# Introduction:

* [Problems](#_Instructions:)

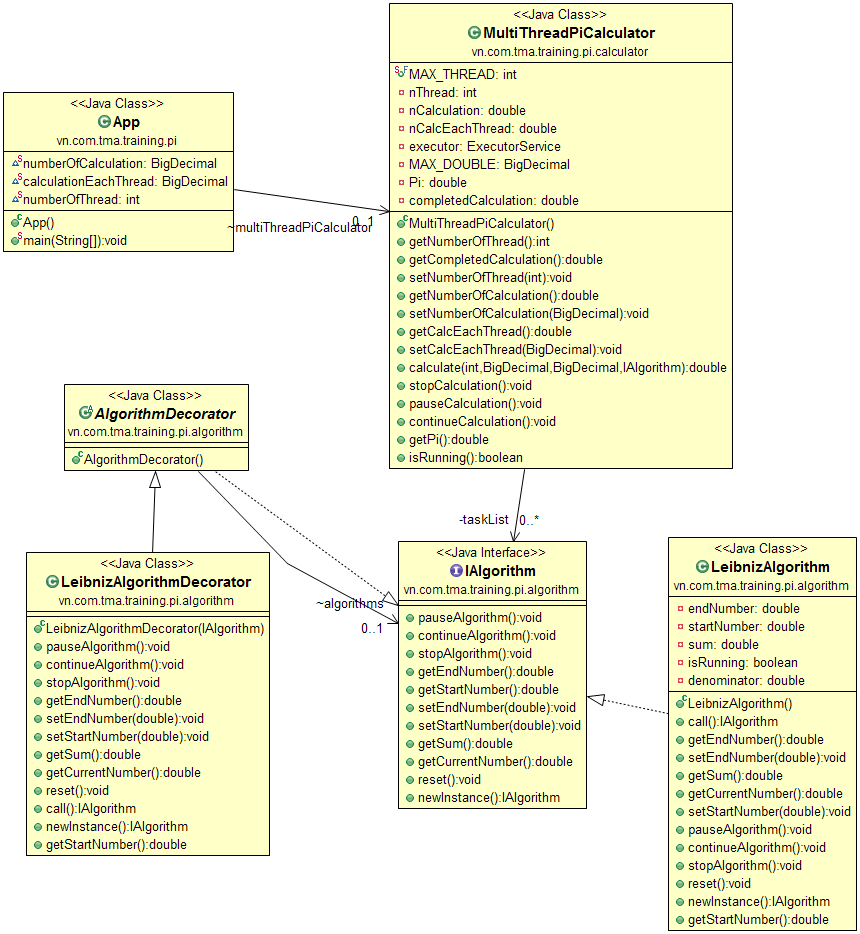
Make a program to calculate number PI, using java and applying design pattern to make program reusable, maintainable and extendable.

* [Solutions](#_Solutions:)

Apply Strategy pattern, make a PiCalculator which the algorithm function was done by other classes. Create interface IAlgorithm that concrete algorithm class must implement all method to communicate with PiCaulator. It help programmer to implement new algorithm easily and user can change algorithm quickly.

A formula will loop the calculation n times. I use multithread: create t threads, each thread will run c calculation (c < n) until finish n calculations.

* High Level Design



* Detail Design

**IAlgorithm****:** An interface of algorithm to calculate PI number. Concrete algorithms will implement this and have own implementation. It extends Callable return a IAlgorithm

**LeibnizAlgorithm:** A concrete class will implement this and have own implementation. The formula is: PI/4 = (-1)^n/(2\*n+1) (n from 0 to infinity) <=> PI/4 = 1 - 1/3 + 1/5 - 1/7 + 1/9 - ... (-1)^n/(n\*2+1) .... (\*)

A Leibniz instance will take 2 parameter is startNumber and endNumber so that each instance can calculate a part of all formula and each part may run in a particular thread to get more efficient.

As we see in (\*) the denominator is odd number (2n+1). Instead of using a multiplication and a addition in for loop, I init the start denominator of loop sequence then next denominator just need to plus to 2. About the numerator, if n is odd then numerator equal 1, and even mean -1. Instead of get power of (-1), I init the start numerator of loop sequence then next numerator will multiply with -1.

**LeibnizAlgorithmDecorator:** is a wrapper class extends AlgorithmDecorator contain a IAlgorithm object. In this class, on the call method, I add new behavior to print information when LeibnizAlgorithm is called.

**MultiThreadPiCalculator:** is the main object of Pi Calculation, user will using this class to calculate PI. This class manipulate some object of IAlgorithm, divide many tasks them pass to thread pool. Thread pool is fixed size and input by user but the size is limited between 1 and MAX\_THREAD. I create amount instance of IAlgorithm (which represent for task, each task process a part of calculations) equal to the amount of thread in thread pool. When thread pool handle task, its will finish all current task then new parameter will update to task instance. This process is looped until all tasks be done. The main method is **public** **double** calculate(**int** numberOfThread, BigDecimal numberOfCalculation,

BigDecimal calculationEachThread, IAlgorithm algorithm)

Take 4 parameter and return the value of calculated number PI. I use BigDecimal to check overflow of double type.

**App:** This class have main class to show console user interface, it handle input and control command. It represent for the client using PiCalculator

# Program Instructions**:**

* [Build](#_Build)
* [Run](#_Run)

Menu -> Run -> Run (CTRL+F11)

Run as a java application

* [Use](#_Use)

When application start, it show request inputting number of calculation (total loop the formula with do). You need to input a number an press enter. Then application request to inputting Number of threads t (0 < t <MAX\_THREAD), it is size of thread pool. Finally, you need to input number of calculation each thread will handle. After you press enter button, the calculating process will begin, it will show some process information base on what IAlgorithm object is passed. A control command guide also appear

-----------------Control-Guide-------------------

Input 'p' to pause calculation

Input 'c' to continue calculation

Input 's' to stop calculation

Input 'next' to stop current calculation an start a new one

Input 'e' to exit program

You need to type exactly command string and press enter. When calculation finished it show a note look like:

Finished all threads. PI: 3.1425916543395442

Result in Math.PI: 3.141592653589793

Eslapsed Time: 6

You need to input command ‘next’ to make a new calculation

# Conclusion:

* Achievement: the program can calculate approximately the number PI with few precision digit. Using multithread and ensure the calculation will be right, return the number of completed calculation if stop when have not finished. Narrow down the instance of task made for calculation.
* [Expansion](#_Expansion): the program have quite loose coupling and each class have a particular functionality, user can change the algorithm to calculate number PI easily. Using decorator pattern made the application can extend more behavior when process. I realize that checking of input parameter have some problem but I doesn’t found a good solution to solve without changing code.