

Lab 10

Advanced Concepts in C#

Encrypting and Decrypting Data

Mục tiêu	2
Phần I Bài tập step by step	3
Bài 10.1	3
Bài 10.2	5
Bài 10.3	8
Bài 10.4	12
Bài 10.5	16
Bài 10.6	24
Bài 10.7	34
Bài 10.8	36
Bài 10.9	38
Bài 10.10	40
Bài 10.11	42
Bài 10.12	44
Bài 10.13	47
Bài 10.14	50
Phần II Bài tập tự làm	53

Mục tiêu

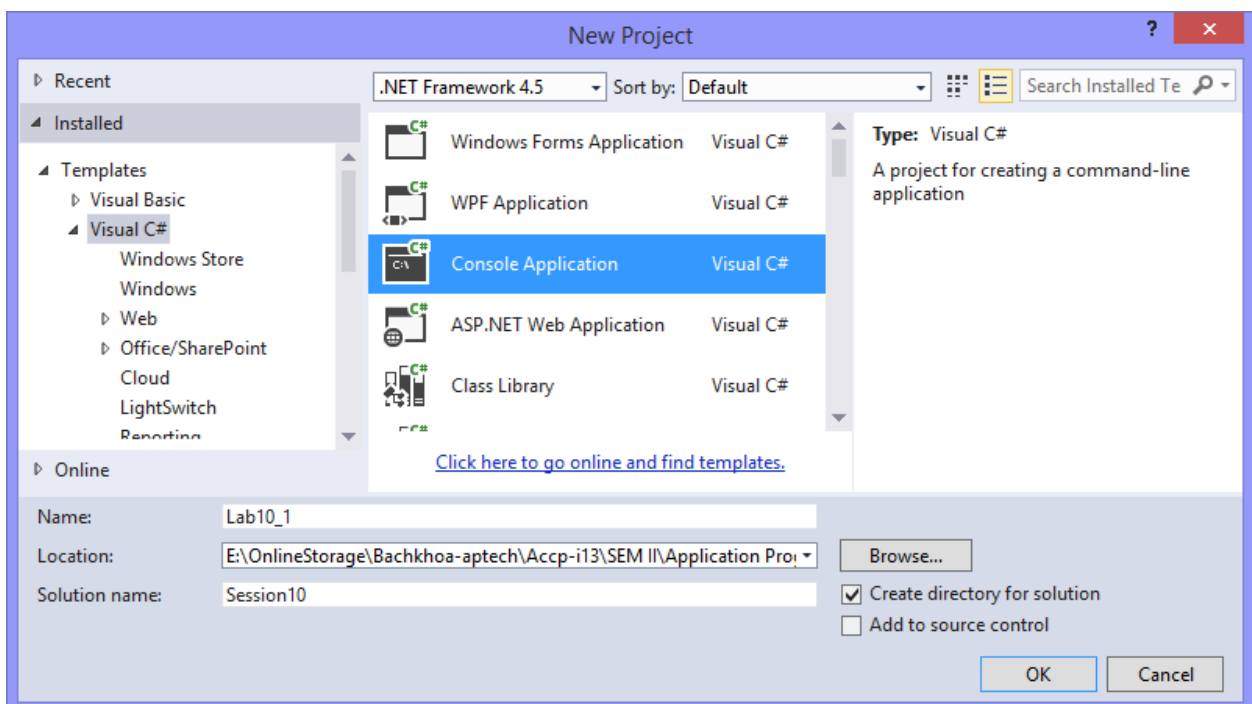
- Sử dụng các delegate có sẵn trong hệ thống
- Sử dụng toán tử lambda và câu lệnh lambda
- Truy cập dữ liệu sử dụng Entity Framework và LINQ
- Tạo WebService với WCF
- Tạo và sử dụng Thread
- Lập song song
- Truy vấn song song sử dụng LINQ
- Phương thức bất động bộ
- Lập trình động (Dynamic)
- Mã hóa và giải mã dữ liệu

Phần I Bài tập step by step

Bài 10.1

Viết chương trình C# minh họa việc sử dụng các delegate định nghĩa sẵn trong hệ thống

BƯỚC 1: Mở Visual Studio 2013, vào menu File -> New -> Project -> chọn loại project “Console Application”, nhập tên project, tên solution -> OK.



BƯỚC 2: Tạo lớp Calculator và code theo gợi ý sau:

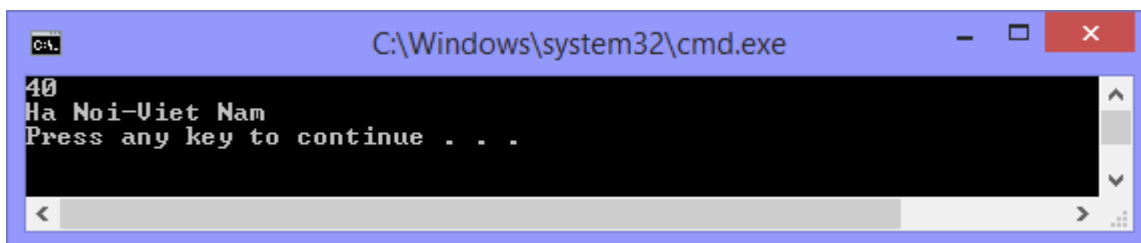
```
//Định nghĩa lớp tính toán
class Calculator
{
    //định nghĩa phương thức cộng 2 số
    public static int Add(int a, int b)
    {
        return a + b;
    }
    //định nghĩa phương thức nối 2 chuỗi
    public static string Join(string a, string b)
```

```
{  
    return a + b;  
}  
}
```

BƯỚC 3: Mở tệp Program.cs và code theo gợi ý sau:

```
/// <summary>  
/// Chương trình minh họa việc sử dụng các Delegate định nghĩa sẵn trong hệ  
/// thống  
/// </summary>  
class Program  
{  
    static void Main(string[] args)  
    {  
        //Sử dụng delegate tham chiếu tới phương thức cộng 2 số nguyên  
        Func<int, int, int> sumNumber = Calculator.Add;  
        //Sử dụng delegate tham chiếu tới phương thức nối 2 chuỗi  
        Func<string, string, string> joinString = Calculator.Join;  
        //In kết quả - gọi delegate  
        Console.WriteLine(sumNumber(10, 30));  
        Console.WriteLine(joinString("Ha Noi-", "Viet Nam"));  
    }  
}
```

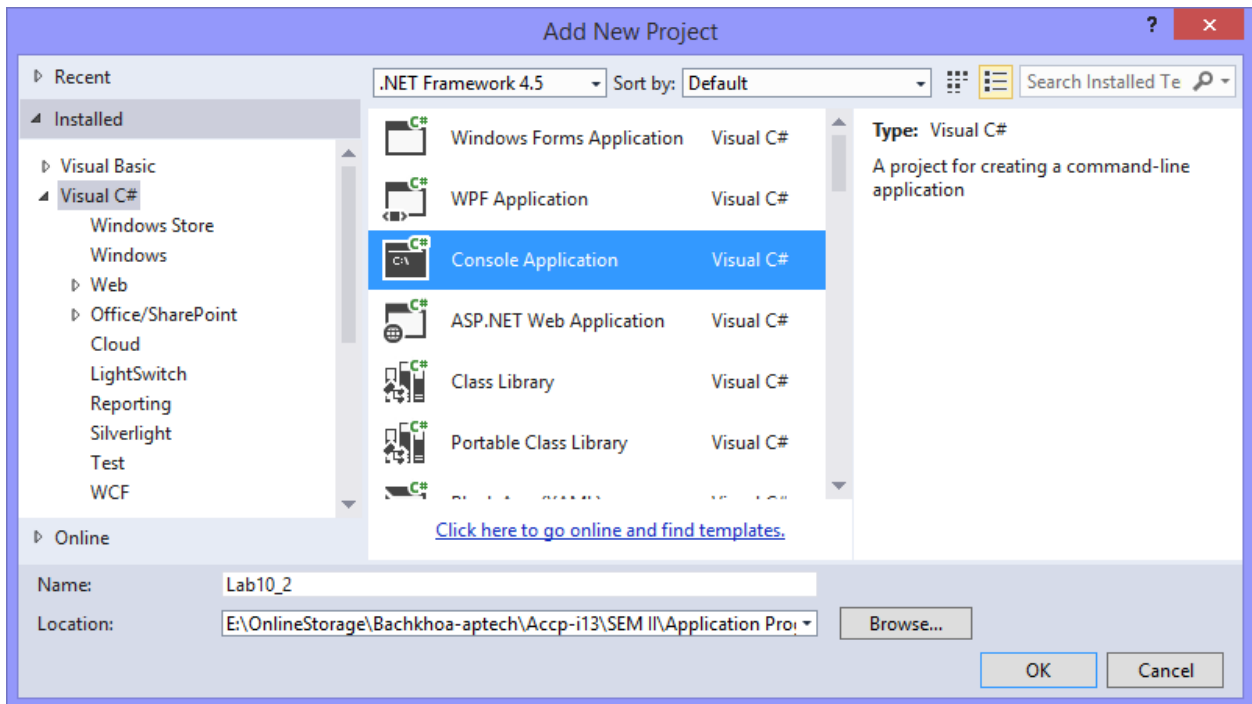
BƯỚC 4: Nhấn Ctrl+F5 để chạy và xem kết quả



