printBoard = "\n";

    for (int row = 0; row < board.length; row++) {
        printBoard = printBoard.concat("\t|");
        for (int col = 0; col < board.length; col++) {
            printBoard = printBoard.concat(
                    String.valueOf(board[row][col]).concat("|"));
        }
        printBoard = printBoard.concat("\n");
    }
    return printBoard;
}
```

The return type is String

OVERRIDING TOSTRING METHOD

```
public static void printBoard(Board board) {
    System.out.println(board);
}
```

```
|1|2|3|
|4|5|6|
|7|8|9|
```

```java
public class Square {

    private int row;
    private int col;
    private char value;

    public Square(int row, int col, char value) {
        this.row = row;
        this.col = col;
        this.value = value;
    }

    public Square(int index, char value) {
        this.row = (index - 1) / 3;
        this.col = (index - 1) % 3;
        this.value = value;
    }

    public int getCol() {...}

    public void setCol(int col) {...}

    public int getRow() {...}

    public void setRow(int row) {...}

    public char getValue() {...}

    public void setValue(char value) {...}

    @Override
    public String toString() {...}
}
```

Overloading constructors

```java
@Override
public String toString() {
    return String.valueOf(this.getValue());
}
```

String.valueOf( ) is used to cast variables to String

valueOf(Object o)	String
valueOf(boolean bln)	String
valueOf(char c)	String
valueOf(char[] chars)	String
valueOf(double d)	String
valueOf(float f)	String
valueOf(int i)	String
valueOf(long l)	String
valueOf(char[] chars, int i, int i1)	String

PLAYER CLASS

```java
public class Player {

    private char player;
    private boolean humanPlayer;

    public Player(char player, boolean humanPlayer) {...}

    public char getPlayer() {...}

    public void setPlayer(char player) {...}

    public boolean isHumanPlayer() {...}

    @Override
    public String toString() {...}

    @Override
    public boolean equals(Object obj) {...}
}
```

OVERRIDING TOSTRING METHOD

```java
@Override
public String toString() {

    String printPlayer = "I am the player "
            + String.valueOf(this.getPlayer()) + " and I am ";

    if (this.isHumanPlayer()) {
        printPlayer = printPlayer.concat("HUMAN");
    } else {
        printPlayer = printPlayer.concat("A ROBOT");
    }
    return printPlayer;
}
```

OVERRIDING EQUALS METHOD

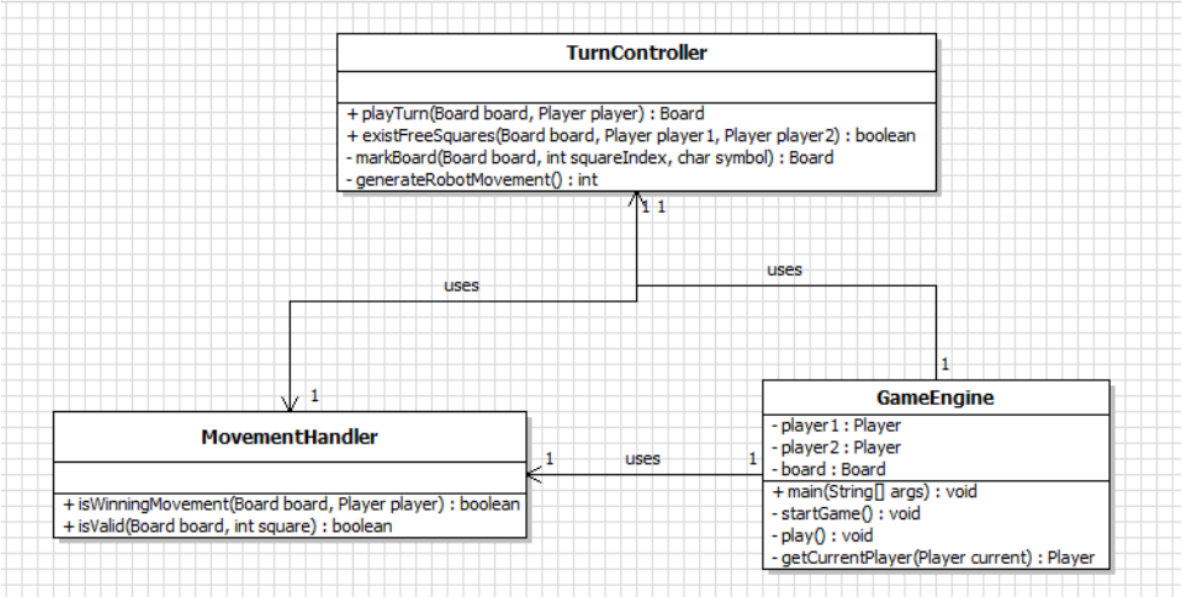
equals method allow us to **override the default way how two user defined objects are equals or not**

Important, do not forget it

If the two objects have the same player (Symbol) are equals

