LUNDS TEKNISKA HÖGSKOLA Institutionen för datavetenskap

Lösningsförslag, kontrollskrivning 3 PTDC

2015–01–12

1.
```
public class FourWay {
    private int[][] a;

    public FourWay(int n) {
        a = new int[n][n];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                a[i][j] = 1;
            }
        }
    }

    public void oneRound() {
        int i = a.length / 2;
        int j = i;
        while (i >= 0 && i < a.length && j >= 0 && j < a.length) {
            int oldValue = a[i][j];
            a[i][j] = oldValue % 4 + 1;
            switch (oldValue) {
            case 1:
                i--;
                break;
            case 2:
                j++;
                break;
            case 3:
                i++;
                break;
            case 4:
                j--;
                break;
            }
        }
    }

    public boolean allOnes() {
        for (int i = 0; i < a.length; i++) {
            for (int j = 0; j < a.length; j++) {
                if (a[i][j] != 1) {
                    return false;
                }
            }
        }
        return true;
    }
}
```

2.
```
public static int[] hyphenate(String word) {
    ArrayList<Integer> resList = new ArrayList<Integer>();
    for (int i = 1; i < word.length() - 1; i++) {
        if (!isVowel(word.charAt(i)) && isVowel(word.charAt(i + 1))) {
            resList.add(i - 1);
        }
    }
    int[] resArray = new int[resList.size()];
    for (int i = 0; i < resList.size(); i++) {
        resArray[i] = resList.get(i);
    }
    return resArray;
}
```

3.
```
public class BulgarianSolitaire {
    private int n;
    private ArrayList<Pile> piles;

    /** Skapar ett patiensobjekt där n kort är slumpmässigt fördelade på ett
        slumpmässigt antal högar */
    public BulgarianSolitaire(int n) {
        this.n = n;
        piles = new ArrayList<Pile>();
        Random rand = new Random();
        int nbrPiles = 1 + rand.nextInt(n);
        for (int i = 0; i < nbrPiles; i++) {
            piles.add(new Pile(1));
        }
        for (int i = nbrPiles; i < n; i++) {
            int pile = rand.nextInt(piles.size());
            piles.get(pile).putCard();
        }
    }

    /** Undersöker om antalet kort i högarna är 1, 2, ..., k */
    public boolean atGoal() {
        int[] count = new int[n + 1];
        for (int i = 0; i < piles.size(); i++) {
            int size = piles.get(i).getSize();
            count[size]++;
        }
        int sum = 0;
        int i = 1;
        while (sum < n && count[i] == 1) {
            sum += i;
            i++;
        }
        return sum == n;
    }
```

```
    /** Tar ett kort från varje hög och lägger i en ny hög */
    public void move() {
        int newSize = piles.size();
        int i = 0;
        while (i < piles.size()) {
            piles.get(i).takeCard();
            if (piles.get(i).getSize() > 0) {
                i++;
            } else {
                piles.remove(i);
            }
        }
        piles.add(new Pile(newSize));
    }

    /** Skriver ut antalet kort i varje hög */
    public void print() {
        for (int i = 0; i < piles.size(); i++) {
            System.out.print(piles.get(i).getSize() + " ");
        }
        System.out.println();
    }
}
```