Bài 10.2

Viết chương trình C# minh họa việc sử dụng biểu thức Lambda, câu lệnh Lambda, và gọi phương thức chứa delegate

BƯỚC 1: Kích chuột phải vào Solution “Session10” chọn Add -> New Project -> nhập tên.

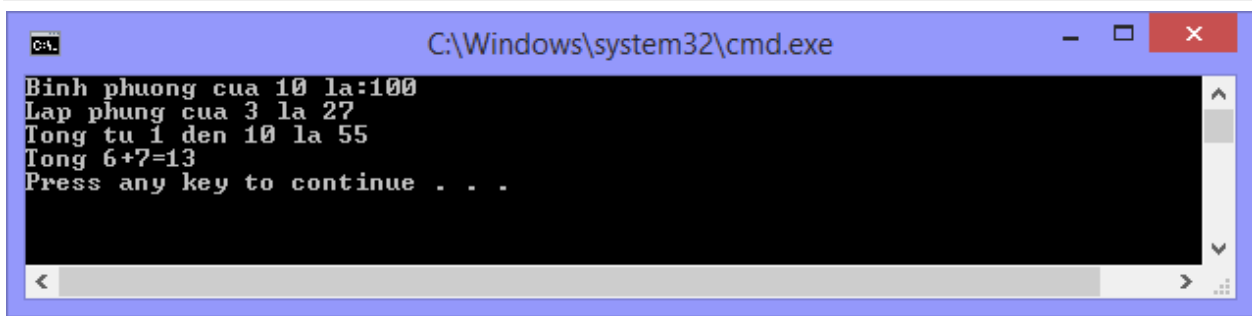


BƯỚC 2: Mở tệp Program.cs và code theo gợi ý sau:

```
//Khai báo delegate nhận 1 tham số nguyên
delegate int MyDelegate1(int x);
//Khai báo delegate nhận 2 tham số nguyên
delegate int MyDelegate2(int x, int y);
class Program
{
    //gọi delegate trong action
    public static int Calculate(MyDelegate2 Add, int x, int y)
    {
        int s = Add(x, y);
        return s;
    }
}
```

```
}  
static void Main(string[] args)  
{  
    //biểu thức Lambda  
    MyDelegate1 cube = x => x * x * x;  
    //câu lệnh Lambda  
    MyDelegate1 sum = x =>  
    {  
        int s = 0;  
        for (int i = 1; i <= x; i++)  
        {  
            s += i;  
        }  
        return s;  
    };  
    //Sử dụng delegate định nghĩa sẵn tham chiếu tới câu lệnh lambda  
    Func<int, int> p = (int x) =>  
    {  
        return x * x;  
    };  
    Console.WriteLine("Bình phương của 10 là:{0}", p(10));  
    //in kết quả  
    Console.WriteLine("Lap phung cua 3 la {0}", cube(3));  
    Console.WriteLine("Tong tu 1 den 10 la {0}", sum(10));  
    //gọi action chứa delegate  
    int total = Calculate((no1, no2) => no1 + no2, 6, 7);  
    Console.WriteLine("Tong 6+7={0}", total);  
}  
}
```

BƯỚC 3: Nhấn Ctrl+F5 để chạy và xem kết quả



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a blue title bar and standard Windows window controls (minimize, maximize, close). The command prompt displays the following text in Vietnamese and English:

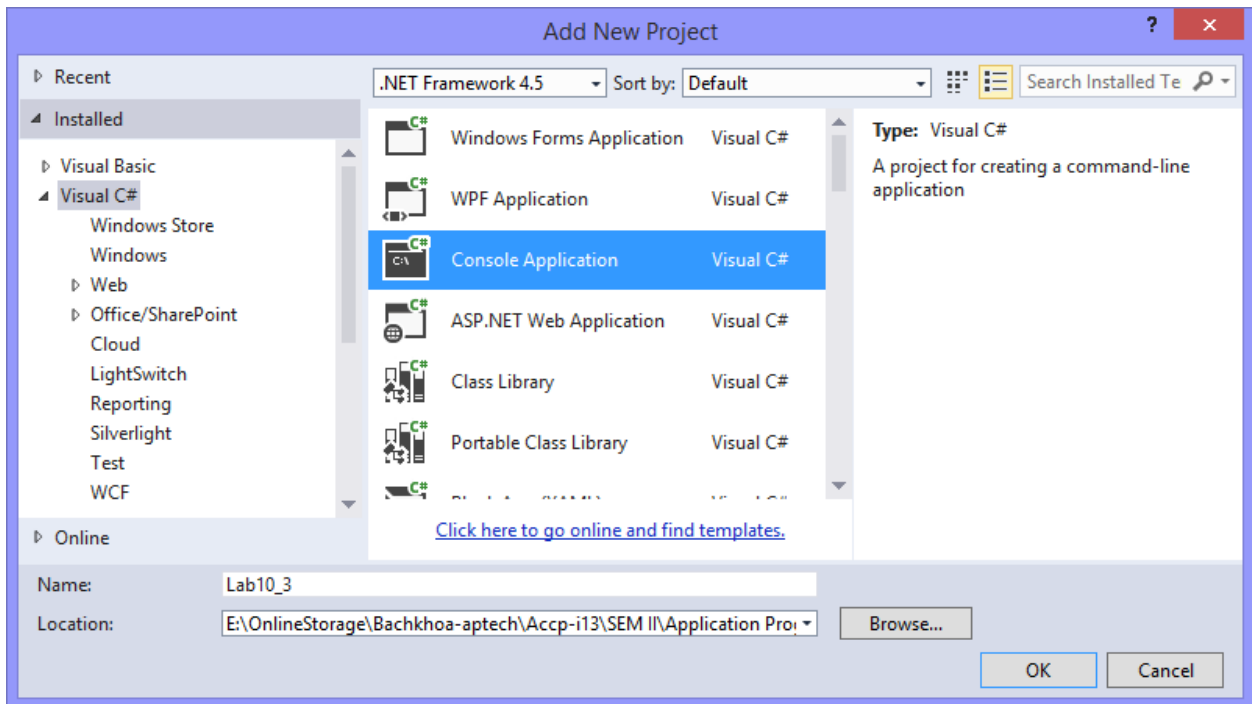
```
Binh phuong cua 10 la:100  
Lap phung cua 3 la 27  
Tong tu 1 den 10 la 55  
Tong 6+7=13  
Press any key to continue . . .
```

The text is displayed in a monospaced font on a black background. The window also features a scrollbar on the right side.