```java
@Override
public boolean equals(Object obj) {
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Player other = (Player) obj;
    if (this.player != other.player) {
        return false;
    }
    return true;
}
```

BUSINESS LOGIC LAYER



GAME ENGINE CLASS

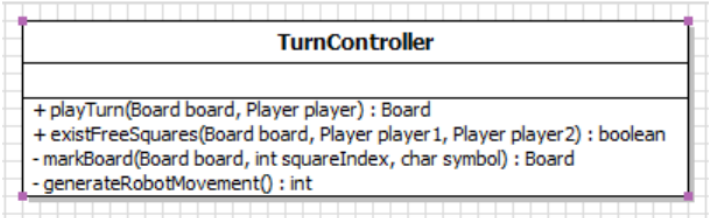
THIS IS THE GAME STARTING POINT

```
import tictactoe.data.Board;
import tictactoe.data.Player;
import tictactoe.ui.UI;

public class GameEngine {

    private static Player player1;
    private static Player player2;
    private static Board board;

    public static void main(String[] args) {...}

    private static void startGame() {...}

    private static void play() {...}

    private static Player getCurrentPlayer(Player current) {...}
}
```

Methods can be private

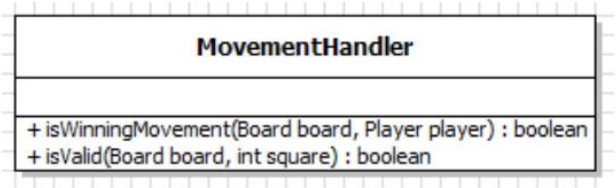
```java
import java.util.Random;
import tictactoe.data.Board;
import tictactoe.data.Player;
import tictactoe.data.Square;
import tictactoe.ui.UI;

public class TurnController {

    public static Board playTurn(Board board, Player player) {...}

    private static Board markBoard(Board board, int squareIndex, char symbol) {...}

    public static boolean existFreeSquares(Board board, Player player1, Player player2) {...}

