Для всех задач необходимо дополнительно создавать класс Application, который будет содержать статический метод main.

**Задача 1**

Во входном файле input.txt содержатся данные о некотором объекте. Необходимо считать данные, которые представлены в формате

<ключ>=<значение>

где значение может быть любым типом. Необходимо реализовать класс FileHelper, содержащий методы readFile(File file), saveFile(File file, <Коллекция>). Первый метод должен считывать файл и возвращать некоторую коллекцию, содержащую данные. Второй файл должен сохранять структуру в формате json. Полученный Json должен быть валиден. Лишние пробелы необходимо удалить. В случае, если значение не задано, в полученный json оно должно быть записано как пустой массив.

В методе main считать файл input.txt, провести необходимые операции и сохранить результат в формате json в файл output.txt.

|  |  |
| --- | --- |
| input.txt | output.txt |
| address=г.Москва  clientId=65737876  value=98.3  isFallback=true  unknownValue=null  name=Дипломатический центр  notifications= | {  "unknownValue": null,  "address": "г.Москва",  "clientId": 65737876,  "name": "Дипломатический центр",  "isFallback": true,  "value": 98.3,  "notifications": []  } |

**Задача 2**

Создать файл application.properties, который будет содержать переменные

output.file=<путь к файлу с выходными данными>

lines.amount=<количество строк>

lines.max.length=<максимальная длина строки>

lines.min.length=<минимальная длина строки>

lines.pattern=<паттерн формирования строки>

lines.message=<сообщение о создании строки (String.format)>

Создать класс Config, содержащий приватные методы getBooleanProperty(String key, Boolean defaultValue), getIntegerProperty(String key, Integer defaultValue), getProperty(String key, String defaultValue). Также добавить поля OUTPUT\_FILE, LINES\_AMOUNT, и прочие, значения которых будут считываться из .properties-файлов во время запуска программы.

Создать класс FileHelper с методом write(List<String> content), который записывает content в OUTPUT\_FILE файл. Создать класс StringHelper с методом generate(), генерирующим строку от LINES\_MIN\_LENGTH до LINES\_MAX\_LENGTH случайных символов из множество LINES\_PATTERN. Метод также должен выводить сгенерированную строку на экран в формате, указанном в LINES\_MESSAGE.

В методе main сгенерировать LINES\_AMOUNT строк и записать их в OUTPUT\_FILE файл.

**Задача 3**

В данной задаче необходимо реализовать класс-генератор файлов FileGen. Класс должен предусматривать возможность генерации нескольких файлов, с заданным размером и расширением.

В конструкторе класса необходимо реализовать передача количества файлов, их размеры, их расширения. Реализовать перегрузку конструктора, позволяющую передавать только часть параметров или не передавать их вовсе.

Реализовать метод initValues(), который в случае, если какой-то из параметров не передан, устанавливает его заданным параметром из application.properties.

Реализовать метод generate(), который генерирует файлы в директорию output и возвращает список List<Files> сгенерированных файлов

Создать файл application.properties, описывающий параметры генерации файлов:

filegen.amount=3

filegen.minsize=100000

filegen.maxsize=1000000

filegen.extensions=TXT,PDF,DOC,XLS

filegen.outputfolder=\\src\\main\\java\\homework\_solution\\lesson10\\task3\\output (поменять на свою)

Конструкторы должны иметь комбинацию следующих параметров:

|  |  |  |
| --- | --- | --- |
| Тип параметра | Если передан в конструкторе | Если не передан в конструкторе |
| List<Integer> sizes | sizes | random от minsize до maxsize, для каждого из файлов |
| List<String> extensions | extensions | случайный из filegen.extensions |
| int amount | amount | взять из filegen.amount |

Имена для генерируемых файлов должны быть в формате

generatedFile<Случайное число от 100000 до 999999>.<расширение>

В методе main сгенерировать файлы и вывести в консоль информацию о них в формате

<Имя файла>: <Размер файла>

**Задача 4**

Необходимо реализовать функционал сохранения зарплатной ведомости и заявки на выдачу средств из кассы банка.