Bài 10.3

Viết chương trình C# minh họa việc thao tác với Collection không sử dụng biểu thức Lambda và có sử dụng biểu thức Lambda

BƯỚC 1: Kích chuột phải vào Solution “Session10” chọn Add -> New Project -> nhập tên.



BƯỚC 2: Tạo lớp Employee với code gợi ý như sau:

```
//Tạo lớp Employee
public class Employee
{
    public int EmpId { get; set; }
    public string EmpName { get; set; }
    public double Salary { get; set; }
    public string City { get; set; }
    //Ghi đè phương thức ToString trả về thông tin chi tiết
    public override string ToString()
    {
```



```
        return String.Format("{0} {1} {2} {3}", EmpId, EmpName, Salary,  
        City);  
    }  
}
```

BƯỚC 3: Mở Program.cs và code theo gợi ý sau:

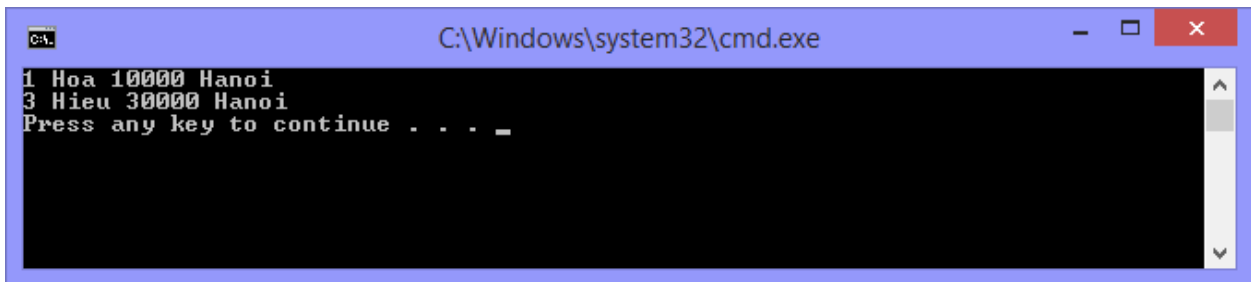
```
class Program  
{  
    ///Không sử dụng biểu thức Lambda  
    //static void Main(string[] args)  
    //{  
    //    List<Employee> employees = new List<Employee>();  
    //    employees.Add(new Employee { EmpId = 1, EmpName = "Hoa", Salary =  
    10000, City = "Hanoi" });  
    //    employees.Add(new Employee { EmpId = 2, EmpName = "Cuong", Salary =  
    20000, City = "Hanam" });  
    //    employees.Add(new Employee { EmpId = 3, EmpName = "Hieu", Salary =  
    30000, City = "Hanoi" });  
  
    //    foreach (Employee emp in FilterByCity(employees, "Hanoi"))  
    //    {  
    //        Console.WriteLine(emp.ToString());  
    //    }  
    //}  
    //public static IEnumerable<Employee> FilterByCity(IEnumerable<Employee>  
    employees, string filterStr)  
    //{  
    //    foreach (Employee emp in employees)  
    //    {  
    //        if (emp.City == filterStr)  
    //            yield return emp;  
    //    }  
    //}  
    /// Có sử dụng biểu thức Lambda
```

```
//Định nghĩa delegate
public delegate bool EmpDelegate(Employee emp);
static void Main(string[] args)
{
    // phần khởi tạo dữ liệu
    List<Employee> employees = new List<Employee>();
    employees.Add(new Employee { EmpId = 1, EmpName = "Hoa", Salary =
    10000, City = "Hanoi" });
    employees.Add(new Employee { EmpId = 2, EmpName = "Cuong", Salary =
    20000, City = "Hanam" });
    employees.Add(new Employee { EmpId = 3, EmpName = "Hieu", Salary =
    30000, City = "Hanoi" });
    Console.WriteLine("Lọc theo City:");
    foreach (Employee emp in Filter(employees, emp => emp.City ==
    "Hanoi"))
    {
        Console.WriteLine(emp.ToString());
    }
    Console.WriteLine("\nLọc theo Salary:");
    foreach (Employee emp in Filter(employees, emp => emp.Salary >=
    20000))
    {
        Console.WriteLine(emp.ToString());
    }
}
//Tạo phương thức lọc dữ liệu
public static IEnumerable<Employee> Filter(IEnumerable<Employee>
employees, EmpDelegate check)
{
    foreach (Employee emp in employees)
    {
        if (check(emp) == true)
            yield return emp;
    }
}
```

```
}
```

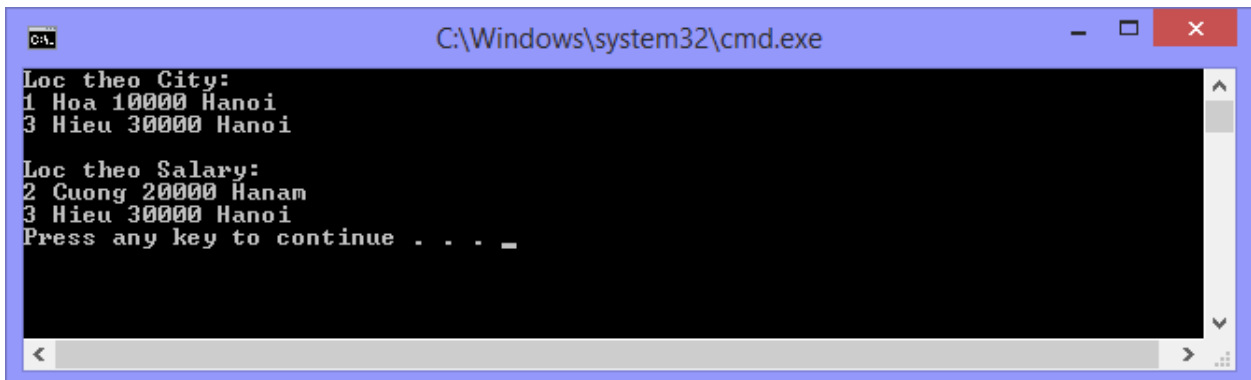
BƯỚC 4: Nhấn Ctrl+F5 để chạy và xem kết quả (Lưu ý, ghi chú từng phần và chạy để kiểm tra kết quả)

- Kết quả phần không sử dụng biểu thức Lambda



```
C:\Windows\system32\cmd.exe
1 Hoa 100000 Hanoi
3 Hieu 300000 Hanoi
Press any key to continue . . . _
```

- Kết quả phần có sử dụng biểu thức Lambda



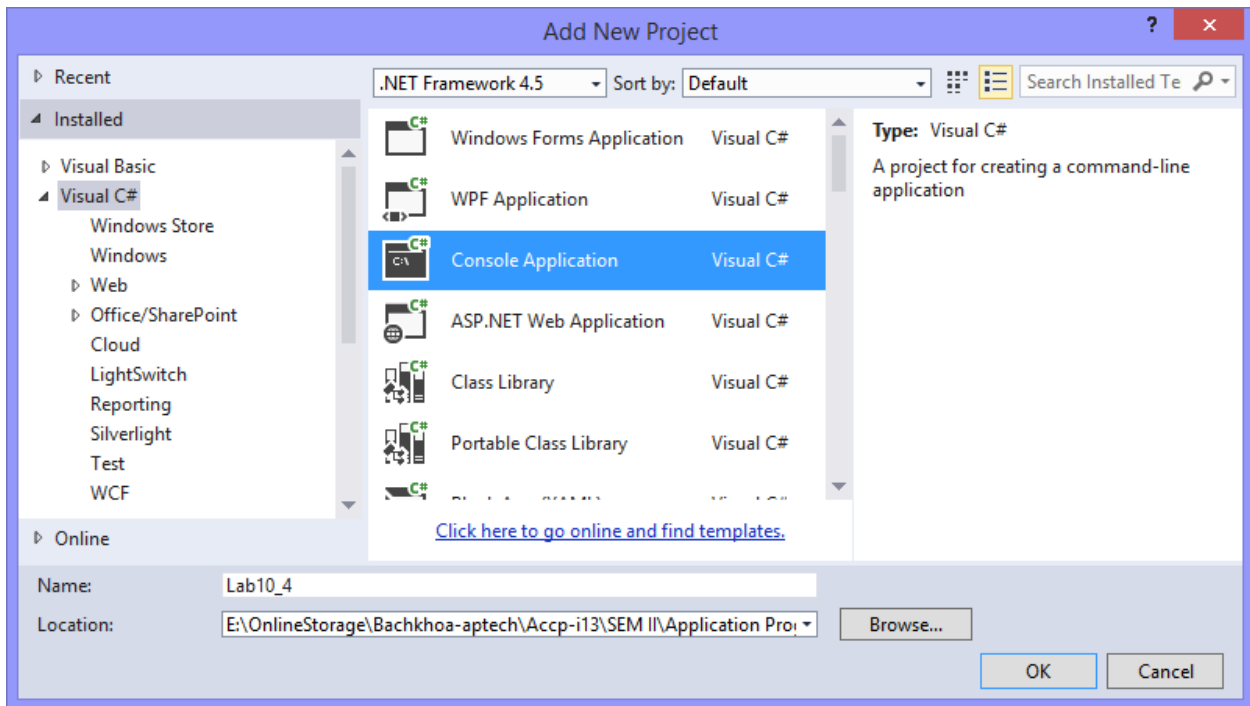
```
C:\Windows\system32\cmd.exe
Loc theo City:
1 Hoa 100000 Hanoi
3 Hieu 300000 Hanoi

Loc theo Salary:
2 Cuong 200000 Hanam
3 Hieu 300000 Hanoi
Press any key to continue . . . _
```

