C# Fundamentals

8. **Generics and Interfaces with a Repository PatternObjective:Requirements:**
    - **Implement a generic in-memory repository to perform CRUD operations.**
    - **Define an interface (e.g., IRepository<T>) with methods like Add, Get, Update, and Delete.**
    - **Create a generic class that implements this interface.**
    - **Use type constraints if necessary (e.g., where T : class or implementing a specific interface).**
    - **Write a simple console UI to demonstrate the repository with a sample entity (e.g., Product or Student).**

Program code:

IEntity.cs

```
public interface IEntity
{
   int Id { get; set; }
}

// IEntity ensures each entity has an Id for repository operations.
```

IRepository.cs

```
using System.Collections.Generic;

public interface IRepository<T> where T : class, IEntity
{
   void Add(T entity);
   T Get(int id);
   // IEnumerable<T> -returning collections when you just want to loop through items
   IEnumerable<T> GetAll();
   void Update(int id, T entity);
   void Delete(int id);
```

```
}

//  Generic Repository (IRepository<T>) provides a reusable data access layer
```

InMemoryRepository.cs

```
using System;
using System.Collections.Generic;

public class InMemoryRepository<T> : IRepository<T> where T : class, IEntity
   // where T : class, IEntity: Ensures T implements IEntity
{
   // _data: Stores entities in a dictionary (key: Id, value: entity)
   // readonly-The reference to the dictionary cannot be changed after initialization (but the contents can still be modified).
   private readonly Dictionary<int, T> _data = new();
   private int _nextId = 1; // Auto-incrementing ID

   public void Add(T entity)
   {
      entity.Id = _nextId++; // Assign unique ID
      _data[entity.Id] = entity;
   }

   public T Get(int id)
   {
      //  retrieve a value from the dictionary. It returns true if the key exists, and false if it doesn't.
      // out var entity - If found, assigns the corresponding value (the entity of type T) to entity.
      _data.TryGetValue(id, out var entity);
      return entity;
   }
```

```
    public IEnumerable<T> GetAll()
    {
        return _data.Values;
    }

    public void Update(int id, T entity)
    {
        if (_data.ContainsKey(id))
        {
            entity.Id = id; // Preserve the ID
            _data[id] = entity;
        }
        else
        {
            Console.WriteLine($"Entity with ID {id} not found.");
        }
    }

    public void Delete(int id)
    {
        if (_data.ContainsKey(id))
        {
            _data.Remove(id);
        }
        else
        {
            Console.WriteLine($"Entity with ID {id} not found.");
        }
    }
}
```

```
//  InMemoryRepository<T> is a simple implementation using a dictionary
```

