Dokumentace projektu

PříHoDa

7. prosince 2024

Složka: .vscode

Složka: C_files

Soubor: $parse_python.py$

Třída: CommentExtractor

- *Metoda:*

$_{init}$*Metoda: visit*$_{F}$*unctionDef Metoda: visit*$_{C}$*lassDef*

Složka: games

Soubor: Board.py

Třída: Board

- Třída reprezentující hrací desku hry
  - Konstruktor třídy Board
  - Inicializace herní desky
  - Vytvoří string hrací desky na výpis do konzole
  - Vrací šachovnici jako list
    Returns: List of Struct : List, kde každý řádek je list obsahující figury na daném řádku
  - Pro možnost přstupovat k poli board jako board[row,col] místo board.board[row][col]
    Args: index: Tuple dvou integerů, (row, col)
    Returns: Figuru na určeném místě na šachovnici, případně None, pokud je prázdné
  - Nastaví políčko na šachovnici jako board[row,col] namísto board.board[row][col]
    Args: index ([int, int]): Tuple dvou integerů, (row, col) value (any): Co se má nastavit na dané políčko

- *Metoda:*

  $_{init}$*Konstruktor třídy Board Metoda:*$_{populateBoard}$*Inicializace herní desky Metoda:*$_{str}$ *Vytvoří string hrací*

  Returns: List of Struct : List, kde každý řádek je list obsahující figury na daném řádku

- *Metoda:*

  $_{getitem}$

  Args: index: Tuple dvou integerů, (row, col)

  Returns: Figuru na určeném místě na šachovnici, případně None, pokud je prázdné

- *Metoda:*

  $_{setitem}$

  Args: index ([int, int]): Tuple dvou integerů, (row, col) value (any): Co se má nastavit na dané políčko

Soubor: ChallengeAccepted.py

Třída: ChallengeAccepted

- *Atribut:*

  $_{init}$*Atribut: getBoard*

- *Atribut: getField*
- *Atribut: moveMole*
- *Atribut: makeMove*
- *Atribut: checkEnd*
- *Atribut: printToTerminal*

Složka: checkers

Soubor: Checkers.py

Třída: Checkers

– Třída reprezentující hru dáma

Attributes: withChoosePiece (bool): True, pokud je v hře možné vybrat figurku, jinak False numberOfPlayers (int): počet hráčů fog (bool): True, pokud je ve hře Fog Of War, jinak False

$_{board(CheckersBoard):hracídeska}{}_{currentPlayer(EnumColors):barvahráčenatahu}{}_{pieceToPlay(Piece):figurka,kterouhráčchcehr}$

- Funkce pro získání hrací desky

  Args: color (Enum Colors, optional): barva hráče na tahu. Výchozí nastavení je na pravidelném střídání.

  Returns: List of List of [int, int]: hrací deska

- Funkce pro vyber figurky, kterou chce hrac hrat

  Args: index ([int,int]): pozice figurky, kterou chce hrac hrat color (Enum Colors, optional): barva hráče na tahu. Výchozí nastavení je na pravidelném střídání.

  Returns: List of [int, int]: seznam dostupných pozic, kam může hráč hrát

- Funkce pro provedení tahu figurkou

  Args: index ([int,int]): pozice, kam chce hráč hrát color (Enum Colors, optional): barva hráče na tahu. Výchozí nastavení je na pravidelném střídání. rightClick (bool, optional): True, pokud se jedná o pravé tlačítko myši. Výchozí nastavení je False.

  Returns: bool: True, pokud se tah podařil, jinak False

- Funkce pro resetování hry

- Funkce pro kontrolu konce hry

  Returns: string: Vrací vítěze "barva won", pokud hra skončila, jinak None

- Funkce pro ukončení tahu

- Funkce pro získání možných tahů pro hráče, primárně pro rozšíření Fog Of War

  Args: color (Enum Colors): barva hráče, pro kterého se mají tahy získat

  Returns: List of [int, int]: seznam možných tahů

- Funkce pro odstranění figurky z hrací desky

  Args: piecePosition ([int, int]): pozice figurky, která má být odstraněna

- Funkce pro výpis stavu hry na terminál

- *Metoda:*

  $_{i}{}^{nit}$*Konstruktor třídy hry dámaMetoda: getBoard*

  Args: color (Enum Colors, optional): barva hráče na tahu. Výchozí nastavení je na pravidelném střídání.

  Returns: List of List of [int, int]: hrací deska

- *Metoda: choosePiece*

  Args: index ([int,int]): pozice figurky, kterou chce hrac hrat color (Enum Colors, optional): barva hráče na tahu. Výchozí nastavení je na pravidelném střídání.

Returns: List of [int, int]: seznam dostupných pozic, kam může hráč hrát*Funkce pro vyber figurky, kterou chce hrac hrat*

Args: index ([int,int]): pozice figurky, kterou chce hrac hrat color (Enum Colors, optional): barva hráče na tahu. Výchozí nastavení je na pravidelném střídání.

Returns: List of [int, int]: seznam dostupných pozic, kam může hráč hrát

- *Metoda: makeMove*

  Args: index ([int,int]): pozice, kam chce hráč hrát color (Enum Colors, optional): barva hráče na tahu. Výchozí nastavení je na pravidelném střídání. rightClick (bool, optional): True, pokud se jedná o pravé tlačítko myši. Výchozí nastavení je False.

  Returns: bool: True, pokud se tah podařil, jinak FalseFunkce pro provedení tahu figurkou

  Args: index ([int,int]): pozice, kam chce hráč hrát color (Enum Colors, optional): barva hráče na tahu. Výchozí nastavení je na pravidelném střídání. rightClick (bool, optional): True, pokud se jedná o pravé tlačítko myši. Výchozí nastavení je False.

  Returns: bool: True, pokud se tah podařil, jinak False

- *Metoda: reset Funkce pro resetování hry*

- *Metoda: checkEnd*

  Returns: string: Vrací vítěze "barva won", pokud hra skončila, jinak NoneFunkce pro kontrolu konce hry

  Returns: string: Vrací vítěze "barva won", pokud hra skončila, jinak None

- *Metoda:*
  $_{endOfTurn}$Funkce pro ukončení tahuMetoda: possibleMoves

  Args: color (Enum Colors): barva hráče, pro kterého se mají tahy získat

  Returns: List of [int, int]: seznam možných tahů

- *Metoda: killPiece*

  Args: piecePosition ([int, int]): pozice figurky, která má být odstraněnaFunkce pro odstranění figurky z hrací desky

  Args: piecePosition ([int, int]): pozice figurky, která má být odstraněna

- *Metoda:*
  $_{printToTerminal}$Funkce pro výpis stavu hry na terminál

Soubor: CheckersBoard.py

Třída: CheckersBoard

- Třída reprezentující šachovnici pro hru dáma. Dědí z třídy Board.

  Attributes: board (List of List of Pieces): hrací deska, kde None reprezentuje prázdné políčko
- Konstruktor třídy CheckersBoard. Vytvoří šachovnici a umístí na ni všechny figurky do počáteční polohy.
- Vrací figuru na určeném místě na šachovnici, nebo None, pokud je políčko prázdné.

  Args: index [int, int]: Tuple dvou integerů, (row, col), oba 0-7
- Nastaví políčko na šachovnici jako board[row,col] namísto board.board[row][col]

  Args: index ([int, int]): Tuple dvou integerů, (row, col), oba 0-7 value (Piece): Instance třídy Piece, nebo None, pokud má být políčko prázdné

  Returns: bool: True, pokud se podařilo nastavit políčko, jinak False
- Vrací string reprezentaci šachovnice. Každé políčko je reprezentováno jako string, který je tvořen z informací o barvě a symbolu figury, nebo jako string ”

  ”,*pokudjepolíčkoprázdné.Políčkajsouoddělenamezerouajednotlivéřádkyjsouoddělenyznakemnovéhořádku*().

  Returns: string: string reprezentace šachovnice
- Nastaví šachovnici do počátečního stavu. Bílý je dole a černý nahoře.
- Vrací list všech figurek dané barvy na šachovnici.

  Args: color (Enum Colors): Barva figurek, které chceme najít (Colors.WHITE nebo Colors.BLACK)

  Returns: List of Pieces: List figurek dané barvy
- Vrací list všech figurek na šachovnici.

  Returns: List of Fields: List figurek na šachovnici

- *Metoda:*

  $_{i}{}^{nit}$*Konstruktor třídy CheckersBoard. Vytvoří šachovnici a umístí na ni všechny figurky do počáteční pol*

- *Metoda:*

  $_{s}etitem$

  Args: index ([int, int]): Tuple dvou integerů, (row, col), oba 0-7 value (Piece): Instance třídy Piece, nebo None, pokud má být políčko prázdné

  Returns: bool: True, pokud se podařilo nastavit políčko, jinak False

- *Metoda:*

  $_{s}tr$

  Returns: string: string reprezentace šachovnice

- *Metoda:*
  $_{p}opulateBoard$*Nastaví šachovnici do počátečního stavu. Bílý je dole a černý nahoře.Metoda: pieceList*
  Args: color (Enum Colors): Barva figurek, které chceme najít (Colors.WHITE nebo Colors.BLACK)
  Returns: List of Pieces: List figurek dané barvy
- *Metoda: getListOfBoard*
  Returns: List of Fields: List figurek na šachovnici *Vrací list všech figurek na šachovnici.*
  Returns: List of Fields: List figurek na šachovnici

Soubor: CheckersMines.py

Třída: CheckersMines

- Hra dáma s minami.
  Attributes: mines (List of [int, int]): seznam pozic min explosion ([int, int]): pozice exploze withChoosePiece (bool): True, pokud je v hře možné vybrat figurku, jinak False numberOfPlayers (int): počet hráčů fog (bool): True, pokud je ve hře Fog Of War, jinak False
- Inicializace hry.
- Umístí miny na náhodné pozice.
- Vrátí hrací desku.
- Zpracuje tah.

- *Metoda:*
  $_{i}nit$*Inicializace hry.Metoda:*$_{placeMines}$*Umístí miny na náhodné pozice.Metoda: getBoardVrátí hrací desku.*
- *Metoda: makeMove Zpracuje tah.*

Soubor: CheckersMinesWithFogOfWar.py

Třída: CheckersMinesWithFogOfWar

- Třída CheckersWithFogOfWar slouží k reprezentaci hry Dáma s mlhou války.
  Attributes: mines (List of [int, int]): seznam pozic min explosion ([int, int]): pozice exploze withChoosePiece (bool): True, pokud je v hře možné vybrat figurku, jinak False numberOfPlayers (int): počet hráčů fog (bool): True, pokud je ve hře Fog Of War, jinak False
- Konstruktor třídy CheckersWithFogOfWar

  - Vrací zakrytou šachovnici
    Args: color (Enum Colors): Barva hráče na tahu
    Returns: Array of Field: zakrytá šachovnice

- *Metoda:*

  $_{i}{}^{nit}$*Konstruktor třídy CheckersWithFogOfWarMetoda: getBoard*
  Args: color (Enum Colors): Barva hráče na tahu
  Returns: Array of Field: zakrytá šachovnice

Složka: pieces

Soubor: Pawn.py

Třída: Pawn

- Třída reprezentující pěšáka
  Attributes: color (Colors): barva figurky position ([int,int]): pozice figurky
  - Funkce na výpis figurky
    Returns: str: P + barva figurky
  - Funkce najde všechny možné tahy figurky
    Args: board (two-dimensional array of ints): hrací deska
    Returns: [int,int]: pole možných tahů
  - Funkce najde všechny jednoduché skoky z pozice
    Args: board (two-dimensional array of ints): hrací deska position ([int,int]], optional):
    $_{d}escription$
    $_{.}DefaultstoNone.$
    Returns: [int,int]: pole možných skoků
  - Funkce na zjištění přeskočených figurek
    Args: endPosition ([int, int]): koncová pozice
    Returns: [int, int]: pozice přeskoečené figurky

- *Metoda:*

  $_{s}{}^{tr}$
  Returns: str: P + barva figurky

- *Metoda: possibleMoves*

  Args: board (two-dimensional array of ints): hrací deska

  Returns: [int,int]: pole možných tahů*Funkce najde všechny možné tahy figurky*

Args: board (two-dimensional array of ints): hrací deska

Returns: [int,int]: pole možných tahů

- *Metoda: possibleJumps*

  Args: board (two-dimensional array of ints): hrací deska position ([int,int]], optional):
  $_{d}escription$
  $_{.}DefaultstoNone.$

  Returns: [int,int]: pole možných skokůFunkce najde všechny jednoduché skoky z pozice

  Args: board (two-dimensional array of ints): hrací deska position ([int,int]], optional):
  $_{d}escription$
  $_{.}DefaultstoNone.$

Returns: [int,int]: pole možných skoků

- *Metoda: trackJumps*

  Args: endPosition ([int, int]): koncová pozice

  Returns: [int, int]: pozice přeskoečené figurkyFunkce na zjištění přeskočených figurek

  Args: endPosition ([int, int]): koncová pozice

  Returns: [int, int]: pozice přeskoečené figurky

Soubor: Piece.py

Třída: Piece

- Třída reprezentující figurku na šachovnici
  Attributes: color (Colors): barva figurky position ([int,int]): pozice figurky
- Konstruktor třídy Piece
  Args: color (Enum Colors): požadovaná barva figurky position ([int,int]): výchozí pozice figurky
- Metoda pro zjištění možných tahů figurky
  Args: board (Board): hrací deska, na které se figurka nachází
- Metoda pro zjištění možných skoků figurky
  Args: board (Board): hrací deska, na které se figurka nachází position ([int, int], optional): pozice, ze které se má skákat. Výchozí hodnota je None, což znamená, že se skáče ze současné pozice figurky
- Metoda pro zjištění možných skoků figurky
  Args: board (Board): hrací deska, na které se figurka nachází endPosition ([int, int]): pozice, kam se má skákat

  - Vlastnost pro získání řádku, na kterém se figurka nachází
    Returns: int: řádek, na kterém se figurka nachází
  - Vlastnost pro získání sloupce, na kterém se figurka nachází
    Returns: int: sloupec, na kterém se figurka nachází

- *Metoda:*

  $_{i}nit$

  Args: color (Enum Colors): požadovaná barva figurky position ([int,int]): výchozí pozice figurky

- *Metoda: possibleMoves*

  Args: board (Board): hrací deska, na které se figurka nachází*Metoda pro zjištění možných tahů figurky*

