**1. Create a console application to print the behavior of an animal on a console screen. Application must be object oriented. Apply both the generic and type specific generic (generic with constraint) approach.**

**a.       Enum and Its Implementation**

**b.       Principal of OOP**

**c.       Generic Interface & it’s implementation**

**d.       Generic and type specific generic approach**

**--Solution-1:**

1.Constants -enum

namespace Ev\_03

{

public enum AnimalType

{

Harvivore, Carnivore, Omnivore

}

public enum Gender

{

Female, Male

}

}  
2.Interface – IGenericBehavior

namespace Ev\_03

{

public interface IGenericBehavior<T>

{

string GetGenericBehavior(T obj);

}

}

3.Interface – ITypeSpecificBehavior

namespace Ev\_03

{

public interface ITypeSpecificBehavior<T>

{

string GetTypeSpecificBehavior<T>(T data) where T : Animal;

}

}  
4.Class – Animal

namespace Ev\_03

{

public class Animal

{

public string Name { get; set; }

public int Age { get; set; }

public Gender Sex { get; set; }

public AnimalType Type { get; set; }

}

}

5.Class – GenericAnimal

namespace Ev\_03

{

public class GenericAnimal<T>:IGenericBehavior<T>

{

public string GetGenericBehavior(T obj)

{

string behavior = string.Empty;

if (obj is Animal)

{

Animal a = obj as Animal;

switch (a.Type)

{

case AnimalType.Harvivore:

behavior = "Lives on harvs, four footed";

break;

case AnimalType.Carnivore:

behavior = "Have canine, meat eater";

break;

case AnimalType.Omnivore:

behavior = "Diverse eating bahvior, adaptable";

break;

default:

behavior = "Unknown behaviour";

break;

}

}

else

{

behavior = "Not an animal";

}

return behavior;

}

}

}  
6.Class - TypeSpecificAnimal  
namespace Ev\_03

{

public class TypeSpecificAnimal<T> : ITypeSpecificBehavior<T>

{

public string GetTypeSpecificBehavior<T>(T data) where T : Animal

{

string behavior = string.Empty;

switch (data.Type)

{

case AnimalType.Harvivore:

behavior = "Lives on harvs, four footed";

break;

case AnimalType.Carnivore:

behavior = "Have canine, meat eater";

break;

case AnimalType.Omnivore:

behavior = "Diverse eating bahvior, adaptable";

break;

default:

behavior = "Unknown behaviour";

break;

}

return behavior;

}

}

}

7.Program Class – main Method

namespace Ev\_03

{

class Program

{

static void Main(string[] args)

{

Animal cow = new Animal

{

Name = "Cow",

Age = 3,

Type = AnimalType.Harvivore,

Sex = Gender.Female

};

GenericAnimal<Animal> animal = new GenericAnimal<Animal>();

Console.WriteLine(animal.GetGenericBehavior(cow));

Animal bear = new Animal

{

Name = "Grizzly",

Age = 4,

Type = AnimalType.Omnivore,

Sex = Gender.Male

};

TypeSpecificAnimal<Animal> obj2 = new TypeSpecificAnimal<Animal>();

Console.WriteLine(obj2.GetTypeSpecificBehavior(bear));

Console.ReadLine();

}

}

}

**2. Create a console application Find out all the Product information by creating At least Three Class Name Category,Product, ProductModel. (Data of 3 table given in desktop as Product.txt file.) You must join three tables.**

**--Solution-2:**

class Category

    {

        public int CategoryID { get; set; }

        public string Name { get; set; }

    }

public class Product

    {

        public int ProductID { get; set; }

        public string Name { get; set; }

        public int CategoryID { get; set; }

        public int ModelID { get; set; }

    }

class Model

    {

        public int ModelID { get; set; }

        public string Name { get; set; }

    }

class Program

    {

        static void Main(string[] args)

        {

            var categories = new Category[]

            {

                new Category{CategoryID = 1,Name = "Shirts"},

                new Category{CategoryID = 2,Name = "Pants"}

            };

            var models = new Model[]

            {

                new Model{ModelID = 1, Name="Man's Shirts"},

                new Model{ModelID = 2, Name="Man's Pants"},

                new Model{ModelID = 3, Name="Woman's Shirts"},

                new Model{ModelID = 4, Name="Woman's Pants"}

            };

            var products = new Product[]

            {

                new Product{ProductID = 1,Name = "Polo Shirts",CategoryID = 1,ModelID = 1},

                new Product{ProductID = 1,Name = "Polo Shirts",CategoryID = 1,ModelID = 3},

                new Product{ProductID = 1,Name = "Formal Pants",CategoryID = 2,ModelID = 2},

                new Product{ProductID = 1,Name = "Formal Pants",CategoryID = 2,ModelID = 4}

            };

            var v = from p in products

                    join c in categories

                    on p.CategoryID equals c.CategoryID

                    join m in models

                    on p.ModelID equals m.ModelID

                    select new { p.ProductID, Model = m.Name, Category = c.Name, p.Name, p.Color,p.Weight, p.Size, p.ListPrice };

            foreach (var x in v)

            {

                Console.WriteLine($"{ x.Name}\t{ x.Model}\t{x.Color}\t{x.Weight}\t{x.Size}\t{ x.ListPrice} ");

            }

            Console.ReadKey();

        }

    }

**3. Create a console application to print multi task.**

**--Solution-3:**

namespace MultyTask\_Asynchronous

{

class Program

{

private static void CreateTaskUsingAction()

{

Console.WriteLine("This Task is created using Action.\n");

}

private static void CreateTaskUsingAsyncAwait()

{

Console.WriteLine("This Task is created using Async and Await.\n");

}

private static async void CreateAsyncTask()

{

await Task.Run(() => CreateTaskUsingAsyncAwait());

}

private static int Add(int a, int b)

{

return a + b;

}

private static async void SolveTheMath(int firstInt, int secondtInt)

{

int result = await Task.FromResult(Add(firstInt, secondtInt));

await Task.Delay(1000);

Console.WriteLine("Result = " + result.ToString());

}

static void Main(string[] args)

{

Console.WriteLine("Start a Task in C# .NET:");

Console.WriteLine("=====================================\n");

Task actionTask = new Task(new Action(CreateTaskUsingAction)); //Creating a Task using Action

actionTask.Wait(1000);

actionTask.Start();

CreateAsyncTask(); //Creating a Task using Async and Await.

Console.Write("\nFirst Integer = ");

int firstInt = int.Parse(Console.ReadLine());

Console.Write("Second Integer = ");

int secondtInt = int.Parse(Console.ReadLine());

SolveTheMath(firstInt, secondtInt); //Creating a Task using FromResult Method.

Console.ReadKey();

}

}

}