Bài 10.4

Viết chương trình C# minh họa việc sử dụng một số phương thức mở rộng LINQ.

BƯỚC 1: Kích chuột phải vào Solution “Session10” chọn Add -> New Project -> nhập tên.



BƯỚC 2: Tạo lớp Film với code gợi ý như sau:

```
class Film
{
    public string FilmId { get; set; }
    public string FilmName { get; set; }
    public int Price { get; set; }
    public override string ToString()
    {
        return FilmId + ":" + FilmName + ":" + Price;
    }
}
```

BƯỚC 3: Mở tệp Program.cs và code theo gợi ý sau:

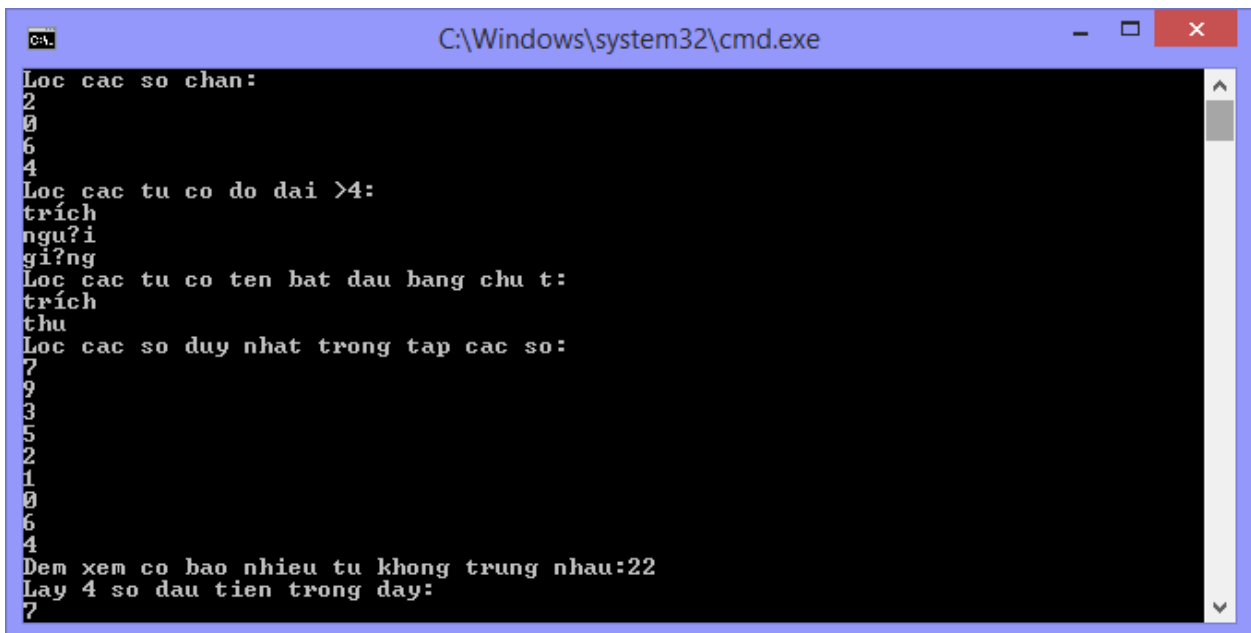
```
class Program
{
    static void Main(string[] args)
    {
        //khai báo chung
        int[] Numbers = { 7, 9, 3, 5, 2, 1, 0, 6, 4, 3, 1 };
        string[] Words = {
            "Chỉ", "trích", "phê", "phán", "người", "khác",
            "giống", "như", "con", "chim", "bồ", "câu", "đưa", "thư",
            "bao", "giờ", "cũng", "quay", "về", "nơi", "xuất", "phát"
        };
        List<Film> ListFilm = new List<Film>()
        {
            new Film{FilmId="F01",FilmName="Điện viên 007",Price=120000},
            new Film{FilmId="F02",FilmName="Tam quốc diễn
            nghĩa",Price=130000},
            new Film{FilmId="F03",FilmName="Thiếu lâm truyền
            kỳ",Price=16000},
            new Film{FilmId="F04",FilmName="Người nhện 2",Price=100000},
            new Film{FilmId="F05",FilmName="Ngân hàng tình
            yêu",Price=340000},
            new Film{FilmId="F06",FilmName="Người đẹp và quái
            thú",Price=230000},
            new Film{FilmId="F07",FilmName="Biệt động Sài
            Gòn",Price=190000},
        };

        //lọc các số chẵn
        IEnumerable<int> querynumber = Numbers.Where(n => n % 2 == 0);
        Show<int>(querynumber, "Loc cac so chan:");
        //lọc các từ có độ dài >4
        IEnumerable<string> queryword = Words.Where(w => w.Length > 4);
        Show<string>(queryword, "Loc cac tu co do dai >4:");
        //lọc các từ có tên bắt đầu bằng chữ t
        IEnumerable<string> queryT = Words.Where(w => w.StartsWith("t"));
    }
}
```

```
Show<string>(queryT, "Loc cac tu co ten bat dau bang chu t:");
//Lọc các số duy nhất trong tập các số
var uniqueNumber = Numbers.Distinct();
Show<int>(uniqueNumber, "Loc cac so duy nhat trong tap cac so:");
//Đếm xem có bao nhiêu từ không trùng nhau
var countDistinct = Words.Distinct().Count();
Console.WriteLine("Dem xem co bao nhieu tu khong trung nhau:" +
countDistinct);
//lấy 4 số đầu tiên trong dãy
var fourNumber = Numbers.Take(4);
Show<int>(fourNumber, "Lay 4 so dau tien trong day:");
//lấy 2 từ đầu tiên trong câu
var twoword = Words.Take(2);
Show<string>(twoword, "Lay 2 tu dau tien trong cau:");
//lấy những từ đầu tiên có chứa chữ t
var searchword = Words.TakeWhile(w => w.Contains('t'));
Show<string>(searchword, "Lay nhung tu dau tien co chua chu t:");
//sắp xếp theo đơn giá, lấy những phim đầu tiên có đơn giá <200000
var queryfilm = ListFilm.OrderBy(f => f.Price)
.Select(x => new { x.FilmId, x.FilmName, x.Price })
.ToList().TakeWhile(t => t.Price < 200000);
//bỏ qua 3 phần từ đầu tiên, lấy tất cả các phần từ còn lại
var skipNumber = Numbers.Skip(3);
Show<int>(skipNumber, "Bo qua 3 phan tu dau tien, lay tat ca cac
phan tu con lai:");
//bỏ qua 4 phần từ đầu tiên lấy 3 phần từ kế tiếp
var skipTakeNumber = Numbers.Skip(4).Take(3);
Show<int>(skipTakeNumber, "Bo qua 4 phan tu dau tien, lay 3 phan tu
ke tiep:");
//bỏ qua 3 phim đầu tiên lấy 3 phim kết tiếp (có thể áp dụng để
phân trang)
var skipTakeFilm = ListFilm.Skip(3).Take(3);
Show<Film>(skipTakeFilm, "Bo qua 3 phim dau tien, lay 3 phim ke
tiep:");
//Sắp xếp giảm dần, sau đó lấy các phần từ <5
```

```
var sortNumber = Numbers.OrderByDescending(x => x).SkipWhile(x => x  
> 5);  
Show<int>(sortNumber, "Sap xep giam dan, sau do lay cac phan tu  
<5:");  
}  
//Định nghĩa phương thức Generic hiển thị dữ liệu  
static void Show<T>(IEnumerable<T> data, string message)  
{  
    Console.WriteLine(message);  
    foreach (var item in data)  
    {  
        Console.WriteLine(item);  
    }  
}  
}
```