  Args: board (Board): hrací deska, na které se figurka nachází

- *Metoda: possibleJumps*

  Args: board (Board): hrací deska, na které se figurka nachází position ([int, int], optional): pozice, ze které se má skákat. Výchozí hodnota je None, což znamená, že se skáče ze současné pozice figurkyMetoda pro zjištění možných skoků figurky

  Args: board (Board): hrací deska, na které se figurka nachází position ([int, int], optional): pozice, ze které se má skákat. Výchozí hodnota je None, což znamená, že se skáče ze současné pozice figurky

- *Metoda: trackJumps*

  Args: board (Board): hrací deska, na které se figurka nachází endPosition ([int, int]): pozice, kam se má skákatMetoda pro zjištění možných skoků figurky

  Args: board (Board): hrací deska, na které se figurka nachází endPosition ([int, int]): pozice, kam se má skákat

- *Metoda: row*

  Returns: int: řádek, na kterém se figurka nacházíVlastnost pro získání řádku, na kterém se figurka nachází

  Returns: int: řádek, na kterém se figurka nachází

- *Metoda: col*

  Returns: int: sloupec, na kterém se figurka nacházíVlastnost pro získání sloupce, na kterém se figurka nachází

  Returns: int: sloupec, na kterém se figurka nachází

Soubor: Queen.py

Třída: Queen

- Třída reprezentující dámu

  Attributes: color (Colors): barva figurky position ([int,int]): pozice figurky
- Konstruktor třídy Queen

  Args: pawn (Pawn): Pěšák, který se má proměnit na dámu
- Vrací string reprezentaci dámy

  Returns: string: string reprezentace dámy
- Vrací seznam možných tahů dámy

  Args: board (CheckersBoard): šachovnice, na které se dáma nachází

  Returns: List of [int, int] : seznam možných tahů dámy
- Vrací seznam možných prvních skoků dámy

  Args: board (CheckersBoard): šachovnice, na které se dáma nachází position ([int, int], optional): pozice, ze které se má skákat. Výchozí hodnota je None, což znamená, že se skáče ze současné pozice dámy

  Returns: List of [int, int]: seznam možných prvních skoků dámy
- Vrací seznam pozic figurek, které dáma přeskočí, než se dostane na koncovou pozici

  Args: endPosition ([int, int]): koncová pozice, na kterou se má dáma dostat

  Returns: List of [int, int]: seznam pozic figurek, které dáma přeskočí, než se dostane na koncovou pozici

- *Metoda:*

  $_{i}nit$

  Args: pawn (Pawn): Pěšák, který se má proměnit na dámu

- *Metoda:*

  $_{s}tr$

  Returns: string: string reprezentace dámy

- *Metoda: possibleMoves*

  Args: board (CheckersBoard): šachovnice, na které se dáma nachází

  Returns: List of [int, int] : seznam možných tahů dámy*Vrací seznam možných tahů dámy*

  Args: board (CheckersBoard): šachovnice, na které se dáma nachází

  Returns: List of [int, int] : seznam možných tahů dámy

- *Metoda: possibleJumps*

  Args: board (CheckersBoard): šachovnice, na které se dáma nachází position ([int, int], optional): pozice, ze které se má skákat. Výchozí hodnota je None, což znamená, že se skáče ze současné pozice dámy

  Returns: List of [int, int]: seznam možných prvních skoků dámyVrací seznam možných prvních skoků dámy

  Args: board (CheckersBoard): šachovnice, na které se dáma nachází position ([int, int], optional): pozice, ze které se má skákat. Výchozí hodnota je None, což znamená, že se skáče ze současné pozice dámy

  Returns: List of [int, int]: seznam možných prvních skoků dámy

- *Metoda: trackJumps*

  Args: endPosition ([int, int]): koncová pozice, na kterou se má dáma dostat

  Returns: List of [int, int]: seznam pozic figurek, které dáma přeskočí, než se dostane na koncovou poziciVrací seznam pozic figurek, které dáma přeskočí, než se dostane na koncovou pozici

  Args: endPosition ([int, int]): koncová pozice, na kterou se má dáma dostat

  Returns: List of [int, int]: seznam pozic figurek, které dáma přeskočí, než se dostane na koncovou pozici

Soubor: $_{i}nit_{.py}$

Složka: __pycache__

Soubor: $_{i}nit_{.py}$

Složka: __pycache__

Soubor: CheckersWithFogOfWar.py

Třída: CheckersWithFogOfWar

- Třída CheckersWithFogOfWar slouží k reprezentaci hry Dáma s mlhou války.
- Konstruktor třídy CheckersWithFogOfWar
- Vrací zakrytou šachovnici
  Args: color (Enum Colors): Barva hráče na tahu
  Returns: Array of Field: zakrytá šachovnice

- *Metoda:*

  $_{i}nit$ *Konstruktor třídy CheckersWithFogOfWar Metoda: getBoard*
  Args: color (Enum Colors): Barva hráče na tahu
  Returns: Array of Field: zakrytá šachovnice

Složka: chess

Soubor: Chess.py

Třída: Chess

- Třída reprezentující hru šachy

  Attributes: withChoosePiece (bool): True, pokud se má hráči zobrazovat možnost vybrat figurku, jinak False

  $_{m}ovesSinceLastImportantMove(int)$:$Počettahůodposledníhodůležitéhotahu_{b}oard(ChessBoard)$:$Šachovnice_{p}layedPiece(Piec$

  – Vrátí šachovnici v aktuálním stavu jako dvourozměrné pole Field

    Args: color (Enum Colors): Barva hráče, pro kterého se má šachovnice vykreslit

    Returns: list of list of Field: šachovnice

  – Funkce pro vyber figurky, kterou chce hrac hrat

    Args: positionToPlay ([int, int]): pozice figurky, kterou chce hrac hrat color (Enum Colors): Barva hrace, ktery chce hrat

    Returns: (list of [int, int]): dostupne pozice, kam muze hrac hrat

  – Provedení tahu hrace

    Args: playedMove ([int, int]): pozice, kam chce hrac hrat color (Enum Colors): Barva hrace, ktery chce hrat rightClick (bool): True, pokud hrac klikl pravym tlacitkem mysi, jinak False

    Returns: bool: tah se zdařil nebo ne string: "Promote"pokud je potreba provest vylepseni pesaka

  – Promote pesaka

    Args: newFigure (string): figurka, na kterou se ma pesak zmenit ("Q", "R", "B", "N")

    Returns: bool: podle toho zda se tah podaril nebo ne string: pokud hra skoncila

  – Konec tahu

    Returns: True: novy stav sachovnice po tahu string: string pokud hra skoncila

  – Vrati mozne tahy pro hrace, primárně pro rozšíření Fog Of War

    Args: color (Enum Colors): barva hrace, pro ktereho se maji tahy vypsat

    Returns: list of [int, int]: seznam moznych tahu

  – Kontrola konce hry

    Returns: string: "Draw by fifty-move rule"pokud bylo 50 tahu bez pohybu pesaku nebo braneni string: "Draw by threefold repetition"pokud se stejna pozice opakovala 3x string: "color won"pokud byl vyhozen kral None: pokud hra neskoncila

  - Vyhození figurky z hrací desky
    Args: piecePosition ([int, int]): pozice figurky, kterou chceme vyhodit
  - Vytiskne hrací desku do konzole

- *Metoda:*

  $_{i}{}^{nit}$*Konstruktor třídy šachůMetoda: getBoard*

  Args: color (Enum Colors): Barva hráče, pro kterého se má šachovnice vykreslit

  Returns: list of list of Field: šachovnice

- *Metoda: choosePiece*

  Args: positionToPlay ([int, int]): pozice figurky, kterou chce hrac hrat color (Enum Colors): Barva hrace, ktery chce hrat

  Returns: (list of [int, int]): dostupne pozice, kam muze hrac hrat*Funkce pro vyber figurky, kterou chce hrac hrat*

  Args: positionToPlay ([int, int]): pozice figurky, kterou chce hrac hrat color (Enum Colors): Barva hrace, ktery chce hrat

  Returns: (list of [int, int]): dostupne pozice, kam muze hrac hrat

- *Metoda: makeMove*

  Args: playedMove ([int, int]): pozice, kam chce hrac hrat color (Enum Colors): Barva hrace, ktery chce hrat rightClick (bool): True, pokud hrac klikl pravym tlacitkem mysi, jinak False

  Returns: bool: tah se zdařil nebo ne string: "Promote"pokud je potreba provest vylepseni pesakaProvedení tahu hrace

  Args: playedMove ([int, int]): pozice, kam chce hrac hrat color (Enum Colors): Barva hrace, ktery chce hrat rightClick (bool): True, pokud hrac klikl pravym tlacitkem mysi, jinak False

  Returns: bool: tah se zdařil nebo ne string: "Promote"pokud je potreba provest vylepseni pesaka

- *Metoda: promote*

  Args: newFigure (string): figurka, na kterou se ma pesak zmenit ("Q", "R", "B", "N")

  Returns: bool: podle toho zda se tah podaril nebo ne string: pokud hra skoncilaPromote pesaka

  Args: newFigure (string): figurka, na kterou se ma pesak zmenit ("Q", "R", "B", "N")

  Returns: bool: podle toho zda se tah podaril nebo ne string: pokud hra skoncila

- *Metoda:*
  $_{e}ndOfMove$
  Returns: True: novy stav sachovnice po tahu string: string pokud hra skoncila

- *Metoda: possibleMoves*
  Args: color (Enum Colors): barva hrace, pro ktereho se maji tahy vypsat
  Returns: list of [int, int]: seznam moznych tahuVrati mozne tahy pro hrace, primárně pro rozšíření Fog Of War
  Args: color (Enum Colors): barva hrace, pro ktereho se maji tahy vypsat
  Returns: list of [int, int]: seznam moznych tahu

- *Metoda: checkEnd*
  Returns: string: "Draw by fifty-move rule"pokud bylo 50 tahu bez pohybu pesaku nebo braneni string: "Draw by threefold repetition"pokud se stejna pozice opakovala 3x string: "color won"pokud byl vyhozen kral None: pokud hra neskoncilaKontrola konce hry
  Returns: string: "Draw by fifty-move rule"pokud bylo 50 tahu bez pohybu pesaku nebo braneni string: "Draw by threefold repetition"pokud se stejna pozice opakovala 3x string: "color won"pokud byl vyhozen kral None: pokud hra neskoncila

- *Metoda: killPiece*
  Args: piecePosition ([int, int]): pozice figurky, kterou chceme vyhoditVyhození figurky z hrací desky
  Args: piecePosition ([int, int]): pozice figurky, kterou chceme vyhodit

- *Metoda:*
  $_{p}rintToTerminal$ *Vytiskne hrací desku do konzole*

Soubor: ChessBoard.py

Třída: ChessBoard

– Třída reprezentující šachovnici s figurkami. Dědí od třídy Board.
Attributes: board (list of list of Piece): Dvourozměrné pole, které reprezentuje šachovnici. Každé políčko obsahuje instanci třídy Piece, nebo None, pokud je políčko prázdné.

– Konstruktor třídy ChessBoard. Vytvoří šachovnici 8x8 a umístí na ni všechny figury ve standardním pořadí.

– Vrací figuru na určeném místě na šachovnici, nebo None, pokud je políčko prázdné.
Args: index [int, int]: Tuple dvou integerů, (row, col), oba 0-7

Returns: Piece: Instance třídy Piece, nebo None, pokud je políčko prázdné

- Nastaví políčko na šachovnici jako board[row,col] namísto board.board[row][col]

  Args: index ([int, int]): Tuple dvou integerů, (row, col), oba 0-7 value (Piece): Instance třídy Piece, nebo None, pokud má být políčko prázdné

  Returns: bool: True, pokud se podařilo nastavit políčko, jinak False

- Vrací string reprezentaci šachovnice. Každé políčko je reprezentováno jako string, který je tvořen z informací o barvě a symbolu figury, nebo jako string "

  $^{"}$,*pokudjepolíčkoprázdné.Políčkajsouoddělenamezerouajednotlivéřádkyjsouoddělenyznakemnovéhořádku*().*Nastavíšac*

- Vrací list všech figurek dané barvy na šachovnici.

  Args: color (Enum Colors): Barva figurek, které chceme najít (Colors.WHITE nebo Colors.BLACK)

  Returns: List of Pieces: List figurek dané barvy

- Vrátí kopii šachovnice. Každá figura z originální šachovnice je nahrazena její kopií.

  Returns: ChessBoard: Kopie šachovnice

- Porovná dvě šachovnice. Tato metoda porovná dvě šachovnice a vrátí True, pokud jsou stejné. Jinak vrátí False.

  Args: board: Šachovnice, která se má porovnat s aktuální šachovnicí

  Returns: Boolean:True, pokud jsou šachovnice stejné, jinak False

- Vrátí šachovnici jako list of Field.

  Returns: List of Struct Field: šachovnice jako list of Field

- *Metoda:*

  $_{i}{}^{nit}$ *Konstruktor třídy ChessBoard. Vytvoří šachovnici 8x8 a umístí na ni všechny figury ve standardním*

  Returns: Piece: Instance třídy Piece, nebo None, pokud je políčko prázdné

- *Metoda:*

  $_{s}{}^{etitem}$

  Args: index ([int, int]): Tuple dvou integerů, (row, col), oba 0-7 value (Piece): Instance třídy Piece, nebo None, pokud má být políčko prázdné

  Returns: bool: True, pokud se podařilo nastavit políčko, jinak False

- *Metoda:*

  $_{s}{}^{tr}$ *Vrací string reprezentaci šachovnice. Každé políčko je reprezentováno jako string, který je tvořen z i*

  Returns: List of Pieces: List figurek dané barvy

- *Metoda: copy*

  Returns: ChessBoard: Kopie šachovnice*Vrátí kopii šachovnice. Každá figura z originální šachovnice je nahrazena její kopií.*

  Returns: ChessBoard: Kopie šachovnice

- *Metoda: compare*

  Args: board: Šachovnice, která se má porovnat s aktuální šachovnicí

  Returns: Boolean:True, pokud jsou šachovnice stejné, jinak FalsePorovná dvě šachovnice. Tato metoda porovná dvě šachovnice a vrátí True, pokud jsou stejné. Jinak vrátí False.

  Args: board: Šachovnice, která se má porovnat s aktuální šachovnicí

  Returns: Boolean:True, pokud jsou šachovnice stejné, jinak False

- *Metoda: getListOfBoard*

  Returns: List of Struct Field: šachovnice jako list of FieldVrátí šachovnici jako list of Field.

