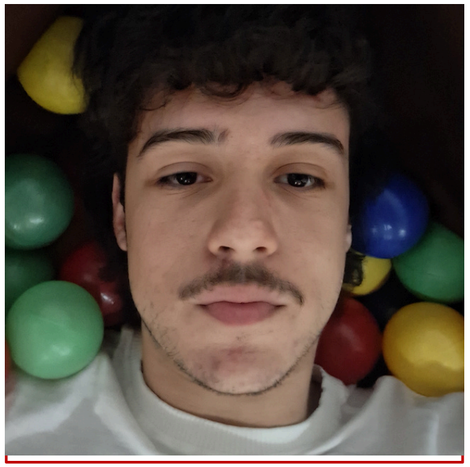
https://www.youtube.com/watch?v=pfTUkTQ3s-k

TCP - Trailer

GRUPO 02



Tabuleiro de Combate de Peças (TCP)

Rayan Raddatz, Enzo Lisboa, Natan Tristão, Gabriel Henrique e Iuri Kali

Facilitador: Rayan Raddatz

Técnicas de Construção de Programas – Turma A

Prof. Karina Kohl

2025/1

INSTITUTO DE INFORMÁTICA
UFRGS

DESCRIÇÃO DO PROBLEMA

Recriação do tradicional jogo de xadrez

Ideal para aplicar os quatro pilares do Paradigma de Orientação a Objeto

Abstração

Encapsulamento

Herança

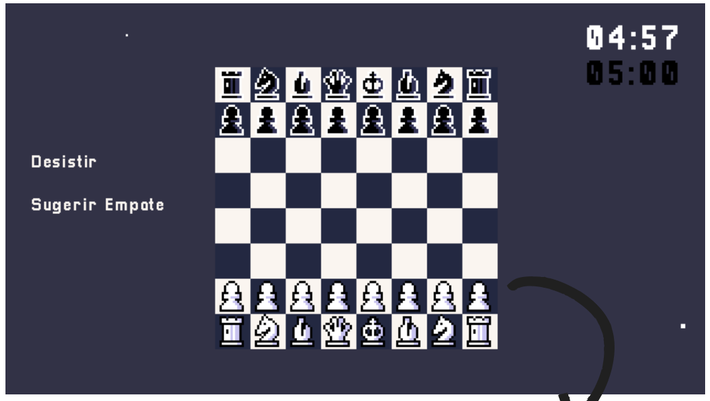
Polimorfismo

REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

Objetivos:

Interface amigável e simples para a usabilidade do usuário

Informar os movimentos válidos de uma peça caso o jogador clique nela



Exibir de forma gráfica um tabuleiro 8x8 e permitir a interação do usuário

Informar o vencedor caso haja um, ou empate caso não tenha um vencedor

INSTITUTO
DE INFORMÁTICA
UFRGS

ARTE

A free pack of chess pieces for games and other projects, including standard and fairy chess pieces. All other designs can be achieved by transforming the existing ones.

Includes PNG images and TTF font (monochrome and color).

Details

- 19 pieces (white, black, and neutral)
- 16x16 resolution
- **Color and monochrome** versions
- **PNG** and **TTF** file formats
- CC0 license

