Задачи за изпълнение:

1. Създаване на клас Employee.java с всички необходими член-променливи, констурктори, аксесори, мутатори и методи.

- разработка на класа Employee.java (задача 1); включва: ООП и програмен дизайн

2. Работа с входните и изходните файлове - Employee#.in и Employee#.out.

- Операции четене/запис на файл (задача 2); включва: работа със стандартни Java библиотеки, работа със стрингове

3. Работа с входните (конзолни) параметри на алгоритъма и инициализация на помощните променливи.

- разработка на класа Scheduler.java (задача 3); включва: ООП и програмен дизайн, работа със статични обекти

4. За всеки ден от седмицата се намира стажанта с най-малко възможности за работно време и му се осигурява работна смяна, действието се повтатя, докато не се изпълни за всички стажанти, или не свършат свободните места.

- имплементиране на метод с алгоритъм за минимално покриване на смените(зад.4); включва: работа с масиви, имплементация на алгоритми.

5. За всеки ден от седмицата се намира стажанта с най-много (вече променено от т.4) свободно време и му се осигуряват още часове, действието се повтаря, докато не свършат свободните места. Проверка дали дневната заетост на всеки стажант е под 8 часа.

- имплементиране на алгоритъма за допълнително заделяне на смени за свободни студенти(зад.5); включва: работа с масиви, имплементация на алгоритми.

6. Проверка дали всички стажанти имат поне 40 работни часа за 2-те седмици, ако не е изпълнена проверката се взима стажантът с най-малък брой часове и се опитва да вземе смени от стажантът с най-голям брой. След като за всички е изпълнено условието се сравняват работните часове на всеки стажант със средния брой часове за всички (средният брой часове = брой работни дни \* брой работни смени \* брой работни часове в смяна \* брой работни места / брой стажанти), като се взима стажантът с най-много работни часове и се опитва да се даде негова смяна на стажантът с най-малко работни часове, това се изпълнява докато е изпълнено

условието: работни часове на стажанта - продължителност на смяната >= среден брой работни часове за всички.

- имплементиране на алгоритъма за допълнително преразпределяне на резултата за получаване на равномерно приемливо разпределение (зад.6); включва: работа с масиви, имплементация на алгоритми.

Здравейте, пак ☺

Имаме завършени стъпките за работа, благодарение на Георги. Остават ни да допълним по възможност (и желание) тестовите ни данни, след което да започнем и имплементацията. Имаме следните 6 задачи за момента:

- зад 1. разработка на класа Employee.java; включва: ООП и програмен дизайн

- зад 2. Операции четене/запис на файл; включва: работа със стандартни Java библиотеки, работа със стрингове

- зад 3. Разработка на класа Scheduler.java (); включва: ООП и програмен дизайн, работа със статични обекти

- зад 4. имплементиране на метод с алгоритъм за минимално покриване на смените; включва: работа с масиви, имплементация на алгоритми.

- зад 5 Имплементиране на алгоритъма за допълнително заделяне на смени за свободни студенти; включва: работа с масиви, имплементация на алгоритми.

- зад 6. Имплементиране на алгоритъма за допълнително преразпределяне на резултата за получаване на равномерно приемливо разпределение; включва: работа с масиви, имплементация на алгоритми.

Всеки да си хване една от тези задачи, като моля захващайте се с най-трудната, която смятате че е по силите ви. Георги има за задача да прегледа примерните тестови данни и да ги обедини в няколко примерни тестови групи със единен формат. Тези тестови данни имат за краен срок ДОВЕЧЕРА, така че Жоре не почвай своята задача преди утре. Има качен в github скелет на задачата, само имплементацията по точките е нужна. Като е готова, пускаме тестовите данни, оправяме бъговете и минаваме на зад. 7 (Пииване на биричка). Ай, късмет на всички ;)