/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Ramin Edjlal.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Timer is Working Reversely\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RS\*\*\*\*\*0.12\*\*4\*\*Managements and Cuation Programing\*\*(+)

\* Timer Order Decreasing Not Work!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RS\*\*\*\*\*0.12\*\*4\*\*Managements and Cuation Programing\*\*(+)

\* Timer Not Worked.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RS\*\*\*\*\*0.12\*\*4\*\*Managements and Cuation Programing\*\*(+)

\* Timer Scheduling For Regard and Set Point Malfunctions.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RS\*\*\*\*\*0.12\*\*4\*\*Managements and Cuation Programing\*\*(+)

\* Timer Set Point of Text Malfunctioned.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RS\*\*\*\*\*0.12\*\*4\*\*Managements and Cuation Programing\*\*(+)

\* Thinking Finished Begin At New Time Text Box.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RS\*\*\*\*\*0.12\*\*4\*\*Managements and Cuation Programing\*\*(+)

\* Timer Changing Start Stop Function Failed.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RS\*\*\*\*\*0.12\*\*4\*\*Managements and Cuation Programing\*\*(+)

\* Timer MalFunction.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RS\*\*\*\*\*0.12\*\*4\*\*Managements and Cuation Programing\*\*(+)

\* Visual Studio Timer and Visualization du to Internet Access Malfunction\*\*RS\*\*\*\*\*0.12\*\*4\*\*Managements and Cuation Programing\*\*(+)

\* Dynamic Timer AStarGreedyt. First Increment or Decrement Malfunction.\*\*\*\*\*\*\*\*\*\*\*\*RS\*\*\*\*\*0.12\*\*4\*\*Managements and Cuation Programing\*\*(+)

\* No Logically Idea For Managements of Dynamic AStarGreedyt. First Max AStarGreedyt.\*\*\*\*\*\*\*RS\*\*\*\*\*0.12\*\*4\*\*Managements and Cuation Programing\*\*(+)

\* Timer Malfunction When Leave Foreground The Program.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RS\*\*\*\*\*0.12\*\*4\*\*Managements and Cuation Programing\*\*(+)

\* Divison By Zero No Reasonly.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RS\*\*\*\*\*0.12\*\*4\*\*Managements and Cuation Programing\*\*(+)

\* 1395/1/16\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Timer Not Worked.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RS\*\*\*\*\*0.12\*\*4\*\*Managements and Cuation Programing\*\*(+):(Not Set in this instatnt of analysis:Similarity is act.)

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.IO;

namespace Refrigtz

{

[Serializable]

public class Timer

{

//Initiate Variables. static and local for three timer.

public static int StoreAllDrawCount = 0;

public static bool UseDoubleTime = false;

public static long AStarGreedytiLevelMax = 0;

public static bool AStarGreadyFirstSearch;

long ConstTimer = 0;

double AStarGreedytMidleTimer = 0;

long AStarGreedytLastTime = 0;

public static bool Text = false;

public long Times = 5 \* 60 \* 1000;

long TimesBegin = 0;

public bool EndTime = false;

Thread t;

public bool Paused = true;

public bool TextChanged = true;

public int Sign = -1;

bool Infinity = false;

static void Log(Exception ex)

{

try

{

Object a = new Object();

lock (a)

{

string stackTrace = ex.ToString();

File.AppendAllText("ErrorProgramRun.txt", stackTrace + ": On" + DateTime.Now.ToString()); // path of file where stack trace will be stored.

}

}

catch (Exception t) { Log(t); }

}

//Constructive Tow Kind of Timer. Decreased timer and Incresed timer.

public Timer(bool SignPositive)

{

Object o = new Object();

lock (o)

{

//For Infinity Timer until end.

if (SignPositive)

{

Times = 0;

Sign = 1;

Infinity = true;

}

}

}

//Initiate Timer.

public void TimerInitiate()

{

Object o = new Object();

lock (o)

{

t = new Thread(new ThreadStart(TimerThread));

t.Start();

}

}

//Main Timer of Threading.

void TimerThread()

{

Object o = new Object();

lock (o)

{

do

{

//When timer stop sleep and checked for 500 ms.

while (Paused)

{

System.Threading.Thread.Sleep(1000);

};

//When timr begin store current time.

long t1 = DateTime.Now.Hour \* 3600000 + DateTime.Now.Minute \* 60000

+ DateTime.Now.Second \* 1000 + DateTime.Now.Millisecond;

do

{

System.Threading.Thread.Sleep(1000);

}

//Cal for every 1 second.

while (DateTime.Now.Hour \* 3600000 + DateTime.Now.Minute \* 60000

+ DateTime.Now.Second \* 1000 + DateTime.Now.Millisecond - t1 < 1000);

//Dec of inc one second.

Times = Times + 1000 \* Sign;

//Local Variabe of Timer changed.

//TextChanged = true;

//While Condition is true for operations.

} while (Times > 0 || Infinity);

}

//Indicating of end timer.