Pieces

- King
- Queen
- Rook
- Bishop
- Knight
- Pawn

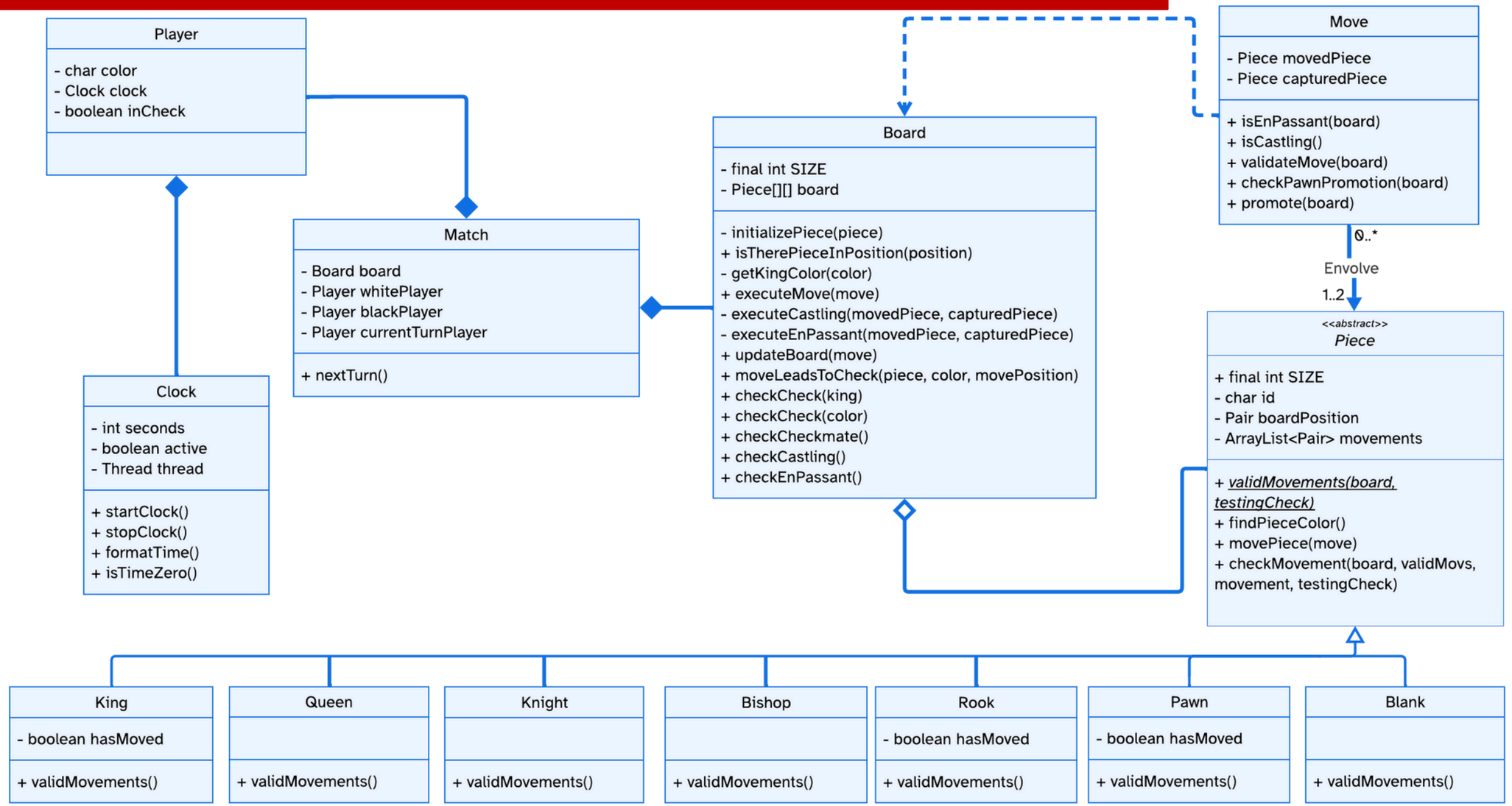
https://spicygame.itch.io/chess-pieces

JAYLIB

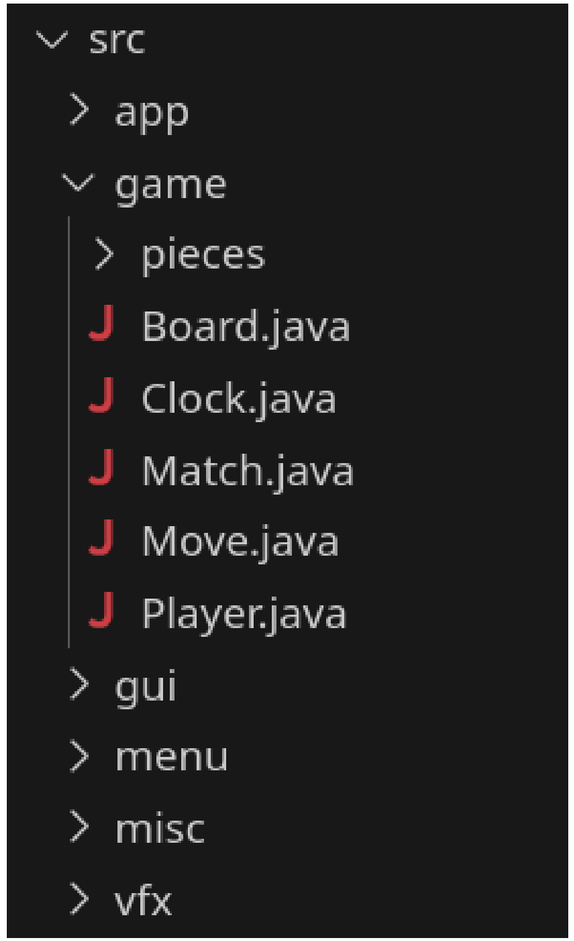
```
rectangle = new Rectangle().x(this.x).y(this.y).width(this.width).height(this.height);
```

```
public class OurRectangle{

    private Rectangle rectangle;
    private float x;
    private float y;
    private float width;
    private float height;

    public OurRectangle(float x, float y, float width, float height){
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
        rectangle = new Rectangle().x(this.x).y(this.y).width(this.width).height(this.height);
