**Курсов проект по Системи за паралелна обработка**

##### **Паралелен алгоритъм за изчисляване на Неперовото число**

#### Изготвил:

Име: Цветислав Викторов

ФН: 80842

Специалност: Компютърни наyки

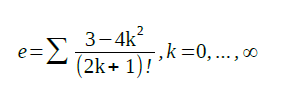
**Ръководител:**

ас. Христо Христов

Проверил: ........................

Дата:20.06.2015г. (ас. Христо Христов)

1. **Условие на задачата**



Трябва да се напише програма за изчисление на числото *e* използвайки цитирания ред, която използва паралелни процеси (нишки) и осигурява пресмятането на ***e***със зададена от потребителя точност.

1. **Изисквания към програмата**

(o) Команден параметър задава точността на пресмятанията. По Ваше желание, точността се изразява или в брой цифри след десетичната запетая или в брой членове на реда. Командният параметър задаващ точността има вида “-p 10000”;

(о) Друг команден параметър задава максималния брой нишки (задачи) на които разделяме работата по пресмятането на *e* – например “–t 1” или

“–tasks 3”;

(о) 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”;

1. **Реализация**

Проектът се състои от четири класа:

- **ThreadManager.JAVA**

- **CalculatorThread.JAVA**

- **E.JAVA**

- **Test.JAVA**

За паралелното решаване на задачата е избран подходът на многонишковото програмиране. Той предоставя възможност за оптимално намаляване на използвания ресурс, тъй като позволява отделните процеси, извършващи изчисленията, да използват един и същ ресурс и да нанасят резултатите върху друг споделен ресурс, който да се ползва без синхронизация, тъй като отделните подпроцеси се нуждаят от различни негови участъци.

Идеята на алгоритъма е всяка нишка изчислява част от числото ***е*** като смята толкова члена на редицата, колкото потребителя ги е задал разделено на броя нишки. Това дава на програмата ускорение.

**CalculatorThread.java**

Този клас наследява class Thread и го използвам за определяне на същността и действието на всяка нишка от програмата. Всяка нишка си има номер с който се идентифицира – **private** **int** index. Според този индекс и според общия брой нишки, на нишките се определят кои и колко члена на редицата да пресметнат.След като е се знае интервала на смятане на всяка нишка, самите сметки се извършват от метода

**public** Apfloat calc(**int** from, **int** to), който обхожда интервала [from, to] с един цикъл и за всяка итерация пресмята съответно числителя и знаменателя, а след това намира тяхното частно и го добавя в общия ресурс за всичките нишки - **private** Apfloat sum ,който се намира в класа Е.

**ТhreadManager.java**

Този клас носи информацията за броя нишки - **private** **int** threads, точността на изчисляването на числото ***е*** ( тоест броя членове на редицата) - **private** **int** digits, **private** E e който носи информация за сумата. В метода се инициализират и вкарват в нишките, от тип CalculatorThread, които се съхраняват в **private** ExecutorService threadServise. Преди да се изведе числото, се изчакват всички нишки да приключат работата си – **while**(!threadServise.isTerminated()) {}

**Test.java**

Чрез него се въвеждат входните данни и се покриват изискванията към програмата(т.2). Създава се обект от тип TreadManager, на който се инициализира с параметрите, зададени от потребителя. Изчислява се времето на изчисление на задачата. Резултатът се записва във файл по подразбиране или в зададения от входните данни.

1. **Експериемент**

Проектът е разработен да Java. За експеримента се изчислява числото ***е*** с 1000 елемента в редицата си. Спрямо броя нишки се определя времето необходимо за изчисление – Т(t), ускорението S(t) = T(1) / T(t), како и ефективността E(t) = S(t) / t, където t е броя нишки.

