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

\* Ramin Edjlal Copyright 1397/04/20 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1397/04/26:Problem in Seirlization Recurisvely of linked list for refrigitz.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

using System;

using System.Collections.Generic;

using System.Text;

using System.Runtime.Serialization.Formatters.Binary;

using System.Runtime.Serialization;

using System.IO;

using RefrigtzDLL;

namespace GalleryStudio

{

[Serializable]

public class RefregizMemmory //:AllDraw

{

public int iii = 0, jjj = 0;

public bool MovementsAStarGreedyHuristicFoundT = false;

public bool IgnoreSelfObjectsT = false;

public bool UsePenaltyRegardMechnisamT = true;

public bool BestMovmentsT = false;

public bool PredictHuristicT = true;

public bool OnlySelfT = false;

public bool AStarGreedyHuristicT = false;

public bool ArrangmentsT = false;

const string SAllDraw = "AllDraw.asd";

public int Kind = 0;

static GalleryStudio.RefregizMemmory Node;

RefrigtzDLL.AllDraw Current = null;

public List<GalleryStudio.RefregizMemmory> NextS = null;

public List<GalleryStudio.RefregizMemmory> NextE = null;

public List<GalleryStudio.RefregizMemmory> NextH = null;

public List<GalleryStudio.RefregizMemmory> NextC = null;

public List<GalleryStudio.RefregizMemmory> NextM = null;

public List<GalleryStudio.RefregizMemmory> NextK = null;

bool NewListOfNextBegins = true;

static void Log(Exception ex)

{

try

{

Object a = new Object();

lock (a)

{

string stackTrace = ex.ToString();

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

}

}

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

}

public RefregizMemmory(bool MovementsAStarGreedyHuristicTFou, bool IgnoreSelfObject, bool UsePenaltyRegardMechnisa, bool BestMovment, bool PredictHurist, bool OnlySel, bool AStarGreedyHuris, bool Arrangments//) : base(MovementsAStarGreedyHuristicTFou, IgnoreSelfObject, UsePenaltyRegardMechnisa, BestMovment, PredictHurist, OnlySel, AStarGreedyHuris, Arrangments

)

{

NextS = new List<RefregizMemmory>();

NextE = new List<RefregizMemmory>();

NextH = new List<RefregizMemmory>();

NextC = new List<RefregizMemmory>();

NextM = new List<RefregizMemmory>();

NextK = new List<RefregizMemmory>();

Object o = new Object();

lock (o)

{

MovementsAStarGreedyHuristicFoundT = MovementsAStarGreedyHuristicTFou;

IgnoreSelfObjectsT = IgnoreSelfObject;

UsePenaltyRegardMechnisamT = UsePenaltyRegardMechnisa;

BestMovmentsT = BestMovment;

PredictHuristicT = PredictHurist;

OnlySelfT = OnlySel;

AStarGreedyHuristicT = AStarGreedyHuris;

ArrangmentsT = Arrangments;

}

}

//async

void RewriteAllDrawRec(BinaryFormatter Formatters, FileStream DummyFileStream, int Order)

{

Object o = new Object();

lock (o)

{

try

{

if (Current != null)

{

//Current.Clone(AllDrawCurrentAccess);

Formatters.Serialize(DummyFileStream, this);

//Kind = Kind;

/\* if (Order == 1)

{

//if (Kind == 1)

{

for (int i = 0; i < Current.SodierMidle; i++)

{

//Formatters.Serialize(DummyFileStream, AllDrawCurrentAccess);

try

{

for (int j = 0; j < Current.SolderesOnTable[i].SoldierThinking[0].AStarGreedy.Count; j++)

{

Object O = new Object();

lock (O)

{

iii = i;

jjj = j;

RefregizMemmory tCurrent = ReterunAstrarGreedysolder(i, j, this);

tCurrent.NewListOfNextBegins = false;

tCurrent.RewriteAllDrawRec(Formatters, DummyFileStream,Current.SolderesOnTable[i].SoldierThinking[0].AStarGreedy[j], Order \* -1);

}

}

}

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

}

}

//else if (Kind == 2)

{

for (int i = 0; i < Current.ElefantMidle; i++)

{

try

{

//Formatters.Serialize(DummyFileStream, AllDrawCurrentAccess);

for (int j = 0; j < Current.ElephantOnTable[i].ElefantThinking[0].AStarGreedy.Count; j++)