  Returns: List of Struct Field: šachovnice jako list of Field

Soubor: ChessMines.py

Třída: ChessMines

- Hra šachy s minami.

  Attributes: mines (list of [int, int]): seznam min explosion ([int, int]): pozice exploze
- Inicializace hry.
- Umístí miny na náhodné pozice.
- Vrátí hrací desku.
- Zpracuje tah.

- *Metoda:*

  ^{init}*Inicializace hry.Metoda:*$_{placeMines}$*Umístí miny na náhodné pozice.Metoda: getBoardVrátí hrací desku.*

- *Metoda: makeMove Zpracuje tah.*

Soubor: ChessMinesWithFogOfWar.py

Třída: ChessMinesWithFogOfWar

- Hra šachy s minami a mlhou války.

  Attributes: fog (bool): True, pokud je mlha války zapnutá, jinak False
- Konstruktor třídy ChessMinesWithFogOfWar

- Vrátí hrací desku.

  Args: color (Enum Colors): Barva hráče, pro kterého se má šachovnice vykreslit

- *Metoda:*

  $_{i}{}^{nit}$Konstruktor třídy ChessMinesWithFogOfWarMetoda: getBoard

  Args: color (Enum Colors): Barva hráče, pro kterého se má šachovnice vykreslit

Složka: pieces

Soubor: Bishop.py

Třída: Bishop

- Třída reprezentující figurku střelce v šachu. Dědí od třídy Piece.

  Attributes: color (Colors): barva figurky position ([int,int]): pozice figurky symbol (str): symbol figurky value (int): hodnota figurky

- Konstruktor třídy Bishop. Volá konstruktor třídy Piece.

  Args: color (Colors): barva figurky position ([int,int]): pozice figurky

- Vytvoří kopii instance třídy Bishop.

  Returns: Bishop: kopie instance třídy Bishop

- Vrátí seznam možných tahů pro střelce.

  Args: board (Board): šachovnice, na které se figurka nachází

- *Metoda:*

  $_{i}{}^{nit}$

  Args: color (Colors): barva figurky position ([int,int]): pozice figurky

- *Metoda: copy*

  Returns: Bishop: kopie instance třídy Bishop*Vytvoří kopii instance třídy Bishop.*

  Returns: Bishop: kopie instance třídy Bishop

- *Metoda: possibleMoves*

  Args: board (Board): šachovnice, na které se figurka nacházíVrátí seznam možných tahů pro střelce.

  Args: board (Board): šachovnice, na které se figurka nachází

Soubor: King.py

Třída: King

- Třída reprezentující figurku krále v šachu. Dědí od třídy Piece.
  Attributes: color (Colors): barva figurky position ([int,int]): pozice figurky symbol (str): symbol figurky hasMoved (bool): True, pokud figurka už byla pohnuta, jinak False
- Konstruktor třídy King. Volá konstruktor třídy Piece.
  Args: color (Colors): barva figurky position ([int,int]): pozice figurky
- Vytvoří kopii instance třídy King.
  Returns: King: kopie instance třídy King
- Zkontroluje, zda je možné provést tah králem a provede ho.
  Args: board - šachovnice end - cílová pozice tahu
- Vrátí seznam možných tahů pro krále.
  Args: board (Board): šachovnice, na které se figurka nachází
  Returns: List of [int, int] : seznam možných tahů krále

- *Metoda:*
  $_{i}nit$
  Args: color (Colors): barva figurky position ([int,int]): pozice figurky

- *Metoda: copy*
  Returns: King: kopie instance třídy King*Vytvoří kopii instance třídy King.*
  Returns: King: kopie instance třídy King

- *Metoda: move*
  Args: board - šachovnice end - cílová pozice tahuZkontroluje, zda je možné provést tah králem a provede ho.
  Args: board - šachovnice end - cílová pozice tahu

- *Metoda: possibleMoves*
  Args: board (Board): šachovnice, na které se figurka nachází
  Returns: List of [int, int] : seznam možných tahů králeVrátí seznam možných tahů pro krále.
  Args: board (Board): šachovnice, na které se figurka nachází
  Returns: List of [int, int] : seznam možných tahů krále

Soubor: Knight.py

Třída: Knight

- Třída reprezentující figurku jezdce v šachu. Dědí od třídy Piece.
  Attributes: color (Colors): barva figurky position ([int,int]): pozice figurky symbol (str): symbol figurky value (int): hodnota figurky
- Konstruktor třídy Knight. Volá konstruktor třídy Piece
  Args: color (Colors): barva figurky position ([int, int]): pozice figurky
- Vytvoří kopii instance třídy Knight.
  Returns: Knight: kopie instance třídy Knight
- Vrátí seznam možných tahů pro jezdce.
  Args: board (dict): šachovnice
  Returns: List of [int,int]: seznam možných tahů

- *Metoda:*

  $_{i}nit$

  Args: color (Colors): barva figurky position ([int, int]): pozice figurky

- *Metoda: copy*

  Returns: Knight: kopie instance třídy Knight *Vytvoří kopii instance třídy Knight.*

  Returns: Knight: kopie instance třídy Knight

- *Metoda: possibleMoves*

  Args: board (dict): šachovnice

  Returns: List of [int,int]: seznam možných tahů Vrátí seznam možných tahů pro jezdce.

  Args: board (dict): šachovnice

  Returns: List of [int,int]: seznam možných tahů

Soubor: Pawn.py

Třída: Pawn

- Třída reprezentující pěšce v šachu. Dědí od třídy Piece.
  Attributes: color (Colors): barva figurky position ([int,int]): pozice figurky symbol (str): symbol figurky value (int): hodnota figurky hasMoved (bool): True, pokud figurka už byla pohnuta, jinak False lastMoveWasDouble (bool): True, pokud poslední tah figurkou byl dvojitý, jinak False
- Konstruktor třídy Pawn. Volá konstruktor třídy Piece.
  Args: color (Colors): barva figurky position ([int, int]): pozice figurky

- Vytvoří kopii instance třídy Pawn.
  Returns: Pawn: kopie instance třídy Pawn
- Zkontroluje, zda je možné provést tah pěšcem a provede ho.
  Args: board (dict): šachovnice end ([int, int]): cílová pozice tahu
- Vrátí seznam možných tahů pro pěšce.
  Args: board (dict): šachovnice
  Returns: List of [int,int]: seznam možných tahů

- *Metoda:*
  $_{i}nit$
  Args: color (Colors): barva figurky position ([int, int]): pozice figurky

- *Metoda: copy*
  Returns: Pawn: kopie instance třídy Pawn*Vytvoří kopii instance třídy Pawn.*
  Returns: Pawn: kopie instance třídy Pawn

- *Metoda: move*
  Args: board (dict): šachovnice end ([int, int]): cílová pozice tahuZkontroluje, zda je možné provést tah pěšcem a provede ho.
  Args: board (dict): šachovnice end ([int, int]): cílová pozice tahu

- *Metoda: possibleMoves*
  Args: board (dict): šachovnice
  Returns: List of [int,int]: seznam možných tahůVrátí seznam možných tahů pro pěšce.
  Args: board (dict): šachovnice
  Returns: List of [int,int]: seznam možných tahů

Soubor: Piece.py

Třída: Piece

- Třída reprezentující jednu figurku na šachovnici.
  Attributes: color (Colors): Barva figurky position ([int,int]): Pozice figurky hasMoved (bool): True, pokud figurka už byla pohnuta, jinak False lastMoveWasDouble (bool): True, pokud poslední tah figurkou byl dvojitý, jinak False value (int): Hodnota figurky
- Konstruktor třídy Piece. Nastaví barvu a pozici figurky.
  Args: color (Colors): Barva figurky position ([int, int]): Pozice figurky

  - Metoda pro pohnutí figurky po šachovnici
    Args: board (Board): Sachovnice, na kterou se tah províde end ([row, col]): Policko, kam se tah províde
  - Vrací řádek, na kterém se figurka nachází
    Returns: int: Řádek figurky
  - Vrací sloupec, na kterém se figurka nachází
    Returns: int: Sloupec figurky
  - Funkce pro vypsání všech možných tahů danou figurkou
    Args: board (Board): Sachovnice, na kterou se tah províde
    Returns: (list of [row, col]): Seznam všech možných tahů ve formátu
  - Vraci kopii objektu. Pouziva se, kdybychom chteli mit kopii objektu, bez toho, aby se menil puvodni objekt.
    Returns: (Piece): Kopie objektu

- *Metoda:*
  $_{i}nit$
  Args: color (Colors): Barva figurky position ([int, int]): Pozice figurky

- *Metoda: move*
  Args: board (Board): Sachovnice, na kterou se tah províde end ([row, col]): Policko, kam se tah províde*Metoda pro pohnutí figurky po šachovnici*
  Args: board (Board): Sachovnice, na kterou se tah províde end ([row, col]): Policko, kam se tah províde

- *Metoda: row*
  Returns: int: Řádek figurkyVrací řádek, na kterém se figurka nachází
  Returns: int: Řádek figurky

- *Metoda: col*
  Returns: int: Sloupec figurkyVrací sloupec, na kterém se figurka nachází
  Returns: int: Sloupec figurky

- *Metoda: possibleMoves*
  Args: board (Board): Sachovnice, na kterou se tah províde
  Returns: (list of [row, col]): Seznam všech možných tahů ve formátuFunkce pro vypsání všech možných tahů danou figurkou
  Args: board (Board): Sachovnice, na kterou se tah províde
  Returns: (list of [row, col]): Seznam všech možných tahů ve formátu

- *Metoda: copy*
  Returns: (Piece): Kopie objektuVraci kopii objektu. Pouziva se, kdybychom chteli mit kopii objektu, bez toho, aby se menil puvodni objekt.
  Returns: (Piece): Kopie objektu

Soubor: Queen.py

Třída: Queen

- Třída reprezentující figurku královny v šachu. Dědí od třídy Piece.

  Attributes: color (Colors): Barva figurky position ([int,int]): Pozice figurky hasMoved (bool): True, pokud figurka už byla pohnuta, jinak False lastMoveWasDouble (bool): True, pokud poslední tah figurkou byl dvojitý, jinak False value (int): Hodnota figurky symbol (str): Symbol figurky
- Konstruktor třídy Queen. Volá konstruktor třídy Piece.
- Vytvoří kopii instance třídy Queen.

  Returns: Queen: kopie instance
- Vrátí seznam možných tahů pro královnu.

  Args: board (dict): šachovnice

  Returns: List of [int, int]: seznam možných tahů figurky

- *Metoda:*

  $_{i}{}^{nit}$Konstruktor třídy Queen. Volá konstruktor třídy Piece. Metoda: copy

  Returns: Queen: kopie instance

- *Metoda: possibleMoves*

  Args: board (dict): šachovnice

  Returns: List of [int, int]: seznam možných tahů figurky*Vrátí seznam možných tahů pro královnu.*

  Args: board (dict): šachovnice

  Returns: List of [int, int]: seznam možných tahů figurky

Soubor: Rook.py

Třída: Rook

- Třída reprezentující figurku věže v šachu. Dědí od třídy Piece.

  Attributes: color (Colors): Barva figurky position ([int,int]): Pozice figurky hasMoved (bool): True, pokud figurka už byla pohnuta, jinak False lastMoveWasDouble (bool): True, pokud poslední tah figurkou byl dvojitý, jinak False value (int): Hodnota figurky symbol (str): Symbol figurky
- Konstruktor třídy Rook. Volá konstruktor třídy Piece.

  Args: color (Colors): Barva figurky position ([int,int]): Pozice figurky
- Vytvoří kopii instance třídy Rook.

  Returns: Rook: kopie instance

– Vrátí seznam možných tahů pro věž.
Args: board (dict): šachovnice
Returns: List of [int, int]: seznam možných tahů figurky

- *Metoda:*
  $_{i}nit$
  Args: color (Colors): Barva figurky position ([int,int]): Pozice figurky
- *Metoda: copy*
  Returns: Rook: kopie instance*Vytvoří kopii instance třídy Rook.*
  Returns: Rook: kopie instance
- *Metoda: possibleMoves*
  Args: board (dict): šachovnice
  Returns: List of [int, int]: seznam možných tahů figurkyVrátí seznam možných tahů pro věž.
  Args: board (dict): šachovnice
  Returns: List of [int, int]: seznam možných tahů figurky

Soubor: $_{i}nit_{.py}$

Složka: __pycache__

Soubor: $_{i}nit_{.py}$

Složka: __pycache__

Složka: chessTrackGame

Soubor: ChessTrackGame.py

Třída: ChessTrackGame

– Hra ChessTrackGame.
Attributes: board (ChessTrackGameBoard): Hrací deska currentPlayer (Enum Colors): Barva hráče, který je na tahu
– Inicializace hry.
– Vrácení hrací desky.
Args: color (Enum Colors): Barva hráče.
Returns: ChessTrackGameBoard: Hrací deska.

  - Provedení tahu.

    Args: position ([int,int]]): Pozice, na kterou se má kámen umístit. color (Enum Colors): Barva kamene. rightClick (bool): True, pokud se jedná o pravé tlačítko myši.

    Returns: bool: True, pokud je tah platný, jinak False.
  - Kontrola konce hry.

    Returns: String: Který hráč vyhrál, pokud hra skončila, jinak None.
  - Vypsání hrací desky do terminálu.

- *Metoda:*

  $_{i}{}^{nit}$*Inicializace hry.Metoda: getBoard*

  Args: color (Enum Colors): Barva hráče.

  Returns: ChessTrackGameBoard: Hrací deska.

- *Metoda: makeMove*

  Args: position ([int,int]]): Pozice, na kterou se má kámen umístit. color (Enum Colors): Barva kamene. rightClick (bool): True, pokud se jedná o pravé tlačítko myši.

  Returns: bool: True, pokud je tah platný, jinak False.*Provedení tahu.*

  Args: position ([int,int]]): Pozice, na kterou se má kámen umístit. color (Enum Colors): Barva kamene. rightClick (bool): True, pokud se jedná o pravé tlačítko myši.

  Returns: bool: True, pokud je tah platný, jinak False.

- *Metoda: checkEnd*

  Returns: String: Který hráč vyhrál, pokud hra skončila, jinak None.Kontrola konce hry.

  Returns: String: Který hráč vyhrál, pokud hra skončila, jinak None.

- *Metoda:*
  $_{p}{}^{rintToTerminal}$*Vypsání hrací desky do terminálu.*

Soubor: ChessTrackGameBoard.py

Třída: ChessTrackGameBoard

  - Hrací deska hry ChessTrackGame.

