Rešitev pete domače naloge (Tetris)

Najtežji del te naloge je ugotoviti, kako predstaviti posamezne like. Glede na to, da moramo izračunati samo končno višino posameznih stolpcev v Tetrisovi »jami«, ne pa tudi njihovo vsebino (z morebitnimi vrzelmi v stolpcih se nam ni treba ukvarjati), zadošča, če vsak lik predstavimo z dvema tabelama tipa `int[]`:

- Prva tabela (imenujmo jo V) pove, kako visoko segajo posamezni stolpci lika, če lik spustimo v prazno jamo. Na primer, za lik z indeksom 0 velja $V = \langle 1, 1, 1, 1 \rangle$, pri liku z indeksom 4 pa imamo $V = \langle 2, 3 \rangle$.
- Druga tabela (dajmo ji ime O) pa pove, kako visoko segajo morebitne praznine pod stolpci, če lik spustimo v prazno jamo. Na primer, za lik z indeksom 0 velja $O = \langle 0, 0, 0, 0 \rangle$ (pod nobenim stolpcem ni praznine), pri liku z indeksom 4 pa imamo $O = \langle 1, 0 \rangle$ (prvi stolpec se prične na višini 1).

Tabele V za posamezne like bomo hranili v tabeli `VISINE` (`VISINE[i]` je kazalec na tabelo V za lik z indeksom `i`), tabele `O` pa v tabeli `ODMIKI`:

```
import java.util.Scanner;

public class Tetris {

    private static final int[][] VISINE = {
        {1, 1, 1, 1},
        {4},
        {2, 2},
        {1, 2, 1},
        {2, 3},
        {2, 2, 2},
        {3, 2},
        {3, 1},
        {1, 1, 2},
        {3, 3},
        {2, 2, 2},
        {1, 3},
        {2, 2, 2},
        {3, 3},
        {2, 1, 1},
        {2, 2, 1},
        {2, 3},
        {1, 2, 2},
        {3, 2}
    };

    private static final int[][] ODMIKI = {
        {0, 0, 0, 0},
        {0},
        {0, 0},
        {0, 0, 0},
        {1, 0},
        {1, 0, 1},
```

```
        {0, 1},
        {0, 0},
        {0, 0, 0},
        {2, 0},
        {0, 1, 1},
        {0, 0},
        {1, 1, 0},
        {0, 2},
        {0, 0, 0},
        {1, 0, 0},
        {0, 1},
        {0, 0, 1},
        {1, 0}
    };
    ...
}
```

V metodi `main` beremo podatke o posameznih spustih in sproti simuliramo spuščanje likov v jamo. Višine posameznih stolpcev bomo hranili v tabeli `stolpci`. Zaradi omejitve $x \in [-1000, 1000]$ bodo koordinate x nepraznih stolpcev zanesljivo znotraj intervala $[-1000, 1003]$, kar pomeni, da lahko dolžino tabele `stolpci` nastavimo na, recimo, 2010 (v resnici bi lahko bila še malo manjša, a malo rezerve ponavadi ne škodi). Višina stolpca s koordinato x je v tabeli `stolpci` zapisana na indeksu $x + 1000$.

```
public class Tetris {
    ...
    private static final int ODMIK_X = 1000;
    private static final int MAX_RAZPON_X = 2010;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int stSpustov = sc.nextInt();
        int[] stolpci = new int[MAX_RAZPON_X];

        for (int i = 0;  i < stSpustov;  i++) {
            int lik = sc.nextInt();
            int x = sc.nextInt() + ODMIK_X;
            postavi(stolpci, lik, x);
        }
        for (int i = 0;  i < stolpci.length;  i++) {
            if (stolpci[i] > 0) {
                System.out.printf("%d: %d%n", i - ODMIK_X, stolpci[i]);
            }
        }
    }
}
```

Ko v jamo spustimo vse like, se preprosto sprehodimo po tabeli `stolpci` in izpišemo koordinate in višine nepraznih stolpcev.

Metoda `postavi` simulira spust lika s podanim indeksom na koordinato x in ustrezno posodobi tabelo `stolpci`. Če ne bi bilo praznin pod stolpci lika (tj. če bi veljalo $O[i] = 0$

za vse stolpce lika), bi novo višino stolpca s koordinato $x + i$ izračunali kot `osnova` $+ V[i]$, pri čemer je `osnova` višina najvišjega med stolpci s koordinatami x, $x + 1$, ..., $x + w - 1$, w pa je širina lika. Zaradi praznin pod stolpci lika pa je lahko število `osnova` tudi nekoliko nižje. Izračunamo ga kot maksimum vsot `stolpci`$[x + i] - O[i]$, kjer i teče od 0 do $w - 1$.

```
public class Tetris {
    ...
    private static void postavi(int[] stolpci, int ixLik, int x) {
        int[] visine = VISINE[ixLik];
        int[] odmiki = ODMIKI[ixLik];

        int osnova = 0;
        for (int i = 0;  i < visine.length;  i++) {
            osnova = Math.max(osnova, stolpci[x + i] - odmiki[i]);
        }

        for (int i = 0;  i < visine.length;  i++) {
            stolpci[x + i] = osnova + visine[i];
        }
    }
}
```