

Lösningsförslag, omkontrollskrivning PTDC

2013-04-03

```
1. public class GameBoard {
    private int[] toSquare;

    public GameBoard(int nbrSquares, int nbrSnakes, int nbrLadders) {
        toSquare = new int[nbrSquares + 1];
        for (int i = 1; i <= nbrSquares; i++) {
            toSquare[i] = i;
        }
        Random rand = new Random();
        for (int i = 0; i < nbrLadders; i++) {
            int start = nextSpecialSquare(rand, nbrSquares);
            toSquare[start] = rand.nextInt(nbrSquares - (start + 1)) + (start + 1);
        }
        for (int i = 0; i < nbrSnakes; i++) {
            int start = nextSpecialSquare(rand, nbrSquares);
            toSquare[start] = rand.nextInt(start - 2) + 2;
        }
    }

    private int nextSpecialSquare(Random rand, int nbrSquares) {
        int square = 0;
        do {
            square = rand.nextInt(nbrSquares - 2) + 2;
        } while (square != toSquare[square]);
        return square;
    }

    public int getStart() {
        return 1;
    }

    public int getFinish() {
        return toSquare.length - 1;
    }

    public int getToSquare(int nbr) {
        return toSquare[nbr];
    }

    public void printSnakesAndLadders() {
        for (int i = 1; i < toSquare.length; i++) {
            if (i != toSquare[i]) {
                System.out.println(i + " " + toSquare[i]);
            }
        }
    }
}
```

```
2. public class SolitairePlay {
    private GameBoard board;
    private Die d;

    public SolitairePlay(GameBoard board, Die d) {
        this.board = board;
        this.d = d;
    }

    public int playOneSolitaireRound() {
        int pos = board.getStart();
        int nbrMoves = 0;
        while (pos < board.getFinish()) {
            d.roll();
            pos = pos + d.getResult();
            if (pos < board.getFinish()) {
                pos = board.getToSquare(pos);
            }
            nbrMoves++;
        }
        return nbrMoves;
    }
}

3. public class Main {
    public static void main(String[] args) {
        GameBoard board = new GameBoard(100, 5, 5);
        Die d = new Die();
        SolitairePlay sp = new SolitairePlay(board, d);
        Scanner scan = new Scanner(System.in);

        System.out.print("Antal spelomgångar: ");
        int n = scan.nextInt();
        int max = Integer.MIN_VALUE;
        int sum = 0;
        for (int i = 0; i < n; i++) {
            int nbrMoves = sp.playOneSolitaireRound();
            sum = sum + nbrMoves;
            if (nbrMoves > max) {
                max = nbrMoves;
            }
        }
        board.printSnakesAndLadders();
        System.out.println("Antal kast i genomsnitt: " + (double) sum / n);
        System.out.println("Högsta antalet kast: " + max);
    }
}
```

```
4. public class HighScoreList {
    private ArrayList<Player> list;

    public HighScoreList() {
        list = new ArrayList<Player>();
    }

    private void add(Player p) {
        int pos = 0;
        while (pos < list.size() && p.getMaxPoints() < list.get(pos).getMaxPoints()) {
            pos++;
        }
        list.add(pos, p);
    }

    private int find(String name) {
        for (int i = 0; i < list.size(); i++) {
            if (list.get(i).getName().equals(name)) {
                return i;
            }
        }
        return -1;
    }

    public boolean update(String name, int points) {
        int pos = find(name);
        Player p = null;
        if (pos != -1) {
            p = list.get(pos);
            if (!p.setMaxPoints(points)) {
                return false;
            }
            list.remove(pos);
        } else {
            p = new Player(name);
            p.setMaxPoints(points);
        }
        add(p);
        return true;
    }

    public ArrayList<Player> getBest(int maxNbr) {
        ArrayList<Player> result = new ArrayList<Player>();
        if (list.size() < maxNbr) {
            maxNbr = list.size();
        }
        for (int i = 0; i < maxNbr; i++) {
            result.add(list.get(i));
        }
        return result;
    }
}
```
