04 Generics

Test your Knowledge

1. Describe the problem generics address.
    a. Keep type safety.
    b. Avoid unwanted boxing.

2. How would you create a list of strings, using the generic List class?
   List<string> myString = new List<string>();

3. How many generic type parameters does the Dictionary class have?
   Two. Tkey and Tvalue.

4. True/False. When a generic class has multiple type parameters, they must all match.
   False.

5. What method is used to add items to a List object?
   Add(item)

6. Name two methods that cause items to be removed from a List.
   Remove(item)
   RemoveAt(item's index)

7. How do you indicate that a class has a generic type parameter?
   Put "<T>" after the class name, and use 'T' as a data type in front of the parameters which are generic type parameters.

8. True/False. Generic classes can only have one generic type parameter.
   False.

9. True/False. Generic type constraints limit what can be used for the generic type.
   True.

10. True/False. Constraints let you use the methods of the thing you are constraining to.
    False.

Practice working with Generics

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

2. Create a Generic List data structure MyList<T> that can store any data type. Implement the following methods.
    a. void Add (T element)
    b. T Remove (int index)
    c. bool Contains (T element)
    d. void Clear ()
    e. void InsertAt (T element, int index)
    f. void DeleteAt (int index)
    g. 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
    a. void Add(T item)
    b. void Remove(T item)
    c. Void Save()
    d. IEnumerable<T> GetAll()
    e. 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