# Лабораторная работа №2: Немного об ОКТМО

Тема: чтение текстовых данных, регулярные выражения, стандартные коллекции.

Работа заключается в чтении массива данных на примере классификатора ОКТМО (список муниципальных образований и населённых пунктов России) и анализе его содержимого без использования баз данных и специализированных библиотек.

Даны текстовые файлы с содержимым государственного классификатора ОКТМО (Общероссийского классификатора территорий муниципальных образований)  
tom\*.txt (текст вставлен из .doc-файлов в официальных архивах)

1. Просмотрите файлы в текстовом редакторе. Найдите регионы, муниципальные районы, поселения (сельсоветы и др.), городские округа и входящие в них населенные пункты. Обратите внимание на коды ОКТМО (часть цифр отвечает за регион, часть за район/городской округ, дальше – поселения и НП.

Короткие коды у муниципальных образований (их пока нужно игнорировать), длинные – у населённых пунктов.

1. Создайте проект Java, добавьте пустые классы OktmoMain (c функцией main), OktmoReader, OktmoData, OktmoAnalyzer.
   1. OktmoData будет хранить всю считанную информацию (списки) и содержать методы обращения к ним
   2. OktmoReader считывает текстовые файлы .csv и добавляет их содержимое в разобранном виде в OktmoData
   3. OktmoAnalyzer просматривает объекты в OktmoData и проводит анализ.
2. Чтение списка НП (это проще, т.к. пока нет иерархии)
   1. Добавьте класс Place (населенный пункт) с полями code (long) и name, status (пока текст)  
      (code= 80 602 418 101 name=Зелёный Клин status=деревня)
   2. Добавьте список объектов Place к классу OktmoData (изначально пустой)
   3. Добавьте метод public void readPlaces(String fileName, OktmoData data) к классу OktmoReader
      1. sample1.txt – пример построчного чтения файла
      2. Для начала примените метод .split к каждой считанной строке (передаём символ-разделитель, получаем массив строк), напечатайте содержимое. Разделитель в txt – символ табуляции \t.
      3. Сконвертируйте первый элемент в число ( Long.parseLong )  
         (пробелы можно убрать методом replace(" ","") в классе String )
      4. Вызывайте для каждой считанной строки конструктор Place и заносите созданных объект в OktmoData (метод addPlace)
      5. Напечатайте содержимое OktmoData после чтения.  
         (\*) (\* здесь и далее = "по желанию")  
         Другие примеры разбора строки:  
         <http://stackoverflow.com/questions/5965767/performance-of-stringtokenizer-class-vs-split-method-in-java>   
         Попробуйте использовать только indexOf и замерьте время чтения файлa двумя способами.
   4. Исправьте ”недоработки” при чтении: отделите статус «д» в отдельное поле (рекомендую методы строки indexOf и substring или поиск первой заглавной буквы по коду символа), пропускайте строки, начинающиеся с “Населенные пункты ….” (метод startsWith у строки).
   5. Напишите тест JUnit для чтения (проверяйте кол-во НП и навание/статус/код 10-го и последнего в списке НП)
   6. Составьте множество статусов (д г пгт и т. д.) – HashSet<String> или TreeSet<String> - поле allStatuses в классе OktmoData . Добавляйте в него элементы при чтении.

Напечатайте после чтения всего объёма данных. В чём отличие HashSet и TreeSet?

* 1. Отсортируйте список населённых пунктов по имени
     1. Cоздайте отдельное поле sortedPlaces, cкопируйте туда весь ArrayList.
     2. Вызовите стандартную процедуру сортировки Collections.sort. Так как объекты Place изначально не сравнимы друг с другом, придётся написать класс-компаратор  
        <http://echuprina.blogspot.ru/2012/02/comparable-comparator.html>  
        <http://www.leveluplunch.com/java/tutorials/007-sort-arraylist-stream-of-objects-in-java8/>
  2. С помощью регулярных выражений найдите все НП, название которых содержит меньше 6 букв и заканчиваются на -ово  
     <http://www.quizful.net/post/Java-RegExp>  
     <http://javagu.ru/portal/dt?last=false&provider=javaguru&ArticleId=GURU_ARTICLE_64538&SecID=GURU_SECTION_63111>
  3. Создайте альтернативную процедуру чтения списка НП без split и indexOf с использованием ровно одного регулярного выражения с группами  
     <http://initialize.ru/regular-expressions-in-java>
  4. Создайте тест на чтение файла вторым способом (сравните время работы) и тест на проверку одинаковости данных (используйте equals для коллекций, написав свой equals для класса Place)
  5. Найдите населённые пункты, с названиями, которые начинаются и заканчиваются га одну и ту же согласную букву ( использовать [^…] , () , \\1, а также [Pattern.CASE\_INSENSITIVE](http://download.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html#CASE_INSENSITIVE) или (?i) в регулярном выражении +[Pattern.UNICODE\_CASE](http://download.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html" \l "CASE_INSENSITIVE) для русских букв)