Задание №1. (10 баллов) Контейнеры, IO потоки, классы Object и String.

Написать программу, которая будет принимать в качестве аргумента имя текстового файла, и выводить CSV файл (http://ru.wikipedia.org/wiki/CSV) с колонками:

1. Слово.
2. Частота.
3. Частота (в %).

CSV файл должен быть упорядочен по убыванию частоты, то есть самые частые слова должны идти в начале. Разделителями считать все символы кроме букв и цифр.

Методические указания:

- Использовать класс *java.lang.StringBuilder* для построения слов.
- Для чтения из файла удобно использовать: *java.io.InputStreamReader*, например:

```
Reader reader = null;
try
{
        reader = new InputStreamReader(new FileInputStream("FILE NAME"));
        //read the data here
}
catch (IOException e)
{
        System.err.println("Error while reading file: " + e.getLocalizedMessage());
}
finally
{
        if (null != reader)
        {
                try
                {
                        reader.close();
                }
                catch (IOException e)
                {
                        e.printStackTrace(System.err);
                }
        }
}
```

- Для определения класса символа использовать метод *Character.isLetterOrDigit*. Для хранения статистики в памяти можно использовать одну из реализаций интерфейса *java.util.Set*, который должен будет хранить объекты специального класса. Данный класс должен содержать слово и счётчик. В случае использования *java.util.HashSet* класс также должен реализовать методы *equals*, *hashCode*.

Теоретические сведения:

Контейнеры стандартной библиотеки расположены в пакете *java.util*. IO классы (потоки ввода-вывода) располагаются в пакете *java.io.*

Основные интерфейсы:

1. *Set* – множество без дубликатов и без доступа по индексу.
2. *Map* – множество пар ключ-значение, где ключи не повторяются.

Их основные реализации:

1. *HashMap, HashSet* — реализации на основе функции hashCode.
2. *TreeMap, TreeSet* – реализация на основе бинарного дерева. Ключи (элементы) должны реализовывать интерфейс *Comporable*, иначе необходимо передавать в контейнер при его создании объект, реализующий интерфейс *Comparator*. Хранимые в данных контейнерах данные упорядочены. Лучшее время поиска, но большее накладные расходы на вставку, чем на основе функции *hashCode*.