

Lösningsförslag, omkontrollskrivning PTDC

2015-05-08

```
1. public class Board {
    private int[] [] board;

    public Board() {
        board = new int[8][8];
    }

    public void clear() {
        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 8; j++) {
                board[i][j] = 0;
            }
        }
    }

    public boolean isFree(int x, int y) {
        return x >= 0 && x < 8 && y >= 0 && y < 8 && board[y][x] == 0;
    }

    public void visit(int x, int y, int moveNbr) {
        board[y][x] = moveNbr;
    }

    public void print() {
        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 8; j++) {
                if (board[i][j] < 10) {
                    System.out.print(" ");
                }
                System.out.print(board[i][j] + " ");
            }
            System.out.println();
        }
    }
}

2. public class Knight {
    private Board board;
    private int x;
    private int y;
    private static final int[] dx = { 2, 2, -1, 1, -2, -2, -1, 1 };
    private static final int[] dy = { -1, 1, -2, -2, -1, 1, 2, 2 };

    public Knight(Board board) {
        this.board = board;
    }

    public void walk(int xStart, int yStart) {
        board.clear();
        x = xStart;
        y = yStart;
    }
}
```

```

        board.visit(x, y, 1);
        for (int moveNbr = 2; moveNbr <= 64; moveNbr++) {
            oneMove(moveNbr);
        }
    }

    private void oneMove(int moveNbr) {
        int minFree = Integer.MAX_VALUE;
        int bestX = -1;
        int bestY = -1;
        for (int i = 0; i < 8; i++) {
            int tryX = x + dx[i];
            int tryY = y + dy[i];
            if (board.isFree(tryX, tryY)) {
                int free = nbrFreeFrom(tryX, tryY);
                if (free < minFree) {
                    minFree = free;
                    bestX = tryX;
                    bestY = tryY;
                }
            }
        }
        x = bestX;
        y = bestY;
        board.visit(x, y, moveNbr);
    }

    private int nbrFreeFrom(int x, int y) {
        int nbrFree = 0;
        for (int i = 0; i < 8; i++) {
            if (board.isFree(x + dx[i], y + dy[i])) {
                nbrFree++;
            }
        }
        return nbrFree;
    }
}

```

```

3. public class WordList {
    private ArrayList<WordElement> words;

    public WordList() {
        words = new ArrayList<WordElement>();
    }

    public void addWord(String word, int pageNbr) {
        int i = findWordIndex(word);
        WordElement we;
        if (i < words.size() && words.get(i).getWord().equals(word)) {
            we = words.get(i);
        } else {
            we = new WordElement(word);
            words.add(i, we);
        }
        we.addPage(pageNbr);
    }

    public int size() {
        return words.size();
    }
}

```

```
    public String getWordLine(int nbr) {
        return words.get(nbr).toString();
    }

    private int findWordIndex(String w) {
        for (int i = 0; i < words.size(); i++) {
            if (words.get(i).getWord().compareTo(w) >= 0) {
                return i;
            }
        }
        return words.size();
    }
}
```

```
public class WLTest {
    public static void main(String[] args) {
        WordList wl = new WordList();
        Scanner scan = new Scanner(System.in);
        while (scan.hasNext()) {
            String word = scan.next();
            int pageNbr = scan.nextInt();
            wl.addWord(word, pageNbr);
        }
        scan.close();

        for (int i = 0; i < wl.size(); i++) {
            System.out.println(wl.getWordLine(i));
        }
    }
}
```