{

Object O = new Object();

lock (O)

{

iii = i;

jjj = j;

RefregizMemmory tCurrent = ReterunAstrarGreedyelephant(i, j, this);

tCurrent.NewListOfNextBegins = false;

tCurrent.RewriteAllDrawRec(Formatters, DummyFileStream,Current.ElephantOnTable[i].ElefantThinking[0].AStarGreedy[j], Order \* -1);

}

}

}

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

}

}

//else if (Kind == 3)

{

for (int i = 0; i < Current.HourseMidle; i++)

{

//Formatters.Serialize(DummyFileStream, AllDrawCurrentAccess);

try

{

for (int j = 0; j < Current.HoursesOnTable[i].HourseThinking[0].AStarGreedy.Count; j++)

{

Object O = new Object();

lock (O)

{

iii = i;

jjj = j;

RefregizMemmory tCurrent = ReterunAstrarGreedyHours(i, j, this);

tCurrent.NewListOfNextBegins = false;

tCurrent.RewriteAllDrawRec(Formatters, DummyFileStream,Current.HoursesOnTable[i].HourseThinking[0].AStarGreedy[j], Order \* -1);

}

}

}

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

}

}

//else if (Kind == 4)

{

for (int i = 0; i < Current.CastleMidle; i++)

{

//Formatters.Serialize(DummyFileStream, AllDrawCurrentAccess);

try

{

for (int j = 0; j < Current.CastlesOnTable[i].CastleThinking[0].AStarGreedy.Count; j++)

{

Object O = new Object();

lock (O)

{

iii = i;

jjj = j;

RefregizMemmory tCurrent = ReterunAstrarGreedyCastle(i, j, this);

tCurrent.NewListOfNextBegins = false;

tCurrent.RewriteAllDrawRec(Formatters, DummyFileStream,Current.CastlesOnTable[i].CastleThinking[0].AStarGreedy[j], Order \* -1);

}

}

}

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

}

}

// else if (Kind == 5)

{

for (int i = 0; i < Current.MinisterMidle; i++)

{

//Formatters.Serialize(DummyFileStream, AllDrawCurrentAccess);

try

{

for (int j = 0; j < Current.MinisterOnTable[i].MinisterThinking[0].AStarGreedy.Count; j++)

{

Object O = new Object();

lock (O)

{

iii = i;

jjj = j;

RefregizMemmory tCurrent = ReterunAstrarGreedyMinister(i, j, this);

tCurrent.NewListOfNextBegins = false;

tCurrent.RewriteAllDrawRec(Formatters, DummyFileStream,Current.MinisterOnTable[i].MinisterThinking[0].AStarGreedy[j], Order \* -1);

}

}

}

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

}

}

//else if (Kind == 6)

{

for (int i = 0; i < Current.KingMidle; i++)

{

//Formatters.Serialize(DummyFileStream, AllDrawCurrentAccess);

try

{

for (int j = 0; j < Current.KingOnTable[i].KingThinking[0].AStarGreedy.Count; j++)

{

Object O = new Object();

lock (O)

{

iii = i;

jjj = j;

RefregizMemmory tCurrent = ReterunAstrarGreedyKing(i, j, this);

tCurrent.NewListOfNextBegins = false;

tCurrent.RewriteAllDrawRec(Formatters, DummyFileStream,Current.KingOnTable[i].KingThinking[0].AStarGreedy[j], Order \* -1);

}

}

}

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

}

}

}

else

{

//if (Kind == 1)

{

for (int i = Current.SodierMidle; i < Current.SodierHigh; i++)

{

//Formatters.Serialize(DummyFileStream, AllDrawCurrentAccess);

try

{

for (int j = 0; j < Current.SolderesOnTable[i].SoldierThinking[0].AStarGreedy.Count; j++)

{

Object O = new Object();

lock (O)

{

iii = i;

jjj = j;

RefregizMemmory tCurrent = ReterunAstrarGreedysolder(i, j, this);

tCurrent.NewListOfNextBegins = false;

tCurrent.RewriteAllDrawRec(Formatters, DummyFileStream,Current.SolderesOnTable[i].SoldierThinking[0].AStarGreedy[j], Order \* -1);

}

}

}

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

}

}

//else if (Kind == 2)

