**Име на изпълнителя на задача 1**

**Employee.java**

private int id //ID на стажанта, >=1

private int workHours //работни часове за двете седмици, >=0

private int [] startHours //начални часове на свободното време на стажанта за двете седмици, >=0 && <=24

private int [] endHours //крайни часове на свободното време на стажанта за двете седмици, >=0 && <=24

private int [] availbaleHours //максимално възможно свободно време на стажанта за двете седмици, >=0 && <=24

private boolean [][] availableShifts //двумерен масив с размерност [14][брой работни смени] с true отбелязваме смяна, през която стажантът е свободен

private int [][] workShifts //двумерен масив с размерност [14][брой работни смени] с номер на работно място отбелязваме смяна, през която стажантът ще работи на съответното работно място

*конструктори:*

public Employee(int id, String filename) //инициализира всички полета по следния начин: workHours=0, startHours и endHours се инициализират като се извика метода за четене от файл, availableHours и availableShifts се изчислява в цикъл, за всички елементи в двумерния масив workShifts се приемат стойност 0. filename е името на файла, от който ще се чете информацията

*аксесори и мутатори:*

//стандартни get и set методи, но включващи проверки!

**Име на изпълнителя на задача 2**

**Employee.java**

public readEmployeeDataFromFile(String filename) : int //метода за четене от файл, попълва член-променливите startHours, endHours за всеки ден от двете седмици. Метода връща стойност 0, ако всичко е наред и 1, ако е възникнала грешка

public writeEmployeeScheduleToFile(String filename) : int //метода за запис във файл, данните които се записват са: общ брой работни часове за двете седмици и номер на смяна за всеки ден от седмицата. Метода връща стойност 0, ако всичко е наред и 1, ако е възникнала грешка.

**Име на изпълнителя на задача 3**

**Scheduler.java**

private static boolean [] workdays //за седемте дни от седмицата, true – работен ден, false – почивен ден

private static int numWorkplaces //брой работни места, >=1

private static int numEmployees //брой стажанти, >=1

private static int workdayStart //начален час на работния ден

private static int workdayEnd //краен час на работния ден

private static int numShifts //брой на смените, >=1

private static int hoursInShift //часове в смяна, >=1

private static int breakBetweenShifts //почвика между смените (в часове), >=0

private static int shiftStart []//началните часове на всички смени

private static int [][][] occupiedWorkplace //тримерен масив с размерност [14][брой смени][брой работни места], 0 означава, че работното място не е заето, ако е заето мястото елемента от масива има стойност id на стажанта, който го е заел

private static LinkedList <Employee> employees //свързан списък с всички стажанти (списъкът е индексиран ;) )

private static int minWorkHours //минималният брой часове за 14-те дни, >=1 && <=maxWorkHours

private static int maxWorkHours //максималният брой часове за 14-те дни,>=1 && <=80

/\*Всички променливи са статични, тъй като те са референция към самия клас, а не към неговите инстанции (обектите от този клас)\*/

static{ ….} //статична инициализация на всички входни параметри, метода връща стойност 0, ако всички данни са коректни, или стойност 1, ако има грешка

public static main(String [] args) : void //това на всички трябва да е ясно какво е :D

**Име на изпълнителя на задача 4**

**Scheduler.java**

public arrangeHours(int day) : void //започвайки от най-заетия, продължавайки към най-свободния стажант се разпределят работни смени за всеки ден (входен параметър day за метода). Метода маркира occupiedWorkplace[day][работна смяна][работно място] в id, ако намери стажант, който да заеме съответно работно място в съответната работна смяна. Също така вдига флаг номер на работно място в член-променливата workShift[day][работна смяна] в обекта от клас Employee, с който се работи в метода. Реално тук се раздава по 1 смяна за всички, които имат възможност да работят в този ден.

**Име на изпълнителя на задача 5**

**Scheduler.java**

public addMoreHours(int day) : void //започвайки от най-свободния, продължавайки към най-заетия стажант се разпределят работни смени за всеки ден (входен параметър day за метода). Метода маркира occupiedWorkplace[day][работна смяна][работно място] в id, ако намери стажант, който да заеме съответно работно място в съответната работна смяна. Също така вдига флаг номер на работно място в член-променливата workShift[day][работна смяна] в обекта от клас Employee, с който се работи в метода. За разлика от метода от задача 4, тук се раздават всички останали смени, т.е възможно е някой да вземе 5 смени, докато друг да остане 1 (тук говорим условно, в нашия случай ще имаме максимум 2 смени, на практика няма да има възможност за подобно нещо.

**Име на изпълнителя на задача 6**

**Scheduler.java**

public rearrangeAveraging() : int //в метода се дефинира 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). В цикъл се проверява стажантът с най-малък брой часове, ако този брой увеличен с продължителността на работната смяна (hoursInShift) е по-малък или равен на средния брой часове (условие (1)), то се търси стажантът с най-много работни часове и се опитва да се направи размяна на смените, ако е невъзможно се взима следващият по брой работни часове (вторият с най-много работни часове), 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 плюс продължителността на смяната (условие (2)). Ако условие (1) не е изпълнено се излиза от цикъла и се проверява коректността на данните. Метода връща стойност 0 – ако всичко е ОК, или 1 – ако има нарушения на ограниченията.