04 Generics  
Test your Knowledge  
1. Describe the problem generics address.

Can address the problem of unspecified datatype or datatype inconsistency. Make sure the data type will always be the same. Generic can be used in class, method, interface.   
2. How would you create a list of strings, using the generic List class?

List<string> list = new list<string>();

GenericList<string> list = new GenericList<string>();

List.Add();

List.Remove();  
3. How many generic type parameters does the Dictionary class have?

Two, <Tkey,Tvalue>  
4. True/False. When a generic class has multiple type parameters, they must all match. False, multiple parameters don’t have to be the same type.   
5. What method is used to add items to a List object?

Add()  
6. Name two methods that cause items to be removed from a List.

List.Remove(); remove the first occurrence of the item

List.Removeat(); remove item at specific index

List.Clear(); remove all items from list  
7. How do you indicate that a class has a generic type parameter?

When class name has <T> as type parameter  
8. True/False. Generic classes can only have one generic type parameter. True  
9. True/False. Generic type constraints limit what can be used for the generic type. True. The WHERE clause limits what can be accessed from another class/interface  
10. True/False. Constraints let you use the methods of the thing you are constraining to. True.

Public class AGenericClass<T> where T:

public void MyMethod<T>(T t) where T : IMyInterface { }

Practice working with Generics  
1. Create a custom Stack class MyStack<T> that can be used with any data type which  
has following methods  
1. int Count()  
2. T Pop()  
3. Void Push()

2. Create a Generic List data structure MyList<T> that can store any data type.  
Implement the following methods.  
1. void Add (T element)  
2. T Remove (int index)  
3. bool Contains (T element)  
4. void Clear ()  
5. void InsertAt (T element, int index)  
6. void DeleteAt (int index)  
7. T Find (int index)

3. Implement a GenericRepository<T> class that implements IRepository<T> interface that will have common /CRUD/ operations so that it can work with any data source such as SQL Server, Oracle, In-Memory Data etc. Make sure you have a type constraint on T were it should be of reference type and can be of type Entity which has one property called Id. IRepository<T> should have following methods  
1. void Add(T item)  
2. void Remove(T item)

3. Void Save()  
4. IEnumerable<T> GetAll()  
5. T GetById(int id)

Explore following topics  
Generics in .NET  
Generic classes and methods  
Collections and Data Structures  
Commonly Used Collection Types  
When to Use Generic Collections