CSE 437
REAL TIME SYSTEM ARCHITECTURES

HOMEWORK 01

REPORT

ÖMER ÇEVİK
161044004

In this project, I created a class which is called *Timer* class which implements *ITimer* interface and overrides its *registerTimer()* overloaded methods.

1. void registerTimer(const Timepoint &tp, const TTimerCallback &cb)

In this method, method calls *TTimerCallback cb()* function at time point *tp*. Using while loop to wait until time point *tp.* Before and after calling *cb()* function calculating the difference of time if *cb()* function runned less or greater than 5 milliseconds. If it is less or greater, printing the *"deadline error"* to screen.

2. void registerTimer(const Millisecs & period, const TTimerCallback &cb)

In this method, method calls *TTimerCallback cb()* function forever. Using while loop to wait forever. Using while loop to wait until period. Before and after calling *cb()* function calculating the difference of time if *cb()* function runned less or greater than 5 milliseconds. If it is less or greater, printing the *"deadline error"* to screen.

3. void registerTimer(const Timepoint &tp, const Millisecs & period, const TTimerCallback &cb)

In this method, method calls *TTimerCallback cb()* function until time point *tp*. Using while loop to wait until time point *tp.* Using while loop to wait until *period.* Before and after calling *cb()* function calculating the difference of time if *cb()* function runned less or greater than 5 milliseconds. If it is less or greater, printing the *"deadline error"* to screen.

4. void registerTimer(const TPredicate &pred, const Millisecs & period, const TTimerCallback &cb)

In this method, method calls *TTimerCallback cb()* function until *TPredicate pred()* returns false. Using while loop to wait until *pred()* returns false. Using while loop to wait until *period.* Before and after calling *cb()* function calculating the difference of time if *cb()* function runned less or greater than 5 milliseconds. If it is less or greater, printing the *"deadline error"* to screen.

5. int main() function

In this function, there are test operations. Creating *Time* class object and calls *four registerTimer()* methods using *std::thread* for each call. I used *rand()* function for *TPredicate* functor.

(*) I couldn't solve this problem using **single** thread. I didn't create time for it. I could create single thread in constructor and in *registerTimer()* methods I could store given parameters in a data structure. Using mutex and locking synchronization principle there might be a solution using single thread. The *join()* of thread might be in *destructor* of *Timer* class.