Cvičenia č. 3, úloha č. 4

Uvažujme nasledujúcu jednoduchú hru pre dvoch hráčov: na stole je $n \geq 1$ zápaliek a každý hráč si môže v rámci každého svojho ťahu zobrať, pre nejaké vopred dané $k \geq 1$, jednu až k z nich. Hráči sa vo svojich ťahoch striedajú a prehráva ten z nich, ktorý zo stola zoberie poslednú zápalku.

V priloženej kostre je už hotová trieda `Matches` v balíku `matches` s metódou `main`, ktorá načítava vstup a vypisuje výstup simulujúci priebeh vyššie opísanej hry (formát vstupu a výstupu je vysvetlený nižšie). V metóde `main` sa využívajú aj triedy, ktorých implementácia sa očakáva od vás.
Ďalej je súčasťou priloženej kostry rozhranie `Player` (tiež v balíku `matches`) deklarujúce jedinú metódu, ktorú by mala poskytovať každá trieda reprezentujúca hráča opísanej hry: metódu

```
int take(int totalMatches, int maxMatches),
```

ktorej vstupnými argumentmi sú momentálny počet zápaliek na stole `totalMatches` a hodnota `maxMatches` udávajúca maximálny povolený počet zápaliek, ktoré si môže hráč zobrať v jednom ťahu. Môžete predpokladať, že `totalMatches` aj `maxMatches` sú pri každom volaní implementácie tejto metódy pre nejakú triedu vždy nenulové prirodzené čísla a že hodnota `maxMatches` je v priebehu jednej hry vždy tá istá.
Napíšte nasledujúce tri triedy implementujúce rozhranie `Player` a reprezentujúce hráčov pre vyššie opísanú hru:

- Trieda `MinimalisticPlayer` by mala reprezentovať hráča, ktorý si za každých okolností zoberie jedinú zápalku. Pri tejto triede sa od vás očakáva iba implementácia metódy `take`.
- Trieda `OptimalPlayer` by mala reprezentovať hráča, ktorý používa optimálnu stratégiu pre opísanú hru (takýto hráč síce nemusí vyhrať vždy, ale vyhrá vždy, keď to je možné). Nech k je maximálny povolený počet zápaliek, ktorý si jeden hráč môže zobrať v jednom ťahu. Hráč typu `OptimalPlayer` sa počas celého priebehu hry snaží brať zo stola také počty zápaliek, aby počet zápaliek, ktorý na stole ostane *po jeho ťahu*, dával po delení číslom $k+1$ zvyšok 1. Akonáhle sa mu totiž takúto situáciu podarí dosiahnuť raz, môže v nasledujúcich ťahoch už stále udržiavať tento zvyšok na hodnote 1, čím napokon donúti jeho protihráča zobrať zo stola poslednú zápalku. V prípade, že hráč typu `OptimalPlayer` nemá možnosť uskutočniť svoj ťah týmto spôsobom, zoberie zo stola práve jednu zápalku.[1] Aj pri tejto triede sa od vás očakáva iba implementácia metódy `take`.
- Trieda `HumanPlayer` by mala reprezentovať človeka, ktorý počty zápaliek priebežne zadáva na konzolu. Mala by obsahovať konštruktor `public HumanPlayer(Scanner scanner, PrintStream out)`, ktorý iba skopíruje referencie `scanner` a `out` do premenných inštancie tejto triedy. Môžete predpokladať, že aj `scanner` aj `out` sú už korektne otvorené. Implementácia metódy `take` by sa pre túto triedu mala správať nasledujúcim spôsobom: do vstupného prúdu `out` sa vypíše text `"Zadaj pocet zapaliek: "` (s medzerou na konci, bez úvodzoviek a bez znaku pre nový riadok); následne sa pomocou skenera `scanner` prečíta jedno celé číslo. Ak používateľ zadá číslo mimo povoleného rozsahu (1 až `maxMatches`), celý proces sa opakuje; v opačnom prípade sa číslo zadané používateľom vráti ako výstup metódy `take`.

Všetky tieto triedy by mali byť súčasťou balíka `matches` a mali by byť uložené v samostatných súboroch. Kód triedy `Matches` a rozhrania `Player` nemeňte.

Vstup, ktorý číta metóda `main` triedy `Matches`, začína dvojicou celých čísel reprezentujúcich počiatočný počet zápaliek na stole a maximálny povolený počet odobratých zápaliek v jednom ťahu. Za nimi nasledujú typy nultého a prvého hráča, ktorými môžu byť `"MINIMALISTIC"`, `"OPTIMAL"`, alebo `"HUMAN"`. Na základe nich metóda `main` vytvorí inštancie príslušných tried a následne odsimuluje priebeh hry s týmito hráčmi a s danými parametrami. V prípade, že je medzi hráčmi človek, pokračuje vstup jeho jednotlivými ťahmi. Na konzolu sa priebežne vypisuje výstup informujúci o stave hry. Ukážky kombinovaných vstupov a výstupov programu možno nájsť nižšie.
Na testovač odovzdávajte ZIP archív obsahujúci priečinok `matches` a v ňom zdrojové súbory všetkých tried tohto balíka (vrátane tých, ktoré sa oproti kostre nezmenili).

[1] Na tejto voľbe z hľadiska optimality stratégie nezáleží, ale je podstatná kvôli kontrole výstupov programu na testovači.

Príklad vstupu a výstupu č. 1:

```
23 4
OPTIMAL
HUMAN
Hrac 0 berie 2 zapaliek. Zostava 21 zapaliek.
Zadaj pocet zapaliek: 4
Hrac 1 berie 4 zapaliek. Zostava 17 zapaliek.
Hrac 0 berie 1 zapaliek. Zostava 16 zapaliek.
Zadaj pocet zapaliek: 10
Zadaj pocet zapaliek: 10
Zadaj pocet zapaliek: 4
Hrac 1 berie 4 zapaliek. Zostava 12 zapaliek.
Hrac 0 berie 1 zapaliek. Zostava 11 zapaliek.
Zadaj pocet zapaliek: 4
Hrac 1 berie 4 zapaliek. Zostava 7 zapaliek.
Hrac 0 berie 1 zapaliek. Zostava 6 zapaliek.
Zadaj pocet zapaliek: 4
Hrac 1 berie 4 zapaliek. Zostava 2 zapaliek.
Hrac 0 berie 1 zapaliek. Zostava 1 zapaliek.
Zadaj pocet zapaliek: 1
Hrac 1 berie 1 zapaliek. Zostava 0 zapaliek.
Vyhrava hrac 0
```

Príklad vstupu a výstupu č. 2:

```
23 4
MINIMALISTIC
OPTIMAL
Hrac 0 berie 1 zapaliek. Zostava 22 zapaliek.
Hrac 1 berie 1 zapaliek. Zostava 21 zapaliek.
Hrac 0 berie 1 zapaliek. Zostava 20 zapaliek.
Hrac 1 berie 4 zapaliek. Zostava 16 zapaliek.
Hrac 0 berie 1 zapaliek. Zostava 15 zapaliek.
Hrac 1 berie 4 zapaliek. Zostava 11 zapaliek.
Hrac 0 berie 1 zapaliek. Zostava 10 zapaliek.
Hrac 1 berie 4 zapaliek. Zostava 6 zapaliek.
Hrac 0 berie 1 zapaliek. Zostava 5 zapaliek.
Hrac 1 berie 4 zapaliek. Zostava 1 zapaliek.
Hrac 0 berie 1 zapaliek. Zostava 0 zapaliek.
Vyhrava hrac 1
```