    }
```

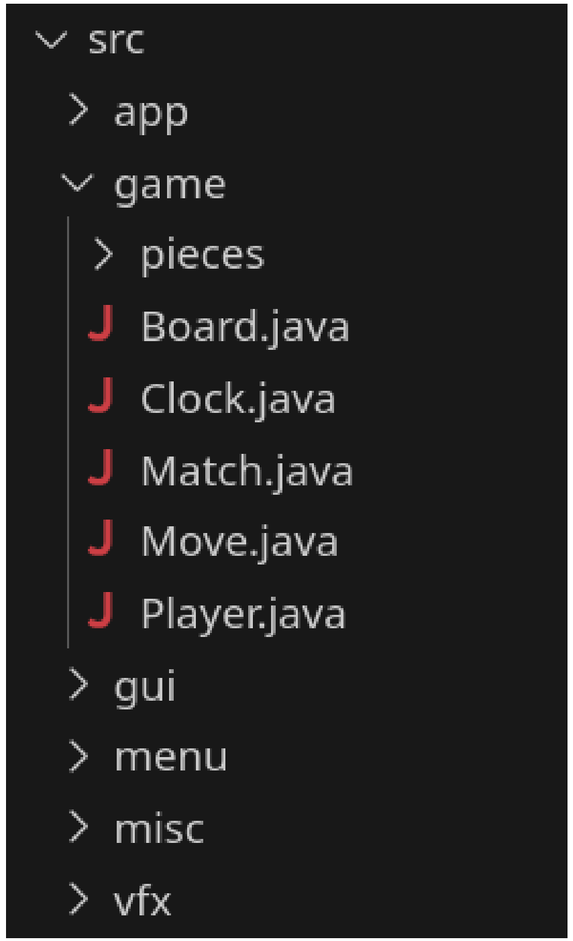
DIAGRAMA DE CLASSES



VISÃO GERAL DO CÓDIGO



VISÃO GERAL DO CÓDIGO



VISÃO GERAL DO CÓDIGO

```java
public abstract class Piece {
    public final int SIZE = 8;
    private char id;
    private Pair boardPosition;
    private ArrayList<Pair> movements = new ArrayList<>();
    private Sprite sprite;

    public Piece(int x, int y, char id) {
        this.boardPosition = new Pair(x, y);
        this.id = id;
    }

    public ArrayList<Pair> getMovements() {
        return movements;
    }

    public Pair getBoardPosition() {
        return boardPosition;
    }
```

