Зад. 8 (Пресмятане на e)

Едно важно за математиката число е Неперовото число (Ойлеровото число), тоест числото ***e***. Използвайки сходящи редове, можем да сметнем стойността на ***e*** с произволно висока точност. Един от сравнително бързо сходящите към ***e*** редове е:

$$(1) \quad e = \sum_{k=0}^{\infty} \frac{2k+1}{(2k)!} = \frac{1}{0!} + \frac{3}{2!} + \frac{5}{4!} + \frac{7}{6!} + \frac{9}{8!} + \frac{11}{10!} + \frac{13}{12!} + \frac{15}{14!} + \ldots$$

Вашата задача е да напишете програма за изчисление на числото ***e*** изпозлвайки цитирания ред, която използва паралелни процеси (нишки) и осигурява пресмятането на ***e*** със зададена от потребителя точност. Изискванията към програмата са следните:

(o) Точността на пресмятанията, в брой цифри след десетичната точка, задаваме с подходящо избран команден параметър – например **"-p 10000";**

(o) Друг команден параметър задава максималния брой нишки (задачи) на които разделяме работата по пресмятането на ***e*** – например **"-t 1"** или **"-tasks 3";**

(o) Програмата извежда подходящи съобщения на различните етапи от работата си, както и времето отделено за изчисление и резултата от изчислението (стойността на ***e***);

(o) Записва резултата от работа си (стойността на ***e***) във изходен файл, зададен с подходящ параметър, например **"-o result.txt"**. Ако този параметър е изпуснат, се избира име по подразбиране;

(o) Да се осигури възможност за „quiet" режим на работа на програмата, при който се извежда само времето отделено за изчисление на ***e***, отново чрез подходящо избран друг команден параметър – например **"-q";**

ЗАБЕЛЕЖКА:

(o) При желание за направата на подходящ графичен потребителски интерфейс (GUI) с помощта на класовете от пакета **javax.swing** задачата може да се изпълни от **двама души**; Разработването на графичен интерфейс не отменя изискването Вашата програма да поддържа изредените командни параметри. В този случай към функцията на параметъра параметъра **"-q"** се добавя изискването **да не запуска** графичния интерфейс. Причината за това е, Вашата програма да може да се тества отдалечено.

Уточнения:

(o) В условието на задачата се говори за разделянето на работата на две или повече нишки. Работата върху съответната задача на една нишка ще служи за еталон, по който да измерваме евентуално ускорение (T1). Тоест в кода реализиращ решенията на задачите трябва да се предвиди и тази възможност – задачата да бъде решавана от единствена нишка (процес); Пускайки програмата да работи върху задачата с помощта на единствена нишка, ще считаме че използваме серийното решение на задачата; Измервайки времето за работа на програмата при работа с „p" нишки - Tp, изчисляваме Sp. Представените на защитата данни за работата на програмата, трябва да отразят и ефективността от работата и, тоест да се изчисли и покаже Ep.

(o) Не се очаква от вас да реализирате библиотека, осигуряваща математически операции със голяма точност. Подходяща за тази цел библиотека е например Apfloat (http://www.apfloat.org).

(o) Командните аргументи (параметри) на терминална (конзолна) Java програма, получаваме във масива String args[] на main() метода, на стартовия клас. За „разбирането" им (анализирането им) може да ползвате и външни библиотеки писани специално за тази цел . Един добър пример за това е: Apache Commons CLI (http://commons.apache.org/cli/).

(o) Интересен е въпросът, кога достигаме зададената точност на изчисленията? Тоест

кога сме сметнали “*e*” със зададените от потребителя брой цифри след десетичната точка. Едно добро ограничение за серийната (последователната) програма е разликата между две поредно изчислени стойности на “*e*” да е произволно малка.