{

for (int i = Current.ElefantMidle; i < Current.ElefantHigh; i++)

{

//Formatters.Serialize(DummyFileStream, AllDrawCurrentAccess);

try

{

for (int j = 0; j < Current.ElephantOnTable[i].ElefantThinking[0].AStarGreedy.Count; j++)

{

Object O = new Object();

lock (O)

{

iii = i;

jjj = j;

RefregizMemmory tCurrent = ReterunAstrarGreedyelephant(i, j, this);

tCurrent.NewListOfNextBegins = false;

tCurrent.RewriteAllDrawRec(Formatters, DummyFileStream,Current.ElephantOnTable[i].ElefantThinking[0].AStarGreedy[j], Order \* -1);

}

}

}

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

}

}

//else if (Kind == 3)

{

for (int i = Current.HourseMidle; i < Current.HourseHight; i++)

{

//Formatters.Serialize(DummyFileStream, AllDrawCurrentAccess);

try

{

for (int j = 0; j < Current.HoursesOnTable[i].HourseThinking[0].AStarGreedy.Count; j++)

{

Object O = new Object();

lock (O)

{

iii = i;

jjj = j;

RefregizMemmory tCurrent = ReterunAstrarGreedyHours(i, j, this);

tCurrent.NewListOfNextBegins = false;

tCurrent.RewriteAllDrawRec(Formatters, DummyFileStream,Current.HoursesOnTable[i].HourseThinking[0].AStarGreedy[j], Order \* -1);

}

}

}

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

}

}

//else if (Kind == 4)

{

for (int i = Current.CastleMidle; i < Current.CastleHigh; i++)

{

//Formatters.Serialize(DummyFileStream, AllDrawCurrentAccess);

try

{

for (int j = 0; j < Current.CastlesOnTable[i].CastleThinking[0].AStarGreedy.Count; j++)

{

Object O = new Object();

lock (O)

{

iii = i;

jjj = j;

RefregizMemmory tCurrent = ReterunAstrarGreedyCastle(i, j, this);

tCurrent.NewListOfNextBegins = false;

tCurrent.RewriteAllDrawRec(Formatters, DummyFileStream,Current.CastlesOnTable[i].CastleThinking[0].AStarGreedy[j], Order \* -1);

}

}

}

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

}

}

//else if (Kind == 5)

{

for (int i = Current.MinisterMidle; i < Current.MinisterHigh; i++)

{

//Formatters.Serialize(DummyFileStream, AllDrawCurrentAccess);

try

{

for (int j = 0; j < Current.MinisterOnTable[i].MinisterThinking[0].AStarGreedy.Count; j++)

{

Object O = new Object();

lock (O)

{

iii = i;

jjj = j;

RefregizMemmory tCurrent = ReterunAstrarGreedyMinister(i, j, this);

tCurrent.NewListOfNextBegins = false;

tCurrent.RewriteAllDrawRec(Formatters, DummyFileStream,Current.MinisterOnTable[i].MinisterThinking[0].AStarGreedy[j], Order \* -1);

}

}

}

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

}

}

// else if (Kind == 6)

{

for (int i = Current.KingMidle; i < Current.MinisterHigh; i++)

{

//Formatters.Serialize(DummyFileStream, AllDrawCurrentAccess);

try

{

for (int j = 0; j < Current.KingOnTable[i].KingThinking[0].AStarGreedy.Count; j++)

{

Object O = new Object();

lock (O)

{

iii = i;

jjj = j;

RefregizMemmory tCurrent = ReterunAstrarGreedyKing(i, j, this);

tCurrent.NewListOfNextBegins = false;

tCurrent.RewriteAllDrawRec(Formatters, DummyFileStream,Current.KingOnTable[i].KingThinking[0].AStarGreedy[j], Order \* -1);

}

}

}

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

}

}

}\*/

}

}

catch (Exception tt)

{

Log(tt);

}

/\*while (t != null)

{

//Formatters.Serialize(DummyFileStream, AllDrawCurrentAccess);

t = AllDrawNextAccess;

}\*/

}

}

public RefregizMemmory CloneSphycose(RefregizMemmory t)

