## Подробно описание на задачите за изпълнение.docx

**Име на изпълнителя на задача 1**

private int [] startHours

private int [] endHours

private int [] availbaleHours

Моля, добави пример за как се попълва масива. Предположенията ми са, че са на ден([0] – ден 1, [1] – ден 2, etc..; за две седмици). Ако е така, значи availableHours става безполезно. Също, как се определя ден 1 от двете седмици (Тази седмица кой ден е Понеделник?). Предполагам е с четност от годината (т.е. на нечетна седмица е 1ден, на четна- 8), но кажи как ти го виждаш.

**availbaleHours се използва за улеснение, иначе може и без него да се мине, да. Ден 1 (индекс 0 на масива) се определя от входния файл, за който има first week и second week, съответно понеделника от втората седмица ще е ден 8 (индекс 7 на масива).**

**Пак не се разбира коя е първа и втора седмица от реалния календар. От файла както си решим ние ще е, там няма грижи.**

**Това няма значение за програмата, това е условност, която се изяснява, когато всеки стажант попълва файла със свободните си часове.**

private boolean [][] availableShifts

Защо е нужен този масив? Ние знаем часовете, в които е свободен и часовете на смените.

**Малко изпреварваме текущата задача и се подготвяме да имаме по-гъвкаво работно време, чрез този масив ще се гарантира избора на най-доброто работно време за всички стажанти. Също така доста се улеснява проверката, дали стажант може да работи в дадена смяна.**

**Ако не ни е нужно за текущата задача, го премахни. Наблягам на думата нужно. Проверката си е стандартна и без него, но ако сметнеш че наистина не може да се направи без този масив, го остави. Моля, добави това към документацията.**

**Нека остане този масив, стандартна или не, проверката ще изглежда сложна. На 90% искам да остане този масив, няма да се натоварим чак толкова много ☺**

public Employee(int id, String filename)

Id и numShifts не се ли зареждат от файла? Не ни ли трябва само името на този файл за инициализация? Той така или иначе се използва за startHours и endHours.

**Методът за четене от файл ще се вика от конструктора, съответно там ще се дефинират и startHours и endHours. Точно това казвам. Това е конструктора, но не знаем кой файл да четем. numShifts го няма никъде във файла, това е параметър, който е нужен за да дефинираме масива availableShifts. Тук има малко промяна обаче, тъй като numShifts е статична променлива от стартовия ни клас, то няма нужда да я предаваме като параметър на конструктора, а просто можем да си я викаме, когато ни е нужно, същото важи за workdayStart и workdayEnd. Съгласен.**

**Съгласен съм и аз – името на файла ще се подава в конструктора.**

„availableShifts се изчислява в цикъл“

Без инициализация на часовете на смените, това няма как да стане. Не можем ли да ги определим в отделен метод, а в конструктора да останат празни?

**????? Нещо не разбрах въпроса. Можем да направим метод (по-скоро трябва да направим), който да изчислява availableShifts и който да се вика в конструктора. Този метод е set метода за променливата availableShifts. Може би това си имал предвид? Да, нещо подобно исках да кажа.**

„*аксесори и мутатори:*“

Всички променливи са private. Не е нужно да се предпазваме от себе си (надявам се).

**Именно поради това, че са private има нужда от аксесори и мутатори. Можем да ги направим public и тогава да нямаме нужда от тези методи, но няма как да се гарантира, че съотборник няма да пипне някоя променлива без да иска!!! Също така в аксесорите и мутаторите ще има проверки, за да не set-нем startHour на -4 часа, или да set-нем availableHours на 50.**

**Направих промяна, която е по-скоро минимална и към момента незначителна, но ще можем да надградим програмата, ако се налага: вместо true и false масива workShifts ще приема стойност – номер на работното място, на което ще работи стажанта.**

**Име на изпълнителя на задача 2**

public readFromFile(String filename)