BƯỚC 4: Nhấn Ctrl+F5 để chạy và xem kết quả



```
C:\Windows\system32\cmd.exe  
Loc cac so chan:  
2  
0  
6  
4  
Loc cac tu co do dai >4:  
trich  
ngu?i  
gi?ng  
Loc cac tu co ten bat dau bang chu t:  
trich  
thu  
Loc cac so duy nhat trong tap cac so:  
7  
9  
3  
5  
2  
1  
0  
6  
4  
Dem xem co bao nhieu tu khong trung nhau:22  
Lay 4 so dau tien trong day:  
7
```

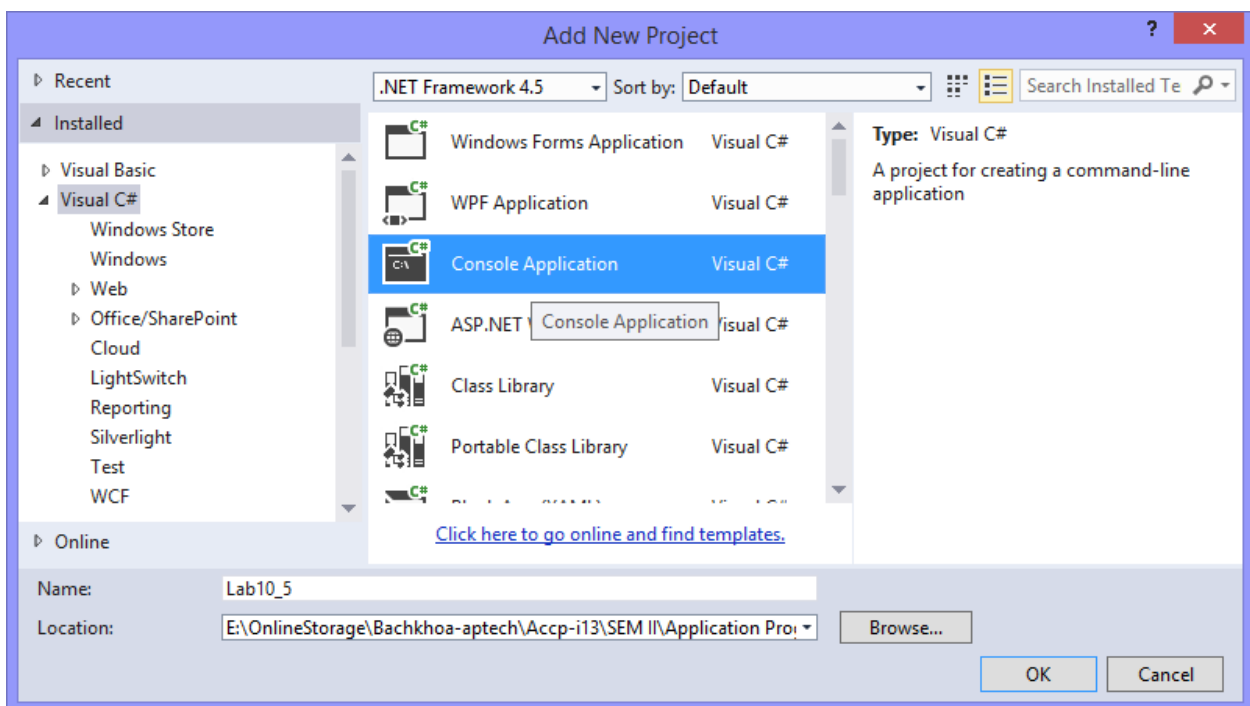
Bài 10.5

Viết chương trình C# và thực hiện những công việc sau:

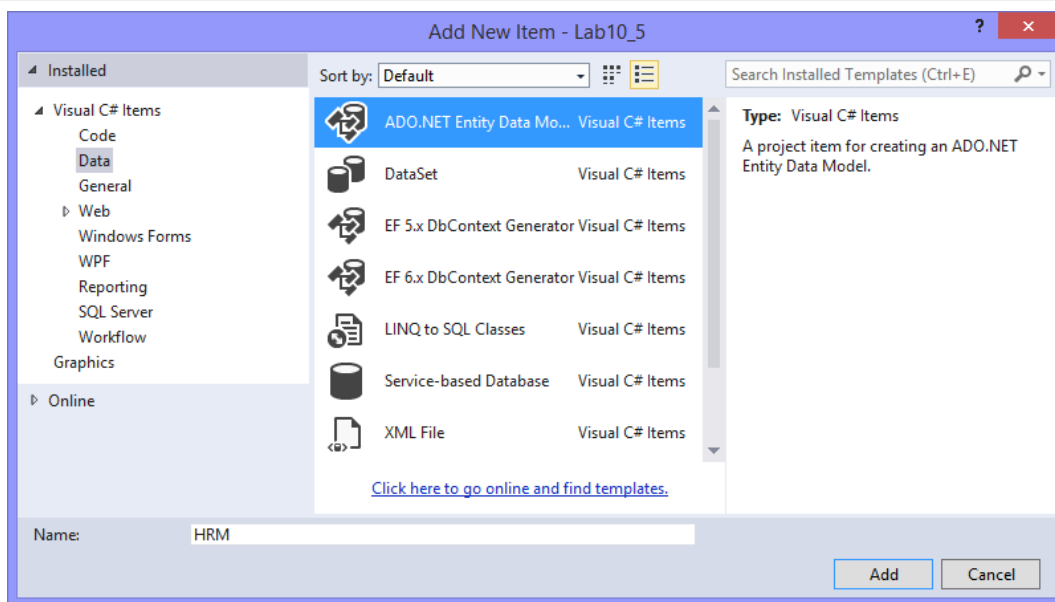
- Tạo Entity Data Model với 2 thực thể Department(DepId, DepName), Employee(EmpId, FirstName, LastName, BirthDay, Sex, Email, Phone, DepId).
- Thực hiện Generate database từ Model vừa tạo.
- Sinh ra
- Thực hiện truy vấn dữ liệu trong database sử dụng LINQ

