* canChangeLine / station
* addTwoFollowingStations
* Element at ולהוריד GET
* מחיקת תחנה כשנשאר אחת בקו
* מספר רץ – מחלקה או XML?
* Get עם תנאי – יש דרך לקרוא ל-DL?
* מחלקת השעון להשאיר ב PL?

*שלב ג - החלפת שכבת הDAL מ DLObject ל DLXML לפחות ישות אחת עם XElement, השאר אפשר בסריאליזציה.*

*האוטובוס יסע בתהליכונים מסוימים - כלומר תהיה נסיעה של האוטובוס כיישות של BO, וכל פעם שתותחל נסיעה לפי לוח זמנים יווצר תהליכון. כמו כן יהיה אלמנט של רנדומנילזציה, כלומר הזמנים לא יהיו מדוייקים לפי לוח הזמנים, אלא בטווח אקראי כלשהו. לכן נצטרך כל הזמן לבדוק איפה הוא בפועל כדי לחשב נסיעה בין תחנות. הנסיעה תהיה סוג של ארוע -לפי תבנית המשקיף ( observer) וכל מי שרוצה לקבל ממנה מידע כדי לדעת את הנסיעה, ירשם לאירוע. א. צריך גם את יישות לוח הזמנים בשכבת הDO. אבל רק אחרי שכל שאר הפרויקט עשוי. ב. תהיה יישות נסיעה של אוטובוס, ושאילתא של נסיעת נוסע מתחנה לתחנה - בשכבת הBL.*

*אם רוצים ישות משתמש – בונוס – אז שומרים את היסטוריית כל הבקשות שלו.*

בסוף:

* בונוסים אפשריים: (אוטובוס) + סימון למחיקה (לא כולל תחנות עוקבות) + שיבוט + הצפנה
* לתעד בונוסים
* לתעד תוכנית
* לעבור על ההוראות של התרגיל
* למחוק קבצים מיותרים

//#region DrivingBuses

//public void addDrivingBus(DrivingBus drivingBus)

//{

// try

// {

// getBus(drivingBus.LicensePlate); // check if the bus exists

// drivingBus.ThisSerial = DataSource.serial++;

// DataSource.DrivingBuses.Add(drivingBus.Clone());

// }

// catch (BusException ex)

// {

// throw new BusException(ex.Message);

// }

//}

//public void removeDrivingBus(DrivingBus drivingBus)

//{

// DrivingBus d = DataSource.DrivingBuses.Find(item => item.ThisSerial == drivingBus.ThisSerial && item.LicensePlate == drivingBus.LicensePlate && item.Line == drivingBus.Line && item.Start == drivingBus.Start);

// if (d == null)

// throw new BusException("The driving bus does not exist.");

// DataSource.DrivingBuses.Remove(d); // remove the old driving bus

//}

//public void updateDrivingBus(DrivingBus drivingBus)

//{

// removeDrivingBus(drivingBus); // remove the old driving bus

// DataSource.DrivingBuses.Add(drivingBus.Clone()); // add the updated driving bus

//}

//public DrivingBus getDrivingBus(int thisSerial, string licensePlate, int line, DateTime start)

//{

// DrivingBus drivingBus = DataSource.DrivingBuses.Find(item => item.ThisSerial == thisSerial && item.LicensePlate == licensePlate && item.Line == line && item.Start == start);

// if (drivingBus == null)

// return null;

// return drivingBus.Clone();

//}

//public IEnumerable<DrivingBus> GetDrivingBuses()

//{

// return from item in DataSource.DrivingBuses

// select item.Clone();

//}

//public IEnumerable<DrivingBus> GetDrivingBuses(Predicate<DrivingBus> condition)

//{

// return from item in DataSource.DrivingBuses

// where condition(item)

// select item.Clone();

//}

//#endregion

מ-BLIMP:

#region DrivingBuses

/// <summary>

/// Func that converts driving bus of BO to driving bus of DO

/// </summary>

/// <param name="drivingBus">driving bus of BO</param>

/// <returns>driving bus of DO</returns>

DrivingBus convertToDrivingBusDO(BO.DrivingBus drivingBus)

{

return new DrivingBus()

{

ThisSerial = drivingBus.ThisSerial,

Line = drivingBus.Line,

LicensePlate = drivingBus.LicensePlate,

ActualStart = drivingBus.ActualStart,

Start = drivingBus.Start,

PreviousStationID = drivingBus.PreviousStationID,

PreviousStationTime = drivingBus.PreviousStationTime,

NextStationTime = drivingBus.NextStationTime

};

}

/// <summary>

/// Func that converts driving bus of DO to driving bus of BO

/// </summary>

/// <param name="drivingBus">driving bus of DO</param>

/// <returns>driving bus of BO</returns>

BO.DrivingBus convertToDrivingBusBO(DrivingBus drivingBus)

{

return new BO.DrivingBus()

{

LicensePlate = drivingBus.LicensePlate,

Line = drivingBus.Line,

Start = drivingBus.Start,

ThisSerial = drivingBus.ThisSerial,

ActualStart = drivingBus.ActualStart,

PreviousStationID = drivingBus.PreviousStationID,

PreviousStationTime = drivingBus.PreviousStationTime,

NextStationTime = drivingBus.NextStationTime

};

}

public void addDrivingBus(BO.DrivingBus drivingBus)

{

try

{

dal.addDrivingBus(convertToDrivingBusDO(drivingBus));

}

catch (BusException ex)

{

throw new BO.BusException(ex.Message, ex);

}

}

public void removeDrivingBus(BO.DrivingBus drivingBus)

{

try

{

dal.removeDrivingBus(convertToDrivingBusDO(drivingBus));

}

catch (BusException ex)

{

throw new BO.BusException(ex.Message, ex);

}

}

public void updateDrivingBus(BO.DrivingBus drivingBus)

{

try

{

dal.updateDrivingBus(convertToDrivingBusDO(drivingBus));

}

catch (BusException ex)

{

throw new BO.BusException(ex.Message, ex);

}

}

public BO.DrivingBus getDrivingBus(int thisSerial, string licensePlate, int line, DateTime start)

{

try

{

return convertToDrivingBusBO(dal.getDrivingBus(thisSerial, licensePlate, line, start));

}

catch (BusException ex)

{

throw new BO.BusException(ex.Message, ex);

}

}

public IEnumerable<BO.DrivingBus> GetDrivingBuses()

{

try

{

return from drivingBus in dal.GetDrivingBuses()

select convertToDrivingBusBO(drivingBus);

}

catch (BusException ex)

{

throw new BO.BusException(ex.Message, ex);

}

}

public IEnumerable<BO.DrivingBus> GetDrivingBuses(Predicate<BO.DrivingBus> condition)

{

try

{

return from item in GetDrivingBuses()

where condition(item)

select item;

}

catch (BO.BusException ex)

{

throw new BO.BusException(ex.Message);

}

}

#endregion