VISÃO GERAL DO CÓDIGO

```java
@Override
public ArrayList<Pair> validMovements(Board board, boolean testingCheck) {

    ArrayList<Pair> newMovements = new ArrayList<>();
    this.hasEnPassant = false;

    int direction = this.getMoveDirection();
    char color = this.findPieceColor();

    Pair up = this.getBoardPosition().add(new Pair(0, direction * 1));
    Pair doubleUp = this.getBoardPosition().add(new Pair(0, direction * 2));

    // The pawn only attacks on its diagonals
    Pair upperRight = this.getBoardPosition().add(new Pair(+1, direction * 1));
    Pair upperLeft = this.getBoardPosition().add(new Pair(-1, direction * 1));

    if (up.isPieceInsideBoard(0, SIZE)) {
        if (!(board.isTherePieceInPosition(up))) {
            this.checkMovement(board, newMovements, up, testingCheck);

            if (!this.hasMoved && doubleUp.isPieceInsideBoard(0, SIZE)) {
                if (!(board.isTherePieceInPosition(doubleUp))) {
                    this.checkMovement(board, newMovements, doubleUp, testingCheck);
                }
            }
        }
    }

    if (upperRight.isPieceInsideBoard(0, SIZE)) {
```

VISÃO GERAL DO CÓDIGO

```java
/*
 * add the moviment to the movs list only if this moviment doesn't lead to a
 * check
 */
public void checkMovement(Board board, ArrayList<Pair> movs, Pair movement, boolean testingCheck) {

    if (testingCheck) {
        if (!board.moveLeadsToCheck(this, this.findPieceColor(), movement)) {
            movs.add(movement);
        }
    } else {
        movs.add(movement);
    }

}
```

VISÃO GERAL DO CÓDIGO

```java
/* Return if a move is valid */
public boolean validateMove(Board board) {

    for (Pair p : this.getMovedPiece().getMovements()) {
        if (p.isEqualsTo(this.getCapturedPiece().getBoardPosition())) {
        return true;
        }
    }

    return false;
}
```

VISÃO GERAL DO CÓDIGO

```java
/* Execute the move and change the positions of the pieces */
public void executeMove(Move move) {

    Piece movedPiece = move.getMovedPiece();
    Piece capturedPiece = move.getCapturedPiece();

    // Set the pieced as move to prevent special moviments after
    if (movedPiece instanceof King) {
        ((King) movedPiece).setHasMoved(true);
    }
    if (movedPiece instanceof Rook) {
        ((Rook) movedPiece).setHasMoved(true);
    }
    if (movedPiece instanceof Pawn) {
        ((Pawn) movedPiece).setHasMoved(true);
    }

    // checking for special moviments
    if (move.isCastling()) {
        this.executeCastling(movedPiece, capturedPiece);
    } else if (move.isEnPassant(this)) {
        this.executeEnPassant(movedPiece, this.getLastMove().getCapturedPiece());
    } else {
        updateBoard(move);
        movedPiece.movePiece(move);
    }
}
```

VISÃO DO REPOSITÓRIO

What is TCP?

TCP is an ***free open-source***, object-oriented implementation of the traditional Chess, made for our final project for the "INF01120" course. This game brings the full experience of playing chess with your friends side-by-side to your computer.

TCP intends to be an beginner-friendly easy-to-play chess game. To starting playing, everything you need to do is clone this repository and build the game with the following commands based on your operation system:

Linux:

```
git clone https://github.com/SW-Engineering-Courses-Karina-Kohl/tcp-final-20251-grupo02.git tcp
cd tcp
make run
```

Windows (VSCode Terminal/PowerShell):

```
git clone https://github.com/SW-Engineering-Courses-Karina-Kohl/tcp-final-20251-grupo02.git tcp
cd tcp
javac -cp lib/jaylib-5.5.0-2.jar -d bin (Get-ChildItem -Recurse -Filter *.java -Path src).FullName
java -cp lib/jaylib-5.5.0-2.jar`;bin app.Main
```