    Attributes: board (list): Hrací deska moves (int): Počet tahů
  - Inicializace hrací desky.
  - Inicializace hrací desky.
  - Otočení hrací desky.

- Umístění kamene na desku.
  Args: position ([int,int]]): Pozice, na kterou se má kámen umístit. color (Enum Colors): Barva kamene.
- Provedení tahu.
  Args: position ([int,int]]): Pozice, na kterou se má kámen umístit. color (Enum Colors): Barva kamene.
- Vrátí seznam seznamů reprezentující hrací desku.
  Returns: list: Seznam seznamů reprezentující hrací desku.
- Kontrola konce hry.
  Returns: str: Výsledek hry.
- Kontrola výherce.
  Returns: str: Výsledek hry.

- *Metoda:*

  $_{i}{}^{nit}$*Inicializace hrací desky.Metoda:*$_{populateBoard}$*Inicializace hrací desky.Metoda:*$_{spinBoard}$*Otočení hrací de*

- *Metoda: makeMove*

  Args: position ([int,int]]): Pozice, na kterou se má kámen umístit. color (Enum Colors): Barva kamene.*Provedení tahu.*

  Args: position ([int,int]]): Pozice, na kterou se má kámen umístit. color (Enum Colors): Barva kamene.

- *Metoda: getListOfBoard*

  Returns: list: Seznam seznamů reprezentující hrací desku.Vrátí seznam seznamů reprezentující hrací desku.

  Returns: list: Seznam seznamů reprezentující hrací desku.

- *Metoda: checkEnd*

  Returns: str: Výsledek hry.Kontrola konce hry.

  Returns: str: Výsledek hry.

- *Metoda:*

  $_{c}heckWinner$

  Returns: str: Výsledek hry.

Soubor: $_{i}nit_{.py}$

Složka: __pycache__

Soubor: ChessWithFogOfWar.py

Třída: ChessWithFogOfWar

- Třída ChessWithFogOfWar slouží k reprezentaci hry Šachy s mlhou války.

  - Konstruktor třídy ChessWithFogOfWar
  - Vrací zakrytou šachovnici
    Args: color (Enum Colors): Barva hráče na tahu
    Returns: Array of Field: zakrytá šachovnice

- *Metoda:*

  $_{init}$*Konstruktor třídy ChessWithFogOfWar Metoda: getBoard*

  Args: color (Enum Colors): Barva hráče na tahu

  Returns: Array of Field: zakrytá šachovnice

Složka: connectFour

Soubor: ConnectFour.py

Třída: ConnectFour

- Hrací deska pro hru ConnectFour
  - Inicializace hry ConnectFour
  - Vrací hrací desku
    Args: color (Enum, optional): Barva hráče. Defaults to None.
    Returns: List of Struct: Hrací deska
  - Provedení tahu
    Args: position ([int,int]): Pozice tahu color (Enum, optional): Barva hráče. Defaults to None. rightClick (bool, optional): True, pokud se jedná o pravé tlačítko myši. Defaults to False.
    Returns: bool: True, pokud se tah podařil, jinak False
  - Zjištění konce hry
    Returns: Enum: Barva vítěze
  - Výpis hrací desky do konzole

- *Metoda:*

  $_{init}$*Inicializace hry ConnectFour Metoda: getBoard*

  Args: color (Enum, optional): Barva hráče. Defaults to None.

  Returns: List of Struct: Hrací deska

- *Metoda: makeMove*

  Args: position ([int,int]): Pozice tahu color (Enum, optional): Barva hráče. Defaults to None. rightClick (bool, optional): True, pokud se jedná o pravé tlačítko myši. Defaults to False.

  Returns: bool: True, pokud se tah podařil, jinak False*Provedení tahu*

Args: position ([int,int]): Pozice tahu color (Enum, optional): Barva hráče. Defaults to None. rightClick (bool, optional): True, pokud se jedná o pravé tlačítko myši. Defaults to False.

Returns: bool: True, pokud se tah podařil, jinak False

- *Metoda: checkEnd*

  Returns: Enum: Barva vítězeZjištění konce hry

  Returns: Enum: Barva vítěze

- *Metoda:*
  $_{p}rintToTerminal$ Výpis hrací desky do konzole

Soubor: ConnectFourBoard.py

Třída: ConnectFourBoard

- Hrací deska pro hru ConnectFour
- Inicializace hry ConnectFour
- Vytvoří string hrací desky na výpis do konzole
- Inicializace herní desky
- Provedení tahu
  Args: position ([int,int]): Pozice tahu up (bool, optional): Směr zjišťování umístění. Defaults to False, tedy dolů.
  Returns: [int,int]: Pozice, kam se má umístit kámen
- Provedení tahu
  Args: position ([int,int]): Pozice tahu color (Enum): Barva hráče
  Returns: bool: True, pokud se tah podařil, jinak False
- Zjistí, zda je herní deska plná
  Returns: bool: True, pokud je deska plná, jinak False
- Zjistí, zda je konec hry
  Args: position ([int,int]): Pozice tahu color (Enum): Barva hráče
  Returns: bool: True, pokud je konec hry, jinak False
- Zjistí, zda hráč vyhrál
  Args: color (Enum): Barva hráče
  Returns: bool: True, pokud hráč vyhrál, jinak False
- Zjistí, zda hráč vyhrál horizontálně
  Args: color (Enum): Barva hráče
  Returns: bool: True, pokud hráč vyhrál horizontálně, jinak False
- Zjistí, zda hráč vyhrál vertikálně
  Args: color (Enum): Barva hráče
  Returns: bool: True, pokud hráč vyhrál vertikálně, jinak False

  - Zjistí, zda hráč vyhrál diagonálně
    Args: color (Enum): Barva hráče
    Returns: bool: True, pokud hráč vyhrál diagonálně, jinak False
  - Vrací šachovnici jako list
    Returns: List of Struct : List, kde každý řádek je list obsahující figury na daném řádku

- *Metoda:*

  $_{init}$*Inicializace hry ConnectFour Metoda:*$_{str}$ *Vytvoří string hrací desky na výpis do konzole Metoda:*$_{popula}$

  Args: position ([int,int]): Pozice tahu up (bool, optional): Směr zjišťování umístění. Defaults to False, tedy dolů.

  Returns: [int,int]: Pozice, kam se má umístit kámen

- *Metoda: makeMove*

  Args: position ([int,int]): Pozice tahu color (Enum): Barva hráče

  Returns: bool: True, pokud se tah podařil, jinak False*Provedení tahu*

  Args: position ([int,int]): Pozice tahu color (Enum): Barva hráče

  Returns: bool: True, pokud se tah podařil, jinak False

- *Metoda: isFull*

  Returns: bool: True, pokud je deska plná, jinak FalseZjistí, zda je herní deska plná

  Returns: bool: True, pokud je deska plná, jinak False

- *Metoda: checkEnd*

  Args: position ([int,int]): Pozice tahu color (Enum): Barva hráče

  Returns: bool: True, pokud je konec hry, jinak FalseZjistí, zda je konec hry

  Args: position ([int,int]): Pozice tahu color (Enum): Barva hráče

  Returns: bool: True, pokud je konec hry, jinak False

- *Metoda: checkWin*

  Args: color (Enum): Barva hráče

  Returns: bool: True, pokud hráč vyhrál, jinak FalseZjistí, zda hráč vyhrál

  Args: color (Enum): Barva hráče

  Returns: bool: True, pokud hráč vyhrál, jinak False

- *Metoda: checkHorizontal*

  Args: color (Enum): Barva hráče

  Returns: bool: True, pokud hráč vyhrál horizontálně, jinak FalseZjistí, zda hráč vyhrál horizontálně

  Args: color (Enum): Barva hráče

  Returns: bool: True, pokud hráč vyhrál horizontálně, jinak False

- *Metoda: checkVertical*

  Args: color (Enum): Barva hráče

  Returns: bool: True, pokud hráč vyhrál vertikálně, jinak FalseZjistí, zda hráč vyhrál vertikálně

  Args: color (Enum): Barva hráče

  Returns: bool: True, pokud hráč vyhrál vertikálně, jinak False

- *Metoda: checkDiagonal*

  Args: color (Enum): Barva hráče

  Returns: bool: True, pokud hráč vyhrál diagonálně, jinak FalseZjistí, zda hráč vyhrál diagonálně

  Args: color (Enum): Barva hráče

  Returns: bool: True, pokud hráč vyhrál diagonálně, jinak False

- *Metoda: getListOfBoard*

  Returns: List of Struct : List, kde každý řádek je list obsahující figury na daném řádkuVrací šachovnici jako list

  Returns: List of Struct : List, kde každý řádek je list obsahující figury na daném řádku

Soubor: $_{i}nit_{.py}$

Složka: __pycache__

Soubor: Enums.py

Třída: Field

- Třída reprezentující políčko na šachovnici
- Enum pro typy figurek
- Enum pro barvy
- Vrací string reprezentaci barvy
  Returns: String: reprezentovaná barva
- Vrátí inverzní barvu
  Returns: Enum Colors: inverzní barva
- Vrátí další barvu
  Returns: Enum Colors: další barva

Třída: Figures

  - Třída reprezentující políčko na šachovnici
  - Enum pro typy figurek
  - Enum pro barvy
  - Vrací string reprezentaci barvy
    Returns: String: reprezentovaná barva
  - Vrátí inverzní barvu
    Returns: Enum Colors: inverzní barva
  - Vrátí další barvu
    Returns: Enum Colors: další barva

- *Atribut: PAWN*
- *Atribut: ROOK*
- *Atribut: BISHOP*
- *Atribut: KNIGHT*
- *Atribut: QUEEN*
- *Atribut: KING*
- *Atribut: X*
- *Atribut: O*
- *Atribut: FLAG*
- *Atribut: EXPLOSION*
- *Atribut: MINE*
- *Atribut: ONE*
- *Atribut: TWO*
- *Atribut: THREE*
- *Atribut: FOUR*
- *Atribut: FIVE*
- *Atribut: SIX*
- *Atribut: SEVEN*
- *Atribut: EIGHT*
- *Atribut: SHADOW*
- *Atribut: MOLE*
- *Atribut: LOGO*

Třída: Colors

- Třída reprezentující políčko na šachovnici
- Enum pro typy figurek
- Enum pro barvy
- Vrací string reprezentaci barvy
  Returns: String: reprezentovaná barva
- Vrátí inverzní barvu
  Returns: Enum Colors: inverzní barva
- Vrátí další barvu
  Returns: Enum Colors: další barva

- *Atribut: WHITE*
- *Atribut: BLACK*
- *Atribut: RED*
- *Atribut: GREEN*
- *Metoda:*
  $_{s}tr$
  Returns: String: reprezentovaná barva
- *Metoda: changeColor*
  Returns: Enum Colors: inverzní barva*Vrátí inverzní barvu*
  Returns: Enum Colors: inverzní barva
- *Metoda: changeColorFour*
  *Returns: Enum Colors: další barva**Vrátí další barvu*
  Returns: Enum Colors: další barva

Soubor: GameTemplate.py

Třída: GameTemplate

- Třída GameTemplate slouží k reprezentaci šablony hry.
- Konstruktor třídy GameTemplate

- *Metoda:*
  $_{i}nit$*Konstruktor třídy GameTemplate*

Soubor: GameTests.py

Třída: GameTests

- Testy na hry
- Testuje, zda se vytvoří hrací pole

  Args: name (string): jméno hry game
  ${}_{c}lass(game) : třídahryTestuje, zdasefigurkadávybrat$
- Args: name (string): jméno hry game
  ${}_{c}lass(game) : třídahryposition([int, int]) : pozicefigurky, sekterousemápohnoutcolor(EnumColor$

Args: name (string): jméno hry game
${}_{c}lass(game) : třídahryposition([int, int]) : pozicefigurky, sekterousemápohnoutcolor(EnumColors) : barvan$

- Testuje, zda se figurka nedá vybrat

  Args: name (string): jméno hry game
  ${}_{c}lass(game) : třídahryposition([int, int]) : pozicefigurky, sekterousemápohnoutcolor(EnumColors) : barvan$
- Args: name (string): jméno hry game
  ${}_{c}lass(game) : třídahrychoose_position([int, int]) : pozicefigurky, sekterousemápohnoutcolor(EnumColors) :$

  Args: name (string): jméno hry game
  ${}_{c}lass(game) : třídahrychoose_position([int, int]) : pozicefigurky, sekterousemápohnoutcolor(EnumColors) :$
- Testuje, zda se pohyb nelze provést

  Args: name (string): jméno hry game
  ${}_{c}lass(game) : třídahrychoose_position([int, int]) : pozicefigurky, sekterousemápohnoutcolor(EnumColors) :$
- Args: name (string): jméno hry game
  ${}_{c}lass(game) : třídahryTestuje, zdajedefinovánovybíránífigurky$

  Args: name (string): jméno hry game
  ${}_{c}lass(game) : třídahry$
- Testuje, zda je definován mlhový efekt

  Args: name (string): jméno hry game
  ${}_{c}lass(game) : třídahryTestuje, zdajedefinovánpočethráčů$
- Args: name (string): jméno hry game
  ${}_{c}lass(game) : třídahryTestuje, zdasehrazahrajedokonce$

  Args: name (string): jméno hry game
  ${}_{c}lass(game) : třídahry$

  - *Metoda: testInitialBoard*

    Args: name (string): jméno hry game
    ${}_{c}lass(game) : třídahryTestuje, zdasevytvoříhracípole$

    Args: name (string): jméno hry game
    ${}_{c}lass(game) : třídahryMetoda: testChoosePiece$

Args: name (string): jméno hry game
${}_{c}lass(game) : třídahryposition([int, int]) : pozicefigurky, sekterousemápohnoutcolor(EnumColors) : barvan$

- *Metoda: testChooseWrongPiece*

  Args: name (string): jméno hry game
  $_{c}lass(game) : třídahryposition([int, int]) : pozicefigurky, sekterousemápohnoutcolor(EnumColors) : barvan$

  Args: name (string): jméno hry game
  $_{c}lass(game) : třídahryposition([int, int]) : pozicefigurky, sekterousemápohnoutcolor(EnumColors) : barvan$

  Args: name (string): jméno hry game
  $_{c}lass(game) : třídahryposition([int, int]) : pozicefigurky, sekterousemápohnoutcolor(EnumColors) : barvan$

- *Metoda: testMakeMove*

  Args: name (string): jméno hry game
  $_{c}lass(game) : třídahrychoose_{p}osition([int, int]) : pozicefigurky, sekterousemápohnoutcolor(EnumColors) :$

  Args: name (string): jméno hry game
  $_{c}lass(game) : třídahrychoose_{p}osition([int, int]) : pozicefigurky, sekterousemápohnoutcolor(EnumColors) :$

  Args: name (string): jméno hry game
  $_{c}lass(game) : třídahrychoose_{p}osition([int, int]) : pozicefigurky, sekterousemápohnoutcolor(EnumColors) :$

- *Metoda: testMakeUnableMove*

  Args: name (string): jméno hry game
  $_{c}lass(game) : třídahrychoose_{p}osition([int, int]) : pozicefigurky, sekterousemápohnoutcolor(EnumColors) :$

  Args: name (string): jméno hry game
  $_{c}lass(game) : třídahrychoose_{p}osition([int, int]) : pozicefigurky, sekterousemápohnoutcolor(EnumColors) :$

  Args: name (string): jméno hry game
  $_{c}lass(game) : třídahry$

- *Metoda: testWithChoosingPiece*

  Args: name (string): jméno hry game
  $_{c}lass(game) : třídahryTestuje, zdajedefinovánovybíránífigurky$

  Args: name (string): jméno hry game
  $_{c}lass(game) : třídahryMetoda: testFog$

  Args: name (string): jméno hry game
  $_{c}lass(game) : třídahry$

- *Metoda: testNumberOfPlayers*

  Args: name (string): jméno hry game
  $_{c}lass(game) : třídahryTestuje, zdajedefinovánpočethráčů$

  Args: name (string): jméno hry game
  $_{c}lass(game) : třídahryMetoda: testSimulateFullGame$

  Args: name (string): jméno hry game
  $_{c}lass(game) : třídahry$

Složka: humanDoNotWorry

Soubor: HumanDoNotWorry.py

Třída: HumanDoNotWorry

- Třída pro hru člověče, nezlob se.
  - Konstruktor třídy hry člověče, nezlob se.
  - Metoda vrátí hrací desku.

