Rešitev 1. domače naloge (Jaka in Darko)

Začetek ni težak: preberemo dimenzije predavalnice in število študentov, ki prosijo za pomoč.

```
import java.util.Scanner;

public class JakaInDarko {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int stVrst = sc.nextInt();
        int stStolpcev = sc.nextInt();
        int stProsenj = sc.nextInt();
        ...
    }
}
```

V prvi različici programa predpostavimo, da Jaka in Darko nikoli nista enako oddaljena od prosilca za pomoč. Da bomo lahko izračunali pot, ki jo prehodita v terminu vaj, bomo vzdrževali njun trenutni položaj (spremenljivke `vrJaka`, `stJaka`, `vrDarko` in `stDarko`) in pot, ki sta jo do danega trenutka prehodila (spremenljivki `potJaka` in `potDarko`):

```
public class JakaInDarko {

    public static void main(String[] args) {
        ...
        // spremenljivke vr... vsebujejo koordinate vrstic,
        // spremenljivke st... pa koordinate stolpcev

        int vrJaka = 0;
        int stJaka = 0;
        int vrDarko = stVrst - 1;
        int stDarko = stStolpcev - 1;

        int potJaka = 0;
        int potDarko = 0;
        ...
    }
}
```

Sedaj pa v zanki beremo položaje študentov, ki prosijo za pomoč, ter posodabljamo položaj asistentov in njuno pot. Če je Jaka bližje prosilcu kot Darko, posodobimo Jakov položaj in razdaljo prištejemo njegovi poti, sicer pa storimo enako za Darka. Ko se zanka izteče, spremenljivki `potJaka` in `potDarko` vsebujeta celotno pot, ki sta jo prehodila asistenta.

```
public class JakaInDarko {

    public static void main(String[] args) {
        ...
```

```
        for (int p = 1;  p <= stProsenj;  p++) {
            // preberemo koordinati prosilca
            int vrStudent = sc.nextInt();
            int stStudent = sc.nextInt();

            // izračunamo Jakovo in Darkovo razdaljo
            int dJaka = razdalja(vrJaka, stJaka, vrStudent, stStudent);
            int dDarko = razdalja(vrDarko, stDarko, vrStudent, stStudent);

            if (dJaka < dDarko) {
                // k študentu priskoči Jaka
                vrJaka = vrStudent;
                stJaka = stStudent;
                potJaka += dJaka;

            } else {
                // k študentu priskoči Darko
                vrDarko = vrStudent;
                stDarko = stStudent;
                potDarko += dDarko;
            }
        }

        // izpišemo pot, ki sta jo prehodila asistenta
        System.out.println(potJaka);
        System.out.println(potDarko);
    }
}
```

Računanje manhattanske razdalje je jasno definiran podproblem, ki ga za nameček še dvakrat rešujemo. Zato si zasluži svojo metodo:

```
public class JakaInDarko {
    ...
    private static int razdalja(int v1, int s1, int v2, int s2) {
        return (Math.abs(v1 - v2) + Math.abs(s1 - s2));
    }
    ...
}
```

Lotimo se še možnosti, da sta asistenta od prosilca enako oddaljena. Ko se tak primer prvič pojavi, se odzove Jaka. Ko nastopi drugič, se na pot odpravi Darko. Pri tretjem takem dogodku je spet na vrsti Jaka, nato spet Darko itd. Logična spremenljivka `jakaNaVrsti` naj pove, kdo se bo odzval, ko bosta asistenta naslednjič enako oddaljena od študenta. Vrednost `true` označuje, da bo to Jaka, vrednost `false` pa pove, da gre za Darka. Ko taka situacija dejansko nastopi, moramo spremenljivko seveda negirati, da bomo pripravljeni na naslednji tovrstni dogodek.

```
import java.util.Scanner;

public class JakaInDarko {
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int stVrst = sc.nextInt();
        int stStolpcev = sc.nextInt();
        int stProsenj = sc.nextInt();

        int vrJaka = 0;
        int stJaka = 0;
        int vrDarko = stVrst - 1;
        int stDarko = stStolpcev - 1;

        int potJaka = 0;
        int potDarko = 0;
        boolean jakaNaVrsti = true;

        for (int p = 1;  p <= stProsenj;  p++) {
            int vrStudent = sc.nextInt();
            int stStudent = sc.nextInt();
            int dJaka = razdalja(vrJaka, stJaka, vrStudent, stStudent);
            int dDarko = razdalja(vrDarko, stDarko, vrStudent, stStudent);

            if (dJaka < dDarko || dJaka == dDarko && jakaNaVrsti) {
                // Jaka se odzove, če je bližji kot Darko ali pa če sta oba
                // enako oddaljena, vendar pa je na vrsti po pravilu menjavanja
                vrJaka = vrStudent;
                stJaka = stStudent;
                potJaka += dJaka;

            } else {
                vrDarko = vrStudent;
                stDarko = stStudent;
                potDarko += dDarko;
            }

            if (dJaka == dDarko) {
                jakaNaVrsti = !jakaNaVrsti;
            }
        }

        System.out.println(potJaka);
        System.out.println(potDarko);
    }

    private static int razdalja(int v1, int s1, int v2, int s2) {
        return (Math.abs(v1 - v2) + Math.abs(s1 - s2));
    }
}
```