04 Generics

Test your Knowledge

1. Describe the problem generics address.

- Generics help address the problems of:
  - Code Duplication
    - With the use of generics, we can write a single implementation that works with any data types. DRY is enforced easily
  - Type Safety
    - With Generics, types are checked at compile time ensuring only correct types are used.
  - Performance Overhead
    - Generics allows skipping the boxing, unboxing process, reducing overhead and also the errors due to type non-safety of explicit unboxing.

2. How would you create a list of strings, using the generic List class?

```
List<string> stringList = new List<string>();
```

3. How many generic type parameters does the Dictionary class have?

- Dictionary class has two generic type parameters, `<TKey>` and `<TValue>`
- TKey must be unique
- TValue epresents the value associated with the key

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

- False
- a generic class can have multiple type parameters and they do not need to match
- Each type parameter can represent same or even different actual types.
- Example: Dictionary class can have parameters of same or different actual types

5. What method is used to add items to a List object?

- `List.Add(item)` method is used to add an item to a list object

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

- `.Remove(item)` and `.RemoveAt(index)`
- `.Remove(item)` will remove first occurence of that item, if it doesn't exist, nothing happens
- `.RemoveAt(index)` will remove the item at specified index, can throw `ArgumentOutOfRangeException`

7. How do you indicate that a class has a generic type parameter?

- A class can me marked with after class name to indicate it has a generic type param.
- The syntax is Angle brackets, with a T enclosed within.

- T is the placeholder for the type that will be decided at compile Time

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

- False
- A generic class can have multiple type parameters, separted by comma inside the angle brackets
- Example: `public class KeyValuePair<TKey, TValue>`

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

- True
- Generic Type Constraints can be used to limit the scope of types used,
- Example: the below constraint will limit the Class to use value types.

```
public class GenericDemo<T> where T: struct
```

- we can use `class` as a constraint, even a specific base clas or interface as one.

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

- True ?
- We can use method or properties of type used as constraint.
- Example, for struct constraint, since it encomapsses broad range of types, we can only use the methods and properties common to all the shared value types
  - e.g. Equals, GetHashCode, etc.

Practice working with Generics

1. Create a custom Stack class `MyStack<T>` that can be used with any data type which has the following methods:
   1. `int Count()`
   2. `T Pop()`
   3. `void Push(T item)`
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 an `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` where it should be of reference type and can be of type `Entity` which has one property called `Id`. `IRepository<T>` should have the 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