**[Документация]**

**Условие:**

*Да се състави програма, изчисляваща най-оптималния график за присъствия на стажанти в практически занятия, по зададени брой работни дни, брой работни смени, продължителност на работния ден, продължителност на работна смяна,брой стажанти в работна смяна и минимален и максимален позволен брой часове на стажант за две седмици.*

**Решение:**

За решението сме използвали следните три класа:

**Employee.java**

1)Декларират се следните член-променливи на класа:

**private** **int** id = Идентификационен номер на стажанта.

**private** **int** workHours = Брой на работните часове за две седмици.

**private** **int** [] startHours = Масив със начални часове на свободното време на стажант за 14 дни.Всяка стойност се проверява дали е >=0 и <=24.

**private** **int** [] endHours = Масив със крайни часове на свободното време на стажант за 14 дни.Всяка стойност се проверява дали е >=0 и <=24.

**private** **int** [] availableHours = Масив с броя свободни часове на ден за 14 дни. Получаваме ги от разликата между endHours и startHours.

**private** **boolean** [][] availableShifts = Двумерен булев масив с размерност [14][numShifts], в който се маркира с true всяка смяна, в която стажанта може да присъства и с false, ако неможе.

**private** **int** [][] workShifts = Двумерен масив с размерност [14][numShifts], в който с номер на работно място се маркира смяна, през която стажантът ще работи на съответното работно място.

2) Конструктор Employee(**int** id, String filename){

инициализира член-променливите на класа:

workHours = 0;

- setDefaultWorkShifts – инициализира workShifts с начални стойности = 0;

- readEmployeeDataFromFile(String filename) – метода за четене от файл, също инициализиращ startHours и endHours;

- setAvailableHoursAndAvailableShifts() - инициализират се availableHours и availableShifts със съответните стойности след вече прочетените от файла startHours и endHours.

}

3) Класът съдържа стандартни Set и Get методи за член-променливите с нужните проверки.

4) Класът съдържа следните други методи:

-writeEmployeeScheduleToFile(String filename) - метода записва номера на смяна за съответен ден във файл с име, подадено като параметър към метода;

-**public** **void** printSpec(**int** day, PrintStream fileWriter) - помощен метод за метода writeEmployeeScheduleToFile, чрез цикъл се записват данните за съответния ден чрез индекса на масива workShifts

**Scheduler.java**

1)Декларират се следните член-променливи на класа:

private static boolean[] *workdays* = Булев масив за седемте дни от седмицата. По време на въвеждане на входните данни маркира избраните работните дни с true, а почивните с false.

private static int *numWorkplaces* = Брой работни места в една работна смяна.

private static int *numEmployees* = Общ брой на стажантите.

private static int *numShifts* = Брой на смените.

private static int *workdayStart* = Начален час на работния ден.

private static int *workdayEnd* = Краен час на работния ден.

private static int *hoursInShift* = Брой часове в една смяна.

private static int *breakBetweenShifts* = Почивка между смените.

private static int *minWorkHours* = Минимален брой работни часове на стажант за 14 дни.

private static int *maxWorkHours* = Максимален брой работни часове на стажант за 14 дни.

private static int [] *shiftStart* = Масив с размерност [numShifts] съдържащ началните часове на работните смени.

private static int[][][] *occupiedWorkplace* = Тримерен масив с размерност [14][numShifts][numWorkplaces], стойност 0 означава, че работното място е свободно, ако е заето мястото има стойност ID на стажанта.

private static LinkedList<Employee> *employees*; - Свързан списък с обектите от клас Employee.java

2)Статична инициализация на всички входни параметри;

static(){

Switch case със седем опции за всеки ден от седмицата. За всеки един пита дали е работен или не, ако е работен го отбелязва в булевия масив с true, ако е почивен с false.

Стандартен вход от конзолата за останалите променливи, с нужните проверки.

}

3)Методи:

- arrangeHours(int day){

Създава се нов свързан списък LinkedList<Employee>employeeTemp и обектите се сортират от най-заетия към най-свободния с помощта на метода sortEmpoyeesByFreeTimeInDay.

Във вложен for цикъл се опитва да се даде по една смяна на всеки стажант с помощта на метода enrollEmployee();

}

– addMoreHours(int day){

Тук помощният метод addMoreHoursForShift е вкаран във for цикъл с дължина numShifts.

}

-rearrangeAveraging(){

Декларират се две помощни променливи за индексите на текущите стажантите с най-малък и най-голям брой часове за 14 дни.

Обявяват се и две променливи, използвани за проверка на времето, през което се изпълнява този метод. В нашето решение това време е 5000 милисекунди. В безкраен while цикъл се опитва да се прехвърлят смени на стажанта с най-малко часове взети от този с най-много, при всеки неуспешен опит индекса на текущия стажант с най-много часове се намаля с 1.Ако се намери стажант, които може да даде смяна на друг се вика метода exchangehours(). От цикъла се излиза когато втората променлива е с 5000 > от първата.

}

– getEmployeesCountInShift(int day, int shift){

Инициализира се променлива occupiedCount = 0;

Метода преброява стажантите в една смяна, проверявайки за стойности различни от 0 в тримерния масив occupiedWorkplaces[][][], ако намери заето място увеличава occupiedCount с 1;

Връща occupiedCount;

}

– addMoreHoursForShift(int day, int shift){

Този метод сравнява броя на окупираните и максимално позволените работни места в всяка смяна. Ако не са окупирани всички места, намира стажанта с най-малко свободно време и пробва да го сложи на свободното място. Това продължава докато не се намери някой, които да заеме мястото или докато не свърши броя на стажантите.

}

– enrollEmployee(Employee employee, int day, int shift){

В този метод се проверява, ако има останали незаети места във всички смени(occupiedWorkplaces[][][] == 0) и дали има стажанти, които биха могли да го заемат, но без вече да са записани в тази група (employee.getWorkShifts()[day][shift] == 0). Ако и двете проверки са успешни, стойността на съответната смяна в occupiedWorkplaces[][][] и workShifts[][] се отбелязва с ID на стажанта, а в workHours се увеличава сумата на работните часове на съответния стажант с цифра продължителността на работната смяна.

}

– dismissEmployee(Employee employee, int day,int shift){

Методът занулява смените на даден стажант в даден ден в occupiedWorkplaces[][][] и workShifts[][] когато бъде извикан, а от workHours изважда цифрата за продължителност на една смяна.

}

– sortEmployeesByFreeTimeInDay (LinkedList<Employee> employees, int day){

Инициализира свързан списък orderedEmployees;

Сортира обектите от клас Employee в новия свързан списък по свободното им време за ден, започвайки от най-заетия.

Връща orderedEmployees;

}

–sortEmployeesByWorkHours(LinkedList<Employee> employees){

Инициализира свързан списък orderedEmployees;

Сортира обектите от клас Employee в новия свързан списък по заетите работни часове за ден, започвайки от най-заетия.

Връща orderedEmployees;

}

-exchangeHours(int indexOfRecipient, int indexOfDonor){

Този метод намира стажанта с най-малко работни часове и този с най-много.Опитва се да вземе смяна от този с най-много и да я даде на другия. Ако не успеят се взима следващия с най-много часове и се повтаря действието. Ако проверката е успешна се извикват методи dissmisEmployee() за даващия смяна и enrollEmployee() за да вмести взимащия.

}

-dismissHours(){

Отнема смени на стажант, докато часовете му за две седмици не станат <= максималните позволени.

}

**Main.java**

Стартира приложението. В края на метода main, writeEmployeesScheduleToFile вкаран във foreach цикъл, записва обработената информация за всички стажанти във файлове с разширение .out;