ASSIGNMENT 1:

Kartikeya Saxena

180101034

kartikey18a@iitg.ac.in

PART(A) : Design and implement a java multithreaded program to estimate the value of PI using Monte-Carlo Simulation. The number of points should be 10^6 and you should compute the same using 4 to 16 threads. The number of threads should be one argument to the program.

SOLUTION :

The program takes the number of threads as the argument and for each thread and throws an error if the argument is not an integer.
Two global variables are maintained :
1- points: count of points generated by threads
2- circlePoints: count of points generated by threads which are within unit distance of origin
Each thread generates a random point inside a square of side length 2 centered at origin and increments the points counter if it is less than 10^6. If the point is within unit distance from the origin it circlePoint is incremented. This process is continued until points is less than 10^6 . Synchronisation is achieved by using a lock object and synchronized statements for points and atomicInteger for circlePoints. Finally the main thread prints the value of PI = 4 * circlePoints / 10^6.

USAGE :

javac monteCarlo
java monteCarlo threadCount[Integer[4-16]]

PART(B) : Design and implement a java multithreaded program for approximating the following integration using composite Simpson 1/3 rule. The number of points should be > 10^6 and compute the value of integration using 4 to 16 threads. The number of threads should be one argument to the program.

$$\int_{-1}^{1} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \, dx$$

SOLUTION :

The program takes the number of threads as the argument and for each thread and throws an error if the argument is not an integer.

$$\int_a^b f(x)dx \approx \frac{h}{3}\left(f_0 + f_n + 4 * \sum_{i=1,3,5}^{n-1} f_i + 2 * \sum_{i=2,4,6}^{n-2} f_i\right)$$

Each thread takes cnt = ceil((n+1)/threadCnt) number of points to calculate the function value f(x) and calculates sum according to the above formula and returns the sum. Thread i calculates the function in the range [i * cnt, min(i * cnt + size, n + 1)). The main thread then adds up these partial sums and prints the result.

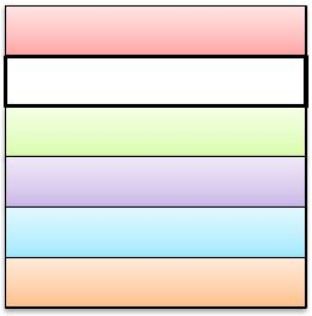
USAGE :

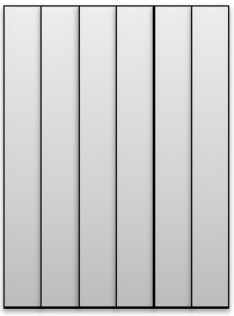
javac simpsonRule.java
java simpsonRule threadCount[Integer[4-16]]

PART(C) : Design and implement a java multithreaded program to compute matrix multiplication C=A x B. Assume both A and B are square matrices of row size N=1000. Initialize all the elements of both A and B matrix with a random number between 0 to 10. Compute both initialization and matrix multiplication computation using 4 to 16 threads. The number of threads should be one argument to the program.

SOLUTION :

The program takes the number of threads as the argument and for each thread and throws an error if the argument is not an integer.
First the main thread initializes the matrix A and B with random numbers between 0 to 10 both inclusive.
The whole rows of matrix A is divided into threadCnt chunks and each thread multiplies its chunk with matrix B. The main thread then prints the matrix A, B and C.





USAGE :

javac matrixMultiplication
java matrixMultiplication threadCount[Integer[4-16]]