Product.cs

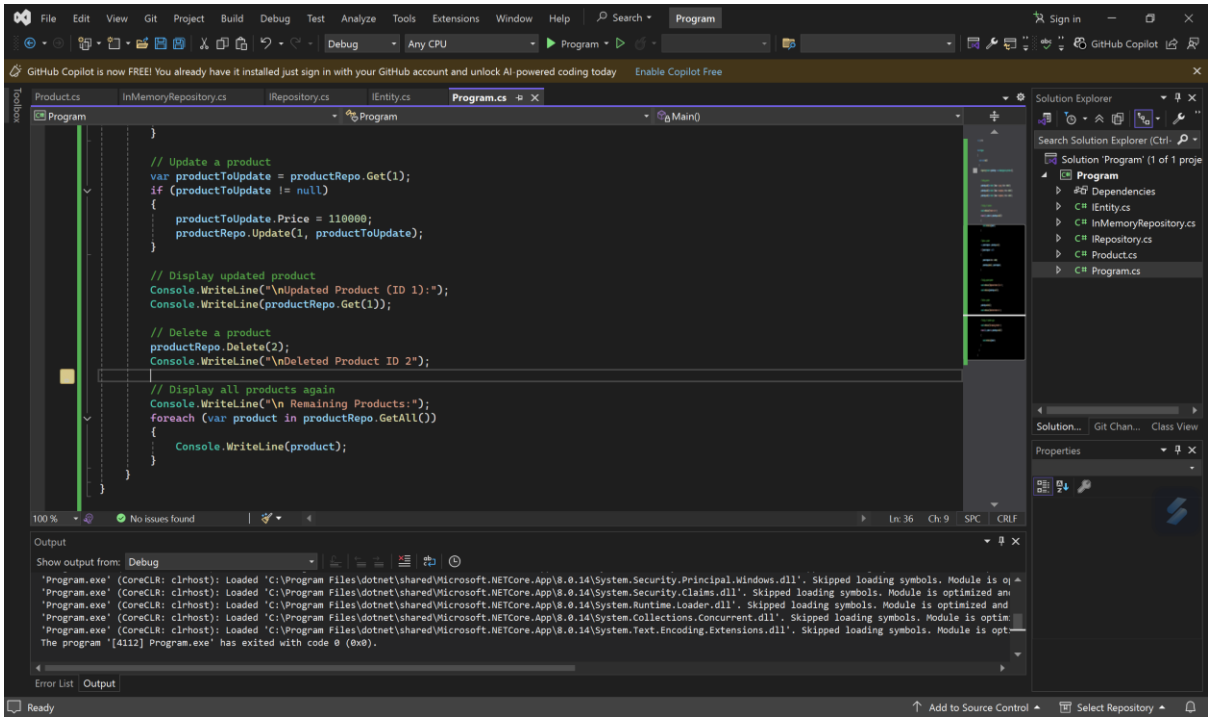
```
public class Product : IEntity
{
    public int Id { get; set; } // required by IEntity
    public string Name { get; set; }
    public double Price { get; set; }

    public override string ToString()
    {
        return $"ID: {Id}, Name: {Name}, Price: Rs.{Price}";
    }
}
```

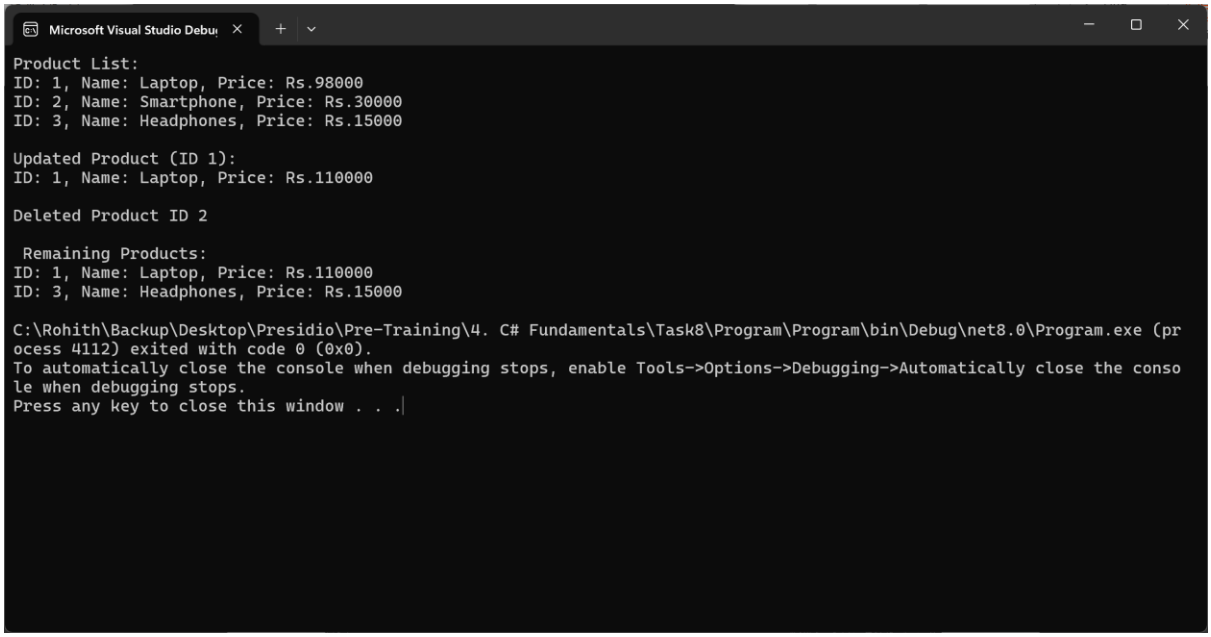
Program.cs

```
using System;

class Program
{
    static void Main()
    {
        IRepository<Product> productRepo = new InMemoryRepository<Product>();

        // Adding products
        productRepo.Add(new Product { Name = "Laptop", Price = 98000 });
        productRepo.Add(new Product { Name = "Smartphone", Price = 30000 });
        productRepo.Add(new Product { Name = "Headphones", Price = 15000 });

        // Display all products
        Console.WriteLine("Product List:");
```

```
        foreach (var product in productRepo.GetAll())
        {
            Console.WriteLine(product);
        }

        // Update a product
        var productToUpdate = productRepo.Get(1);
        if (productToUpdate != null)
        {
            productToUpdate.Price = 110000;
            productRepo.Update(1, productToUpdate);
        }

        // Display updated product
        Console.WriteLine("\nUpdated Product (ID 1):");
        Console.WriteLine(productRepo.Get(1));

        // Delete a product
        productRepo.Delete(2);
        Console.WriteLine("\nDeleted Product ID 2");

        // Display all products again
        Console.WriteLine("\n Remaining Products:");
        foreach (var product in productRepo.GetAll())
        {
            Console.WriteLine(product);
        }
    }
}
```

