Practice6

Везде в коде, при реализации функциональных интерфейсов использовать лямбда выражения или ссылки на методы.

Помните, что контейнеры Hash/TreeSet, Hash/TreeMap не контролируют потенциальное появление дубликатов при изменении никак не отслеживают изменения, то есть, если изменение состояния ключа какой ни будь пары может произойти, то контейнер это никак не отслеживает.

Задание 6

Пакет: ua.nure.name.Practice6
Класс: **Part6** (принимает параметры командной строки)
Входную информацию загружать из файла который лежит в корне проекта, назвать **part6.txt**

Реализовать консольное приложение, (**ua.nure.name.Practice6.Part6**) которое анализирует текст. Формат входных параметров командной строки для приложения (программа должна понимать как **короткие**, так и **длинные** опции):

-i (или --input) путь к входному файлу;
-t (или --task) наименование подзадачи.

Всего параметров командной строки 4, т.е. две пары <**имя_опции значение_опции**>. Порядок следования опций с их значениями прооизвольный.
Примеры параметров командной строки для запуска приложения

```
-i input.txt -t frequency
 --input input.txt --task length
```

В методе **Demo.main** смоделировать пеердачу параметров командной строки в метод **Part6.main**, пример:

```
System.out.println("~~~~~~~~~~~~~Part6");
Part6.main(new String[] {"--input", "part6.txt", "--task", "length"});
Part6.main(new String[] {"--input", "part6.txt", "--task", "frequency"});
Part6.main(new String[] {"--input", "part6.txt", "--task", "duplicates"});
```

Подзадач всего три **frequency**, **length**, **duplicates**.

Под словами понимать непрерывную последовательность букв **латинского** алфавита.

Part 1 (frequency)

Во входном файле найти три слова, которые встречаются наиболее часто (**при совпадении частот – те, которые встречаются раньше**), и распечатать их отсортированными по алфавиту в обратном порядке в формате: слово ==> частота

Пример вывода

```
panda ==> 15
ezhik ==> 20
apple ==> 19
```

Part 2 (length)

Во входном файле найти три самых длинных слова и распечатать их в формате: слово ==> количество букв в слове. Список должен быть отсортирован по убыванию количества букв в слове.

Если у двух слов количество букв совпадает, то слово, которое было в исходном файле раньше, должно в результирующем списке находиться также раньше.

Пример вывода

```
anesthetist ==> 11
kitchen ==> 7
bird ==> 4
```

Part 3 (duplicates)

Во входном файле найти первые три слова, которые имеют дубликаты, и напечатать их инверсию в верхнем регистре.

Пример вывода

ADNAP
TAC
ENIGREBUA

Задание 7

Пакет: ua.nure.name.Practice6
Класс: **Part7**, **Range**

Написать класс **Range**, который бы представлял собой промежуток чисел **[n, m]**, где **n<m**. Класс должен реализовывать интерфейс **Iterable**. Итератор реализовать таким образом, чтобы он проходил он начала до конца промежутка. В конструктор передавать дополнительный параметр **reverse**.

Класс **Range** должен иметь два конструктора:

```
public Range(int n, int m) { ... }
public Range(int n, int m, boolean reverse) { ... }
```

Пример.

```
Range range = new Range(3, 10);
for (Integer el : range) {
        System.out.printf("%d ", el);
}
System.out.println();
// result: 3 4 5 6 7 8 9 10

range = new Range(3, 10, true);
for (Integer el : range) {
        System.out.printf("%d ", el);
}
System.out.println();
// result: 10 9 8 7 6 5 4 3
```

Продемонстрировать работу приложения (**Part7.main**).