{

t = new RefregizMemmory(MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT);

t.iii = iii;

t.jjj = jjj;

t.MovementsAStarGreedyHuristicFoundT = MovementsAStarGreedyHuristicFoundT;

t.IgnoreSelfObjectsT = IgnoreSelfObjectsT;

t.UsePenaltyRegardMechnisamT = UsePenaltyRegardMechnisamT;

t.BestMovmentsT = BestMovmentsT;

t.PredictHuristicT = PredictHuristicT;

t.OnlySelfT = OnlySelfT;

t.AStarGreedyHuristicT = AStarGreedyHuristicT;

t.ArrangmentsT = ArrangmentsT;

t.Kind = Kind;

Current.Clone(t.Current);

for (int i = 0; i < NextS.Count; i++)

{

t.NextS.Add(new RefregizMemmory(MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT));

NextS[i].CloneSphycose(t.NextS[i]);

}

for (int i = 0; i < NextE.Count; i++)

{

t.NextE.Add(new RefregizMemmory(MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT));

NextE[i].CloneSphycose(t.NextE[i]);

}

for (int i = 0; i < NextH.Count; i++)

{

t.NextH.Add(new RefregizMemmory(MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT));

NextH[i].CloneSphycose(t.NextH[i]);

}

for (int i = 0; i < NextC.Count; i++)

{

t.NextC.Add(new RefregizMemmory(MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT));

NextC[i].CloneSphycose(t.NextC[i]);

}

for (int i = 0; i < NextM.Count; i++)

{

t.NextM.Add(new RefregizMemmory(MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT));

NextM[i].CloneSphycose(t.NextM[i]);

}

for (int i = 0; i < NextK.Count; i++)

{

t.NextK.Add(new RefregizMemmory(MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT));

NextK[i].CloneSphycose(t.NextK[i]);

}

t.NewListOfNextBegins = NewListOfNextBegins;

return t;

}

public void RewriteAllDraw(int Order)

{

Object oo = new Object();

lock (oo)

{

//Current = new RefrigtzDLL.AllDraw(MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT);

FileStream DummyFileStream = null;

try

{

//RefregizMemmory t = p;

FileInfo DummyFileInfo = new FileInfo(SAllDraw);

DummyFileInfo.Delete();

DummyFileStream = new FileStream(SAllDraw, System.IO.FileMode.OpenOrCreate, System.IO.FileAccess.Write);

BinaryFormatter Formatters = new BinaryFormatter();

DummyFileStream.Seek(0, SeekOrigin.Begin);

//Formatters.Serialize(DummyFileStream, t);

RewriteAllDrawRec(Formatters, DummyFileStream, Order);

DummyFileStream.Flush(); DummyFileStream.Close();

}

catch (Exception o)

{

Console.WriteLine(o.Message.ToString());

}

}

}

public AllDraw Load(int Order)

{

Object o = new Object();

lock (o)

{

//Node.AllDrawNextAccessS = null;

//Node.AllDrawNextAccessE = null;

//Node.AllDrawNextAccessH = null;

//Node.AllDrawNextAccessC = null;

//Node.AllDrawNextAccessM = null;

//Node.AllDrawNextAccessK = null;

//Node.AllDrawCurrentAccess = null;

AllDraw t = null;

try

{

FileStream DummyFileStream = new FileStream(SAllDraw, System.IO.FileMode.OpenOrCreate, System.IO.FileAccess.ReadWrite);

BinaryFormatter Formatters = new BinaryFormatter();

Console.WriteLine("Loading...");

DummyFileStream.Seek(0, SeekOrigin.Begin);

t = LoadrEC(Order, this, DummyFileStream, Formatters);

DummyFileStream.Flush();

DummyFileStream.Close();

}

catch (IOException tt) { Log(tt); }

return t;

//return Node.al;

}

}

public AllDraw LoadrEC(int Order, GalleryStudio.RefregizMemmory Last, FileStream DummyFileStream, BinaryFormatter Formatters)

{

RefregizMemmory Dummy = null;

Object o = new Object();

lock (o)

{

//Node.AllDrawNextAccessS = null;

//Node.AllDrawNextAccessE = null;

//Node.AllDrawNextAccessH = null;

//Node.AllDrawNextAccessC = null;

//Node.AllDrawNextAccessM = null;

//Node.AllDrawNextAccessK = null;

//Node.AllDrawCurrentAccess = null;

try

{

//NEWNOD = Node.AllDrawCurrentAccess;

while (DummyFileStream.Position < DummyFileStream.Length)

