

**Use Except to print the first character of product names that aren't also the first character of customer names.**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class MainClass {
public static void Main() {

List products = GetProductList();
List customers = GetCustomerList();

var productFirstChars =
from p in products
select p.ProductName[0];
var customerFirstChars =
from c in customers
select c.CompanyName[0];

var productOnlyFirstChars =
productFirstChars.Except(customerFirstChars);

Console.WriteLine("First letters from Product names, but not
from Customer names:");
foreach (var ch in productOnlyFirstChars) {
Console.WriteLine(ch);
}
}
static List GetProductList() {
List empTree = new List();
```

```
empTree.Add(new Product { ProductName = "A", Category = "0",
UnitPrice = 12, UnitsInStock = 5, Total = 36, OrderDate = new
DateTime(2005, 1, 1), Id = 1 });
empTree.Add(new Product { ProductName = "B", Category = "0",
UnitPrice = 2, UnitsInStock = 4, Total = 35, OrderDate = new
DateTime(2005, 1, 1), Id = 1 });
empTree.Add(new Product { ProductName = "C", Category = "0",
UnitPrice = 112, UnitsInStock = 3, Total = 34, OrderDate = new
DateTime(2005, 1, 1), Id = 1 });
empTree.Add(new Product { ProductName = "D", Category = "0",
UnitPrice = 112, UnitsInStock = 0, Total = 33, OrderDate = new
DateTime(2005, 1, 1), Id = 1 });
empTree.Add(new Product { ProductName = "E", Category = "0",
UnitPrice = 1112, UnitsInStock = 2, Total = 32, OrderDate =
new DateTime(2005, 1, 1), Id = 1 });
empTree.Add(new Product { ProductName = "F", Category = "0",
UnitPrice = 11112, UnitsInStock = 0, Total = 31, OrderDate =
new DateTime(2005, 1, 1), Id = 1 });
return empTree;
}
static List GetCustomerList() {
List empTree = new List();
empTree.Add(new Product { ProductName = "A", Category = "0",
UnitPrice = 12, UnitsInStock = 5, Total = 36, OrderDate = new
DateTime(2005, 1, 1), Id = 1 });
empTree.Add(new Product { ProductName = "B", Category = "0",
UnitPrice = 2, UnitsInStock = 4, Total = 35, OrderDate = new
DateTime(2005, 1, 1), Id = 1 });
empTree.Add(new Product { ProductName = "C", Category = "0",
UnitPrice = 112, UnitsInStock = 3, Total = 34, OrderDate = new
DateTime(2005, 1, 1), Id = 1 });
empTree.Add(new Product { ProductName = "D", Category = "0",
UnitPrice = 112, UnitsInStock = 0, Total = 33, OrderDate = new
DateTime(2005, 1, 1), Id = 1 });
empTree.Add(new Product { ProductName = "E", Category = "0",
UnitPrice = 1112, UnitsInStock = 2, Total = 32, OrderDate =
new DateTime(2005, 1, 1), Id = 1 });
```

```
empTree.Add(new Product { ProductName = "F", Category = "0",
UnitPrice = 11112, UnitsInStock = 0, Total = 31, OrderDate =
new DateTime(2005, 1, 1), Id = 1 });
```

```
List l = new List();
```

```
l.Add(new Customer { CompanyName = "A", Region = "R1",
UnitsInStock = 1, Orders = empTree, CustomerId = 0 });
```

```
l.Add(new Customer { CompanyName = "B", Region = "R2",
UnitsInStock = 2, Orders = empTree, CustomerId = 1 });
```

```
l.Add(new Customer { CompanyName = "C", Region = "R3",
UnitsInStock = 3, Orders = empTree, CustomerId = 2 });
```

```
l.Add(new Customer { CompanyName = "D", Region = "R4",
UnitsInStock = 4, Orders = empTree, CustomerId = 3 });
```

```
l.Add(new Customer { CompanyName = "E", Region = "R5",
UnitsInStock = 5, Orders = empTree, CustomerId = 4 });
```

```
return l;
```

```
}
```

```
}
```

```
class Customer : IComparable {
```

```
public string CompanyName { get; set; }
```

```
public string Region { get; set; }
```

```
public List Orders { get; set; }
```

```
public int UnitsInStock { get; set; }
```

```
public int CustomerId { get; set; }
```

```
public override string ToString() {
```

```
return String.Format("Id: {0}, Name: {1}, Region: {3}",
this.CustomerId, this.CompanyName, this.Region);
```

```
}
```

```
int IComparable.CompareTo(Customer other) {
```

```
if (other == null)
```

```
return 1;
```

```
if (this.CustomerId > other.CustomerId)
```

```
return 1;
```

```
if (this.CustomerId < other.CustomerId) return -1; return 0; }
```

```
} class Product : IComparable {
```

```
public string ProductName { get; set; }
public string Category { get; set; }
public int UnitPrice { get; set; }
public int UnitsInStock { get; set; }
public int Total { get; set; }
public DateTime OrderDate { get; set; }
public int Id { get; set; }

public override string ToString() {
return String.Format("Id: {0}, Name: {1} , Category: {3}",
this.Id, this.ProductName, this.Category);
}
int IComparable.CompareTo(Product other) {
if (other == null)
return 1;
if (this.Id > other.Id)
return 1;

if (this.Id < other.Id) return -1; return 0; } } [/csharp]
```