Balíček CreateLevel

Obsahuje šablonu a odvozené třídy.

Šablona obsahuje následující konstruktor.

- mode - herní režim, level - úroveň,
- savedTimer - poslední uložený časovač,
- solveBoard - vyřešené sudoku (používá se v Sudoku Killer pro porovnání požadované sumy v zóně), levelGameBoard - vyřešené sudoku s odstraněnými buňkami,
- lastSaveGameBoard - zde se ukládají uživatelem zadané hodnoty,
- operations - seznam operací pro použití tlačítek "Undo" a "Redo".

```
2 usages   kramkvol
public CreateLevelTemplate(String mode, String level) {
    this.mode = mode;
    this.level = level;
    this.savedTimer = 0;
    this.solveBord = generateSolveSudoku();
    this.levelGameBoard = generateLevelGameBoard();
    this.lastSaveGameBoard = getCopy(levelGameBoard);
    this.operations = new ArrayList<>();
}
```

- Odvozené třídy od šablony se liší logikou generování počáteční desky (levelGameBoard).
- V režimu Classic mohou být odstraněné buňky v různých částech pole, zatímco v režimu Killer jsou buňky odstraňovány v závislosti na zóně, ke které patří.

- Nejprve se vybírá jakákoli buňka ze zóny 2, a když jsou tyto zóny vyčerpány, buňky se odstraňují ze zóny 3 a tak dále.

Sudoku Classic

```java
@Override
public int[][] removeCells(int[][] board, int numToRemove) {
    int[][] newBoard = getCopy(board);
    SecureRandom random = new SecureRandom();
    for (int i = 0; i < numToRemove; i++) {
        int row = random.nextInt( bound: 9);
        int col = random.nextInt( bound: 9);
        if (newBoard[row][col] != 0) {
            newBoard[row][col] = 0;
        }
    }
    return newBoard;
}
```

Sudoku Killer

```java
@Override
public int[][] removeCells(int[][] board, int numToRemove) {
    int[][] newBoard = getCopy(board);
    SecureRandom random = new SecureRandom();
    List<Cell> cellsWithPattern2 = new ArrayList<>();
    List<Cell> cellsWithPattern3 = new ArrayList<>();
    List<Cell> cellsWithPattern4 = new ArrayList<>();
    List<Cell> cellsWithPattern5 = new ArrayList<>();

    for (Cell c : cells) {
        switch (c.getPattern()) {
            case 2 -> cellsWithPattern2.add(c);
            case 3 -> cellsWithPattern3.add(c);
            case 4 -> cellsWithPattern4.add(c);
            case 5 -> cellsWithPattern5.add(c);
            default -> {}
        }
    }

    for (int counter = 0; counter < numToRemove; counter++) {
        if (counter <= 12) {
            if (!cellsWithPattern2.isEmpty()) {
                int cellForRemove = random.nextInt(cellsWithPattern2.size());
                newBoard[cellsWithPattern2.get(cellForRemove).getRow()][cellsWithPattern2.get(cellForRemove).getCol()] = 0;
                cellsWithPattern2.remove(cellForRemove);
            }
        } else if (counter <= 24) {
            if (!cellsWithPattern3.isEmpty()) {
                int cellForRemove = random.nextInt(cellsWithPattern3.size());
                newBoard[cellsWithPattern3.get(cellForRemove).getRow()][cellsWithPattern3.get(cellForRemove).getCol()] = 0;
                cellsWithPattern3.remove(cellForRemove);
            }
        } else if (counter <= 56) {
            if (!cellsWithPattern4.isEmpty()) {
                int cellForRemove = random.nextInt(cellsWithPattern4.size());
                newBoard[cellsWithPattern4.get(cellForRemove).getRow()][cellsWithPattern4.get(cellForRemove).getCol()] = 0;
                cellsWithPattern4.remove(cellForRemove);
            }
        } else if (counter <= 81 && !cellsWithPattern5.isEmpty()) {
                int cellForRemove = random.nextInt(cellsWithPattern5.size());
                newBoard[cellsWithPattern5.get(cellForRemove).getRow()][cellsWithPattern5.get(cellForRemove).getCol()] = 0;
                cellsWithPattern5.remove(cellForRemove);
            }

    }
    return newBoard;
}
```

Pro pochopení, ve které zóně se buňka nachází, používám static List cells. Seznam obsahuje 81 buněk. V balíčku logic existuje třída Cell a má následující konstruktor:

- row - řádek,
- col - sloupec,
- square - jedna z 9 částí pole sudoku,
- pattern - zóna od 2 do 5

```
public Cell(int row, int col){
    this.row = row;
    this.col = col;
    this.square = getSquare(row, col);
    this.pattern = getPattern(row, col);
}
```

- Umístění zón buněk vypadá takto:

{4, 4, 5}, {4, 4, 5}, {5, 5, 5}	{4, 4, 2}, {4, 4, 2}, {3, 3, 3}	{4, 4, 5}, {4, 4, 5}, {5, 5, 5}
{4, 4, 3}, {4, 4, 3}, {2, 2, 3}	{5, 5, 2}, {2, 5, 2}, {2, 5, 5}	{3, 2, 2}, {3, 4, 4}, {3, 4, 4}
{5, 5, 5}, {4, 4, 5}, {4, 4, 5}	{3, 3, 3} {4, 4, 2}, {4, 4, 2},	{5, 5, 5}, {4, 4, 5}, {4, 4, 5}