Создать файл application.properties, содержащий следующие значения:

|  |  |  |
| --- | --- | --- |
| Ключ | Значение | Описание |
| employee.lastname.length | 12 | длина генерируемой части фамилии |
| employee.firstname.length | 12 | длина генерируемой части имени |
| employee.name.pattern | QWERTYUIOPASDFGHJKLZXCVBNMOqwertyuiopasdfghjklzxcvbnm | допустимые символы при генерации фамилии и имени |
| employee.salary.min | 10000.00 | минимальная сумма зп при генерации |
| employee.salary.max | 50000.00 | максимальная сумма зп при генерации |
| passport.series.digits | 4 | длина генерируемой серии паспорта |
| passport.number.digits | 6 | длина генерируемого номера паспорта |
| salary.date.format | dd.MM.yyyy HH:mm:ss | формат даты в зарплатной ведомости |
| salary.employee.count | 5 | количество сотрудников в зарплатной ведомости |
| salary.output.file | src\\main\\....\\salary.txt | Полный путь для файла зарплатной ведомости |
| cash.output.file | src\\main\\....\\cash.txt | Полный путь для файла заявки на выдачу средств из кассы банка |
| cash.nominals | 5000,1000,500,100,50,10,5,2,1,0.5,0.1,0.05,0.01 | Номиналы банкнот и монет, содержащихся в кассе |

Создать интерфейс Generatable<T>, содержащий метод T generate().

Создать интерфейс Save, содержащий метод void save().

Создать класс Passport, имплементирующий интерфейс Generatable. Класс Passport должен содержать поля, описывающие серию и номер, а метод генерации должен инициализировать эти поля случайной строкой, состоящей из чисел. Длина полей должна быть равна passport.series.digits и passport.number.digits соответственно.

Создать класс Employee, описывающий сотрудника. Класс должен содержать поля, описывающие фамилию, имя, паспорт, размер заработной платы сотрудника. Имплементировать интерфейс Generatable, реализовав метод, который генерирует паспорт, фамилию и имя, размер заработной платы. Использовать параметры, определенные в application.properties.

Создать класс SalaryDocument, описывающий зарплатную ведомость. Класс должен содержать список сотрудников и реализовывать методы generate и save, генерирующие и сохраняющие в файл зарплатную ведомость. При генерации зарплатной ведомости она должна получать случайный uuid и помещаться в очередь обработки, время операции – текущее плюс один день. При сохранении в файл также содержимое выводить в консоль. Переопределить метод toString() для формирования в нужном формате и пользоваться им при выводе и записи в файл.

Создать класс CashDocument, описывающий заявку на выдачу средств из кассы банка. Класс должен содержать структуру, описывающую номиналы банкнот и монет и их количество, а также uuid зарплатной ведомости, которую данная заявка обрабатывает. Имплементировать интерфейсы Generatable и Save. Метод generate должен взять из очереди зарплатную ведомость, сгенерировать на ее основе заявку на выдачу. Сумму из зарплатной ведомости необходимо разбить по номиналам. Для разбиения реализовать жадный алгоритм.

Создать класс DocumentQueue, реализующий работу с очередью зарплатных ведомостей. Механизм обработки ведомостей – FIFO.

В методе main протестировать созданное решение.

Форматы файлов и вывода (примеры):

Зарплатная ведомость:

Зарплатная ведомость.  
uuid: ce6dec69-8385-456b-a4fe-7e2598eea5ff  
  
Время операции: 17.09.2020 18:11:53.  
  
Список сотрудников:  
AutoLCiiYhSkGLvDt AutoFykSWgUeDAduC, 1525 423742, 21465,21  
AutoLmcwqdaOSwUgQ AutoFWePVTcZJDUaw, 0151 025061, 20631,54  
AutoLQsoVrxncnEzX AutoFOoIvtivclDMb, 6797 414964, 46029,07  
AutoLUHwSATHqImHo AutoFfSGUODHYvNlO, 5728 502792, 25546,15  
AutoLOCrnapZIpcZG AutoFtNwLnZLbULkp, 3145 895102, 28040,93  
Общая сумма: 141712,90

Заявка на выдачу средств из кассы

Заявка на выдачу средств из кассы.  
uuid: 8952a0a6-eb89-4d4d-b081-e138d9002bae  
  
Обоснование: Зарплатная ведомость ce6dec69-8385-456b-a4fe-7e2598eea5ff  
  
Время операции: 17.09.2020 18:11:53.  
  
Сумма операции: 141712,90  
0,10 - 4 шт.  
0,50 - 1 шт.  
2,00 - 1 шт.  
10,00 - 1 шт.  
100,00 - 2 шт.  
500,00 - 1 шт.  
1000,00 - 1 шт.  
5000,00 - 28 шт.