|  |
| --- |
| **EASJ Notes** |
| Object-Oriented Pro-gramming with C# |
| Object-Oriented Programming, Part IV |

|  |
| --- |
| By Per Laursen  25-01-2019 |

[Introduction 2](#_Toc532285857)

[Exercises 3](#_Toc532285858)

[OOP.4.1 3](#_Toc532285859)

# Introduction

TBD

# Indexers

TBD

# Making a Class Enumerable

TBD

# Operator Overloading

TBD

# Extension Methods

TBD

# Anonymous Types

TBD

# Reflection

TBD

# Exercises

|  |  |
| --- | --- |
| **Exercise** | OOP.4.1 |
| **Solution** | **ImprovedCatalog** |
| **Purpose** | See the index operator and enumeration interface in action. |
| **Description** | The project contains several interfaces and classes related to the concept of a **catalog**. A catalog is simply a container for a collection of values of one specific type. A catalog will typically support essential CRUD operations, and possibly other general-purpose methods and properties.  The folder **CatalogBaseClasses** contains interface definitions and imple­menta­tions of several variants of catalogs. These classes and interfaces are then used in the classes in the **Model** folder, and in the **Run** method in the **Tester** class. |
| **Steps** | 1. Study the interfaces and class definitions in the **CatalogBaseClasses** folder, and be sure you understand the role of each interface and class.(**NB**: implementation of two of the classes is not complete yet – see below). 2. Study the **DomainModel** class. Note that the type of the two properties referring to catalog objects is set to **IAll<…>**. 3. Go to the **Tester** class. In the **Run** method, we are initially only calling the test method **TestIterateOverAll**. Take a look at **TestIterateOverAll**, and make sure you understand what it tests. 4. In the **Run** method, now uncomment the section consisting of calls of **TestRead** (also take a look at the **TestRead** method itself). As it stands, this code cannot be compiled. Why not…? 5. In the **DomainModel** class, change the type of the two properties to **Catalog<…,…>**, and see what effect this has. 6. Perform steps 4 and 5 again for the next code section. In this case, you need to change the type of the properties to **IndexableCatalog<…,…>**. However, running the test will reveal that the index operator does not work properly yet. Open **IndexableCatalog** and implement it correctly. 7. Perform steps 4 and 5 again for the last code section. In this case, you need to change the type of the properties to **EnumerableCatalog<…,…>**. Running the test will reveal that the enumeration interface has not been implemented correctly yet. Open **EnumerableCatalog** and implement it correctly. 8. In all the steps above, we changed the type of the two properties in **Domain­Model** to a class type. Why did we not use the interface types instead? Try it, and see what happens... Can (or should) we “fix” this? |

|  |  |
| --- | --- |
| **Exercise** | OOP.4.2 |
| **Solution** | **ImprovedCatalog** |
| **Purpose** | See the index operator and enumeration interface in action. |
| **Description** | The project contains several interfaces and classes related to the concept of a **catalog**. A catalog is simply a container for a collection of values of one specific type. A catalog will typically support essential CRUD operations, and possibly other general-purpose methods and properties.  The folder **CatalogBaseClasses** contains interface definitions and imple­menta­tions of several variants of catalogs. These classes and interfaces are then used in the classes in the **Model** folder, and in the **Run** method in the **Tester** class. |
| **Steps** | 1. Study the interfaces and class definitions in the **CatalogBaseClasses** folder, and be sure you understand the role of each interface and class.(**NB**: implementation of two of the classes is not complete yet – see below). 2. Study the **DomainModel** class. Note that the type of the two properties referring to catalog objects is set to **IAll<…>**. 3. Go to the **Tester** class. In the **Run** method, we are initially only calling the test method **TestIterateOverAll**. Take a look at **TestIterateOverAll**, and make sure you understand what it tests. 4. In the **Run** method, now uncomment the section consisting of calls of **TestRead** (also take a look at the **TestRead** method itself). As it stands, this code cannot be compiled. Why not…? 5. In the **DomainModel** class, change the type of the two properties to **Catalog<…,…>**, and see what effect this has. 6. Perform steps 4 and 5 again for the next code section. In this case, you need to change the type of the properties to **IndexableCatalog<…,…>**. However, running the test will reveal that the index operator does not work properly yet. Open **IndexableCatalog** and implement it correctly. 7. Perform steps 4 and 5 again for the last code section. In this case, you need to change the type of the properties to **EnumerableCatalog<…,…>**. Running the test will reveal that the enumeration interface has not been implemented correctly yet. Open **EnumerableCatalog** and implement it correctly. 8. In all the steps above, we changed the type of the two properties in **Domain­Model** to a class type. Why did we not use the interface types instead? Try it, and see what happens... Can (or should) we “fix” this? |