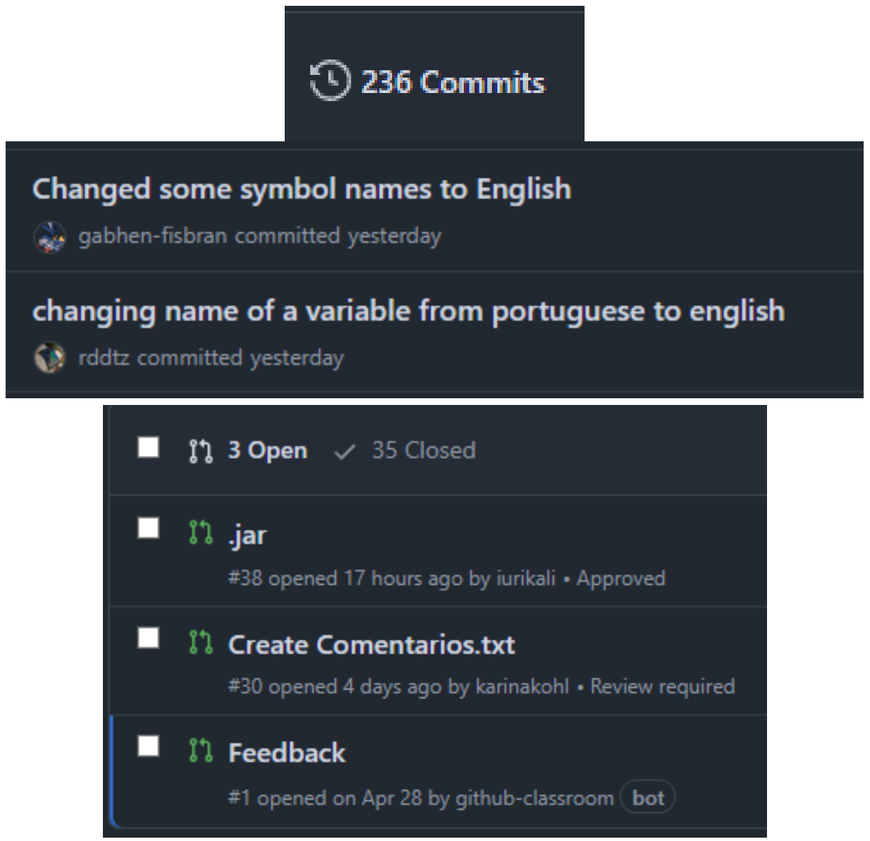
```
JC=javac
JAYLIB=lib/jaylib-5.5.0-2.jar
JAYUNIT=lib/junit-platform-console-standalone-1.11.0.jar

run: src/*/*.java
	javac -g -Xlint:all -deprecation -cp "$(JAYLIB):$(JAYUNIT)" -d bin `find . -name "*.java" -not -name ".*"`
	java -cp $(JAYLIB):bin app.Main

clean:
	rm -r bin
```

jar
refactorAndTests
chess-features
hot-fix
som
last-ui-new
last-ui
feedback
renaming
promocaoPeao
development
modificacoes
relogio2

INSTITUTO DE INFORMÁTICA UFRGS

https://github.com/SW-Engineering-Courses-Karina-Kohl/tcp-final-20251-grupo02

VISÃO DO REPOSITÓRIO



Tarefas

._subtarefas/Tarefas	Menu	Jogo	Movimentos	Cheque-Mate	Fim
1	Novo Jogo	Tabuleiro	Movimento para cada peça	Veficicação de possíveis defesas	Vencedor
2	Sair do Jogo	Peças	Captura de peças	Derrota/Vitória	Novo Jogo
3	Configurações	Turnos	Promoção	Afogamento	
4		Relógio	Movimentos especiais	Repetição	