Output:


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
Program
Sign in
Debug
Any CPU
Program
GitHub Copilot
GitHub Copilot is now FREE! You already have it installed just sign in with your GitHub account and unlock AI-powered coding today
Enable Copilot Free
Product.cs
InMemoryRepository.cs
IRepository.cs
IEntity.cs
Program.cs
Program
Program
Main()
}
// Update a product
var productToUpdate = productRepo.Get(1);
if (productToUpdate != null)
{
productToUpdate.Price = 110000;
productRepo.Update(1, productToUpdate);
}
// Display updated product
Console.WriteLine("\nUpdated Product (ID 1):");
Console.WriteLine(productRepo.Get(1));
// Delete a product
productRepo.Delete(2);
Console.WriteLine("\nDeleted Product ID 2");
// Display all products again
Console.WriteLine("\n Remaining Products:");
foreach (var product in productRepo.GetAll())
{
Console.WriteLine(product);
}
}
}
100 %
No issues found
Ln: 36
Ch: 9
SPC
CRLF
Solution Explorer
Search Solution Explorer (Ctrl
Solution 'Program' (1 of 1 proje
Program
Dependencies
IEntity.cs
InMemoryRepository.cs
IRepository.cs
Product.cs
Program.cs
Solution...
Git Chan...
Class View
Properties
Output
Show output from: Debug
'Program.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.14\System.Security.Principal.Windows.dll'. Skipped loading symbols. Module is o
'Program.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.14\System.Security.Claims.dll'. Skipped loading symbols. Module is optimized an
'Program.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.14\System.Runtime.Loader.dll'. Skipped loading symbols. Module is optimized and
'Program.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.14\System.Collections.Concurrent.dll'. Skipped loading symbols. Module is optim
'Program.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.14\System.Text.Encoding.Extensions.dll'. Skipped loading symbols. Module is opt
The program '[4112] Program.exe' has exited with code 0 (0x0).
Error List
Output
Ready
Add to Source Control
Select Repository


Microsoft Visual Studio Debu
Product List:
ID: 1, Name: Laptop, Price: Rs.98000
ID: 2, Name: Smartphone, Price: Rs.30000
ID: 3, Name: Headphones, Price: Rs.15000
Updated Product (ID 1):
ID: 1, Name: Laptop, Price: Rs.110000
Deleted Product ID 2
Remaining Products:
ID: 1, Name: Laptop, Price: Rs.110000
ID: 3, Name: Headphones, Price: Rs.15000
C:\Rohith\Backup\Desktop\Presidio\Pre-Training\4. C# Fundamentals\Task8\Program\Program\bin\Debug\net8.0\Program.exe (pr
ocess 4112) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .