using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication13

{

class Program

{

static void Main(string[] args)

{

List<Product> products = new List<Product>();

MainMenu(ref products);

}

public static void MainMenu(ref List<Product> products)

{

int ans;

Console.Clear();

DecorationLine();

Console.WriteLine("1.ADD Products");

Console.WriteLine("2.SEARCH PRODUCTS");

DecorationLine();

Console.WriteLine("Enter your Choice:");

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

if (ans == 1)

{

Console.Clear();

DecorationLine();

Console.WriteLine("\tADD PRODUCT");

DecorationLine();

Product.AddProduct(ref products);

DecorationLine();

MainMenuMessage();

MainMenu(ref products);

}

if (ans == 2)

{

Console.Clear();

DecorationLine();

Console.WriteLine("\tSEARCH PRODUCT");

DecorationLine();

Product.SearchProduct(ref products);

DecorationLine();

MainMenuMessage();

MainMenu(ref products);

}

}

public static void MainMenuMessage()

{

Console.WriteLine("Press enter to go to the main menu");

Console.ReadLine();

}

public static void DecorationLine()

{

Console.ForegroundColor = ConsoleColor.DarkMagenta;

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.ResetColor();

}

}

}

//////////

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication13

{

partial class Product

{

public int id;

public string name;

public int price;

public static Product AddProduct(ref List<Product> products)

{

int id, price;

string name;

Console.WriteLine("Enter product id:");

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

Console.WriteLine("Enter product name:");

name = Console.ReadLine();

Console.WriteLine("Enter price:");

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

var Product = new Product() { id = id, name = name, price = price };

if (IsProductexists(ref products, id))

{

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.WriteLine("Product already exists");

}

else

{

products.Add(Product);

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.WriteLine("Product added Successfully");

Console.ResetColor();

}

return Product;

}

public static bool IsProductexists(ref List<Product> products, int id)

{

return products.Where(p => p.id == id).ToList().Count > 0;

}

public static void SearchProduct(ref List<Product> products)

{

Console.Clear();

Console.WriteLine("SEARCH PRODUCT");

int id;

Console.WriteLine("Enter id:");

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

if (products.Where(p => p.id == id).ToList().Count > 0)

{

var productfound = from p in products

where p.id == id

select p;

Console.WriteLine("Id:{0},NAME:{1},Price:{2}", productfound.First().id, productfound.First().name, productfound.First().price);

}

else

{

Console.WriteLine("PRODUCT NOT FOUND");

}

Console.WriteLine("Enter go to main menu:");

Console.ReadLine();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication15

{

[AttributeUsage(AttributeTargets.All)]

public class DeveloperAttribute:Attribute

{

//private fields.

private string name;

private string level;

private bool reviewed;

//this constructor defines two required parameters:name and level.

public DeveloperAttribute(string name,string level)

{

this.name = name;

this.level = level;

this.reviewed = false;

}

public virtual string Name

{

get { return name; }

}

public virtual string Level

{

get { return level; }

}

public virtual bool Reviewed

{

get { return reviewed; }

set { reviewed = value; }

}

}

[Developer(name:"jeevi",level:"jnr.developer",Reviewed =true)]

class Program

{

static int Factorial(int n)

{

int i, result = 1;

for (i = 1; i <= n; i++)

result = result \* i;

return result;

}

static int Power(int a,int b)

{

int count = 1;

for (int i = 1; i <= b; i++)

count = count \* a;

return count;

}

static int Square(int a)

{

int res;

return res=a\*a;

}

static void Main(string[] args)

{

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication15

{

[AttributeUsage(AttributeTargets.Class)]

class FoodTypeAttribute : Attribute

{

private string category;

public FoodTypeAttribute(string category)

{

this.category = category;

}

public virtual string Category

{

get { return category; }

}

}

[FoodType(category: "veg")]

class Apple

{

}

[FoodType(category: "nonveg")]

class Chkn

{

}

[FoodType(category: "spinach")]

class Spinach

{

}

class Program

{

static void Main(string[] args)

{

}

}

}

Validations:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication13

{

partial class Product

{

public int id;

public string name;

public int price;

public static Product AddProduct(ref List<Product> products)

{

String errorMessage = string.Empty;

int id, price;

string name;

Console.WriteLine("Enter product id:");

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

Console.WriteLine("Enter product name:");

name = Console.ReadLine();

Console.WriteLine("Enter price:");

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

var Product = new Product() { id = id, name = name, price = price };

//validation for id should not exist already

if (IsProductexists(ref products, id))

errorMessage = errorMessage + "product already exists\n";

//products name cannot starts with space

if (Product.name[0]==' ')

errorMessage = errorMessage + "products name cannot starts with space\n";

//products name should starts with PRD-

if (!Product.name.StartsWith("PRD-"))

errorMessage = errorMessage + "products name should starts with PRD-\n";

//validation price should not ne negative

if (Product.price<0)

errorMessage = errorMessage + "price cannot be negative\n";

if(errorMessage!="")

{

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine(errorMessage);

Console.ResetColor();

}

else

{

products.Add(Product);

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("product added successfully");

Console.ResetColor();

}

return Product;

}

public static bool IsProductexists(ref List<Product> products, int id)

{

return products.Where(p => p.id == id).ToList().Count > 0;

}

public static void SearchProduct(ref List<Product> products)

{

Console.Clear();

Console.WriteLine("SEARCH PRODUCT");

int id;

Console.WriteLine("Enter id:");

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

if (products.Where(p => p.id == id).ToList().Count > 0)

{

var productfound = from p in products

where p.id == id

select p;

Console.WriteLine("Id:{0},NAME:{1},Price:{2}", productfound.First().id, productfound.First().name, productfound.First().price);

}

else

{

Console.WriteLine("PRODUCT NOT FOUND");

}

Console.WriteLine("Enter go to main menu:");

Console.ReadLine();

}

}

}

888888888888888

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication16

{

public static class MyExtensions

{

public static int EmpStartingwithA(this List<string> products)

{

var q = products.Where(p => p.StartsWith("A")).ToList().Count;

return q;

}

public static List<string> EndsWithH(this List<string> names)

{

return names.Where(p => p.ToUpper().EndsWith("H")).ToList();

}

public static int LastElement(this Stack<int> stack)

{

return stack.ElementAt(stack.Count - 1);

}

}

public class Program

{

private static readonly bool e;

static void Main(string[] args)

{

List<string> employees = new List<string>()

{

"Meghanadh","Anand","Anil","Charan","Kiran","Mahesh","Rajesh","Sarath"

};

foreach (var e in employees.EndsWithH()) ;

Console.WriteLine(e);

Console.WriteLine(employees.EmpStartingwithA());

Console.ReadLine();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication16

{

public static class MyExtensions

{

public static int EmpStartingwithA(this List<string> products)

{

var q = products.Where(p => p.StartsWith("A")).ToList().Count;

return q;

}

public static List<string> EndsWithH(this List<string> names)

{

return names.Where(p => p.ToUpper().EndsWith("H")).ToList();

}

public static int LastElement(this Stack<int> stack)

{

return stack.ElementAt(stack.Count - 1);

}

}

public class Program

{

static void Main(string[] args)

{

Stack<int> data = new Stack<int>();

data.Push(10);

data.Push(23);

data.Push(6);

Console.WriteLine(data.LastElement());

Console.ReadLine();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication16

{

public static class MyExtensions

{

public static int EmpStartingwithA(this List<string> strings)

{

var q = strings.Where(p => p.StartsWith("A")).ToList().Count;

return q;

}

public static List<string> EndsWithH(this List<string> names)

{

return names.Where(p => p.ToUpper().EndsWith("H")).ToList();

}

public static int LastElement(this Stack<int> stack)

{

return stack.ElementAt(stack.Count - 1);

}

public static string ToWord(this int p)

{

string ans;

switch(p)

{

case 1:ans= "one";

break;

case 2:

ans="two";

break;

case 3:

ans="three";

break;

case 4:

ans= "four";

break;

case 5:

ans= "five";

break;

default:ans = "greater than five";

break;

}

return ans;

}

}

public class Program

{

static void Main(string[] args)

{

int p=4;

Console.WriteLine(p.ToWord());

Console.ReadLine();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication17

{

public static class MyProgram

{

public static int ToCube(this int p)

{

return p \* p \* p;

}

}

class Program

{

static void Main(string[] args)

{

int p = 4;

Console.WriteLine(p.ToCube());

Console.ReadLine();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication17

{

class Program

{

public delegate void Print(string value);

public delegate void Algebra(int val);

static void Main(string[] args)

{

Print message = delegate (string val)

{

Console.WriteLine("Hi, {0}", val);

};

message("Capgemini");

Print myOrder = delegate (string val)

{

Console.WriteLine("{0} {1} {2}", val, val, val);

};

myOrder("Order");

Algebra multiple100 = delegate (int val)

{

Console.WriteLine("{0}", val \* 100);

};

multiple100(4);

multiple100(5);

multiple100(8);

Console.ReadLine();

}

}

}

Delegate example 2:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication17

{

class Program

{

public static void Add(int a,int b)

{

Console.WriteLine("{0}", a + b);

}

public static void Mul(int a, int b)

{

Console.WriteLine("{0}", a \* b);

}

public static void Div(int a, int b)

{

Console.WriteLine("{0}", a / b);

}

public delegate void MyDelegate(int a, int b);

static void Main(string[] args)

{

MyDelegate md = new MyDelegate(Add);

md += Mul;

md += Div;

md(2, 5);

Console.ReadLine();

}

}

}

Refelectoinss

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication19

{

class Program

{

static void Main(string[] args)

{

Assembly myassembly = Assembly.LoadFrom("D:CSharp\\.dll");

var mytypes = myassembly.GetTypes();

foreach(var p in mytypes)

{

foreach(var d in p.GetMembers())

{

Console.WriteLine("{0}\t:{1}", p.Name, d.Name);

}

}

Console.ReadLine();

}

}

}

///////////////////

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication20

{

class Program

{

static int Add(int a=0,int b=0,int c=0)

{

return a + b + c;

}

static void Main(string[] args)

{

Console.WriteLine(Add(a: 5, c: 2));//optional parameters

Console.WriteLine(Add(a: 5, c: 2,b:8));//named arguements

Console.ReadLine();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*dynamic

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication20

{

class Maths

{

static int Add(int a, int b)

{

return a + b;

}

}

class Program

{

static void Main(string[] args)

{

dynamic p = new Maths();

Console.WriteLine(p.Add(9, 8));

Console.ReadLine();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*tasksss egssssss

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication21

{

class Program

{

static void PrintFirstNum()

{

for (int i = 1; i <= 100; i++)

Console.WriteLine(i);

}

static void PrintBigNum()

{

for(int i=1000;i<=1100;i++)

Console.WriteLine(i);

}

static void Main(string[] args)

{

var taskA =new Task(()=>PrintFirstNum());

var taskB = new Task(() => PrintBigNum());

taskA.Start();

taskB.Start();

Console.ReadLine();

}

}

}