Test č. 2, úloha č. 3

Napíšte generickú triedu `IncreasinglyRepetitiveIterator<E>` (v nepomenovanom balíku) implementujúcu rozhranie `Iterator<E>` a poskytujúcu konštruktor, ktorého jediným argumentom je inštancia `iterator` typu `Iterator<E>`.
Inštancia vytvorená týmto konštruktorom bude reprezentovať iterátor, ktorý sa správa podobne ako iterátor `iterator` z argumentu konštruktora, avšak prvky prechádzané iterátorom `iterator` zakaždým vracia aj niekoľkokrát za sebou. *Presnejšie*: prvok, ktorý je iterátorom `iterator` vrátený ako k-ty v poradí (počítané od 1) vráti váš iterátor v presne k po sebe idúcich volaniach metódy `next`; prípadným ďalším volaním metódy `next` sa váš iterátor posunie na prvok vrátený iterátorom `iterator` ako $(k+1)$-vý v poradí.
V prípade, že metóda `next` už podľa horeuvedenej špecifikácie nemôže vrátiť žiaden ďalší prvok, malo by jej volanie mať za následok vyhodenie výnimky typu `NoSuchElementException`. Metóda `hasNext` by sa mala správať konzistentne s metódou `next`.

Príklad. Predpokladajme, že argumentom konštruktora je iterátor cez zoznam obsahujúci prvky

`1, 2, 3, 4, 5.`

Volania metódy `next` vášho iterátora by potom mali postupne vracať prvky

`1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5.`

Pri pokuse o ďalšie volanie metódy `next` by mala vzniknúť výnimka typu `NoSuchElementException`.

Maximálny počet prvkov typu `E`, ktorý si budete pamätať v rámci jednej inštancie vašej triedy, by nemal závisieť od počtu prvkov prechádzaných iterátorom `iterator`. Špeciálne teda úlohu *neriešte* vygenerovaním všetkých budúcich výstupov metódy `next` už v konštruktore.

Na testovač odovzdávajte súbor `IncreasinglyRepetitiveIterator.java` s kódom vašej triedy.