Cvičenia č. 4, úloha č. 3

Napíšte generickú triedu `Multiset<E>` v nepomenovanom balíku, ktorá bude reprezentáciou *multimnožiny* – čiže množiny, ktorá môže obsahovať aj viacnásobné výskyty svojich prvkov. Každému prvku `e` typu `E` teda zodpovedá (prípadne aj nulové) prirodzené číslo udávajúce počet jeho výskytov v multimnožine; hovoríme, že multimnožina *obsahuje* prvok `e`, ak je tento jeho počet výskytov v danej multimnožine nenulový. Trieda `Multiset<E>` by mala poskytovať:[1]

- Konštruktor bez parametrov, ktorý vytvorí prázdnu multimnožinu. Všetky prvky typu `E` sa v takejto multimnožine vyskytujú nulakrát.
- Metódu `public void add(E e)`, ktorá pridá jeden výskyt prvku `e` do reprezentovanej multimnožiny.
- Metódu `public void remove(E e)`, ktorá z reprezentovanej multimnožiny odoberie jeden výskyt prvku `e`. V prípade, že sa prvok `e` v multimnožine vôbec nevyskytuje (t. j. multimnožina obsahuje nula výskytov tohto prvku), nemalo by mať volanie tejto metódy žiaden efekt.
- Metódu `public boolean contains(E e)`, ktorá vráti `true` práve vtedy, keď reprezentovaná multimnožina obsahuje aspoň jeden výskyt prvku `e`.
- Metódu `public int getMultiplicity(E e)`, ktorá vráti počet výskytov prvku `e` v reprezentovanej multimnožine; v prípade, že `e` nie je prvkom reprezentovanej multimnožiny, tak táto metóda vždy vráti nulu.

Môžete predpokladať, že prvky typu `E` majú korektne sa správajúce metódy `equals` a `hashCode`; korektne sa teda bude správať aj `HashMap` s kľúčmi typu `E`.

Na testovač odovzdávajte súbor `Multiset.java` obsahujúci zdrojový kód triedy `Multiset<E>`.

[1] Všetky porovnávania prvkov typu `E` by sa mali realizovať pomocou ich metódy `equals`. Nie je ale nutné takéto porovnávania písať do kódu explicitne; výhodnejšie je pri implementácii triedy `Multiset<E>` využiť už existujúce dátové štruktúry.