    Args: color (Enum Colors, optional): Barva hráče na tahu. Defaults to None.

    Returns: List of list of field: Hrací deska
  - Metoda zvolí figurku, kterou se bude hrát.

    Args: position (int): Pozice figurky color (Enum Colors, optional): Barva hráče. Defaults to None.

    Returns: bool: True, pokud se podařilo zvolit figurku, jinak False
  - Metoda provede tah figurkou.

    Args: position ([int,int]): Pozice figurky color (Enum Colors, optional): Barva hráče. Defaults to None. rightClick (bool, optional): True, pokud se jedná o pravé tlačítko myši. Defaults to False.

    Returns: bool: True, pokud se podařilo provést tah, jinak False
  - Metoda provede standardní tah figurkou.

    Args: row (int): řádek col (int): sloupec
  - Metoda umístí figurku do cíle.

    Args: board (Board): Hrací deska piece (Piece): Figurka
  - Metoda zkontroluje, zda hra skončila.

    Returns: string: Vrátí barvu hráče, který vyhrál, jinak None
  - Metoda hodí kostkou.

    Returns: int: Hodnota kostky
  - Metoda vytiskne hrací desku na obrazovku.

- *Metoda:*

  $_{init}$ Konstruktor třídy hry člověče, nezlob se. Metoda: getBoard

  Args: color (Enum Colors, optional): Barva hráče na tahu. Defaults to None.

  Returns: List of list of field: Hrací deska

- *Metoda: choosePiece*

  Args: position (int): Pozice figurky color (Enum Colors, optional): Barva hráče. Defaults to None.

Returns: bool: True, pokud se podařilo zvolit figurku, jinak False*Metoda zvolí figurku, kterou se bude hrát.*

Args: position (int): Pozice figurky color (Enum Colors, optional): Barva hráče. Defaults to None.

Returns: bool: True, pokud se podařilo zvolit figurku, jinak False

- *Metoda: makeMove*

  Args: position ([int,int]): Pozice figurky color (Enum Colors, optional): Barva hráče. Defaults to None. rightClick (bool, optional): True, pokud se jedná o pravé tlačítko myši. Defaults to False.

  Returns: bool: True, pokud se podařilo provést tah, jinak FalseMetoda provede tah figurkou.

  Args: position ([int,int]): Pozice figurky color (Enum Colors, optional): Barva hráče. Defaults to None. rightClick (bool, optional): True, pokud se jedná o pravé tlačítko myši. Defaults to False.

  Returns: bool: True, pokud se podařilo provést tah, jinak False

- *Metoda:*

  $_{m}akeStandartMove$

  Args: row (int): řádek col (int): sloupec

- *Metoda:*

  $_{p}laceToFinal$

  Args: board (Board): Hrací deska piece (Piece): Figurka

- *Metoda: checkEnd*

  Returns: string: Vrátí barvu hráče, který vyhrál, jinak NoneMetoda zkontroluje, zda hra skončila.

  Returns: string: Vrátí barvu hráče, který vyhrál, jinak None

- *Metoda: rollDice*

  Returns: int: Hodnota kostkyMetoda hodí kostkou.

  Returns: int: Hodnota kostky

- *Metoda:*

  $_{p}rintToTerminal$*Metoda vytiskne hrací desku na obrazovku.*

Soubor: HumanDoNotWorryBoard.py

Třída: HumanDoNotWorryBoard

– Třída HumanDoNotWorryBoard slouží k reprezentaci hrací desky hry člověče, nezlob se.

– Konstruktor třídy HumanDoNotWorryBoard.

  - Vrátí textovou reprezentaci instance třídy HumanDoNotWorryBoard.
  - Metoda, která zjistí, zda je alespoň jedna figurka zadané barvy nasazena.
    Args: color (Colors): Barva figurky
  - Metoda, která naplní hrací desku figurkami.
  - Metoda, která vrátí seznam seznamů reprezentující hrací desku.
    Returns: List of List of Field: Seznam seznamů reprezentující hrací desku

- *Metoda:*
  $_{init}$*Konstruktor třídy HumanDoNotWorryBoard. Metoda:* $_{str}$ *Vrátí textovou reprezentaci instance třídy H*
  Args: color (Colors): Barva figurky

- *Metoda:*
  $_{populateBoard}$*Metoda, která naplní hrací desku figurkami. Metoda: getListOfBoard*
  Returns: List of List of Field: Seznam seznamů reprezentující hrací desku

Složka: pieces

Soubor: BlackPiece.py

Třída: BlackPiece

  - Třída BlackPiece slouží k reprezentaci jedné černé figurky.
  - Konstruktor třídy BlackPiece.
    Args: position ([int,int]): Pozice figurky
  - Vrací textovou reprezentaci instance třídy BlackPiece.
    Returns: str: Textová reprezentace instance třídy BlackPiece

- *Metoda:*
  $_{init}$
  Args: position ([int,int]): Pozice figurky

- *Metoda:*
  $_{str}$
  Returns: str: Textová reprezentace instance třídy BlackPiece

Soubor: GreenPiece.py

Třída: GreenPiece

- Třída GreenPiece slouží k reprezentaci jedné zelené figurky.
- Konstruktor třídy GreenPiece.
  Args: position ([int,int]): Pozice figurky
- Vrací textovou reprezentaci instance třídy GreenPiece.
  Returns: str: Textová reprezentace instance třídy GreenPiece

- *Metoda:*

  $_{i}nit$

  Args: position ([int,int]): Pozice figurky

- *Metoda:*

  $_{s}tr$

  Returns: str: Textová reprezentace instance třídy GreenPiece

Soubor: Piece.py

Třída: Piece

- Třída Piece slouží k reprezentaci jedné herní figurky.
- Konstruktor třídy Piece.
  Args: color (Colors): Barva figurky position ([int,int]): Pozice figurky
- Metoda vrátí figurku domů.
- Metoda vrátí seznam možných tahů figurky.
  Args: number (int): Počet oček hoděných na kostce
  Returns: List of [int,int]: Seznam možných tahů figurky
- Metoda provede standardní tah figurkou.
  Args: board (Board): Hrací deska position ([int,int]): Pozice figurky

- *Metoda:*

  $_{i}nit$

  Args: color (Colors): Barva figurky position ([int,int]): Pozice figurky

- *Metoda: returnHome Metoda vrátí figurku domů.*

- *Metoda: possibleMoves*

  Args: number (int): Počet oček hoděných na kostce

  Returns: List of [int,int]: Seznam možných tahů figurky*Metoda vrátí seznam možných tahů figurky.*

  Args: number (int): Počet oček hoděných na kostce

  Returns: List of [int,int]: Seznam možných tahů figurky

- *Metoda: makeStandartMove*

  Args: board (Board): Hrací deska position ([int,int]): Pozice figurkyMetoda provede standardní tah figurkou.

  Args: board (Board): Hrací deska position ([int,int]): Pozice figurky

Soubor: RedPiece.py

Třída: RedPiece

- Třída RedPiece slouží k reprezentaci jedné červené figurky.
- Konstruktor třídy RedPiece.
  Args: position ([int,int]): Pozice figurky
- Vrací textovou reprezentaci instance třídy RedPiece.
  Returns: str: Textová reprezentace instance třídy RedPiece

- *Metoda:*

  $_{i}nit$

  Args: position ([int,int]): Pozice figurky

- *Metoda:*

  $_{s}tr$

  Returns: str: Textová reprezentace instance třídy RedPiece

Soubor: WhitePiece.py

Třída: WhitePiece

- Třída WhitePiece slouží k reprezentaci jedné bílé figurky.
- Konstruktor třídy WhitePiece.
  Args: position ([int,int]): Pozice figurky
- Vrací textovou reprezentaci instance třídy WhitePiece.
  Returns: str: Textová reprezentace instance třídy WhitePiece

- *Metoda:*

  $_{i}nit$

  Args: position ([int,int]): Pozice figurky

- *Metoda:*

  $_{s}tr$

  Returns: str: Textová reprezentace instance třídy WhitePiece

Soubor: $_{i}nit_{.py}$

Složka: __pycache__

Soubor: $_{i}nit_{.py}$

Složka: __pycache__

Soubor: ListOfGames.py

Třída: Game

- Třída reprezentující hru v seznamu her pro frontend
- Třída reprezentující seznam her pro frontend
- Vrací seznam her pro frontend
  Returns: List[Game]: seznam her

Třída: ListOfGames

- Třída reprezentující hru v seznamu her pro frontend
- Třída reprezentující seznam her pro frontend
- Vrací seznam her pro frontend
  Returns: List[Game]: seznam her

- *Metoda: getListOfGames*
  Returns: List[Game]: seznam her*Vrací seznam her pro frontend*
  Returns: List[Game]: seznam her

Složka: mathGame

Soubor: MathGame.py

Třída: MathGame

- Třída reprezentující hru MathGame
- Konstruktor třídy matematické hry. Vytvoří novou hru a nastaví počáteční hodnoty.
- Vrátí šachovnici ve formě dvourozměrného pole objektů Field
  Args: color (Enum Colors, optional): Barva hráče. Defaults to None.
  Returns: list: dvourozměrné pole objektů Field

  - Vrátí možné tahy pro danou pozici

    Args: position ([int,int]): pozice figury color (Enum Colors, optional): Barva figury. Defaults to None.

    Returns: List of List of int: List možných tahů figury, prázdný list pokud tah není možný
  - Přesune figuru na jinou pozici

    Args: move ([int,int]): nová pozice figury color (Enum Colors, optional): Barva figury. Defaults to None. rightClick (bool, optional): True, pokud hráč klikl pravým tlačítkem myši, jina False. Defaults to False.

    Returns: bool: True, pokud se tah podařil, jinak False List of List of int: List možných tahů figury, pokud se dá pokračovat v pohybu
  - Zkontroluje, zda hra skončila

    Returns: String: Vrací výsledek hry, pokud hra skončila, jinak None
  - Vytiskne aktuální stav hry na terminál

- *Metoda:*

  initKonstruktor třídy matematické hry. Vytvoří novou hru a nastaví počáteční hodnoty. Metoda: getBoa

  Args: color (Enum Colors, optional): Barva hráče. Defaults to None.

  Returns: list: dvourozměrné pole objektů Field

- *Metoda: choosePiece*

  Args: position ([int,int]): pozice figury color (Enum Colors, optional): Barva figury. Defaults to None.

  Returns: List of List of int: List možných tahů figury, prázdný list pokud tah není možný*Vrátí možné tahy pro danou pozici*

  Args: position ([int,int]): pozice figury color (Enum Colors, optional): Barva figury. Defaults to None.

  Returns: List of List of int: List možných tahů figury, prázdný list pokud tah není možný

- *Metoda: makeMove*

  Args: move ([int,int]): nová pozice figury color (Enum Colors, optional): Barva figury. Defaults to None. rightClick (bool, optional): True, pokud hráč klikl pravým tlačítkem myši, jina False. Defaults to False.

  Returns: bool: True, pokud se tah podařil, jinak False List of List of int: List možných tahů figury, pokud se dá pokračovat v pohybuPřesune figuru na jinou pozici

  Args: move ([int,int]): nová pozice figury color (Enum Colors, optional): Barva figury. Defaults to None. rightClick (bool, optional): True, pokud hráč klikl pravým tlačítkem myši, jina False. Defaults to False.

  Returns: bool: True, pokud se tah podařil, jinak False List of List of int: List možných tahů figury, pokud se dá pokračovat v pohybu

- *Metoda: checkEnd*

  Returns: String: Vrací výsledek hry, pokud hra skončila, jinak NoneZkontroluje, zda hra skončila

  Returns: String: Vrací výsledek hry, pokud hra skončila, jinak None

- *Metoda:*
  $_{p}rintToTerminal$ *Vytiskne aktuální stav hry na terminál*

Soubor: MathGameBoard.py

Třída: MathGameBoard

- Třída reprezentující šachovnici pro matematickou hru. Šachovnice je 8x8 a obsahuje figury typu Colors a "TASK". Figury typu Colors jsou na šachovnici reprezentovány barvou, figury typu "TASK"jsou úkoly, které je potřeba splnit.
- Konstruktor třídy herní desky pro matematickou hru. Vytvoří novou šachovnici a nastaví počáteční hodnoty.
- Vrací figuru na určeném místě na šachovnici, nebo None, pokud je políčko prázdné.

  Args: index [int, int]: Tuple dvou integerů, (row, col), oba 0-7
- Nastaví políčko na šachovnici jako board[row,col] namísto board.board[row][col]

  Args: index ([int, int]): Tuple dvou integerů, (row, col), oba 0-7 value (Piece): Instance třídy Piece, nebo None, pokud má být políčko prázdné

  Returns: bool: True, pokud se podařilo nastavit políčko, jinak False
- Vrací string reprezentaci šachovnice. Každé políčko je reprezentováno jako string, který je tvořen z informací o barvě a symbolu figury, nebo jako string "

  $_{"}, pokud je políčko prázdné. Políčka jsou oddělena mezerou a jednotlivé řádky jsou odděleny znakem nového řádku().$

  Returns: String: String reprezentace šachovnice
- Nastaví šachovnici do počátečního stavu. Bílý je vpravo dole a černý vlevo nahoře.
- Vrací počet zbývajících úkolů na šachovnici.

  Returns: int: Počet zbývajících úkolů
- Vrací pozici figury dané barvy.

  Args: color (Enum Colors): Barva figurek, které chceme najít (Colors.WHITE nebo Colors.BLACK)

  Returns: [int, int]: Pozice figury na šachovnici
- Vrací možné tahy figury na dané pozici.

  Args: position ([int, int]): Pozice figury na šachovnici

  Returns: List of [int, int]: List možných tahů figury

  - Přesune figuru na jiné místo na šachovnici.
    Args: move ([int, int]): Nová pozice figury color (Enum Colors): Barva figury, kterou chceme přesunout
  - Vrací list všech figurek na šachovnici.
    Returns: List of Fields: List figurek na šachovnici

- *Metoda:*

  $_{i}{}^{nit}$*Konstruktor třídy herní desky pro matematickou hru. Vytvoří novou šachovnici a nastaví počáteční h*

- *Metoda:*

  $_{s}{}^{etitem}$

  Args: index ([int, int]): Tuple dvou integerů, (row, col), oba 0-7 value (Piece): Instance třídy Piece, nebo None, pokud má být políčko prázdné

  Returns: bool: True, pokud se podařilo nastavit políčko, jinak False

- *Metoda:*

  $_{s}{}^{tr}$

  Returns: String: String reprezentace šachovnice

- *Metoda:*

  $_{p}{}^{opulateBoard}$*Nastaví šachovnici do počátečního stavu. Bílý je vpravo dole a černý vlevo nahoře.Metoda:*

  Returns: int: Počet zbývajících úkolů

- *Metoda: getPosition*

  Args: color (Enum Colors): Barva figurek, které chceme najít (Colors.WHITE nebo Colors.BLACK)

  Returns: [int, int]: Pozice figury na šachovnici*Vrací pozici figury dané barvy.*

  Args: color (Enum Colors): Barva figurek, které chceme najít (Colors.WHITE nebo Colors.BLACK)

  Returns: [int, int]: Pozice figury na šachovnici

- *Metoda: getPosibleMoves*

  Args: position ([int, int]): Pozice figury na šachovnici

  Returns: List of [int, int]: List možných tahů figuryVrací možné tahy figury na dané pozici.

  Args: position ([int, int]): Pozice figury na šachovnici

  Returns: List of [int, int]: List možných tahů figury

- *Metoda: movePiece*

  Args: move ([int, int]): Nová pozice figury color (Enum Colors): Barva figury, kterou chceme přesunoutPřesune figuru na jiné místo na šachovnici.

  Args: move ([int, int]): Nová pozice figury color (Enum Colors): Barva figury, kterou chceme přesunout

- *Metoda: getListOfBoard*

  Returns: List of Fields: List figurek na šachovniciVrací list všech figurek na šachovnici.

  Returns: List of Fields: List figurek na šachovnici

Soubor: $_{i}nit_{.py}$

Složka: __pycache__

Složka: mines

Soubor: Mines.py

Třída: Mines

- Třída reprezentující hru Miny
- Inicializace hry Miny
- Provede tah hráče

  Args: position ([int, int]): pozice, kam se má hráč pohnout color (Enum Colors): barva na tahu rightClick (bool): True, pokud hráč klikl pravým tlačítkem myši, jinak False

  Returns: bool: úspěšnost tahu
- Provede tah

  Args: position ([int, int]): pozice, kam se má hráč pohnout color (Enum Colors): barva na tahu

  Returns: bool: úspěšnost tahu
- Zkontroluje, zda hra skončila

  Returns: None: pokud hra neskončila string: výsledek hry
- Umístí vlajku na danou pozici

  Args: position ([int, int]): pozice, kam se má vlajka umístit color (Enum Colors): barva na tahu

  Returns: bool: úspěšnost umístění vlajky
- Vrátí herní desku

  Args: color (Enum Colors, optional): Barva, která je na tahu. Výchozí je None.

  Returns: list: herní deska
- Vypíše herní desku do konzole

- *Metoda:*

  $_{i}nit$ *Inicializace hry Miny Metoda: makeMove*

Args: position ([int, int]): pozice, kam se má hráč pohnout color (Enum Colors): barva na tahu rightClick (bool): True, pokud hráč klikl pravým tlačítkem myši, jinak False

Returns: bool: úspěšnost tahu

- *Metoda: makeUncoverMove*

  Args: position ([int, int]): pozice, kam se má hráč pohnout color (Enum Colors): barva na tahu

  Returns: bool: úspěšnost tahu*Provede tah*

  Args: position ([int, int]): pozice, kam se má hráč pohnout color (Enum Colors): barva na tahu

  Returns: bool: úspěšnost tahu

- *Metoda: checkEnd*

  Returns: None: pokud hra neskončila string: výsledek hryZkontroluje, zda hra skončila

  Returns: None: pokud hra neskončila string: výsledek hry

- *Metoda: placeFlag*

  Args: position ([int, int]): pozice, kam se má vlajka umístit color (Enum Colors): barva na tahu

  Returns: bool: úspěšnost umístění vlajkyUmístí vlajku na danou pozici

  Args: position ([int, int]): pozice, kam se má vlajka umístit color (Enum Colors): barva na tahu

  Returns: bool: úspěšnost umístění vlajky

- *Metoda: getBoard*

  Args: color (Enum Colors, optional): Barva, která je na tahu. Výchozí je None.

  Returns: list: herní deskaVrátí herní desku

  Args: color (Enum Colors, optional): Barva, která je na tahu. Výchozí je None.

  Returns: list: herní deska

- *Metoda:*
  $_{p}rintToTerminal$ *Vypíše herní desku do konzole*

Soubor: MinesBoard.py

Třída: MinesBoard

– Třída reprezentující hrací desku hry Miny

  - Inicializace hrací desky Miny
    Args: numberOfMines (int): počet min na hrací desce
  - Naplní hrací desku minami
  - Umístí miny na hrací desku
  - Spočítá miny kolem symbolu
    Args: row (int): řádek col (int): sloupec
    Returns: int: počet min kolem symbolu
  - Spočítá zbývající neoznačené miny na hrací desce
    Returns: int: počet zbývajících neoznačených min
  - Spočítá počet vlajek na hrací desce
    Returns: int: počet vlajek
  - Provede tah na hrací desce
    Args: row (int): řádek col (int): sloupec
    Returns: bool: True, pokud tah byl úspěšný, jinak False
  - Odkryje dostupné symboly na hrací desce po odkrytí políčka
    Args: row (int): řádek col (int): sloupec
  - Umístí vlajku na hrací desku
    Args: row (int): řádek col (int): sloupec
    Returns: bool: False, pokud se vlajka neumístí int: -1, pokud se správná vlajka odstraní, 1, pokud se vlajka umístí správně a 0, pokud se vlajka odstraní nebo přidá na špatné místo
  - Vygeneruje textovou reprezentaci hrací desky
    Returns: string: textová reprezentace hrací desky

- *Metoda:*
  $_{i}nit$
  Args: numberOfMines (int): počet min na hrací desce

- *Metoda:*
  $_{p}opulateBoard$*Naplní hrací desku minami Metoda:*$_{placeMines}$*Umístí miny na hrací desku Metoda: countMi*
  Args: row (int): řádek col (int): sloupec
  Returns: int: počet min kolem symbolu

- *Metoda: minesRemaining*
  Returns: int: počet zbývajících neoznačených min*Spočítá zbývající neoznačené miny na hrací desce*
  Returns: int: počet zbývajících neoznačených min

- *Metoda: flagsPlanted*
  Returns: int: počet vlajekSpočítá počet vlajek na hrací desce
  Returns: int: počet vlajek

- *Metoda: makeMove*

  Args: row (int): řádek col (int): sloupec

  Returns: bool: True, pokud tah byl úspěšný, jinak FalseProvede tah na hrací desce

  Args: row (int): řádek col (int): sloupec

  Returns: bool: True, pokud tah byl úspěšný, jinak False

- *Metoda:*

  $_{s}howBoard$

  Args: row (int): řádek col (int): sloupec

- *Metoda: placeFlag*

  Args: row (int): řádek col (int): sloupec

  Returns: bool: False, pokud se vlajka neumístí int: -1, pokud se správná vlajka odstraní, 1, pokud se vlajka umístí správně a 0, pokud se vlajka odstraní nebo přidá na špatné místoUmístí vlajku na hrací desku

  Args: row (int): řádek col (int): sloupec

  Returns: bool: False, pokud se vlajka neumístí int: -1, pokud se správná vlajka odstraní, 1, pokud se vlajka umístí správně a 0, pokud se vlajka odstraní nebo přidá na špatné místo

- *Metoda:*

  $_{s}tr$

  Returns: string: textová reprezentace hrací desky

Soubor: $_{i}nit_{.py}$

Složka: __pycache__

Složka: pexeso

Soubor: Pexeso.py

Třída: Pexeso

- Třída reprezentující hru Pexeso
- Inicializace hry Pexeso
- Vrátí hrací plochu
  Args: color (Enum Colors): barva hráče
  Returns: list: hrací plocha

  - Provede tah

    Args: position ([int, int]): pozice, kterou chce hráč otočit color (Enum Colors): barva na tahu rightClick (bool): True, pokud hráč klikl pravým tlačítkem myši, jinak False

    Returns: bool: úspěšnost tahu
  - Zkontroluje, zda hra skončila

    Returns: None: pokud hra neskončila string: výsledek hry
  - Funkce pro vypsání hrací plochy do terminálu

- *Metoda:*

  $_{i}{}^{nit}$*Inicializace hry Pexeso Metoda: getBoard*

  Args: color (Enum Colors): barva hráče

  Returns: list: hrací plocha

- *Metoda: makeMove*

  Args: position ([int, int]): pozice, kterou chce hráč otočit color (Enum Colors): barva na tahu rightClick (bool): True, pokud hráč klikl pravým tlačítkem myši, jinak False

  Returns: bool: úspěšnost tahu*Provede tah*

  Args: position ([int, int]): pozice, kterou chce hráč otočit color (Enum Colors): barva na tahu rightClick (bool): True, pokud hráč klikl pravým tlačítkem myši, jinak False

  Returns: bool: úspěšnost tahu

- *Metoda: checkEnd*

  Returns: None: pokud hra neskončila string: výsledek hryZkontroluje, zda hra skončila

  Returns: None: pokud hra neskončila string: výsledek hry

- *Metoda:*
  $_{p}{}^{rintToTerminal}$*Funkce pro vypsání hrací plochy do terminálu*

Soubor: PexesoBoard.py

Třída: PexesoBoard

- Reprezentace hrací desky hry Pexeso
- Inicializace hrací desky
- Vygeneruje náhodné kartičky na hrací desku
- Zamíchá kartičky na hrací desce
- Schovej všechny kartičky

  - Vrátí textovou reprezentaci hrací desky
  - Provede tah
  - Vrátí, zda je kartička na dané pozici odhalena
    Args: position ([int, int]): pozice kartičky
    Returns: bool: zda je kartička odhalena
  - Vrátí, zda je kartička na dané pozici uhodnuta
    Args: position ([int, int]): pozice kartičky
    Returns: bool: zda je kartička uhodnuta
  - Vrátí hrací plochu

- *Metoda:*

  $_{i}{}^{nit}$*Inicializace hrací desky Metoda:*$_{populateBoard}$*Vygeneruje náhodné kartičky na hrací desku Metoda: sh*

- *Metoda: hideCards Schovej všechny kartičky*

- *Metoda:*

  $_{s}{}^{tr}$*Vrátí textovou reprezentaci hrací desky Metoda: makeMoveProvede tah*

- *Metoda: isRevealed*

  Args: position ([int, int]): pozice kartičky

  Returns: bool: zda je kartička odhalena*Vrátí, zda je kartička na dané pozici odhalena*

  Args: position ([int, int]): pozice kartičky

  Returns: bool: zda je kartička odhalena

- *Metoda: isCompleted*

  Args: position ([int, int]): pozice kartičky

  Returns: bool: zda je kartička uhodnutaVrátí, zda je kartička na dané pozici uhodnuta

  Args: position ([int, int]): pozice kartičky

  Returns: bool: zda je kartička uhodnuta

- *Metoda: getListOfBoard Vrátí hrací plochu*

Soubor: PexesoCard.py

Třída: PexesoCard

  - Reprezentace jedné kartičky hry Pexeso
  - Inicializace kartičky
    Args: symbol (Field): symbol kartičky odhalena (bool): zda je kartička odhalena

  - Vrátí symbol kartičky
    Returns: Field: symbol kartičky
  - Porovná symboly dvou kartiček
    Args: other (PexesoCard): druhá kartička
    Returns: bool: zda jsou symboly stejné
  - Otočí kartičku
  - Schovej kartičku
  - Odhal kartičku
  - Označ kartičku jako vyřešenou

- *Metoda:*
  $_{i}nit$
  Args: symbol (Field): symbol kartičky odhalena (bool): zda je kartička odhalena

- *Metoda: getSymbol*
  Returns: Field: symbol kartičky *Vrátí symbol kartičky*
  Returns: Field: symbol kartičky

- *Metoda: equals*
  Args: other (PexesoCard): druhá kartička
  Returns: bool: zda jsou symboly stejnéPorovná symboly dvou kartiček
  Args: other (PexesoCard): druhá kartička
  Returns: bool: zda jsou symboly stejné

- *Metoda: turn Otočí kartičku*
- *Metoda: hide Schovej kartičku*
- *Metoda: reveal Odhal kartičku*
- *Metoda: match Označ kartičku jako vyřešenou*

Soubor: $_{i}nit_{.py}$

Složka: __pycache__

Složka: ticTacToe

Soubor: TicTacToe.py

Třída: TicTacToe

  - Třída reprezentující hru Piškvorky

- Inicializace hry
- Vrátí hrací desku

  Args: color (Enum Colors, optional): barva hráče na tahu. Výchozí nastavení je na pravidelném střídání.

