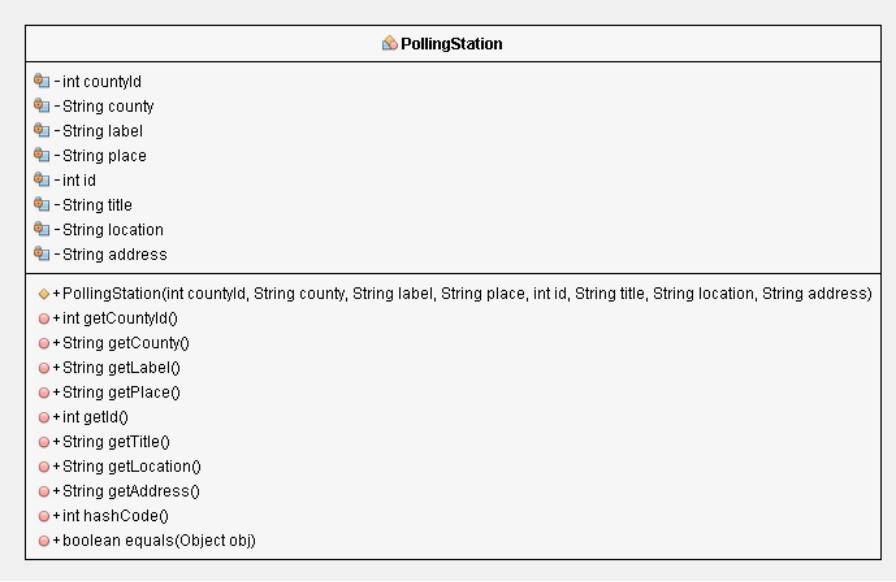
Zadaci za vježbu iz napredne funkcionalnosti kolekcija i kolekcijskih tokova

U sljedećem nizu zadataka potrebno je nadograditi programski kod primjera koji modelira aplikaciju za analizu rezultata lokalnih izbora 2021. godine. Rezultati izbora za grad Zagreb će se dohvaćati sa sljedeće adrese jer službene stranice Državnog izbornog povjerenstva više nisu dostupne. Rezultati se nalaze u komprimiranom nizu XLS datoteka za pojedinu županiju i grad Zagreb. U sljedećim podzadacima će prvo trebati dohvatiti rezultate i pohraniti ih u odgovarajuće strukture podataka te nakog toga napisati dijelove koda koji će na traženi način analizirati rezultate ovih izbora. Za rad s XLS datotekama će biti potrebno koristiti pomoćnu knjižnicu JXL pa je stoga potrebno napraviti Maven ili Gradle projekt te navesti navedenu knjižnicu kao ovisnost u projekt. Za dodavanje navedene knjižnice u Maven projekt u pom.xml je potrebno dodati sljedeće:

```
<dependencies>
    <dependency>
        <groupId>net.sourceforge.jexcelapi</groupId>
        <artifactId>jxl</artifactId>
        <version>2.6.12</version>
    </dependency>
</dependencies>
```

1. (Ponavljanje upotrebe vlastitih klasa s Javinim okvirom kolekcija) Napišite klasu `PollingStation` kojom ćemo modelirati biračko mjesto. Ova klasa ima 8 atributa od kojih se prva 4 odnose na županiju (*county*) i mjesto (*place*), a preostala 4 na samo biračko mjesto (*polling station*). UML dijagram ove klase je prikazan na slici ispod. Ona ima konstruktor koji prima vrijednosti navedenih 8 atributa te *gettere* za njih. Osim toga, ova klasa treba imati implementirane metode `hashCode` i `equals`, a pri njihovoj implementaciji uzmite u obzir da su **objekti ove klase jedinstveno određeni sa sljedeća 3 atributa**: `countyId`, `place` i `id`.



2. (Ponavljanje datoteka) U metodi `main` klase `Main` dohvatite ZIP arhivu s rezultatima izbora za grad Zagreb (oznaka županije `countyId = 21`) s poveznice https://gitlab.tel.fer.hr/ZUP_21.zip te ju pohranite na čvrsti disk pod istim imenom `ZUP_21.zip`.
3. U metodi `main` klase `Main` otvorite InputStream prve datoteke u ZIP arhivi te stvorite novi objekt tipa `jxl.Workbook` pozivanjem statičke metode `getWorkbook(InputStream is)` ove klase.
4. Napravite klasu `PollingResults` koja ima dva atributa tipa `Map<PollingStation, Map<String, Integer>>`. Prvi od njih se naziva `mayorResults` i u njemu će biti pohranjeni rezultati izbora za gradonačelnika, a drugi se zove `assemblyResults` i u njemu će biti pohranjeni rezultati izbora za gradsku skupštinu. Napravite jedan jedini konstruktor `PollingResults(Workbook workbook)` koji kao argument prima objekt tipa `Workbook` koji predstavlja XLS dokument s rezultatima izbora. Svaki XLS dokument se sastoji od više listova (*sheets*), a do pojedinog lista se dolazi pozivom metode `getSheet(int index)` klase

`Workbook`. U prvom listu dohvaćenog XLS dokumenta se nalaze rezultati izbora za gradsku skupštinu, a u drugom listu se nalaze rezultati izbora za gradonačelnika. U ovoj klasi napišite predložak privatne statičke metode `fillResults(Sheet sheet)` koja kao rezultat vraća rezultate izbora u obliku mape tipa `Map<PollingStation, Map<String, Integer>>`. Ovu metodu pozovite dva puta u konstruktoru ove klase za postavljanje vrijednosti njezinih atributa. Na kraju napravite *gettere* za oba atributa.