EndTime = true;

}

//Access to Private Timer Value of Long.

public long TimesAccess

{

get { return Times; }

set { Times = value; }

}

public long TimesConstAccess

{

get { return ConstTimer; }

set { ConstTimer = value; }

}

//AStarGreedyt First MAx Level Condition checked.

public int AStarGreedytiLevelMaxInitiate(Timer TimerColor, int AStarGreedyti)

{

Object o = new Object();

lock (o)

{

//int PowerEx = 4;

int Increase = 0;//Initaiate

Increase = 1;

//When Ok.

if (Sign != 1)

{

/\*if ((System.Math.Pow((AStarGreedytiLevelMax - StoreAllDrawCount) \* AStarGreedytMidleTimer, PowerEx) + System.Math.Pow(TimerColor.TimesAccess, PowerEx) > System.Math.Pow((AStarGreedytiLevelMax - StoreAllDrawCount) \* AStarGreedytMidleTimer, PowerEx) + System.Math.Pow((AStarGreedyti - StoreAllDrawCount) \* AStarGreedytMidleTimer, PowerEx)))

{

Increase = 1;

}

else//When is Cancled.

{

if ((System.Math.Pow((AStarGreedytiLevelMax - StoreAllDrawCount) \* AStarGreedytMidleTimer, PowerEx) + System.Math.Pow(TimerColor.TimesAccess, PowerEx) < System.Math.Pow((AStarGreedytiLevelMax - StoreAllDrawCount) \* AStarGreedytMidleTimer, PowerEx) + System.Math.Pow((AStarGreedyti - StoreAllDrawCount) \* AStarGreedytMidleTimer, PowerEx)))

{

Increase = -1;

}

}\*/

if (Times - 120000 < 0)

Increase = -1;

else

Increase = 1;

//Value

}

return Increase;

}

}

//Set AStarGreedyt First Level Max Variables.

public void SetAStarGreedytTimer()

{

Object o = new Object();

lock (o)

{

if (AStarGreedytLastTime == 0)

AStarGreedytLastTime = 0;

else

AStarGreedytLastTime = Times - AStarGreedytLastTime;

if (StoreAllDrawCount == 0)

AStarGreedytMidleTimer = 0;

}

}

//Cal Midle (Avarage) AStarGreedyt First Some static variables.

public void MidleAStarGreedytTimer(int AStarGreedyti)

{

Object o = new Object();

lock (o)

{

try

{

long Dummy = AStarGreedytLastTime;

AStarGreedytLastTime = Times - AStarGreedytLastTime;

//Division By Zero No Reasonaly.

AStarGreedytMidleTimer = ((Dummy \* (AStarGreedyti - StoreAllDrawCount)) + AStarGreedytLastTime) / ((AStarGreedyti - StoreAllDrawCount + 1));

}

catch (DivideByZeroException t)

{

Log(t);

}

}

}

//Strat timer function.

public void StartTime()

{

TextChanged = true;

Object o = new Object();

lock (o)

{

if (Sign != 1)

{

//Resume Suspended MAin Thread.

TimerInitiate();

//When Begin Timer Valuee is Zero cal.

if (TimesBegin == 0)

TimesBegin = DateTime.Now.Hour \* 3600000 + DateTime.Now.Minute \* 60000

+ DateTime.Now.Second \* 1000 + DateTime.Now.Millisecond;

}

//Set to Thread Paused.

Paused = false;

}

}

//Stop Timer.

public void StopTime()

{

Object o = new Object();

lock (o)

{

if (Sign != 1)

{

//When AStarGreedyt First is not act or Double time is not act.

if (!AStarGreadyFirstSearch || !UseDoubleTime)

{

//Cal Remaining timer value.

long Remaining = Times;

//When Remaining timer is greter than zero.

if (Remaining > 0)

Remaining = 0;

//When Regrad timer is valuable.

if ((DateTime.Now.Hour \* 3600000 + DateTime.Now.Minute \* 60000

+ DateTime.Now.Second \* 1000 + DateTime.Now.Millisecond - TimesBegin) < 5000)

Times = 5 \* 60 \* 1000 + 60000 + Remaining;

else

Times = 5 \* 60 \* 1000 + Remaining;

//Const timer value.

ConstTimer = 5 \* 60 \* 1000 + Remaining;

}

else

{

//Same as else.

long Remaining = Times;

if (Remaining > 0)

Remaining = 0;

if ((DateTime.Now.Hour \* 3600000 + DateTime.Now.Minute \* 60000

+ DateTime.Now.Second \* 1000 + DateTime.Now.Millisecond - TimesBegin) < 10000)

Times = 10 \* 60 \* 1000 + 60000 + Remaining;

else

Times = 10 \* 60 \* 1000 + Remaining;

ConstTimer = 10 \* 60 \* 1000 + Remaining;

}

TimesBegin = 0;

Paused = true;

//Suspend timer.

t.Abort();

}

Paused = true;

TextChanged = false;

}

}

public String ReturnTime()

{

//Cal and return timer string.

Object o = new Object();

lock (o)

{

long T = Times;

//Cal and return timer string.

String Houre = "0";

if (T >= 3600000)

{

Houre = ((System.Convert.ToInt64(T / 3600000))).ToString();

T = (T - System.Convert.ToInt64(T / 3600000) \* 3600000);

}

String Minute = "0";

if (T >= 60000)

{

Minute = ((System.Convert.ToInt64(T / 60000))).ToString();

T = (T - System.Convert.ToInt64(T / 60000) \* 60000);

}

String Second = (T / 1000).ToString();

return Houre + ":" + Minute + ":" + Second;

}

}

}

}