https://iuri-kali.itch.io/tcp

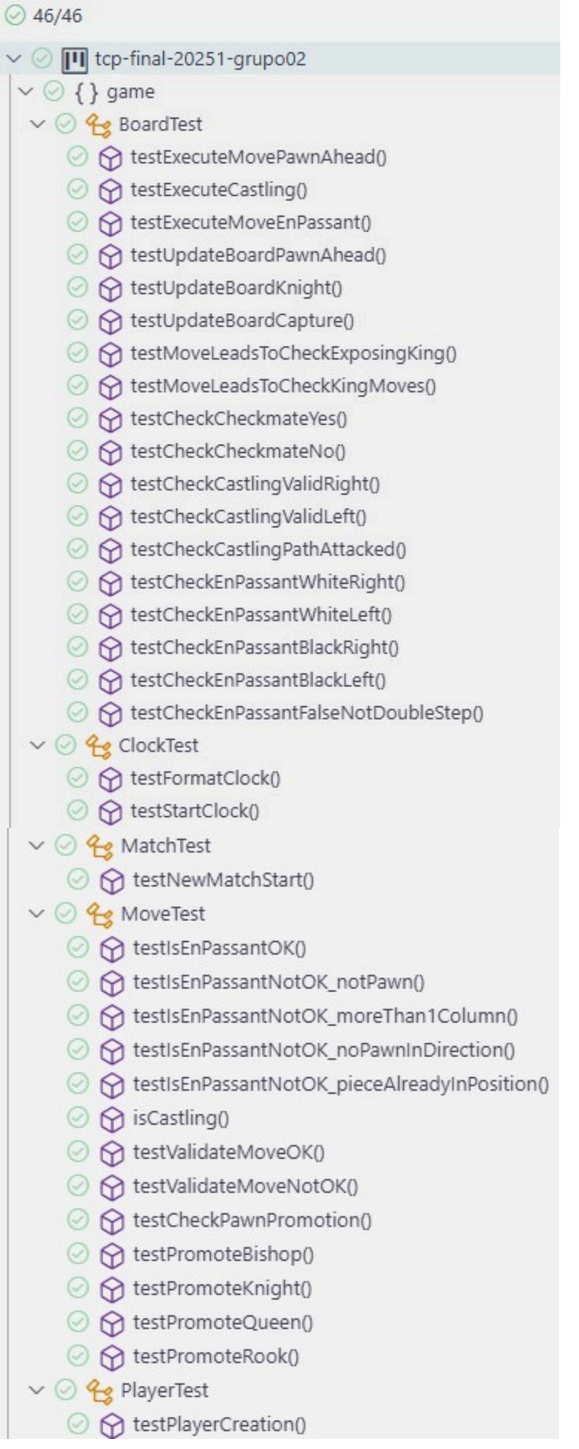
INSTITUTO DE INFORMÁTICA UFRGS

https://github.com/SW-Engineering-Courses-Karina-Kohl/tcp-final-20251-grupo02

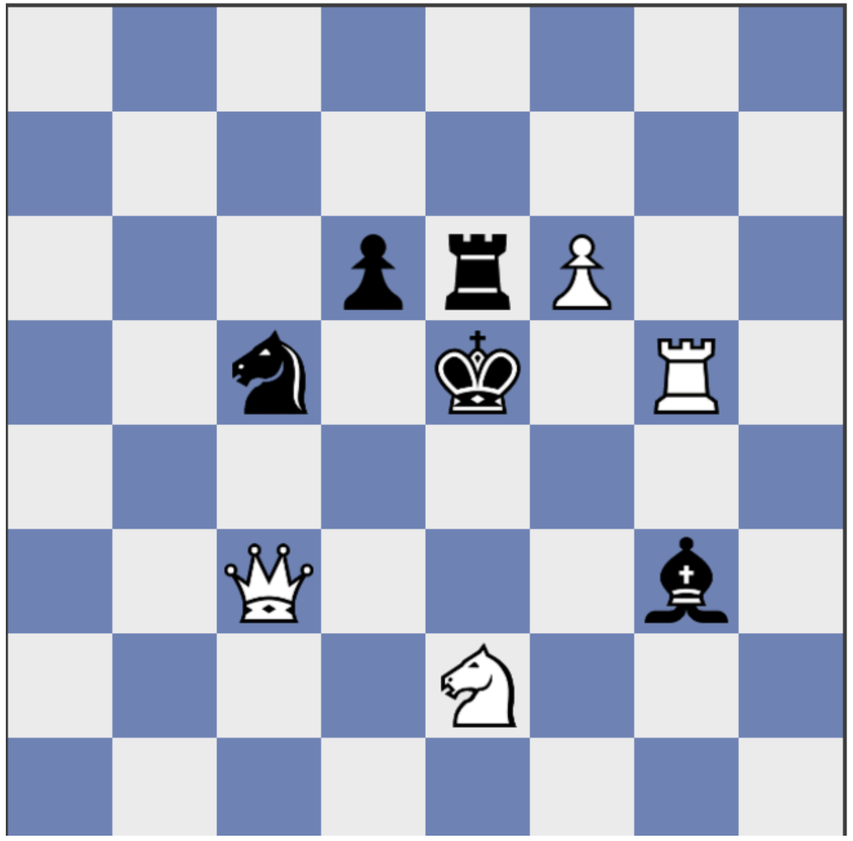
TESTES

- Para todas as classes do pacote game
- Encontrar os movimentos válidos da peça
- Validação de movimentos, incluindo especiais
- Atualização do tabuleiro e da peça
- Detecção de xeque e xeque-mate
- Inicialização da partida e alternância entre jogadores
- Controle do relógio

```
Test run finished after 7280 ms
[         9 containers found      ]
[         0 containers skipped    ]
[         9 containers started    ]
[         0 containers aborted    ]
[         9 containers successful ]
[         0 containers failed     ]
[        46 tests found           ]
[         0 tests skipped         ]
[        46 tests started         ]
[         0 tests aborted         ]
[        46 tests successful      ]
[         0 tests failed          ]
```



TESTES



```
    Board board = new Board(useUI:false, initPieces:false);
    board.setPieceInPosition(x:3, y:2, new Pawn(x:3, y:2, id:'p', initUI:false));
    board.setPieceInPosition(x:4, y:2, new Rook(x:4, y:2, id:'r', initUI:false));
    board.setPieceInPosition(x:5, y:2, new Pawn(x:5, y:2, id:'P', initUI:false));
    board.setPieceInPosition(x:2, y:3, new Knight(x:2, y:3, id:'k', initUI:false));
    board.setPieceInPosition(x:6, y:3, new Rook(x:6, y:3, id:'R', initUI:false));
