Коментари към решението на задачата

Pair of employees who have worked together

Create an application that identifies the pair of employees who have worked together on common projects for the longest period of time…

Данните се прочитат от файла в списък от прости обекти от следния тип:

```
class InputData {
      private Integer projId;
      private Integer empId;
      private LocalDate fromDate;
      private LocalDate toDate;
...
}
```

Тъй като всички обекти са immutable, класът има само конструктор с параметри и get-методи.

Решението на задачата се съдържа в класа DataProcessor

Първата стъпка от алгоритъма е списъка с обекти InputData да се прехвърли в Map с ключ empId и стойност List<InputData>, приема се, че един служител може да "влиза"и "излиза" от даден проект по няколко пъти.

```
Map<Integer, List<InputData>> periodsPerEmployee;  // key: empId
```

След като сме разделили данните по този начин, решението на задачата се свежда в двукратно итериране по ключовете на този map и сравняване на периодите за работа по един и същи проект. Итерирането се провежда по различни ключове (empId), като за да се избегне повтарянето на резултати всяко empId се сравнява само с по-големи от него. Данните се събират в map с ключ двойката (empId1, empId2) (текущите стойности на двете итерации) и стойност списък от проект-период, в които са работили заедно (projId, period):

```
Map<OrderedPair, List<ProjectPeriod>> coworkingPeriods;
```

За този map се използват два помощни класа, които се намират в същия файл на класа DataProcessor (понеже са от локално значение и няма смисъл да са public):

- класът OrderedPair се използва за ключ на map-а, затова имплементира Comparable. В конструктора се осигурява подредеността на двойката числа.

```
class OrderedPair implements Comparable<OrderedPair> {
      Integer empId1, empId2;
      ...
}
```

- за стойности се използва класа:

```
class ProjectPeriod {
      Integer projId;
      Integer days;
      …
}
```

И двата класа са от локално значение, затова за по-кратки записи съм си позволил да използвам директно полетата, без get методи.

Изчисляването на периодите на съвместна работа по даден проект (полето days) се прави по познатия алгоритъм: от всяка двойка периоди се взема по-голямата от началните дати и по-малката от крайните и се смятат дните между тях. Ако резултатът не е отрицателен, има общ период. Към него се добавя един ден (предполагам, че крайната дата се включва в периода на работа по проекта).

След като сме получили Map<OrderedPair, List<ProjectPeriod>> coworkingPeriods, вече можем директно да получим искания резултат за максимален брой дни съвместна работа: в ключа имаме неповтарящи се двойки empId и към тях – списък с проектите и периодите, когато са работили заедно. Трябва само да съберем периодите от всеки списък и да намерим максималния резултат. Това става в метода

```
public List<CoworkerPeriod> getResult()
```

За пренос на данните се използва класът:

```
public class CoworkerPeriod {
      private Integer empId1;
      private Integer empId2;
      private Integer days;
...
}
```

В полето days се намира сумата от всички периоди на обща работа. Не се отчитат почивни дни (събота, неделя, празници). Също не се проверява за логически грешки във входните данни (отрицателни периоди или застъпващи се периоди на работа на един човек по даден проект).
Методът getResult() не връща само един резултат (най-дългия период на обща работа), а списък, защото може да се окаже, че няколко двойки служители са работили по еднакъв (максимален) брой дни заедно. Съответно, ако няма никакви резултати, списъкът е празен.

Аналогично, методът

```
public List<TableData> getTableData ()
```

връща данни, подходящи за въвеждане в таблицата, която трябва да се покаже:

```
class TableData implements Comparable<TableData> {
      private Integer empId1;
      private Integer empId2;
      private Integer projId;
      private Integer days;
...
}
```

Всичко това е в основния клас DataProcessor.