{

Dummy = (RefregizMemmory)Formatters.Deserialize(DummyFileStream);

//Dummy.CloneSphycose(Last);

/\*{

//Last = Node;

if (Dummy.NextS.Count > 0 && Dummy.NewListOfNextBegins)

{

do

{

if (DummyFileStream.Position < DummyFileStream.Length)

Dummy = (RefregizMemmory)Formatters.Deserialize(DummyFileStream);

else

break;

Last.NextS.Add(Dummy);

} while (!Dummy.NewListOfNextBegins);

for (int i = 0; i < Last.NextS.Count; i++)

Last.NextS[i].Load(Order \* -1, Last.NextS[i]);

}

else

if (Dummy.NextE.Count > 0 && Dummy.NewListOfNextBegins)

{

do

{

if (DummyFileStream.Position < DummyFileStream.Length)

Dummy = (RefregizMemmory)Formatters.Deserialize(DummyFileStream);

else

break;

Last.NextE.Add(Dummy);

} while (!Dummy.NewListOfNextBegins);

for (int i = 0; i < Last.NextE.Count; i++)

Last.NextE[i].Load(Order \* -1, Last.NextE[i]);

}

else

if (Dummy.NextH.Count > 0 && Dummy.NewListOfNextBegins)

{

do

{

if (DummyFileStream.Position < DummyFileStream.Length)

Dummy = (RefregizMemmory)Formatters.Deserialize(DummyFileStream);

else

break;

Last.NextH.Add(Dummy);

} while (!Dummy.NewListOfNextBegins);

for (int i = 0; i < Last.NextH.Count; i++)

Last.NextH[i].Load(Order \* -1, Last.NextH[i]);

}

else

if (Dummy.NextC.Count > 0 && Dummy.NewListOfNextBegins)

{

do

{

if (DummyFileStream.Position < DummyFileStream.Length)

Dummy = (RefregizMemmory)Formatters.Deserialize(DummyFileStream);

else

break;

Last.NextC.Add(Dummy);

} while (!Dummy.NewListOfNextBegins);

for (int i = 0; i < Last.NextC.Count; i++)

Last.NextC[i].Load(Order \* -1, Last.NextC[i]);

}

else

if (Dummy.NextM.Count > 0 && Dummy.NewListOfNextBegins)

{

do

{

if (DummyFileStream.Position < DummyFileStream.Length)

Dummy = (RefregizMemmory)Formatters.Deserialize(DummyFileStream);

else

break;

Last.NextM.Add(Dummy);

} while (!Dummy.NewListOfNextBegins);

for (int i = 0; i < Last.NextM.Count; i++)

Last.NextM[i].Load(Order \* -1, Last.NextM[i]);

}

else

if (Dummy.NextK.Count > 0 && Dummy.NewListOfNextBegins)

{

do

{

if (DummyFileStream.Position < DummyFileStream.Length)

Dummy = (RefregizMemmory)Formatters.Deserialize(DummyFileStream);

else

break;

Last.NextK.Add(Dummy);

} while (!Dummy.NewListOfNextBegins);

for (int i = 0; i < Last.NextK.Count; i++)

Last.NextK[i].Load(Order \* -1, Last.NextK[i]);

}

}

\*/

}

}

catch (IOException tt) { Log(tt); }

//return CreateAllDrawFromMemmory(Last, new AllDraw(MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT), Order);

return Dummy.Current;

//return Node.al;

}

}

AllDraw CreateAllDrawFromMemmory(RefregizMemmory t, AllDraw Last,int Order)

{

Object o = new Object();

lock (o)

{

if (t == null)

return null;

else

t.Current = Last;

try

{

/\*if (t.NextS.Count > 0)

{

for (int i = 0; i < t.NextS.Count; i++)

{

t.NextS[i].Current.Clone(Last.SolderesOnTable[t.iii].SoldierThinking[0].AStarGreedy[t.jjj]);

t.NextS[i].CreateAllDrawFromMemmory(t.NextS[i], Last.ElephantOnTable[t.iii].ElefantThinking[0].AStarGreedy[t.jjj], Order \* -1);

}

}

else

if (t.NextE.Count > 0)

{

for (int i = 0; i < t.NextE.Count; i++)