BƯỚC 1: Mở SQL Server tạo một cơ sở dữ liệu có tên HRM

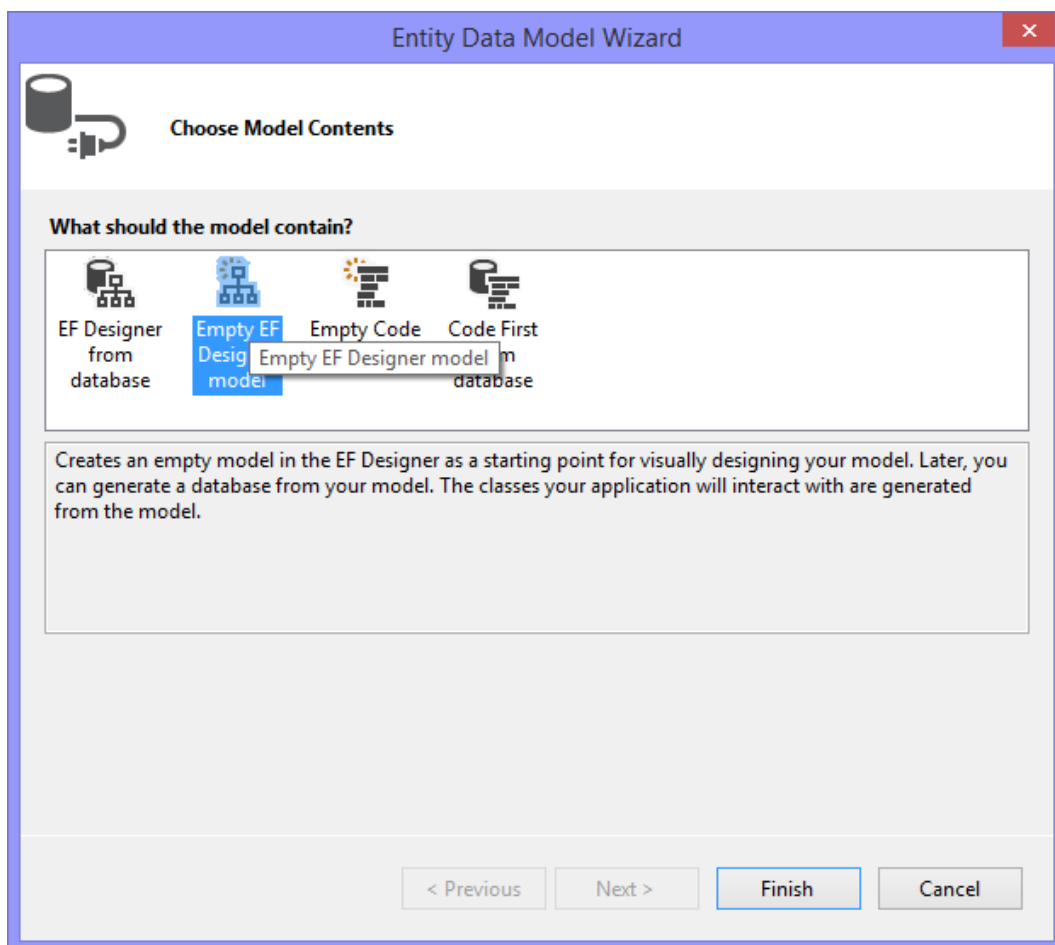
BƯỚC 2: Kích chuột phải vào Solution “Session10” chọn Add -> New Project -> nhập tên.



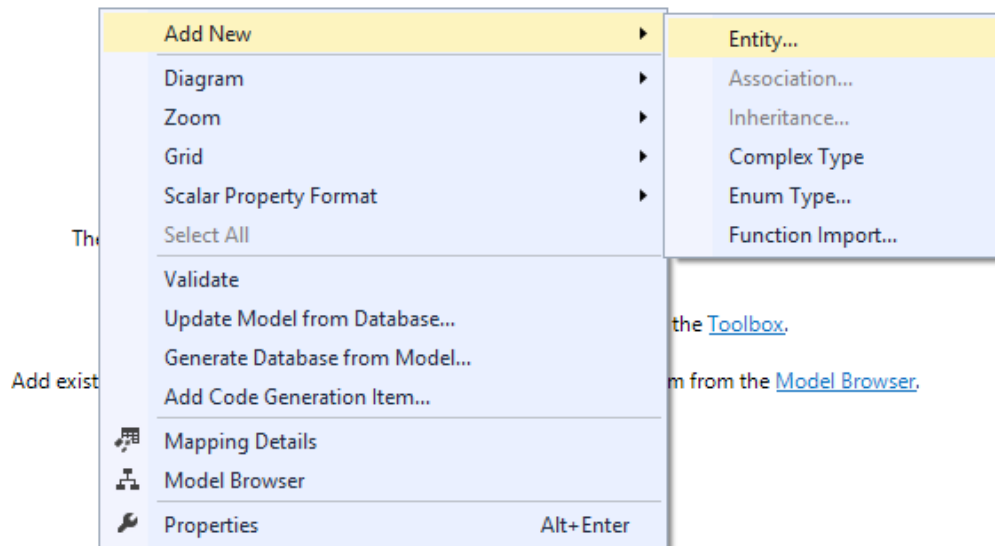
BƯỚC 3: Kích chuột phải vào Project “Session10_5” chọn Add->New Item -> chọn như hình dưới



Tiếp theo



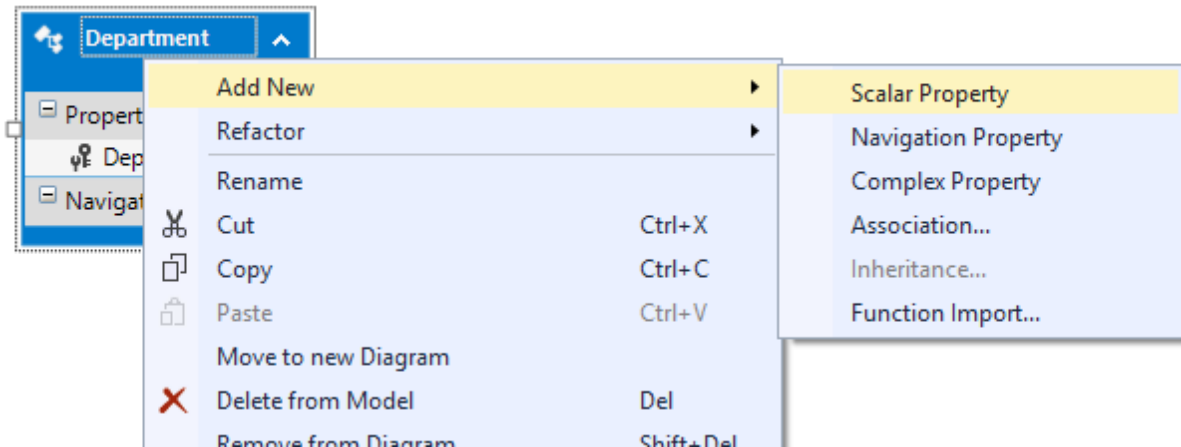
Ấn finish -> cửa sổ thiết kế Entity hiện ra -> kích chuột phải vào cửa sổ và chọn Add New
-> Entity.



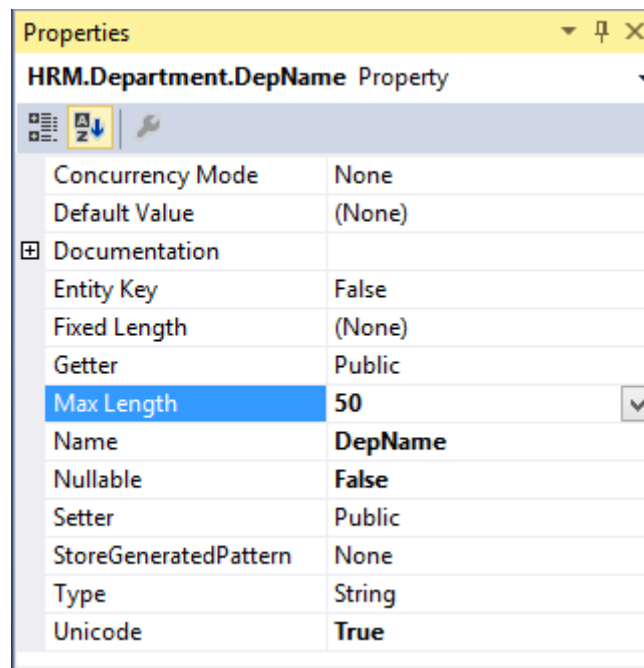
Nhập tên như hình dưới

The image shows the 'Add Entity' dialog box. It has a title bar with a close button. The dialog is divided into two main sections: 'Properties' and 'Key Property'. In the 'Properties' section, there are three fields: 'Entity name' (containing 'Department'), 'Base type' (a dropdown menu showing '(None)'), and 'Entity Set' (containing 'Departments'). In the 'Key Property' section, there is a checked checkbox 'Create key property', a 'Property name' field (containing 'Depld'), and a 'Property type' dropdown menu (showing 'String'). At the bottom right, there are 'OK' and 'Cancel' buttons.

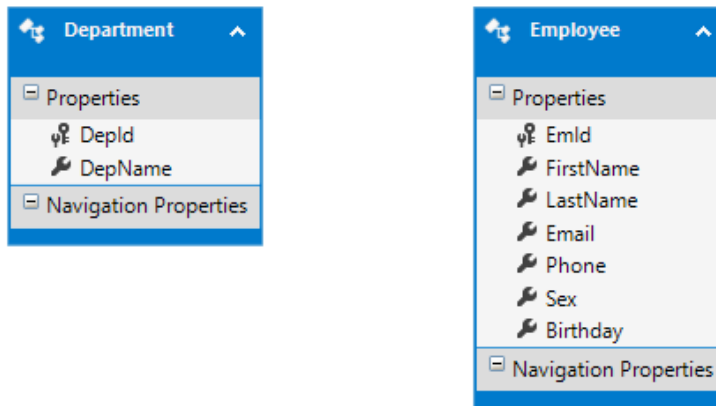
Entity Department sinh ra -> kích chuột phải vào entity để tạo thuộc tính -> chọn Add New
-> Scalar Property



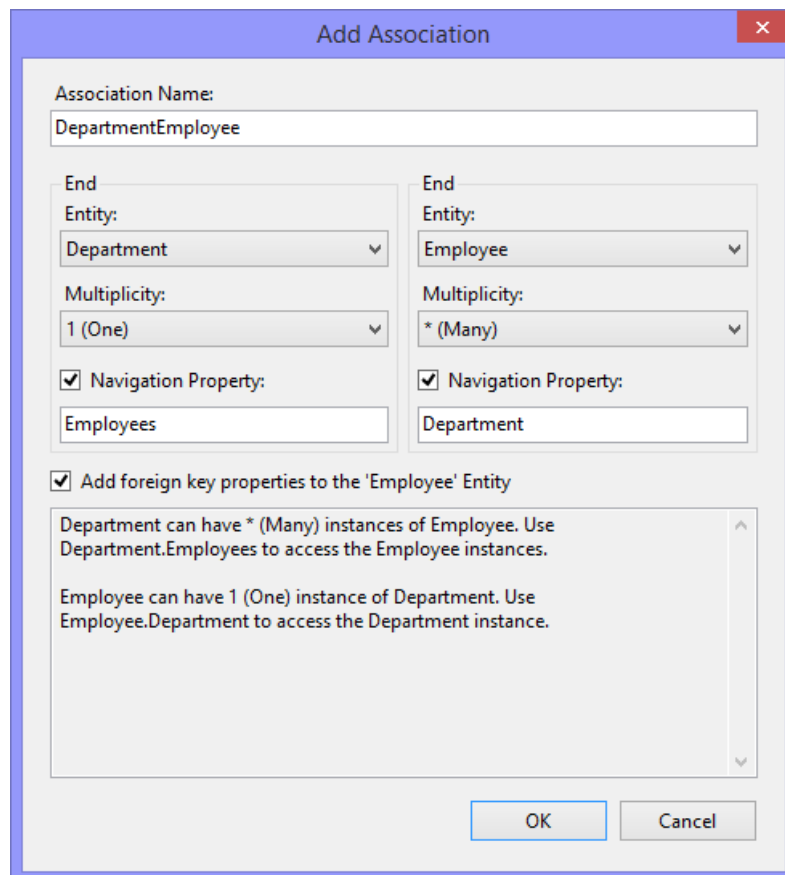
Nhập tên thuộc tính -> tại cửa sổ properties thay đổi kiểu dữ liệu và độ dài cho phù hợp



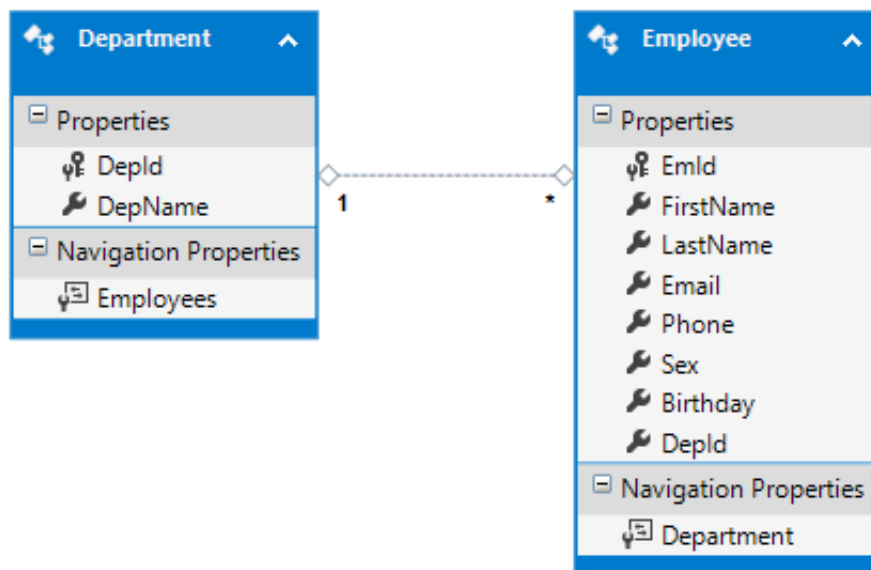
Thêm mới một thực thể Employee với các thuộc tính như hình dưới



Kích chuột phải vào cửa sổ thiết kế chọn Add New -> Association để tạo liên kết giữa 2 thực thể.



Kết quả



Sinh ra kịch bản DDL để tạo các bảng trong database -> kích chuột phải vào màn hình design các Entity -> Chọn Generate Database from Model -> Finish -> Một tệp tin sẽ sinh ra HRM.edmx.sql

- Tạo database HRM trong SQL Server
- Chạy tệp HRM.edmx.sql để sinh ra cấu trúc các bảng

