# Pi Calculator

Tran Huy Nghi

# Introduction:

## Problems

Build a program to calculate an approximation to the constant Pi (*π*).

Your solution must be developed in Java, and must be able to be built using either Maven or Ant. You may make use of any additional libraries required in order to aid development (e.g. Spring, JUnit etc.), so long as these libraries are not used in the calculation itself.

# Solutions

## Idea

* ***n****: number that user enter to calculate with the formula. (total number of calculation)*
* ***Task****: a scope to calculate.*
* ***x :*** number of calculation on each thread
* User’s input: n: total number of calculation, and the number of calculation on each thread.
* Application will calculate number of Task base on n and number x. Create List to save Tasks.
* Default number Thread (equal number of core in CPU). When application starts each thread will calculate a Task; after task finished, it saves result into List result and gets next Task from List.
* When finish all tasks in list then plus all result in list result to have number PI.
* When the application stopped, get result back.

*Example:* N = 19 - Process = 4 – Thread bound = 4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | … |
| Task 1 | | | | Task 2 | | | | Task 3 | | | | Task 4 | | | | List <Task > | | | |

## 2.2 Project Structure

There are 5 main packages in the programs:

|  |  |
| --- | --- |
| Package | Description |
| **com.tma.decorator** | FormulaDecorator and TimeFormulaPI used to create FormulaPI and decorator with run time. |
| **com.tma.factory** | *FormulaFactory*: Create Formula based on given information. |
| **com.tma.model** | *Number****:*** number N, **x**. |
| **com.tma.pi** | *App*: class create GUI and create formula based on given information. |
| **com.tma.service** | *Formula*: Interface of formula  *FormulaPI*: Executable calculation, create tasks from N, **x** and save into List<Task>  *ThreadPi:* Calculate pi with tasks. |

## 2.3 Detail design

### 2.3.1 Class diagram

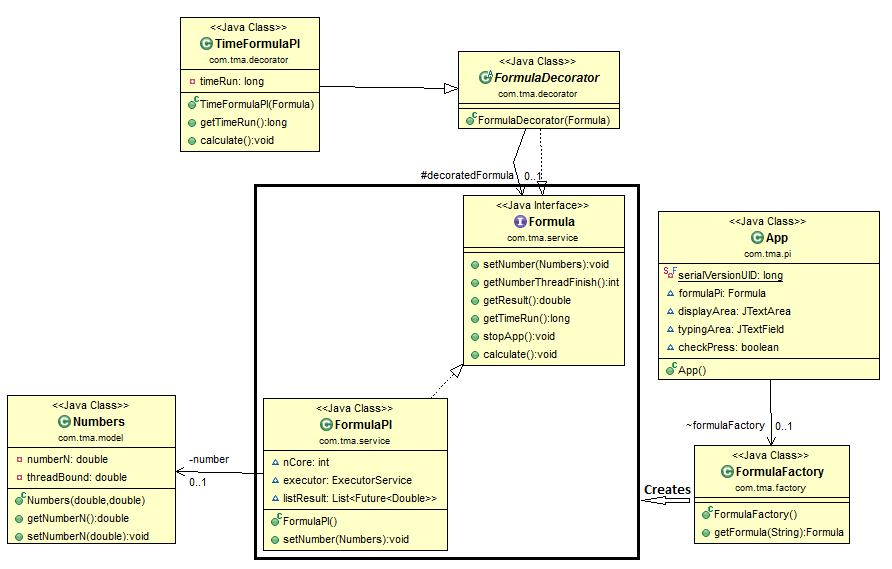
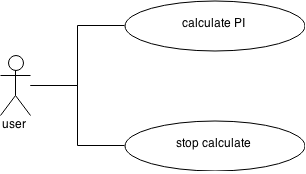


Figure 1: Class diagram

### 2.3.2 Usecase



The program let user can calculate the PI and stop the calculating process.

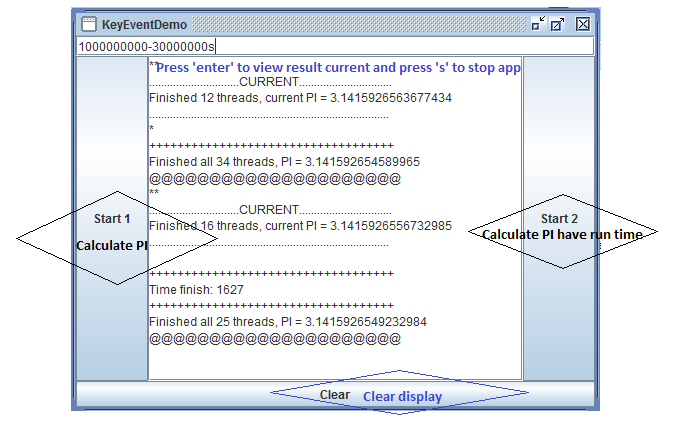
### 2.3.3 Design pattern

Factory pattern: expand application to calculate many formula other.

Decorator pattern: to decorator formula pi with time run and many other

# Program Instructions:

## GUI

****

## Run:

**Step 1:** Run Application

**Step 2:** Input n and thread bound in textbox (temple input n-threadBound: 1000-300)

**Step 3**: Click button ‘Start 1’ to calculate pi, click button ‘Start 2’ to calculate pi have time run

**Step 4:** View Result

To Stop App: press ‘s’

To view result current, press ‘Enter’

When Application run finish, have notify: @@@@@

**Step 5:** Click button “Clear” to clear display. Back step 1

# 4. Conclusion:

## 4.1 Expansion

My program can expand very easy such as:

* + Application can expand by adding many formulas with factory pattern and decorate for formula with decorator pattern.

## 4.2 Achievement

Calculate PI fast base on using multithreads.