{

t.NextE[i].Current.Clone(Last.ElephantOnTable[t.iii].ElefantThinking[0].AStarGreedy[t.jjj]);

t.NextE[i].CreateAllDrawFromMemmory(t.NextE[i], Last.ElephantOnTable[t.iii].ElefantThinking[0].AStarGreedy[t.jjj], Order \* -1);

}

}

if (t.NextH.Count > 0)

{

for (int i = 0; i < t.NextH.Count; i++)

{

t.NextH[i].Current.Clone(Last.HoursesOnTable[t.iii].HourseThinking[0].AStarGreedy[t.jjj]);

t.NextH[i].CreateAllDrawFromMemmory(t.NextH[i], Last.HoursesOnTable[t.iii].HourseThinking[0].AStarGreedy[t.jjj], Order \* -1);

}

}

if (t.NextC.Count > 0)

{

for (int i = 0; i < t.NextC.Count; i++)

{

t.NextC[i].Current.Clone(Last.CastlesOnTable[t.iii].CastleThinking[0].AStarGreedy[t.jjj]);

t.NextC[i].CreateAllDrawFromMemmory(t.NextC[i], Last.CastlesOnTable[t.iii].CastleThinking[0].AStarGreedy[t.jjj], Order \* -1);

}

}

if (t.NextM.Count > 0)

{

for (int i = 0; i < t.NextM.Count; i++)

{

t.NextM[i].Current.Clone(Last.MinisterOnTable[t.iii].MinisterThinking[0].AStarGreedy[t.jjj]);

t.NextM[i].CreateAllDrawFromMemmory(t.NextM[i], Last.MinisterOnTable[t.iii].MinisterThinking[0].AStarGreedy[t.jjj], Order \* -1);

}

}

if (t.NextK.Count > 0)

{

for (int i = 0; i < t.NextK.Count; i++)

{

t.NextK[i].Current.Clone(Last.KingOnTable[t.iii].KingThinking[0].AStarGreedy[t.jjj]);

t.NextK[i].CreateAllDrawFromMemmory(t.NextK[i], Last.KingOnTable[t.iii].KingThinking[0].AStarGreedy[t.jjj], Order \* -1);

}

}

\*/

}

catch (IOException tt) { Log(tt); }

return t.Current;

}

}

/\* public void DeleteObject(RefregizMemmory p)

{

RefregizMemmory t = new RefregizMemmory(MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT);

t = Node;

if ((t.c) != (p.AllDrawCurrentAccess.RefrigtzDLL.AllDrawName))

{

if (t != null)

while ((t.AllDrawNextAccess.AllDrawCurrentAccess.RefrigtzDLL.AllDrawName) != (p.AllDrawCurrentAccess.RefrigtzDLL.AllDrawName))

{

if (t.AllDrawNextAccess != null)

t = t.AllDrawNextAccess;

else

if ((t.AllDrawCurrentAccess.RefrigtzDLL.AllDrawName) != (p.AllDrawCurrentAccess.RefrigtzDLL.AllDrawName))

{

t = null;

break;

}

}

if (t != null)

{

if (t.AllDrawNextAccess != null)

t.AllDrawNextAccess = t.AllDrawNextAccess.AllDrawNextAccess;

else

t.AllDrawNextAccess = null;

}

}

else

{

t = t.AllDrawNextAccess;

Node = t;

}

}

\*/

/\*public void AddObject(RefregizMemmory p)

{

RefregizMemmory t = new catch (IOException tt) {Log(tt); }MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT

);

t = p.AllDrawNodeAccess;

while (t.AllDrawNextAccess != null)

t = t.AllDrawNextAccess;

if (t.AllDrawCurrentAccess == null)

t.AllDrawCurrentAccess = p.AllDrawCurrentAccess;

else

t.AllDrawNextAccess = p;

}\*/

public RefregizMemmory AllDrawNodeAccess

{

get

{ return Node; }

set

{ Node = value; }

}

public AllDraw AllDrawCurrentAccess

{

get

{ return Current; }

set

{ Current = value; }

}

public int OrderPlateCurrentAccess

{

get

{ return Current.OrderP; }

set

{ Current.OrderP = value; }

}

public RefregizMemmory ReterunAstrarGreedysolder(int i, int j, RefregizMemmory t)