Липсва формат на данните във файла. Или това ще се реши от човека създаващ този метод? Името може да е по-ясно (пр. loadEmployeesFromFile())

**ОК, променяме името на readEmployeeDataFromFile(String filename), метода чете информация само за 1 стажант, затова е единственото число. Формата на данните го казах какъв ще е на срещата, затова не сметнах за нужно наново да го пиша. Няма как да очакваш който и да е да е запомнил дословно всичко от срещата, затова си правим записки. Моля, включи формата в документа.**

**ОК.**

**Формат на текстовите файлове:**

**ИМЕ:**

**Employee#.in (filename = “Employee”, с конкатенация се добавят номера (ID) на стажанта и разширението .in, което е константно)**

**СЪДЪРЖАНИЕ:**

**FirstWeek**

**Monday startHour endHour**

**Tuesday startHour endHour**

**Wednesday startHour endHour**

**Thursday startHour endHour**

**Friday startHour endHour**

**Saturday startHour endHour**

**Sunday startHour endHour**

**SecondWeek**

**Monday startHour endHour**

**Tuesday startHour endHour**

**Wednesday startHour endHour**

**Thursday startHour endHour**

**Friday startHour endHour**

**Saturday startHour endHour**

**Sunday startHour endHour**

public writeToFile(String filename)

“общ брой работни часове” имаш предвит сумарно? Не е ли отделно за всеки ден? Също трябва да записва Id. Също липсва формат.

**ОК, и тук променяме името на writeEmployeeScheduleToFile(String filename). Записваме и сумарните часове, все пак това е важно за да се знае дали е изпълнил квотата (40-80 работни часа за двете седмици).**

**Формат на текстовите файлове:**

**ИМЕ:**

**Employee#.out (filename = “Employee”, с конкатенация се добавят номера (ID) на стажанта и разширението .out, което е константно)**

**СЪДЪРЖАНИЕ:**

**Hours scheduled for the two weeks: workHours**

**FirstWeek**

**Monday shiftNumber, shiftNumber, shiftNumber, …. , shiftNumber**

**Tuesday shiftNumber, shiftNumber, shiftNumber, …. , shiftNumber**

**Wednesday shiftNumber, shiftNumber, shiftNumber, …. , shiftNumber**

**Thursday shiftNumber, shiftNumber, shiftNumber, …. , shiftNumber**

**Friday shiftNumber, shiftNumber, shiftNumber, …. , shiftNumber**

**Saturday shiftNumber, shiftNumber, shiftNumber, …. , shiftNumber**

**Sunday shiftNumber, shiftNumber, shiftNumber, …. , shiftNumber**

**SecondWeek**

**Monday shiftNumber, shiftNumber, shiftNumber, …. , shiftNumber**

**Tuesday shiftNumber, shiftNumber, shiftNumber, …. , shiftNumber**

**Wednesday shiftNumber, shiftNumber, shiftNumber, …. , shiftNumber**

**Thursday shiftNumber, shiftNumber, shiftNumber, …. , shiftNumber**

**Friday shiftNumber, shiftNumber, shiftNumber, …. , shiftNumber**

**Saturday shiftNumber, shiftNumber, shiftNumber, …. , shiftNumber**

**Sunday shiftNumber, shiftNumber, shiftNumber, …. , shiftNumber**

**Име на изпълнителя на задача 3**

Тук не съм сигурен че разбрах какво се очаква като резултат от тази задача. Гледам, че се създава нов клас с променливи. Това да не би да е задача 1, но за графика (Scheduler)?

**Да, нещо такова. Променено е името на главния ни клас (съдържащ main метода), за да показва какво прави (Start беше доста тъпо име).**

private static boolean [] workdays

Защо тук имаме работните дни, а за студента нямаме учебни. Мисля си, че решението на студента (с начало и край на смените) е по уместна тука, така елиминираме също numShifts, hoursInShift и breakBetweenShifts.

