**Зад. 1 (Threaded калкулатор)**

Разглеждаме контекстно-свободната граматика **Г = <N, T, Expr, P>**, чийто продукционни правила имат вида:

**Num -> 0 | ... | 9 | 0Num | 1Num | ... | 9Num**
**Expr -> Expr + Term | Expr - Term | Term**
**Term -> Term * Factor | Term / Factor | Factor**
**Factor -> Num | (Expr)**

Граматиката описва прост аритметичен израз, допускащ употребата на скоби.

Задачата е да се напише програма, която интерпретира (пресмята) аритметични изрази извеждани от граматиката Г. Тъй като езика на Г разполага само с цели положителни числа, то за операцията деление се подразбира целочислено деление. Програмата да разпределя по подходящ начин целочислените изчисления между две или повече нишки (задачи).

Изискванията към програмата са следните:

(о) Допуска работа с произволно големи цели числа. За целта може да се използва класа **BigInteger** на **Java**, намиращ се в пакета **java.math**;

(о) Чете израза или изразите от входен текстов файл, чието име задаваме като команден параметър на програмата – например **“-f file.txt”**;

(о) Записва резултата от работа си върху входния аритметичен израз във изходен файл, зададен с подходящ параметър, например **“-o result.txt”**. Ако този параметър е изпуснат, се избира име по подразбиране;

(о) Втори команден параметър задава максималния брой нишки (задачи) които могат да извършват изчисления – 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;

Примери за подходящи съобщения:

„**Thread-<num> started.**“,

„**Thread-<num> stopped.**“,

„**Thread-<num> execution time was (millis): <num>**“,

„**Threads used in current run: <num>**“,

„**Total execution time for current run (millis): <num>**“ и т.н.;

(о) Да се осигури възможност за **„quiet“** режим на работа на програмата, при който се извежда само времето отделено за пресмятане на аритметичния израз (в зависимост от стойността на параметъра **-t**), отново чрез подходящо избран друг команден параметър, например **“-q”** (или **„-quiet“**);

ЗАБЕЛЕЖКА:

(о) Програмата може да изчисли даден аритметичен израз като преди това построи дърво на извод на този израз, според описаната граматика. В зависимост от подадения команден параметър за брой задачи, тя може да раздели работата по изчисленията, ако височинната на дървото на извод позволява това.

(о) 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 пуска** графичният интерфейс. Причината за това е, че Вашата програма трябва да позволява отдалечено тестване, а то ще се извършва в **terminal**.

**Уточнения (hints) към задачата:**

(о) В условието на задачата се говори за разделянето на работата на две или повече нишки. Работата върху съответната задача, в случаят в който е зададен **„-t 1“** (т.е. цялата задача се решава от една нишка) ще служи за еталон, по който да измерваме евентуално ускорение (т.е. това е **T1**). В кода реализиращ решението на задачата 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 използването на **„p“** нишки – намираме **Tp** и съответно можем да изчислим **Sp**. Представените на защитата данни за работата на програмата, трябва да отразят и ефективността от работата и, тоест да се изчисли и покаже **Ep**.

Като обобщение - данните събрани при тестването на програмата Ви, трябва да отразяват **Tp**, **Sp** и **Ep**. Желателно е освен табличен вид, да добавите и графичен вид на **Tp**, **Sp**, **Ep**, в три отделни графики.

(о) Не се очаква от Вас да реализирате библиотека, осигуряваща математически операции със комплексни числа. Подходяща за тази цел е например **Apache Commons Math3 (http://commons.apache.org/proper/commons-math/userguide/complex.html).** При изчисленията, свързани с генерирането на множеството на Манделброт (задачите за фрактали), определено ще имате нужда от нея.

(о) 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)**. Ако програмата Ви има нужда от работа с големи числа, можете да използвате нея.

Разбира се **BigInteger** и **BigDecimal** класовете в **java.math** са също възможно решение – въпрос на избор и вкус.

Преди да направите избора, проверете дали избраната библиотека не използва също нишки – това може да доведе до неочаквани и доста интересни резултати; ;)

(о) Не се очаква от Вас да търсите (пишете) библиотека за генериране на **.png** изображения. Java има прекрасна за нашите цели вградена библиотека, която може да се ползва. Примерен проект, показващ генерирането на чернобялата и цветната версия на фрактала на Манделброт /множество на Манделброт за формула (2)/, цитирани в задачите за фрактали, е качена на **http://rmi.yaht.net/docs/example.projects/ - pfg.zip**.

(о) Командните аргументи (параметри) на терминална Java програма, получаваме във масива **String args[]** на **main()** метода, намиращ се в стартовият 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/)**.