

Re: Help with class design

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2005-05/msg02778.html>

- *From:* "Joanna Carter \ (TeamB\)" <joannac@xxxxxxxxxxxxxxxxxx>
 - *Date:* Sun, 15 May 2005 20:16:50 +0100
-

"JSheble" <jsheble-NOSPAM@xxxxxxxxxxxxx> a écrit dans le message de news:
#Kyv#uWWFHA.580@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

- > I come from a Delphi background, and am currently trying to port or convert
- > some of our Delphi classes to C#. I've got a good handle on basic class
- > design, but am a bit lost with some of the more advanced things I need to
- > do, and was hoping somebody could point me to an online resource that can
- > help me better understand what I need to do.

There really is no difference in how you would design classes in C# as opposed to Delphi. If you got the design right in Delphi, then it should be fine in C# :-)

- > For example, one of the classes is a Shipment class. A Shipment can have
- > any number of Packages, so I have designed a Shipment Class and a Package
- > class. I should be able to retrieve any of the packages by it's index or
- I
- > should be able to use a single package as a property. For example (these
- > examples are not working code, merely pseudo-code to demonstrate what I
- > need):

You need to start off by asking yourself whether a Package can exist outside of a Shipment e.g. can a Package be stored in a Warehouse or transported in a DeliveryVehicle.

If a Package can exist outside of a Shipment then the relationship between them is one of Aggregation and can be expressed thus :

```
class Package
{
...
}
```

```
class PackageList
{
void Add(Package item)...
void Remove(Package item)...
```

Re: Help with class design

```
Package this[int idx]...  
...  
}
```

```
class Shipment  
{  
  PackageList Packages  
{  
  get {...}  
  set {...}  
}  
}
```

But if the Packages cannot exist outside of a Shipment then the relationship is one of Composition and should be expressed thus :

```
class Package  
{  
...  
}  
  
class Shipment  
{  
  Package AddPackage()...  
  void RemovePackage(Package item)...  
  int PackageCount  
{  
  get {...}  
  }...  
  PackageEnumerator GetPackageEnumerator()...  
...  
}
```

The essential difference is that the Composition relationship should not allow adding or otherwise manipulating Packages without going through methods of the Shipment.

```
> oShip = new Shipment();  
> oShip.AddPackage(sPackageID, dPackageWeight);  
> oShip.AddPackage(sDifferentPackageID, dPackageWeight);
```

PackageWeight should be a property of the Package class and you should not need to pass such details to the Add method of the Shipment class. Assuming you are using Composition, you should retrieve a new instance of the Package class from the Shipment and set properties like Weight, etc on that instance. I suspect though that you should be using Aggregation and therefore you should create an instance of Package, set the properties either in the constructor or otherwise, and then add the instance to the Shipment.

```
> // loop though all packages
```

Re: Help with class design

Re: Help with class design

```
> foreach(Package oPack in oShipment)
> {
> MessageBox.Show(oPack.PackageID);
> }
```

I am not sure whether you really should make the Shipment 'enumerable', that implies it not really much more than a list ??

Joanna

—

Joanna Carter (TeamB)
Consultant Software Engineer

-
- *Follow-Ups:*
 - ◆ *Re: Help with class design*
 - ◇ *From: JSheble*

 - *References:*
 - ◆ *Help with class design*
 - ◇ *From: JSheble*

 - Prev by Date: *Re: Help with class design*
 - Next by Date: *Re: Class Hierarchy design problems...*
 - Previous by thread: *Re: Help with class design*
 - Next by thread: *Re: Help with class design*
 - Index(es):
 - ◆ *Date*
 - ◆ *Thread*