

## Re: Class Hierarchy design problems...

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2005-05/msg02756.html>

---

- *From:* "Mark Broadbent" <nospam@xxxxxxxxxx>
  - *Date:* Sun, 15 May 2005 15:23:27 +0100
- 

David, Nick thanks for the input.

I agree (as you are both suggesting) that perhaps my problem is down to an incorrect design, although I am not entirely sure it is as clean cut as that.

The addition of a Remove method on the Items class would probably be a good addition too i.e. Remove(Item item), however I do still think it makes 'sense' to have methods that fire on an object (in this scenario) which operate in the context of the hierarchy.

For instance I would be saying that ... In that basket, choose an item (the apple), and remove it.

You may both disagree entirely?

The question I would like to have answered though is this situation possible?

The biggest problem is that the indexer is determining the subsequent hierarchy due to its returned type (and therefore the methods would need to exist on that type (which causes the problem of being able to reference the items class).

The only work around I can think off is to create a new type called for instance ItemEditor which I can instantiate and pass a reference to the collection stored in the Items class . I could therefore return the ItemEditor via the Items indexer (providing these methods). The problem arises then that I would need to create an Item property in the Items editor to return an object of type Item.

e.g.

```
class ItemEditor{
    Item item;
    Hashtable itemlist;
    public ItemEditor(Hashtable itemcollection, Item item){...} //note
the constructor sets its localvars to these references so I can operate on
them
    public Item
    Item{ //item
property to return type Item
return item;
}
    public void
    Remove(){ //remove
```

## Re: Class Hierarchy design problems...

method that can operate on collection within Items class

```
itemlist.Remove(item.Name);  
}  
}
```

//therefore class Items would need to be...

```
class Items{  
    Hashtable itemlist;  
    public Add(Item item){...}  
    public Item this[string item]{  
        get { Item i = itemlist[item];  
            return new ItemEditor(itemlist, item);  
        }  
    }  
}
```

//Therefore the following would work as thus :-

```
//-----
```

```
Item i = Basket.Items["Orange"].Item;  
Basket.Items["Apple"].Remove(); //method on class ItemEditor is fired
```

```
item i = new ("Pear");  
Basket.Items.Add(item);  
//-----
```

Is there a way to