5. Dovršite kod privatne statičke metode `fillResults(Sheet sheet)`. Do broja redaka i stupaca jednog lista tipa `Sheet` se dolazi pozivom njegove metode `getRows()`, odnosno `getColumns()`. Sadržaj jedne ćelije lista se dohvaća pozivom metode `getCell(int columnIndex, int rowIndex).getContents()` nad objektom tipa `Sheet`.
   Otvorite i proučite izgled listova XLS dokumenta. U prvom retku lista se od 14. stupca (`columnIndex >= 13`, `rowIndex=0`) nalaze imena koalicija za gradsku skupštinu, odnosno imena kandidata za gradonačelnika.
   U preostalim recima lista (`rowIndex > 0`) se u prvih 8 stupaca (`columnIndex < 8`) nalaze podaci o biračkom mjestu u točno onom redoslijedu kojeg prihvaća konstruktor klase `PollingStation` koja predstavlja biračko mjesto, a od 14. stupca (`columnIndex >= 13`) se nalaze podaci o glasovima koje je skupila određena koalicija, odnosno kandidat. Za svaki redak u listu dodajte jedan zapis u povratnu mapu na način da ključ predstavlja biračko mjesto, a vrijednost je nova mapa tipa `Map<String, Integer>` u kojoj je ključ ime koalicije, odnosno kandidata, a vrijednost broj skupljenih glasova. U `main` metodi klase `Main` ispišite jedan od atributa klase `PollingResults`.
6. U metodi `main` klase `Main` za neku koaliciju ili kandidata, ispišite 10 id-jeva biračkih mjesta na kojima je osvojio najveći broj glasova u odnosu na ostala biračka mjesta. Pri tome to ostvarite **u jednoj liniji programskog koda isključivo korištenjem kolekcijskih tokova i lambda izraza**. Za modeliranje parova ključ-vrijednost iskoristite klasu `java.util.AbstractMap.SimpleEntry`. Primijetite da u ispisanim biračkim mjestima ima onih s istim brojem glasova. Što bi bilo drugačije da smo za sortiranje koristili isti komparator s kolekcijom `TreeSet`?
7. (**Napredan zadatak!!!**) U metodi `main` klase `Main` za sve koalicije ili kandidate ispišite biračko mjesto na kojem su ostvarili svoj najveći postotak (na dvije decimale). Pri tome to **ostvarite isključivo korištenjem kolekcijskih tokova i lambda izraza**.
8. Napravite statičku metodu `Map<String, Integer> getCandidateVotes(Map<PollingStation, Map<String, Integer>> results)` u klasi `Main` koja prima mapu s rezultatima za skupštinu ili gradonačelnika, a vraća ukupni broj glasova koji je osvojila određena koalicija ili kandidat. Ovu metodu implementirajte na način da se povratna mapa tipa `Map<String, Integer>` **puni pozivom podrazumijevane (*default*) metode `merge`** iz sučelja `Map`. Pozovite ovu metodu i ispišite njen rezultat.
9. Primijetite da prethodno dobiveni rezultati koje je ostvarila određena koalicija ili kandidat nisu sortirani po broju ostvarenih glasova. Stoga u metodi `main` klase `Main` sortirajte parove mape dobivene u prethodnom zadatku po broju dobivenih glasova u listu tipa `List<Entry<String, Integer>>`. Sortiranje **ostvarite u jednoj liniji programskog koda isključivo korištenjem kolekcijskih tokova i lambda izraza** te na kraju ispišite rezultat.
10. U metodi `main` klase `Main` ispišite kandidate i postotke glasova koje su osvojili (na dvije decimale). Pri tome to ostvarite **u dvije linije programskog koda isključivo korištenjem kolekcijskih tokova i lambda izraza**. U prvoj liniji izračunajte ukupan broj glasova svih koalicija ili kandidate zajedno te ovu brojku iskoristite u drugom lambda izrazu prilikom računanja postotka pojedine koalicije ili kandidata.
11. Primijetite da neka biračka mjesta imaju isti naziv (*title*). U klasi `Main` napravite statičku metodu `List<Entry<String, Integer>> getTitlesSortedPerVotes(Map<PollingStation, Map<String, Integer>> results)` koja će sumirati glasove svih koalicija ili kandidata po biračkim mjestima s istim nazivom te vratiti listu sortiranih parova (po sumiranom broju glasova) kod kojih je ključ naziv biračkog mjesta, a vrijednost sumirani broj glasova. Prilikom implementacije ove metode koristite pomoćnu mapu `TreeMap<String, Integer>` koju ćete **puniti pozivom podrazumijevane (*default*) metode `compute` iz sučelja `Map` te lambda izraze što je više moguće**. Pozovite ovu metodu i ispišite njen rezultat.
12. Implementirajte prethodnu metodu samo s kolekcijskim tokovima bez korištenja pomoćne mape.

Rješenja zadataka: https://github.com/FER-OOP/Lectures/tree/master/Exercises/Homework-13