Cvičenia č. 5, úloha č. 7

Pripomeňme si, že pod *variáciou bez opakovania k-tej triedy z prvkov množiny S* rozumieme ľubovoľnú k-prvkovú postupnosť prvkov S, v ktorej sa žiadne dva prvky množiny S neopakujú.

Stiahnite si priloženú kostru generickej triedy `Variations<E extends Comparable<E>>` (v nepomenovanom balíku), ktorá má realizovať hľadanie všetkých variácií bez opakovania danej triedy z prvkov danej množiny. Typový parameter `E` triedy `Variations` reprezentuje typ prvkov množiny, z ktorej sa variácie budú vytvárať; typ `E` pritom musí implementovať rozhranie `Comparable<E>`. Inštancia triedy `Variations` dostane v argumentoch svojho konštruktora množinu `set` prvkov typu `E` a celé číslo `k`. Takáto inštancia potom bude reprezentovať iterátor cez všetky variácie bez opakovania `k`-tej triedy z prvkov množiny `set` – jej metóda `next` bude vždy vracať nasledujúcu variáciu v poradí opísanom nižšie (ak ešte nejaká existuje).
Jedna variácia sa bude vždy reprezentovať ako zoznam typu `List<E>`. Poradie, v ktorom sa jednotlivé variácie budú generovať, bude dané lexikografickým usporiadaním na k-ticiach prvkov typu `E` vzhľadom na prirodzené usporiadanie $\leq$ prvkov typu `E`. Pri tomto usporiadaní je k-tica $(a_1, \ldots, a_k)$ menšia ako k-tica $(b_1, \ldots, b_k)$ práve vtedy, keď existuje $j \in \{1, \ldots, k\}$ také, že $a_j < b_j$ a pre $i = 1, \ldots, j-1$ je $a_i = b_i$. Napríklad štvorica celých čísel $(1, 2, 5, 6)$ je pri tomto usporiadaní menšia ako $(1, 3, 2, 5)$ a tá je menšia ako $(2, 0, 3, 1)$.[1]

V triede `Variations`, ktorá bude implementovať `Iterator<List<E>>`, naprogramujte:

- Konštruktor, ktorý ako argumenty vezme množinu `set` typu `Set<E>` a celé číslo `k`. Ak `set == null` alebo `k < 0`, dôjde k vyhodeniu výnimky typu `IllegalArgumentException` (táto funkcionalita je už v kostre implementovaná). V opačnom prípade sa vykonajú vhodné inicializačné úkony pre iterátor generujúci všetky variácie bez opakovania `k`-tej triedy z prvkov množiny `set` v poradí opísanom vyššie. Korektný iterátor sa vytvorí aj v prípade, že je `k` väčšie ako počet prvkov množiny `set`; pôjde ale potom o iterátor, ktorého metóda `hasNext` bude od začiatku vracať `false`, pretože v takomto prípade žiadna variácia bez opakovania `k`-tej triedy neexistuje.

  Prípadná neskoršia zmena množiny, ktorá bola použitá ako argument konštruktora, by nijak nemala ovplyvniť vnútorný stav inštancie triedy `Variations` a výstupy jej metód.

- Metódu `next`, ktorá vráti nasledujúcu variáciu bez opakovania `k`-tej triedy z prvkov množiny `set` reprezentovanú ako zoznam prvkov typu `E`. V prípade, že už žiadna ďalšia variácia neexistuje, dôjde k vyhodeniu výnimky typu `NoSuchElementException`.

- Metódu `hasNext`, ktorá vráti `true` práve vtedy, keď takáto ďalšia variácia existuje (čiže keď metóda `next` pri svojom ďalšom volaní nevyhodí výnimku).

V rámci inštancie triedy `Variations` si môžete pamätať všetky prvky množiny `set` (prípadne aj vo viacerých reprezentáciách), ako aj nejaký malý konštantný počet generovaných variácií (ideálne jednu). *Nepamätajte* si v nej ale zoznam *všetkých* variácií – úlohu teda *neriešte* vygenerovaním všetkých variácií už v konštruktore. V metóde `next` nespúšťajte generovanie variácií zakaždým odznova.

Príklad. Predpokladajme, že vytvoríme inštanciu triedy `Variations<Integer>` pre množinu `set` obsahujúcu čísla `1, 3, 5, 7` a pre `k = 2`. Metóda `next` by potom mala postupne vracať zoznamy `[1, 3]`, `[1, 5]`, `[1, 7]`, `[3, 1]`, `[3, 5]`, `[3, 7]`, `[5, 1]`, `[5, 3]`, `[5, 7]`, `[7, 1]`, `[7, 3]` a `[7, 5]`. Prípadné ďalšie volanie metódy `next` by malo vyústiť vo vyhodenie výnimky typu `NoSuchElementException`.

Na testovač odovzdávajte súbor `Variations.java` obsahujúci zdrojový kód vami doplnenej triedy.

[1] Ide teda o obdobu bežného lexikografického usporiadania na reťazcoch, kde ale namiesto symbolov uvažujeme prvky typu `E`.