Для выполнения всех дополнительных заданий самостоятельно необходимо будет изучить:

1. Составление диаграмм классов в UML.
2. Stream API
3. Возможно регулярные выражения для работы с классами Matcher и Pattern.

**Задание**

Написать парсер логов (программа, которая считает из файла данные и преобразует в объекты программы)

Лог файл имеет следующий формат:

ip username date event status

Где:

ip - ip адрес с которого пользователь произвел событие.

user - имя пользователя (одно или несколько слов разделенные пробелами).

date - дата события в формате day.month.year hour:minute:second.

event - одно из событий:

* LOGIN - пользователь залогинился,
* DOWNLOAD\_PLUGIN - пользователь скачал плагин,
* WRITE\_MESSAGE - пользователь отправил сообщение,
* SOLVE\_TASK - пользователь попытался решить задачу,
* DONE\_TASK - пользователь решил задачу.

Для событий SOLVE\_TASK и DONE\_TASK существует дополнительный параметр,

который указывается через пробел, это номер задачи.

status - статус:

* OK - событие выполнилось успешно,
* FAILED - событие не выполнилось,
* ERROR - произошла ошибка.

Все ключи разделены между сабой табуляцией.

Пример строки из лог файла:

"146.34.15.5 Eduard Petrovich Morozko 05.01.2021 20:22:55 DONE\_TASK 48 FAILED".

Класс, который будет отвечать за парсинг логов называется LogParser.

1. Добавь в класс LogParser конструктор с параметром Path logDir, где logDir - директория с логами (логов может быть несколько, все они должны иметь расширение log).

LogParser должен распарсить кажду строчку из файлов с логами в ArrayList<Log>.

Сигнатуру класса Log необходимо составить самостоятельно в соответствии с заданием.

После чего записать в отдельный файл только тех студентов, у которых есть выполненные задания.

Возможно будет полезным классы:

java.util.regex.Matcher;  
java.util.regex.Pattern;

**Дополнительное задание 1.**

Составить диаграмму классов для данного задания. Например в Star UML.

**Дополнительное задание 2.**

Добавить логирование там, где вы считаете это нужно.

**Дополнительное задание 3.**

Реализовать интерфейс

public interface IPQuery {  
 int getNumberOfUniqueIPs(Date after, Date before);  
  
 Set<String> getUniqueIPs(Date after, Date before);  
  
 Set<String> getIPsForUser(String user, Date after, Date before);  
  
 Set<String> getIPsForEvent(Event event, Date after, Date before);  
  
 Set<String> getIPsForStatus(Status status, Date after, Date before);  
}

1.1. Метод getNumberOfUniqueIPs(Date after, Date before) должен возвращать количество уникальных IP адресов за выбранный период. Здесь и далее, если в методе есть параметры Date after и Date before, то нужно возвратить данные касающиеся только данного периода (включая даты after и before).

Если параметр after равен null, то нужно обработать все записи, у которых дата меньше или равна before.

Если параметр before равен null, то нужно обработать все записи, у которых дата больше или равна after.

Если и after, и before равны null, то нужно обработать абсолютно все записи (без фильтрации по дате).

1.2. Метод getUniqueIPs() должен возвращать множество, содержащее все не повторяющиеся IP. Тип в котором будем хранить IP будет String.

1.3. Метод getIPsForUser() должен возвращать IP, с которых работал переданный пользователь.

1.4. Метод getIPsForEvent() должен возвращать IP, с которых было произведено переданное событие.

1.5. Метод getIPsForStatus() должен возвращать IP, события с которых закончилось переданным статусом.

**Дополнительное задание 4**

**Реализовать все с использованием Stream API.**