{

if (t.Current.SolderesOnTable[i].SoldierThinking[0].AStarGreedy.Count > j && j > 0)

{

Kind = 1;

t.NextS.Add(new RefregizMemmory(MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT));

t.Current.SolderesOnTable[i].SoldierThinking[0].AStarGreedy[j].Clone(t.NextS[j].Current);

}

return t.AllDrawNextS(j);

}

public RefregizMemmory ReterunAstrarGreedyelephant(int i, int j, RefregizMemmory t)

{

if (t.Current.ElephantOnTable[i].ElefantThinking[0].AStarGreedy.Count > j && j > 0)

{

Kind = 2;

t.NextE.Add( new RefregizMemmory(MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT));

t.Current.ElephantOnTable[i].ElefantThinking[0].AStarGreedy[j].Clone(t.NextE[j].Current);

}

return t.AllDrawNextE(j);

}

public RefregizMemmory ReterunAstrarGreedyHours(int i, int j, RefregizMemmory t)

{

if (t.Current.HoursesOnTable[i].HourseThinking[0].AStarGreedy.Count > j && j > 0)

{

Kind = 3;

t.NextH.Add(new RefregizMemmory(MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT));

t.Current.HoursesOnTable[i].HourseThinking[0].AStarGreedy[j].Clone(t.NextH[j].Current);

}

return t.AllDrawNextH(j);

}

public RefregizMemmory ReterunAstrarGreedyCastle(int i, int j, RefregizMemmory t)

{

if (t.Current.CastlesOnTable[i].CastleThinking[0].AStarGreedy.Count > j && j > 0)

{

Kind = 4;

t.NextC.Add(new RefregizMemmory(MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT));

t.Current.CastlesOnTable[i].CastleThinking[0].AStarGreedy[j].Clone(t.NextC[j].Current);

}

return t.AllDrawNextC(j);

}

public RefregizMemmory ReterunAstrarGreedyMinister(int i, int j, RefregizMemmory t)

{

if (t.Current.MinisterOnTable[i].MinisterThinking[0].AStarGreedy.Count > j && j > 0)

{

Kind = 5;

t.NextM.Add(new RefregizMemmory(MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT));

t.Current.MinisterOnTable[i].MinisterThinking[0].AStarGreedy[j].Clone(t.NextM[j].Current);

}

return t.AllDrawNextM(j);

}

public RefregizMemmory ReterunAstrarGreedyKing(int i, int j, RefregizMemmory t)

{

if (t.Current.KingOnTable[i].KingThinking[0].AStarGreedy.Count > j && j > 0)

{

Kind = 6;

t.NextK.Add( new RefregizMemmory(MovementsAStarGreedyHuristicFoundT, IgnoreSelfObjectsT, UsePenaltyRegardMechnisamT, BestMovmentsT, PredictHuristicT, OnlySelfT, AStarGreedyHuristicT, ArrangmentsT));

t.Current.KingOnTable[i].KingThinking[0].AStarGreedy[j].Clone(t.NextK[j].Current);

}

return t.AllDrawNextK(j);

}

public RefregizMemmory AllDrawNextS(int i)

{

return NextS[i];

}

public RefregizMemmory AllDrawNextE(int i)

{

return NextE[i];

}

public RefregizMemmory AllDrawNextH(int i)

{

return NextH[i];

}

public RefregizMemmory AllDrawNextC(int i)

{

return NextC[i];

}

public RefregizMemmory AllDrawNextM(int i)

{

return NextM[i];

}

public RefregizMemmory AllDrawNextK(int i)

{

return NextK[i];

}

/\*public RefregizMemmory AllDrawNextAccessS

{

get

{ return NextS; }

set

{ NextS = value; }

}

public RefregizMemmory AllDrawNextAccessE

{

get

{ return NextE; }

set

{ NextE = value; }

}

public RefregizMemmory AllDrawNextAccessH

{

get

{ return NextH; }

set

{ NextH = value; }

}

public RefregizMemmory AllDrawNextAccessC

{

get

{ return NextC; }

set

{ NextC = value; }

}

public RefregizMemmory AllDrawNextAccessM

{

get

{ return NextM; }

set

{ NextM = value; }

}

public RefregizMemmory AllDrawNextAccessK

{

get

{ return NextK; }

set

{ NextK = value; }

}

\*/

}

}