    private static int generateRobotMovement() {...}
}
```

TURN CONTROLLER CLASS



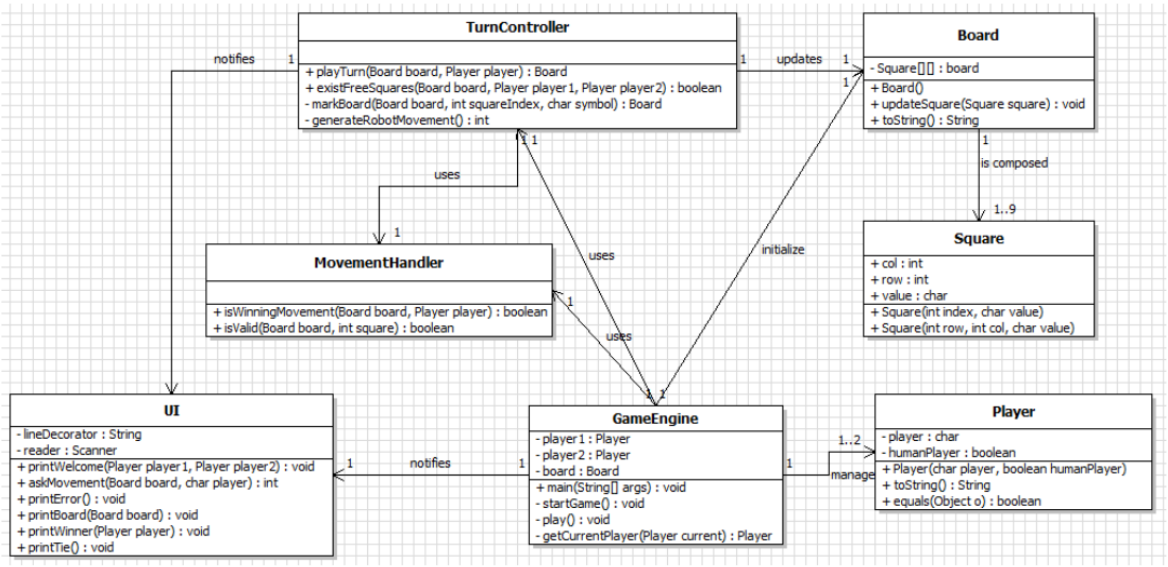
Class	Method signature	Function
TurnController	playTurn (Board , Player)	Handle movement turn Call movement validator Call board modifier
	markBoard (Board , int , char)	Modify board after a valid play
	existFreeSquares (Board , Player , Player)	Chek if there are available squares to play
	generateRobotMovement ( )	Generate random robot movement

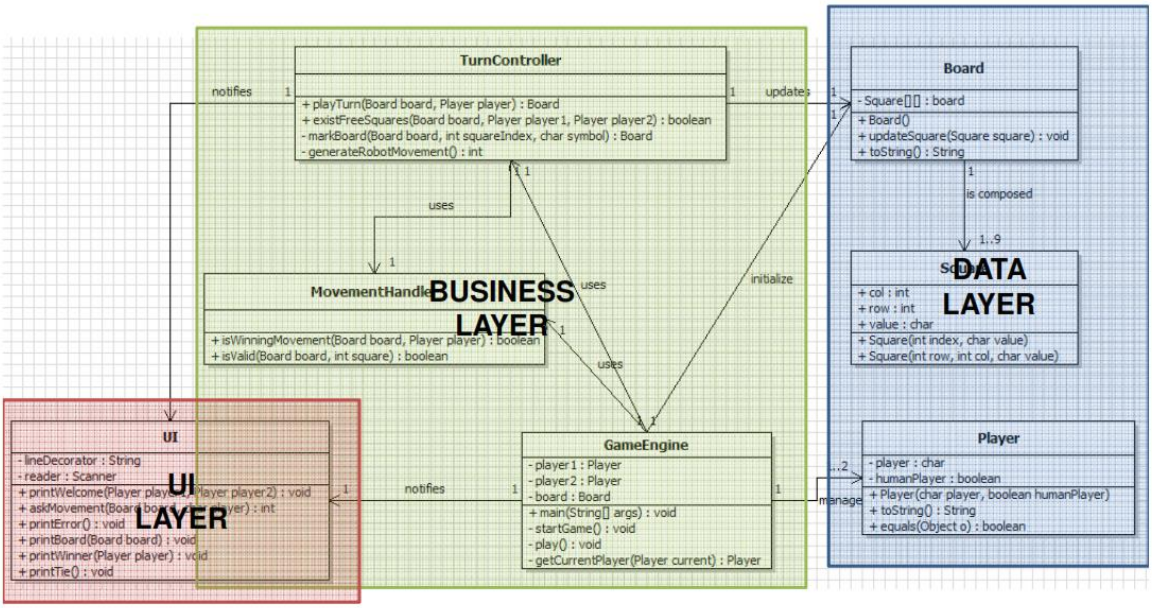
MOVEMENTHANDLER CLASS

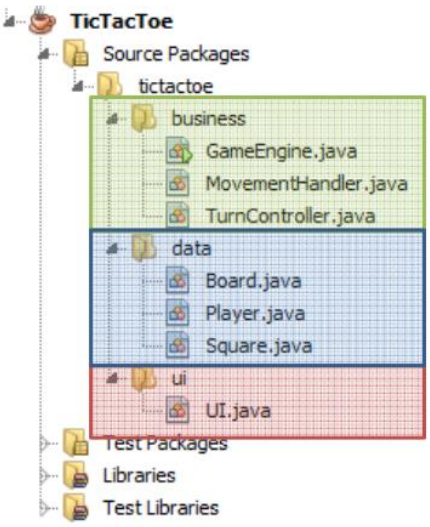


Class	Method signature	Function
MovementHandler	isValid (Board , int)	Check if a square selection is available to be marked
	isWinningMovement (Board , Player)	Check if the last movement causes the player's victory

UML CLASS DIAGRAM







- Three layers
  - Data
  - Business logic
  - UI