  Returns: List: hrací deska
- Metoda na zahrání tahu

  Args: index ([int,int]): pozice kterou chce hráč obsadit player (Enum Colors, optional): Hráč co má být na tahu. Výchozí je nastaveno na střídání. rightClick (bool, optional): True, pokud se jedná o pravé tlačítko myši. Defaults to False.

  Returns: Boolean: true pokud se tah podařil, jinak false
- Kontroluje jestli hra skončila

  Returns: String: "Draw"pokud je remíza String: "barva won"pokud někdo vyhrál None: pokud hra neskončila
- Resetuje hru
- Vytiskne hrací desku do konzole
- Kontroluje jestli někdo vyhrál

  Returns: Enum Colors: barva hráče, který vyhrál None: pokud nikdo nevyhrál
- Kontroluje jestli je remíza

  Returns: Boolean: true pokud je remíza, jinak false

- *Metoda:*

  $_{i}{}^{nit}$*Inicializace hry Metoda: getBoard*

  Args: color (Enum Colors, optional): barva hráče na tahu. Výchozí nastavení je na pravidelném střídání.

  Returns: List: hrací deska

- *Metoda: makeMove*

  Args: index ([int,int]): pozice kterou chce hráč obsadit player (Enum Colors, optional): Hráč co má být na tahu. Výchozí je nastaveno na střídání. rightClick (bool, optional): True, pokud se jedná o pravé tlačítko myši. Defaults to False.

  Returns: Boolean: true pokud se tah podařil, jinak false*Metoda na zahrání tahu*

  Args: index ([int,int]): pozice kterou chce hráč obsadit player (Enum Colors, optional): Hráč co má být na tahu. Výchozí je nastaveno na střídání. rightClick (bool, optional): True, pokud se jedná o pravé tlačítko myši. Defaults to False.

  Returns: Boolean: true pokud se tah podařil, jinak false

- *Metoda: checkEnd*

  Returns: String: "Draw"pokud je remíza String: "barva won"pokud někdo vyhrál None: pokud hra neskončilaKontroluje jestli hra skončila

  Returns: String: "Draw"pokud je remíza String: "barva won"pokud někdo vyhrál None: pokud hra neskončila

- *Metoda: reset Resetuje hru*

- *Metoda:*
  $_{p}rintToTerminal$ *Vytiskne hrací desku do konzole Metoda:*$_{checkWinner}$ *Returns : EnumColors : barvahráče*

- *Metoda:*
  $_{c}heckDraw$

  Returns: Boolean: true pokud je remíza, jinak false

Soubor: TicTacToeBoard.py

Třída: TicTacToeBoard

- Třída reprezentující hrací desku hry Piškvorky
- Konstruktor třídy TicTacToeBoard
- Inicializace herní desky
- Vytiskne hrací desku do konzole
- Vrací seznam políček na hrací desce

Returns: List of Struct Field:
$_{d}escription$

- *Metoda:*
  $_{i}nit$ *Konstruktor třídy TicTacToeBoard Metoda:*$_{populateBoard}$ *Inicializace herní desky Metoda:*$_{str}$ *Vytiskne*

Returns: List of Struct Field:
$_{d}escription$

Soubor: $_{i}nit_{.py}$

Složka: __pycache__

Soubor: $_{i}nit_{.py}$

Složka: __pycache__

Soubor: GameView.py

Třída: ClickableLabel

- Třída ClickableLabel slouží k vytvoření klikatelného labelu, který může vyslat signál o kliknutí na dané políčko
- Konstruktor třídy ClickableLabel
- Metoda, která se zavolá při kliknutí na dané políčko
- Třída GameView slouží k zobrazení hry na grafickém rozhraní pomocí PyQt5
- Konstruktor

  Args: game (game): Hra, kterou chceme zobrazit
- Funkce pro nastavení obrázku figurky na dané pozici

  Args: row (int): řádek col (int): sloupec filePath (string): cesta k obrázku
- Funkce pro vytvoření herní desky
- Funkce pro zobrazení otázky
- Funkce pro zpracování odpovědi na otázku

  Args: correct (bool): Byla odpověď správná?
- Funkce pro obsluhu kliknutí na políčko

  Args: row (int): řádek col (int): sloupec button (string): tlačítko, které bylo stisknuto
- Funkce pro výběr figurky

  Args: row (int): řádek col (int): sloupec
- Funkce pro provedení tahu

  Args: row (int): řádek col (int): sloupec
- Funkce pro zvýraznění políčka

  Args: row (int): řádek col (int): sloupec
- Funkce pro aktualizaci herní desky

  Args: isFirst (bool, optional): Je to první aktualizace?. Defaults to False.
- Funkce pro odstranění herní desky

  - Funkce pro zobrazení dialogového okna s výsledkem hry
    Args: message (string): Výsledek hry
  - Funkce pro výběr figurky, na kterou se má pesák změnit

- *Atribut: clicked*
- *Metoda:*

  $_{i}{}^{nit}$*Konstruktor třídy ClickableLabel Metoda: mousePressEventMetoda, která se zavolá při kliknutí na da*

Třída: GameView

- – Třída ClickableLabel slouží k vytvoření klikatelného labelu, který může vyslat signál o kliknutí na dané políčko
  - Konstruktor třídy ClickableLabel
  - Metoda, která se zavolá při kliknutí na dané políčko
  - Třída GameView slouží k zobrazení hry na grafickém rozhraní pomocí PyQt5
  - Konstruktor
    Args: game (game): Hra, kterou chceme zobrazit
  - Funkce pro nastavení obrázku figurky na dané pozici
    Args: row (int): řádek col (int): sloupec filePath (string): cesta k obrázku
  - Funkce pro vytvoření herní desky
  - Funkce pro zobrazení otázky
  - Funkce pro zpracování odpovědi na otázku
    Args: correct (bool): Byla odpověď správná?
  - Funkce pro obsluhu kliknutí na políčko
    Args: row (int): řádek col (int): sloupec button (string): tlačítko, které bylo stisknuto
  - Funkce pro výběr figurky
    Args: row (int): řádek col (int): sloupec
  - Funkce pro provedení tahu
    Args: row (int): řádek col (int): sloupec
  - Funkce pro zvýraznění políčka
    Args: row (int): řádek col (int): sloupec
  - Funkce pro aktualizaci herní desky
    Args: isFirst (bool, optional): Je to první aktualizace?. Defaults to False.
  - Funkce pro odstranění herní desky

  - Funkce pro zobrazení dialogového okna s výsledkem hry
    Args: message (string): Výsledek hry
  - Funkce pro výběr figurky, na kterou se má pesák změnit

- *Metoda:*
  $_{i}nit$
  Args: game (game): Hra, kterou chceme zobrazit

- *Metoda: set*
  $_{p}iece$
  $_{i}mage$
  Args: row (int): řádek col (int): sloupec filePath (string): cesta k obrázku

- *Metoda: create*
  $_{b}oardFunkce\ pro\ vytvoření\ herní\ desky\ Metoda:\ show_{q}uestionFunkce\ pro\ zobrazení\ otázky\ Metoda:\ hand$

- *Metoda: handle*
  $_{s}quare$
  $_{c}lick$
  Args: row (int): řádek col (int): sloupec button (string): tlačítko, které bylo stisknuto

- *Metoda: choose*
  $_{p}iece$
  Args: row (int): řádek col (int): sloupec

- *Metoda: make*
  $_{m}ove$
  Args: row (int): řádek col (int): sloupec

- *Metoda: highlight*
  $_{s}quare$
  Args: row (int): řádek col (int): sloupec

- *Metoda: update*
  $_{b}oard$
  Args: isFirst (bool, optional): Je to první aktualizace?. Defaults to False.

- *Metoda: remove*
  $_{b}oardFunkce\ pro\ odstranění\ herní\ desky\ Metoda:\ game_{e}ndedArgs : message(string) : Výsledekhry$

- *Metoda: promote*
  $_{p}awnFunkce\ pro\ výběr\ figurky,\ na\ kterou\ se\ má\ pesák\ změnit$

Soubor: GetResource.py

Třída: GetResource

- Třída GetResource slouží k získání cesty k obrázku, který reprezentuje daný zdroj.
- Metoda na základě zadaného zdroje vrátí cestu k obrázku, který reprezentuje daný zdroj.
  Args: resource (str): Zdroj, pro který chceme získat cestu k obrázku.
  Returns: str: cesta k obrázku, který reprezentuje zadaný zdroj.

- *Metoda: getResource*

  Args: resource (str): Zdroj, pro který chceme získat cestu k obrázku.

  Returns: str: cesta k obrázku, který reprezentuje zadaný zdroj.*Metoda na základě zadaného zdroje vrátí cestu k obrázku, který reprezentuje daný zdroj.*

  Args: resource (str): Zdroj, pro který chceme získat cestu k obrázku.

  Returns: str: cesta k obrázku, který reprezentuje zadaný zdroj.

Soubor: main.py

Soubor: MainView.py

Třída: MainView

- Třída MainView slouží k zobrazení hlavního menu aplikace.
- Konstruktor třídy
- Spustí hru dle jména hry
  Args: game (Game): objekt Game, který obsahuje název hry a objekt hry

- *Metoda:*

$_{i}{}^{nit}$*Konstruktor třídy Metoda:* $start_game_{Args:game(Game):objektGame,kterýobsahujenázevhryaobjekthry}$

Složka: questions

Soubor: GenerateQuestion.py

Třída: GenerateQuestion

- Třída na generování otázek Pro generování otázek použij metodu generateQuestion Pro kontrolu odpovědí použij metodu checkAnswer s parametrem answer Pro získání správné odpovědi použij funkci doupovcuvOperator Výsledky se zaokrouhlují na celá čísla!!!

  - Konstruktor třídy GenerateQuestion
  - Metoda na generování otázek
    Args: n (int): Číslo otázky, defaultně náhodné
    Returns: string, string (questionText, questionLatex): otázka

- *Metoda:*

  $_{init}$ Konstruktor třídy GenerateQuestion Metoda: generateQuestion

  Args: n (int): Číslo otázky, defaultně náhodné

  Returns: string, string (questionText, questionLatex): otázka

Složka: generators

Soubor: AnalyticGeometryQuestionGenerator.py

Třída: AnalyticGeometryQuestionGenerator

- Generátor otázek na analytickou geometrii
  - Konstruktor třídy AnalyticGeometryQuestionGenerator
  - Generování náhodné otázky na analytickou geometrii
    Args: n (int): Číslo otázky, defaultně náhodné
    Returns: AnalyticGeometryQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí

- *Metoda:*

  $_{init}$ Konstruktor třídy AnalyticGeometryQuestionGenerator Metoda: generateQuestion

  Args: n (int): Číslo otázky, defaultně náhodné

  Returns: AnalyticGeometryQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí

Soubor: DerivativeQuestionGenerator.py

Třída: DerivativeQuestionGenerator

- Generátor otázek na derivace
  - Konstruktor třídy DerivativeQuestionGenerator
  - Generování náhodného polynomu
    Args: degree (int): stupeň polynomu
    Returns: sympy symbol: polynom

– Generování náhodné otázky na derivace
Args: n (int): Číslo otázky, defaultně náhodné
Returns: DerivativeQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí

- *Metoda:*

  ${}_{i}{}^{nit}$*Konstruktor třídy DerivativeQuestionGenerator Metoda: generatePolynomial*

  Args: degree (int): stupeň polynomu

  Returns: sympy symbol: polynom

- *Metoda: generateQuestion*

  Args: n (int): Číslo otázky, defaultně náhodné

  Returns: DerivativeQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí*Generování náhodné otázky na derivace*

  Args: n (int): Číslo otázky, defaultně náhodné

  Returns: DerivativeQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí

Soubor: FractionQuestionGenerator.py

Třída: FractionQuestionGenerator

– Generátor otázek na zlomky
– Konstruktor třídy FractionQuestionGenerator
– Vygenerování náhodné dvojice čísel
Returns: int, int: (numerator, denominator), kde numerator je čitatel v rozmezí 10-100 a denominator jmenovatel v rozmezí 1-10
– Zjednodušení zlomku
Args: numerator (int): čitatel denominator (int): jmenovatel Returns: int, int: (numerator, denominator), kde numerator je čitatel a denominator jmenovatel zjednodušeného zlomku
– Výpočet nejmenšího společného násobku
Args: a (int): první číslo b (int): druhé číslo
Returns: int: Nejmenší společný násobek
– Výpočet největšího společného dělitele
Args: a (int): první číslo b (int): druhé číslo Returns: int: Největší společný dělitel
– Převedení zlomku na string
Args: numerator (int): čitatel denominator (int): jmenovatel
Returns: String: reprezentace zlomku v latexu

  - Převedení zlomku na string
    Args: numerator (int): čitatel denominator (int): jmenovatel
    Returns: String: reprezentace zlomku ve formátu "a/b
  - Vygenerování náhodné otázky na zlomky
    Args: n (int): Číslo otázky, defaultně náhodné
    Returns: FractionQuestionGenerator: Samo sebe s vygenerovanou otázkou a odpovědí

- *Metoda:*

  $_{i}{}^{nit}$Konstruktor třídy FractionQuestionGenerator Metoda: generateFraction

  Returns: int, int: (numerator, denominator), kde numerator je čitatel v rozmezí 10-100 a denominator jmenovatel v rozmezí 1-10

- *Metoda: simplifyFraction*

  Args: numerator (int): čitatel denominator (int): jmenovatel Returns: int, int: (numerator, denominator), kde numerator je čitatel a denominator jmenovatel zjednodušeného zlomku*Zjednodušení zlomku*

  Args: numerator (int): čitatel denominator (int): jmenovatel Returns: int, int: (numerator, denominator), kde numerator je čitatel a denominator jmenovatel zjednodušeného zlomku

- *Metoda: lowestCommonMultiple*

  Args: a (int): první číslo b (int): druhé číslo

  Returns: int: Nejmenší společný násobekVýpočet nejmenšího společného násobku

  Args: a (int): první číslo b (int): druhé číslo

  Returns: int: Nejmenší společný násobek

- *Metoda: greatestCommonDivisor*

  Args: a (int): první číslo b (int): druhé číslo Returns: int: Největší společný dělitelVýpočet největšího společného dělitele