    board.setPieceInPosition(x:2, y:5, new Queen(x:2, y:5, id:'Q', initUI:false));
    board.setPieceInPosition(x:6, y:5, new Bishop(x:6, y:5, id:'b', initUI:false));
    board.setPieceInPosition(x:4, y:6, new Knight(x:4, y:6, id:'K', initUI:false));

    Queen whiteQueen = new Queen(x:4, y:3, id:'Q', initUI:false);

    ArrayList<Pair> returnedMovs = whiteQueen.validMovements(board, testingCheck:false);

    ArrayList<Pair> expectedMovs = new ArrayList<>();
    expectedMovs.add(new Pair(x:3, y:2));
    expectedMovs.add(new Pair(x:4, y:2));
    expectedMovs.add(new Pair(x:2, y:3));
    expectedMovs.add(new Pair(x:3, y:3));
    expectedMovs.add(new Pair(x:5, y:3));
    expectedMovs.add(new Pair(x:3, y:4));
    expectedMovs.add(new Pair(x:4, y:4));
    expectedMovs.add(new Pair(x:5, y:4));
    expectedMovs.add(new Pair(x:4, y:5));
    expectedMovs.add(new Pair(x:6, y:5));

    Set<Pair> setReturned = new HashSet<>(returnedMovs);
    Set<Pair> setExpected = new HashSet<>(expectedMovs);

    assertEquals(setExpected, setReturned);
}
```

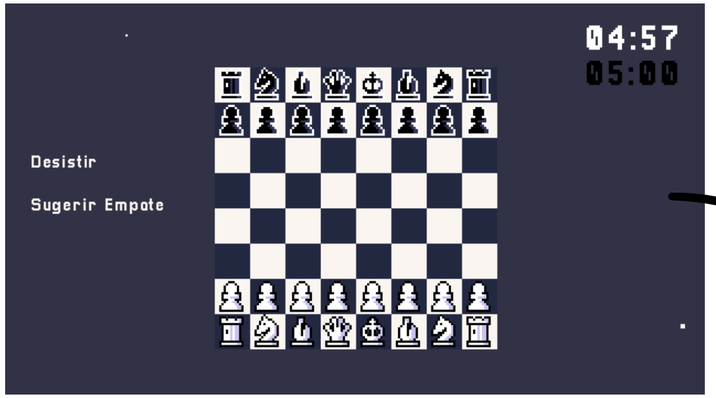
DEMONSTRAÇÃO

CONCLUSÃO

- Melhor organização, principalmente numa melhor divisão no que cada um vai fazer
- Aprendemos a utilizar melhor o git para versionamento do código, mais sobre POO

Adicionar mais opções, como:

- Mudar as cores das peças e do tabuleiro;
- Adicionar incremento de tempo por jogada.



- Adicionar um histórico de jogadas;
- Aplicar empate por afogamento.

Link para o jogo