Прочитане на входните данни от CSV файл

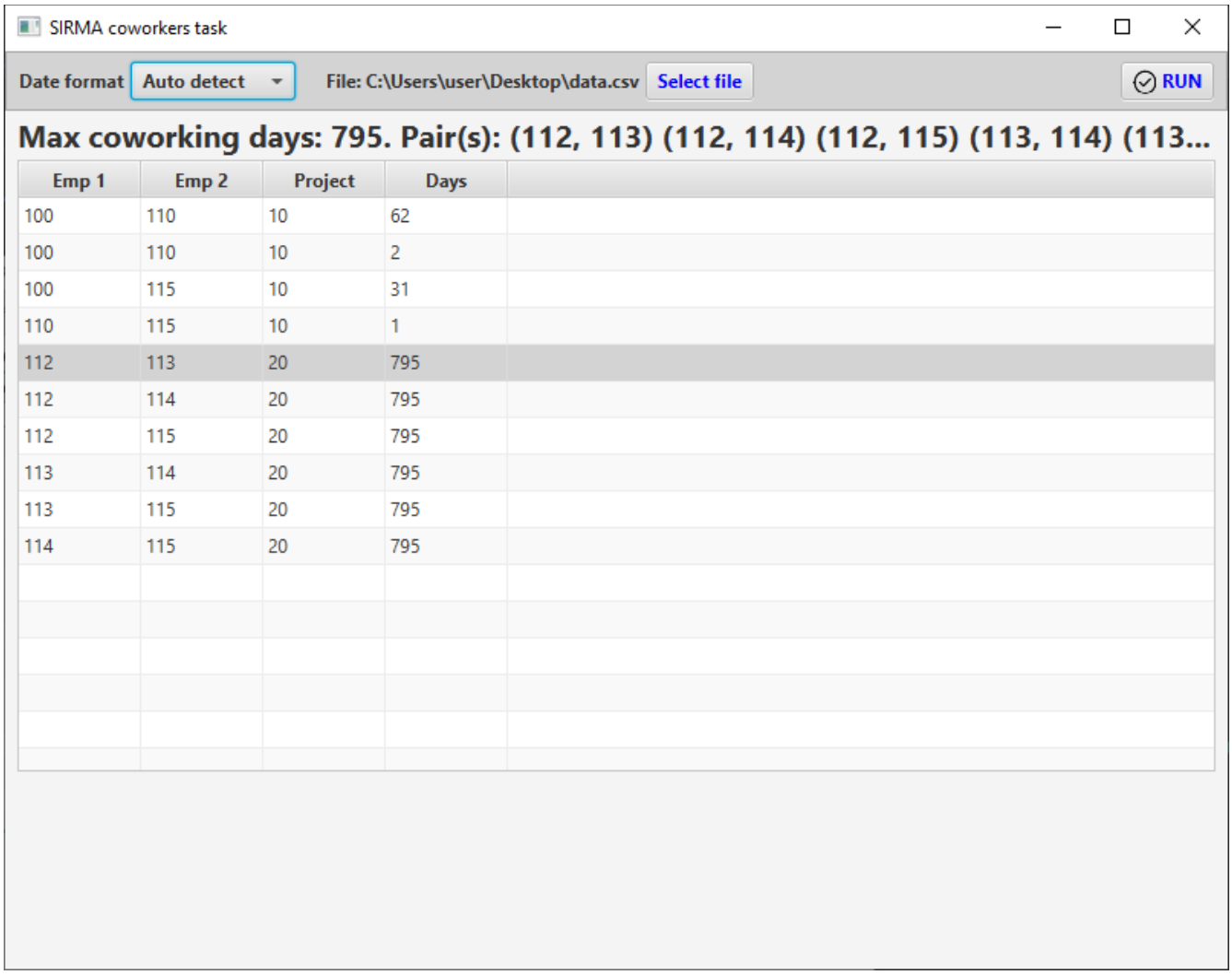
За прочитане на данните се грижи класът DataLoader.
Данните се четат от текстов CSV файл. На всеки ред (по задание) има четири стойности: empId, projectId, dateFrom и dateTo. За последната данна, освен, че може да бъде валидна дата има следните допълнителни правила: тя може да бъде стринговете "NULL" или "null", да бъде празен стринг или да отсъства изобщо. В тези случаи dateTo се приема за текущата дата. Празните редове във входния файл се пренебрегват.
DataLoader получава като аргументи в конструктора име на файл и формат на датата. Поддържат се три формата "yyyy-MM-dd", "MM/dd/yyyy" и "dd.MM.yyyy". За генерирането на съответните DateTimeFormatter-и се грижи специален клас, DateFormatFactory, който е реализиран като Singleton само със static методи и полета. Същият клас има опция за автоматично разпознаване на формата на датата по примерен стринг (използват се Pattern matchers). Ако DataLoader-ът не получи като аргумент валиден формат на датата, той

при прочитане на първия ред от файла извиква метода detectFormatter на DateFormatFactory и после продължава с форматера, който е получил.
DataLoader може да "гръмне" по различни начини (липсващ файл, грешен формат на данните...), Във всички тези случаи се хвърля изключение, съдържащо информация за реда и полето, в което е възникнала грешката.

MVC

За такова малко приложение (една команда) използването на MVC не е много оправдано, но все пак, за да се раздели изчислителната работа от графичния интерфейс, съм сложил един малък Controller, който обработва командата "RUN" и пренася данните от изчислителните класове към визуалните.

Потребителски интерфейс



Интерфейсът е реализиран на JavaFX. Потребителят избира от падащо меню формата на запис на датите (опцията "Auto detect" до голяма степен обезсмисля този избор, но за целите на задачата...). Второто поле е избор на файл. След като файлът е избран се натиска бутона "RUN" и програмата прочита файла и показва резултатите. Ако възникне грешка, тя се изписва в долния край с червено. Ако всичко е наред, под менюто се изписва резултата (макс. брой дни и двойката/двойките служители), а в таблицата се появяват данните, подредени по първи служител, втори служител, проект и брой дни (в обратен ред).