  Args: a (int): první číslo b (int): druhé číslo Returns: int: Největší společný dělitel

- *Metoda: fractionToString*

  Args: numerator (int): čitatel denominator (int): jmenovatel

  Returns: String: reprezentace zlomku v latexuPřevedení zlomku na string

  Args: numerator (int): čitatel denominator (int): jmenovatel

  Returns: String: reprezentace zlomku v latexu

- *Metoda: fractionToAnswer*

  Args: numerator (int): čitatel denominator (int): jmenovatel

  Returns: String: reprezentace zlomku ve formátu "a/b

  Převedení zlomku na string

  Args: numerator (int): čitatel denominator (int): jmenovatel

  Returns: String: reprezentace zlomku ve formátu "a/b"

- *Metoda: generateQuestion*

  Args: n (int): Číslo otázky, defaultně náhodné

  Returns: FractionQuestionGenerator: Samo sebe s vygenerovanou otázkou a odpovědíVygenerování náhodné otázky na zlomky

  Args: n (int): Číslo otázky, defaultně náhodné

  Returns: FractionQuestionGenerator: Samo sebe s vygenerovanou otázkou a odpovědí

Soubor: InfinitiveSeriesQuestionGenerator.py

Třída: InfinitiveSeriesQuestionGenerator

- Generátor otázek na konvergenci nekonečných řad
- Konstruktor třídy nekonečných řad
- Generování náhodné otázky na konvergenci nekonečných řad
  Args: n (int): Číslo otázky, defaultně náhodné
  Returns: InfinitiveSeriesQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí

- *Metoda:*

  $_{i}{}^{nit}$Konstruktor třídy nekonečných řad Metoda: generateQuestion

  Args: n (int): Číslo otázky, defaultně náhodné

  Returns: InfinitiveSeriesQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí

Soubor: IntegralQuestionGenerator.py

Třída: IntegralQuestionGenerator

- Generátor otázek na určení hodnoty integrálu
- Konstruktor třídy IntegralQuestionGenerator

  - Generování náhodné otázky na určení hodnoty integrálu
    Args: n (int): Číslo otázky, defaultně náhodné
    Returns: IntegralQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí

- *Metoda:*

  ${}_{i}{}^{nit}$*Konstruktor třídy IntegralQuestionGenerator Metoda: generateQuestion*

  Args: n (int): Číslo otázky, defaultně náhodné

  Returns: IntegralQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí

Soubor: KardinalNumberQuestionGenerator.py

Třída: KardinalNumberQuestionGenerator

- Generátor otázek na kardinální čísla
  - Konstruktor třídy otázek na kardinální čísla
  - Generování náhodné otázky na kardinální čísla
    Args: n (int): Číslo otázky, defaultně náhodné
    Returns: KardinalNumberQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí

- *Metoda:*

  ${}_{i}{}^{nit}$*Konstruktor třídy otázek na kardinální čísla Metoda: generateQuestion*

  Args: n (int): Číslo otázky, defaultně náhodné

  Returns: KardinalNumberQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí

Soubor: LinearEquationSystemQuestionGenerator.py

Třída: LinearEquationSystemQuestionGenerator

- Generátor otázek na lineární soustavy rovnic
  - Konstruktor třídy soustav lineárních rovnic
  - Vygenerování lineární soustavy rovnic
    Args: num
    ${}_{e}quations(int)$: *početchtěnýchrovnicnum*
    ${}_{v}ariables(int)$: *početchtěnýchproměnných*
    Returns: sympy equations, sympy symbols: Vygenerované rovnice a jejich proměnné

- Metoda na konverzi rovnic do latexu
  Args: equations (sympy equations): vstupní rovnice
  Returns: string: latexový zápis rovnic ve formátu string
- Metoda na generování otázky
  Args: n (int): Číslo otázky, defaultně náhodné
  Returns: LinearEquationSystemQuestionGenerator: Funkce vrací sebe sama s vygenerovanou otázkou

- *Metoda:*

  $_{i}{}^{nit}$*Konstruktor třídy soustav lineárních rovnic Metoda: generateLinearEquationSystem*
  Args: num
  $_{e}$*quations*(*int*) : *početchtěnýchrovnicnum*
  $_{v}$*ariables*(*int*) : *početchtěnýchproměnných*

Returns: sympy equations, sympy symbols: Vygenerované rovnice a jejich proměnné

- *Metoda: convert*
  $_{t}{}^{o}$
  $_{l}$*atex*

  Args: equations (sympy equations): vstupní rovnice

  Returns: string: latexový zápis rovnic ve formátu string

- *Metoda: generateQuestion*

  Args: n (int): Číslo otázky, defaultně náhodné

  Returns: LinearEquationSystemQuestionGenerator: Funkce vrací sebe sama s vygenerovanou otázkou*Metoda na generování otázky*

  Args: n (int): Číslo otázky, defaultně náhodné

  Returns: LinearEquationSystemQuestionGenerator: Funkce vrací sebe sama s vygenerovanou otázkou

Soubor: MatrixQuestionGenerator.py

Třída: MatrixQuestionGenerator

- Generátor otázek na matice
- Konstruktor třídy matice
- Generování regulární matice
  Args: n (int): Velikost matice, defaultně náhodně mezi 2 a 4
  Returns: numpy array int: Náhodná regulární matice
- Výpočet determinantu matice
  Args: matrix (numpy array int): matice, ve tvaru numpy array intů
  Returns: float: Hodnota determinantu matice

  - Výpočet inverzní matice
    Args: matrix (numpy array int): matice
    Returns: numpy array int: inverzní matice
  - Výpočet řádu matice
    Args: matrix (numpy array int): matice
    Returns: int: řád matice
  - Výpočet vlastních čísel matice
    Args: matrix (numpy array int): matice
    Returns: float: součet vlastní čísla matice
  - Metoda pro vygenerování matice ve formátu LaTeX
    Args: matrix (numpy.ndarray): matice
    Returns: string: matice ve formátu LaTeX
  - Metoda na generování otázek na matice
    Returns: MatrixQuestionGenerator: Vrací sebe sama s vygenerovanými otázkami

- *Metoda:*
  $_{i}{}^{nit}$*Konstruktor třídy matice Metoda: generateRegularMatrix*
  Args: n (int): Velikost matice, defaultně náhodně mezi 2 a 4
  Returns: numpy array int: Náhodná regulární matice

- *Metoda: calculateDeterminant*
  Args: matrix (numpy array int): matice, ve tvaru numpy array intů
  Returns: float: Hodnota determinantu matice*Výpočet determinantu matice*
  Args: matrix (numpy array int): matice, ve tvaru numpy array intů
  Returns: float: Hodnota determinantu matice

- *Metoda: calculateInverseMatrix*
  Args: matrix (numpy array int): matice
  *Returns: numpy array int: inverzní matice**Výpočet inverzní matice*
  Args: matrix (numpy array int): matice
  Returns: numpy array int: inverzní matice

- *Metoda: calculateRank*
  Args: matrix (numpy array int): matice
  *Returns: int: řád matice**Výpočet řádu matice*
  Args: matrix (numpy array int): matice
  Returns: int: řád matice

- *Metoda: calculateEigenvalues*

  Args: matrix (numpy array int): matice

  Returns: float: součet vlastní čísla maticeVýpočet vlastních čísel matice

  Args: matrix (numpy array int): matice

  Returns: float: součet vlastní čísla matice

- *Metoda: getLatexMatrix*

  Args: matrix (numpy.ndarray): matice

  Returns: string: matice ve formátu LaTeXMetoda pro vygenerování matice ve formátu LaTeX

  Args: matrix (numpy.ndarray): matice

  Returns: string: matice ve formátu LaTeX

- *Metoda: generateQuestion*

  Returns: MatrixQuestionGenerator: Vrací sebe sama s vygenerovanými otázkamiMetoda na generování otázek na matice

  Returns: MatrixQuestionGenerator: Vrací sebe sama s vygenerovanými otázkami

Soubor: OrdinalNumberQuestionGenerator.py

Třída: OrdinalNumberQuestionGenerator

- Generátor otázek na ordinální čísla
- Konstruktor třídy otázek na ordinální čísla
- Generování náhodné otázky na uspořádaná čísla
  Args: n (int): Číslo otázky, defaultně náhodné
  Returns: OrdinalNumberQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí

- *Metoda:*

  $_{init}$Konstruktor třídy otázek na ordinální čísla Metoda: generateQuestion

  Args: n (int): Číslo otázky, defaultně náhodné

  Returns: OrdinalNumberQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí

Soubor: RegularLanguageQuestionGenerator.py

Třída: RegularLanguageQuestionGenerator

- Generátor otázek na regulární jazyky

  - Konstruktor třídy regulárních jazyků
  - Generování náhodné otázky na regulární jazyky
    Args: n (int): Číslo otázky, defaultně náhodné
    Returns: RegularLanguageQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí

- *Metoda:*

  $_{i}nit$ Konstruktor třídy regulárních jazyků Metoda: generateQuestion

  Args: n (int): Číslo otázky, defaultně náhodné

  Returns: RegularLanguageQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí

Soubor: SetQuestionGenerator.py

Třída: SetQuestionGenerator

- Generátor otázek na množiny
  - Konstruktor třídy otázek na množiny
  - Generování náhodné otázky na množiny
    Args: n (int): Číslo otázky, defaultně náhodné
    Returns: SetQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí

- *Metoda:*

  $_{i}nit$ Konstruktor třídy otázek na množiny Metoda: generateQuestion

  Args: n (int): Číslo otázky, defaultně náhodné

  Returns: SetQuestionGenerator: Vrací samo sebe s vygenerovanou otázkou a odpovědí

Soubor: $_{i}nit_{.py}$

Složka: __pycache__

Soubor: Question.py

Třída: Question

- Třída předpisu otázky
  - Konstruktor třídy Question
  - Metoda na výpis otázky s odpovědí
    Returns: string: Vrátí otázku a odpověď

  - Metoda na kontrolu odpovědi
    Args: string: answer - zadaná odpověď
    Returns: bool: true pokud je odpověď správná
  - Metoda na získání odpovědi
    Returns: string: odpověď
  - Funkce na generování otázky
    Returns: string, string: Vrací otázku

- *Metoda:*
  $_{i}{}^{nit}$*Konstruktor třídy Question Metoda:*$_{s}{}^{tr\ Returns:string:Vrátíotázkuaodpověď}$'

- *Metoda: checkAnswer*
  Args: string: answer - zadaná odpověď
  Returns: bool: true pokud je odpověď správná*Metoda na kontrolu odpovědi*
  Args: string: answer - zadaná odpověď
  Returns: bool: true pokud je odpověď správná

- *Metoda: doupovcuvOperator*
  Returns: string: odpověď'Metoda na získání odpovědi
  Returns: string: odpověď

- *Metoda: generateQuestion*
  Returns: string, string: Vrací otázkuFunkce na generování otázky
  Returns: string, string: Vrací otázku

Soubor: QuestionTests.py

Třída: QuestionTests

  - Testy na generátory otázek
  - Testuje, zda se generuje nenulový počet otázek
    Args: name (string): jméno generátoru generator
    $_{c}lass(Question) : třídagenerátoruTestuje, zdasevygenerujeotázka$
  - Args: name (string): jméno generátoru generator
    $_{c}lass(Question) : třídagenerátoruTestuje, zdasevygenerujeodpověď$

Args: name (string): jméno generátoru generator
$_{c}lass(Question) : třídagenerátoru$

– Testuje, zda je odpověď správná
Args: name (string): jméno generátoru generator
$_{c}lass(Question) : třídagenerátoruTestuje, zdajeodpověďšpatná$
  - Args: name (string): jméno generátoru generator
    $_{c}lass(Question) : třídagenerátoru$

- *Atribut: allClasses*
- *Metoda: testNumberOfQuestions*

  Args: name (string): jméno generátoru generator
  $_{c}lass(Question)$: *třídagenerátoruTestuje*, *zdasegenerujenenulovýpočetotázek*

  Args: name (string): jméno generátoru generator
  $_{c}lass(Question)$: *třídagenerátoruMetoda: testGenerateQuestion*

Args: name (string): jméno generátoru generator
$_{c}lass(Question)$: *třídagenerátoru*

- *Metoda: testDoupovcuvOperator*

  Args: name (string): jméno generátoru generator
  $_{c}lass(Question)$: *třídagenerátoruTestuje*, *zdasevygenerujeodpověď*

  Args: name (string): jméno generátoru generator
  $_{c}lass(Question)$: *třídagenerátoruMetoda: testCheckAnswer*

  Args: name (string): jméno generátoru generator
  $_{c}lass(Question)$: *třídagenerátoru*

- *Metoda: testWrongAnswer*

  Args: name (string): jméno generátoru generator
  $_{c}lass(Question)$: *třídagenerátoruTestuje*, *zdajeodpověďšpatná*

  Args: name (string): jméno generátoru generator
  $_{c}lass(Question)$: *třídagenerátoruMetoda: testTimeSet*

- *Metoda: testHoderova*

Soubor: $_{i}nit_{.py}$

Složka: __pycache__

Soubor: QuestionView.py

Třída: MathQuestion

- Třída MathQuestion slouží k zobrazení matematické otázky.
- Konstruktor třídy

  Args: question (Question): Otázka, která se má zobrazit color (Colors): Barva hráče, který má odpovídat fullscreen (bool): Zda se má zobrazit na celou obrazovku callback (function): Funkce, která se má zavolat po zodpovězení otázky
- Metoda na aktualizaci časovače
- Metoda na oznámení, že čas vypršel
- Metoda na kontrolu odpovědi

- Metoda pro ukončení okna galantní cestou
- Metoda na vytvoření rovnic ve formátu LaTeX
  Args: latexEq (string): rovnice, které se mají vykreslit v LaTeXu

- *Metoda:*

  $_{i}nit$

  Args: question (Question): Otázka, která se má zobrazit color (Colors): Barva hráče, který má odpovídat fullscreen (bool): Zda se má zobrazit na celou obrazovku callback (function): Funkce, která se má zavolat po zodpovězení otázky

- *Metoda: update*
  $_{t}imerMetoda\ na\ aktualizaci\ časovače\ Metoda:\ time_{o}utMetoda\ na\ oznámení,\ že\ čas\ vypršel\ Metoda:\ check_{a}nsu$

Složka: resources

Soubor: KatexHtmlTemplate.py

Složka: __pycache__

Soubor: Tests.py

Složka: __pycache__