**Работните дни оказват кога имаме работен ден, защото програмата ни предвижда да се въвеждат различни работни дни, например:**

**Is Monday workday? Y/N Y //workdays[0]=true;**

**Is Tuesday workday? Y/N N //workdays[1]=false;**

**Is Wednesday workday? Y/N Y //workdays[2]=true;**

**и така нататък**

private static boolean [][][] occupiedWorkplace

Това е прекалено сложно. Не може ли едномерен масив, като за индекс имаме номера на смяната? (Забел.: Същото може да се направи и при смените на студента)

**Не мисля, че е сложно, но дори да е сложно, този масив ни е крайно необходим, за да можем да раздаваме смени на стажантите. И всъщност според мен трябва даже още да го усложним като от boolean го направим int, за да знаем кой стажант в кой ден в коя смяна на кое работно място работи. В противен случай тази информация трябва да я събираме от доста места, което няма да е лесно за имплементиране и за четене, също така могат да възникнат доста лоши резултати при малка грешка. Много места звучи малко смешно на фона на 3-мерен масив, но добре, да речем че така е по-лесно. Пък и вероятно повечето методи ще се съобразят с този масив, така или иначе.**

**Фактор е четимостта на кода, все пак ще трябва да го обясняваме. Смешно или не, трудно е да се прочете if(условие 1 && условие 2 && (условие 3 || (условие 4 && условие 5))) …, съвсем елементарно е if(occupiedWorkspace[day][i][j]>0) …**

private static LinkedList <Employee> employees

Тука само ми се стори странно това, че минахме от масиви на свързани списъци. Не че има нещо нередно, просто ми е странно...

**Ще гарантираме по-лесното изпълнение на програмата, тъй като Employee е по-обемен клас, не е като примитивните променливи, при масива трябва да имаме съседни клетки памет, а стажантите могат да са повече и да забавяме излишно програмата. Но това не е голям проблем, ако мислиш, или ако другите мислят, че е по-добре с масив, за мен е ОК. Не, нека го оставим така. Излишна промяна ще е според мен**

**Съгласен.**

public initialization()

Класа е статичен, значи е необходим статичен инициализиращ блок (static initialization block)

**Прав си ☺ Карал съм наизуст. Само и в документацията ти („Подробно описание на задачите за изпълнение.docx“) си го отрази че е статичен (не съм пипал нищо по нея).**

**Да променям го това, променяйки този текст.**

**Име на изпълнителя на задача 4**

public arrangeHours(int day) : void

Защо за ден, нали каза че работиш на по 14 дена? Това е като алгоритъма в а) от документацията.

**Не е точно така … Може да се направи да е за всичките дни и това няма да влияе на изпълнението на програмата, просто кода на метода ще се утежни. По-добре в цикъл да викаме метода, отколкото да викаме цикъл в метода ;) Освен това занапред можем да решим да повишим бързодействието на програмата, като добавим нишки и всяка нишка да отговаря за 1 ден от седмицата ☺ Моля те, недей ползва нишки за non-realtime програма... Пълен overkill e, и не си виждал ужасите, които нишките могат да създадат.**

**Няма да ползваме.**

„ //започвайки от най-заетия, продължавайки към най-свободния стажант”

Зает == студент с най-малък availableShifts масив, нали? **ДА, но по-скоро**

**зает==студент с най-малък availableHours**

Така даваш приоритет на тези с по-малко часове, което може да причини обратни на очакваното резултати: хората с по-малко свободно време ще имат повече часове от тези с повече.

**В този етап даваме само по 1 смяна.Ок, и това моля добави към документацията.**

„Метода маркира occupiedWorkplace[day][работна смяна][работно място] в true, ако намери стажант, който да заеме съответно работно място в съответната работна смяна.“

Т.е. работи с идеята да запълни първите дни от седмицата, и да остави последните свободни. Това ще наруши условието за равномерно разпределение.

