Zadaci za vježbu iz teme 8 (Generici)

1. Napišite klasu `KeyValueBasic` koja modelira zapis „ključ-vrijednost“. Ima sljedeće članske varijable: `key` tipa `int` i `value` tipa `String`. Napišite konstruktor koji inicijalizira članske varijable, oba *gettera*, toString() s ispisom stanja članskih varijabli i *setter* za `value`. Ilustrirajte primjer uporabe navedene klase putem metode `main`.

2. Napišite klasu `KeyValueParameterized` koja ima iste funkcionalnosti poput `KeyValueBasic` samo što umjesto int i String koristi parametrizirane tipove `K` i `V`. Ilustrirajte primjer uporabe navedene klase putem metode `main` i objasnite zašto je ovo rješenje bolje od `KeyValueBasic`.

3. Po uzoru na klasu `KeyValueParameterized`, napišite klasu `KeyValueMapEntry` na način da implementira sučelje `java.util.Map.Entry<K,V>`. Prije implementacije proučite *JavaDoc* sučelja: https://docs.oracle.com/javase/8/docs/api/java/util/Map.Entry.html. Identificirajte razlike u odnosu na `KeyValueParameterized`.

4. Odaberite primitivni tip po želji. Deklarirajte i inicijalizirajte njegovu lokalnu varijablu (odaberite vrijednost po vlastitom nahođenju) unutar metode `main`. Nakon toga:

    a. Ručno napravite nepromjenjivi objekt koji omata vrijednost te se nalazi na hrpi.

    b. Ručno dohvatite vrijednost objekta iz omotača koja će se sada ponovno naći na stogu.

    c. Ponovite korake a i b ali koristite automatski pristup.

    d. U varijablu koja prima referencu na omotač pridružite `null` te potom ponovite korak b. Što se dogodilo?

5. Napišite statičku metodu `calculateAverageKey` koja prima varijabilan broj objekata klase `KeyValueMapEntry` koju ste napisali u trećem zadatku. Metoda računa i vraća prosječnu vrijednost ključeva. Posljedično, metoda može raditi samo s objektima čiji ključevi („key“) predstavljaju brojeve. Uočiti kako nije bitno kojeg su tipa vrijednosti („value“) u objektima klase `KeyValueMapEntry`. Pretpostavite da objekti mogu imati različite tipove vrijednosti. Metoda mora imati odgovarajuću ogradu koja će osigurati da radi s „pravilno“ parametriziranim objektima. Pretpostavite da se metodi `calculateAverageKey` ne može proslijediti vrijednost `null` niti da ključ može biti `null`.

6. Promijenite metodu `calculateAverageKey` na način da u obzir uzima objekte klase `KeyValueMapEntry` koji imaju iste tipove vrijednosti. Drugim riječima, npr. ako je prvi objekt parametriziran kao `<Integer,String>` tada i ostali objekti koji se prosljeđuju metodi `calculateAverageKey` moraju biti parametrizirani kao `<Integer,String>`.

7. Napišite klasu `CountableKeyValueMapEntry` koja nasljeđuje `KeyValueMapEntry` te prilikom inicijalizacije slijedno numerira svoje objekte kroz člansku varijablu `id` tipa int. Dodatno, prilagodite parametrizaciju na način da tip ključa više nije parametriziran već je `Integer`. Ilustrirajte primjer uporabe navedene klase putem metode `main` na studijskom slučaju Prve hrvatske nogometne lige: ključ je pozicija na ljestvici (https://prvahnl.hr/prva-liga/ljestvica/), vrijednost je ime kluba.

Rješenja zadataka dostupna su na sljedećoj poveznici:
https://github.com/FER-OOP/Lectures/tree/master/Exercises/Homework-08

Komentari:
1. Jednostavno rješenje koje je dobro za jednostavne primjene.

2. Ovo rješenje je bolje od prvog jer sada imamo slobodu odabira tipova. Drugim riječima, nismo prisiljeni koristiti `integer` i `String` već možemo koristiti tipove koje mi želimo.
3. Sučelje `Map.Entry<K,V>` propisuje gettere koje već imamo. Razlika je u definiciji settera: prema JavaDoc-u, setter za `value` mora vratiti staru vrijednost članske varijable `value`. Sjetite se ovog sučelja kada ćemo učiti o kolekcijama.
4. Primitivni tipovi uključuju: *byte, short, int, long, char, boolean, float* i *double.*
    a. Radi se *boxing.*
    b. Radi se *unboxing.*
    c. Radi se automatski *boxing/unboxing* bez eksplicitnog poziva metoda. Prevodilac će sam odraditi taj posao za nas.
    d. Dogodila se iznimka NullPointerException, nije moguće odmotati null u smislenu primitivnu vrijednost.
5. Parametrizirat ćemo metodu na način da ćemo osigurati da su tipovi ključeva podtipovi klase Number. S obzirom na to kako u zadatku nije propisano kojeg tipa moraju biti vrijednosti (te tipovi vrijednosti mogu biti različiti od objekta do objekta), koristit ćemo zamjenski tip "?".
6. Dodatno ćemo parametrizirati metodu kako bismo ogradili tip za vrijednost. Drugim riječima, u potpisu metode nam neće stajati „?" već parametar poput „V". Time smo osigurali da objekti budu usklađeni po tipovima za ključ i vrijednost. Uočite da u rješenju unutar for petlje svejedno koristimo „?", prije svega zato što ni na koji način ne koristimo vrijednost. U protivnom bismo koristili „V".
7. Za brojač ćemo koristiti statičku varijablu. Uočiti kako kod nasljeđivanja u klasi `CountableKeyValueMapEntry` koristimo: `CountableKeyValueMapEntry<V> extends KeyValueMapEntry<Integer, V>` kako bismo ispunili zahtjev iz zadatka (posljednja rečenica).