**Задание № 3**

**Тема: Java 8**

**Условие:**

1. Создайте ветку task-3 и переключитесь на нее
2. Создать папку task-3 Данная папка должна явятся корнем maven приложения (те, в этой папке должен находиться pom.xml) с artifactId – task-3 groupId – com.epam.winter-java-lab
3. Необходимо разработать приложение, которое будет обрабатывать данные о кредитах в банке.
4. В рабочей директории будут содержаться файлы (в обязательном порядке): db.json, settings.json (далее будет пояснение) – если их нет, вывести в консоль сообщение об ошибке!
   1. db.json – наша база данных, содержащая информацию о пользователях, кредитах, погашениях, акциях, и тд. Данные, которые хранит непосредственно банк. (далее будет структура)
   2. settings.json – файл для настройки работы нашего приложения, тут будет содержаться то, что и как должно обработать наше приложение (далее будет структура)
   3. Рабочая директория – папка c названием data, лежащая на уровне с jar (а не та, где расположены .java/.class файлы)
5. Помимо обязательных файлов в рабочей директории могут содержаться и другие файлы (а могут и не содержаться). Нас будут интересовать файлы с названием “db\_[тут любая строка состоящая только из букв и цифр].json”. Данные, хранящиеся в филиалах банка. Другие файлы даже не рассматриваем! (далее будет структура)
6. При запуске приложения, мы должны просканировать всю директорию на наличие файлов формата “db\_\*.json” из предыдущего пункта, а после, построчно (не каждую строку – а каждую запись), перенести в файл db.json, попутно удаляя запись (не файл) из файлов “db\_\*.json” (на случай того, чтобы в следующий раз при запуске приложения, у нас не было дубликатов записей)
7. После перезаписи всех данных в файл db.json – можно переходить к основному функционалу
8. Файл db.json будет иметь такой формат:

{

users: [

{id: 0, name: "Alex", secondName: "Smith", sex: "MALE", birthday: "1984-06-24"},

...

},

credits: [

{id: 0, userId: 0, date: "2011-01-20", period: "DAY", money: 75000, rate: 10.6},

{id: 1, userId: 0, date: "2017-04-15", period: "WEEK", money: 42100, rate: 10.6},

{id: 2, userId: 0, date: "2016-02-01", period: "MONTH", money: 35400, rate: 10.6},

{id: 3, userId: 0, date: "2013-05-07", period: "YEAR", money: 41700, rate: 10.6},

...

],

transactions: [

{id: 0, date: "2007-04-12", creditId: 6, money: 1400},

...

]

}

Где :

1. users – массив со всеми пользователями, пользователя могут быть описаны ТОЛЬКО в этом файле
   1. id – его уникальный идентификатор
   2. name – имя (может быть только в одно слово)
   3. secondName – фамилия (может быть только в одно слово)
   4. sex – пол (может быть только MALE/FEMALE/ANY)
2. credits – массив со всеми кредитами
   1. id - его уникальный идентификатор
   2. userId - уникальный идентификатор клиента, взявшего кредит
   3. date – дата взятия кредита
   4. period – период, через который будет начислена ставка (может быть только DAY/WEEK/MONTH/YEAR, не забываем про високосные года и последние дни месяца! отсчет начинается от date)
   5. money – сколько же взяли денег
   6. rate – ставка, она указывается в процентах и рассчитывается от ТЕКУЩЕГО долга, те, если вам нужно выплатить еще 100 (Br) – то ваш процент считается от 100
3. transactions – массив, содержащий вносимые клиентами суммы, для погашения кредита
   1. id - уникальный идентификатор
   2. date – дата (даты, которые будут позже кредита, если каким-то образом, существует транзакция, которая была сделала по кредиту, который еще не взяли – такие транзакции не рассматривать)
   3. creditId – id кредита, по которому погашается долг (на одного пользователя можно взять несколько кредитов)
   4. money – количество денег
4. Файл settings.json будет иметь такой формат:

{

settings: {

dateFrom: 2010-04-06,

dateTo: 2015-07-06,

showFor: {

type: “ID”

users: [1, 2, 3, 4, 5]

},

sortBy: “NAME”,

useDepartments: [

“gomel”,

“minsk”

]

}

}

Где:

* 1. dateFrom – время с которого нужно начинать отсчет. Но, если для пользователя, запись по кредиту не попадает во временные рамки (даже, если у него есть транзакции в этот период) – для него расчетов не делаем, но, отображаем в финальных результатах (смотрите далее) (может отсутствовать – означает, что нужно брать все записи с самого начала)
  2. dateTo – время до которого нужно рассчитать долги (может отсутствовать – означает, что нужно рассчитать все записи до последней имеющейся транзакции)
  3. showFor – отобразить для (может быть отсутствовать – тогда для всех):
     1. type – по id, либо по ИМЯ(пробел)ФАМИЛИЯ (может быть ID/NAME)
     2. users – в зависимости от type, либо список id, либо список ИМЯ(пробел)ФАМИЛИЯ
  4. sortBy – отсортировать в порядке возрастания (может быть NAME/DEBT/AGE)
  5. useDepartments – определяет, какие файлы формата “db\_\*.json” нужно использовать в программе (пункт 2), соответственно если массив пуст – никаких, если отсутствует – все что есть в директории (соответствующего формата). Пример для “minsk” - “db\_minsk.json”
  6. Если какого-нибудь из параметров не будет в файле конфигураций (кроме тех, у которых явно указано, что они могут отсутствовать) – ошибка, с названием параметра, которого не хватает!

1. Файл db\_\*.json будет иметь такой формат:

{

transactions: [

{id: 0, date:"2018-06-07", creditId:5, money: 1400},

...

]

}

Тоже самое, что и в db.json

1. Наш заказчик – очень скупой, поэтому выделил нам очень мало оперативной памяти, а данных может быть очень много, поэтому нам нужно разработать DAO, позволяющее работать с файлами, не держа все постоянно в памяти (это намного дольше, но банку важнее собственное благосостояние, нежели время ожидания клиентов)
2. Использовать JAVA 8!
3. Для работы с json – рекомендуется использовать gson (но не обязательно)
4. Результатом работы приложения, по итогу должно быть:
   1. Заполненный файл db.json (данными из файлов “db\_\*.json”)
   2. Опустошенные файлы “db\_\*.json” – который учувствовали в обработке данных (не удаленные или пустые, а опустошённые!, те в них будут содержаться пустые json объекты)
   3. Отформатированный (по колонкам, с названием колонки) список кредитов, выведенный в КОНСОЛЬ, в котором должно быть:
      * 1. Id кредита
        2. Id пользователя
        3. Имя(пробел)Фамилия пользователя
        4. Количество транзакций (только обработанных, а не всех)
        5. Размер долга (0, если расплатился)
        6. Период кредита
        7. Статус (IN PROGRESS / DONE)
        8. Если статус DONE – добавить дату погашения кредита (прим. DONE – (2018-05-05))
        9. Список должен быть отсортирован по параметру, переданному через файл settings.json
5. Если кредит не попал во временные рамки, но, пользователя необходимо отобразить – во всех колонках (кроме имени) - null
6. Изменить README.md добавив в него строку “Задание #3: Банк – Date: ” + 2 июля 2019 04:07 (дата, месяц, год, час, минута) (естественно, тут ваша дата)
7. Закомитать ваш код в ветку task-3 и запушить на удаленный сервер.
8. Создать pull request