**Това изобщо не е така, нещо не си разбрал! Винаги раздаваме по 1 смяна на стажант за всеки ден, разбира се ако е възможно.**

„Също така вдига флаг true в член-променливата workShift[day][работна смяна] в обекта от клас Employee, с който се работи в метода“

Не е ли по-лесно да записваме номера(id) на студента, който заема позицията? Така ще е по-лесно да се разбере кой къде е и на практика елиминираме нуждата от workShift в Employee класа.

**Малко по-горе записах, че правим нещо подобно с тримерния масив, както и промяната от bool на int на масива workShifts, в случая все още са необходими и двата масива, за да извършваме проверките в пъти по-лесно, отколкото ако ги елиминираме. Ок, оставяме ги.**

**Име на изпълнителя на задача 5**

започвайки от най-свободния, продължавайки към най-заетия стажант се разпределят работни смени за всеки ден.

Не разбирам какво значи най-свободен тук. Май има противочерие с точка 4 (двата метода работят на противоположен принцип).

**Методите НЕ работят на противоположен принцип. Просто се взима човека с най-висок availableHours и му се дават още смени, ако има възможност. Може би ще ти се изясни след като видиш по-горните ми отговори, надявам се да е така … Мисля че разбрах.**

**Име на изпълнителя на задача 6**

„константа, определяща средния брой работни часове за стажант (средният брой часове = брой работни дни \* брой работни смени \* брой работни часове в смяна \* брой работни места / брой стажанти).“

Не е ли: средният брой часове = (∑ (продължителност на смяната \* броя стажанти в смяната)) / брой стажанти ?

**Искаме да намерим средния брой работни часове, които са възможни, не които са изпълнени до момента, това е разликата, между моя израз и твоя. По принцип не би трябвало да има разлика в изпълнението, ако държиш ще тестваме програмата и с двете константи. Да тестваме по двата начина би било трудно, пък и сме избрали този подход вече. Просто напиши малко по-ясно какво намира формулата, понеже аз се обърках първия път.**

**Сменя се 1 ред код, няма да е много трудно. Но както казах твоята формула намира средното време на заетост, докато моето е максималното средно време на заетост, т.е. ако всички работни места за заети за всички смени, двете стойности ще са еднакви, но ако има незаети работни места за някоя смяна, то средното време на заетост ще е по-малко от максималното средно време на заетост. Надявам се да си разбрал.**

„докато не се намери някой, с когото да се направи размяна на смени, стига работните му часове за седмица да са поне колкото средният брой работни часове“

Така ще се получи deadlock. Прост пример: sredno = 28; E1 = 24, E2 = 28 => прехвърляме една смяна от E2 на E1, и получаваме E1=28, E2=24. Действието се повтаря до безкрай. Решението е да няма <= и >= за сравненията със средния брой.

**Това не се нарича deadlock, ами безкраен цикъл. Иначе усетих проблема и затова се опитах да сложа още проверки, но май не съм успял да обхвана проблема. Тогава можем да променим и за втория проверката да е с + продължителност на смяна.**

**Пример1: средно = 28, Е1 = 24, Е2 = 32 => ще има размяна на часове, ако е възможно.**

**Пример2: средно = 28, Е1 = 24, Е2 = 28 => няма да се разменят, тъй като Е2 не отговаря на проверката. Така ще сработи, да.**

Малко на сляпо ще кажа че всичко по алгоритъма изглежда наред. Нямаме още тестови данни обаче, което означава че това мое твърдение е напълно необосновано.

Задачи за изпълнение.docx

Разпределението изглежда добре направено, като обем и като ниво на трудност.

## „2014-03-29 18.56.30.jpg“

## „2014-03-29 18.56.48.jpg“

Както гледам са описани доста добре тези UML-и в документа „Подробно описание на задачите за изпълнение.docx“, така че ако смяташ че няма да ни трябват може да ги изтрием.

**Имай предвид, че Start.java се преименова на Scheduler.java. Също така някои променливи търпят промени!**