BƯỚC 4: Mở tệp Program.cs ra và code cho hàm Main theo gợi ý sau:

```
static void Main(string[] args)
{
    //Khởi tạo Entity
    HRMContainer hrm = new HRMContainer();
    //tạo 4 phòng ban
    Department dep1 = new Department() { DepId = "D01", DepName = "Phong Hanh Chinh" };
    Department dep2 = new Department() { DepId = "D02", DepName = "Phong Nhan Su" };
    Department dep3 = new Department() { DepId = "D03", DepName = "Phong Ky Thuat" };
```

```
Department dep4 = new Department() { DepId = "D04", DepName = "Phong
Marketing" };
//thêm phòng ban
hrm.Departments.Add(dep1);
hrm.Departments.Add(dep2);
hrm.Departments.Add(dep3);
hrm.Departments.Add(dep4);
//tạo 5 nhân viên
Employee emp1 = new Employee() { EmId = "E01", FirstName = "Tran Van",
LastName = "Tuan", Email = "tuan@gmail.com", Phone = "0977689877", Sex =
true, BirthDay = new DateTime(1990, 1, 20), DepId = "D01" };
Employee emp2 = new Employee() { EmId = "E02", FirstName = "Hoang Van",
LastName = "Hoa", Email = "hoa@gmail.com", Phone = "0977689877", Sex =
true, BirthDay = new DateTime(1990, 1, 20), DepId = "D01" };
Employee emp3 = new Employee() { EmId = "E03", FirstName = "Le Van",
LastName = "Hai", Email = "hai@gmail.com", Phone = "0977689877", Sex =
true, BirthDay = new DateTime(1990, 1, 20), DepId = "D02" };
Employee emp4 = new Employee() { EmId = "E04", FirstName = "Nguyen Van",
LastName = "Long", Email = "long@gmail.com", Phone = "0977689877", Sex =
true, BirthDay = new DateTime(1990, 1, 20), DepId = "D02" };
Employee emp5 = new Employee() { EmId = "E05", FirstName = "Ho Van",
LastName = "Hanh", Email = "hanh@gmail.com", Phone = "0977689877", Sex =
true, BirthDay = new DateTime(1990, 1, 20), DepId = "D02" };
//thêm nhân viên
hrm.Employees.Add(emp1);
hrm.Employees.Add(emp2);
hrm.Employees.Add(emp3);
hrm.Employees.Add(emp4);
hrm.Employees.Add(emp5);
//lưu
hrm.SaveChanges();
//truy vấn dữ liệu
var listE = from e in hrm.Employees select e;
Console.WriteLine("Hien thi danh sach nhan vien:");
foreach (var item in listE)
```

```
{
    Console.WriteLine(item.FirstName + " " + item.LastName + "- Phong:"
        + item.Department.DepName);
}
//lấy các nhân viên có tên bắt đầu bằng chữ H
var listH = from e in hrm.Employees where e.LastName.StartsWith("H")
select e;
Console.WriteLine("Hien thi danh sach nhan vien co ten bat dau bang chu
H:");
foreach (var item in listH)
{
    Console.WriteLine(item.FirstName + " " + item.LastName);
}
//Sử dụng phương thức mở rộng của LINQ để lấy dữ liệu
//lấy tất cả tên nhân viên có chứa chữ a
var listA = hrm.Employees.Where(x => x.LastName.Contains("a"));
Console.WriteLine("Hien thi danh sach nhan vien co ten chua chu a:");
foreach (var item in listA)
{
    Console.WriteLine(item.FirstName + " " + item.LastName);
}
//lấy 3 nhân viên đầu
var list3 = hrm.Employees.Take(3);
Console.WriteLine("Hien thi danh sach 3 nhan vien dau tien:");
foreach (var item in list3)
{
    Console.WriteLine(item.FirstName + " " + item.LastName);
}
}
```

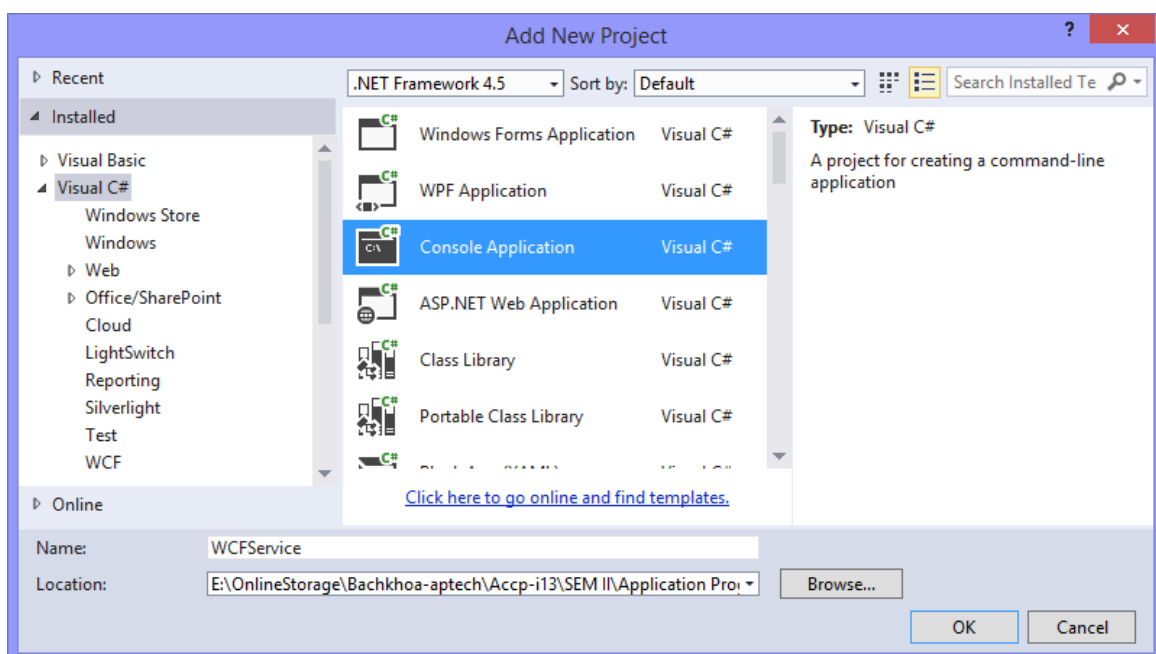
BƯỚC 5: Ctrl + F5 để chạy và kiểm tra kết quả

Bài 10.6

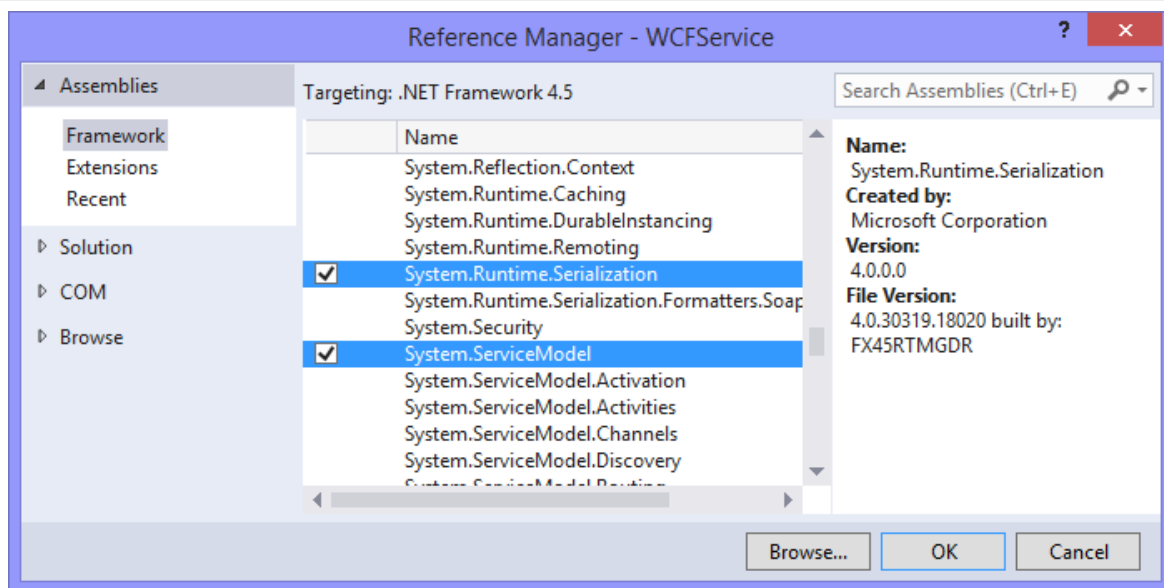
Viết chương trình C# và thực hiện những công việc sau:

- Tạo WCF cung cấp dịch vụ trả về danh sách sản phẩm, dịch vụ trả về sản phẩm theo mã số
- Tạo ứng dụng client để gọi dịch vụ trên.

Bước 1: Kích chuột phải vào Solution “Session10” chọn Add -> New Project -> nhập tên.



Bước 2: Kích chuột phải vào mục References -> Add References để thêm thư viện vào



Bước 3: Tạo tệp tin IProductService.cs và code lớp Product, giao diện IProductService theo gợi ý sau:

//Định nghĩa giao diện cung cấp các phương thức (service)

[ServiceContract]

interface IProductService

{

 [OperationContract]

 List<Product> GetProducts();

 [OperationContract]

 Product GetProduct(int productId);

}

//Định nghĩa lớp ràng buộc dữ liệu

[DataContract]

public class Product

{

 [DataMember]

 public int Id { get; set; }

 [DataMember]

 public string Name { get; set; }

 [DataMember]

 public int Price { get; set; }

```
}
```

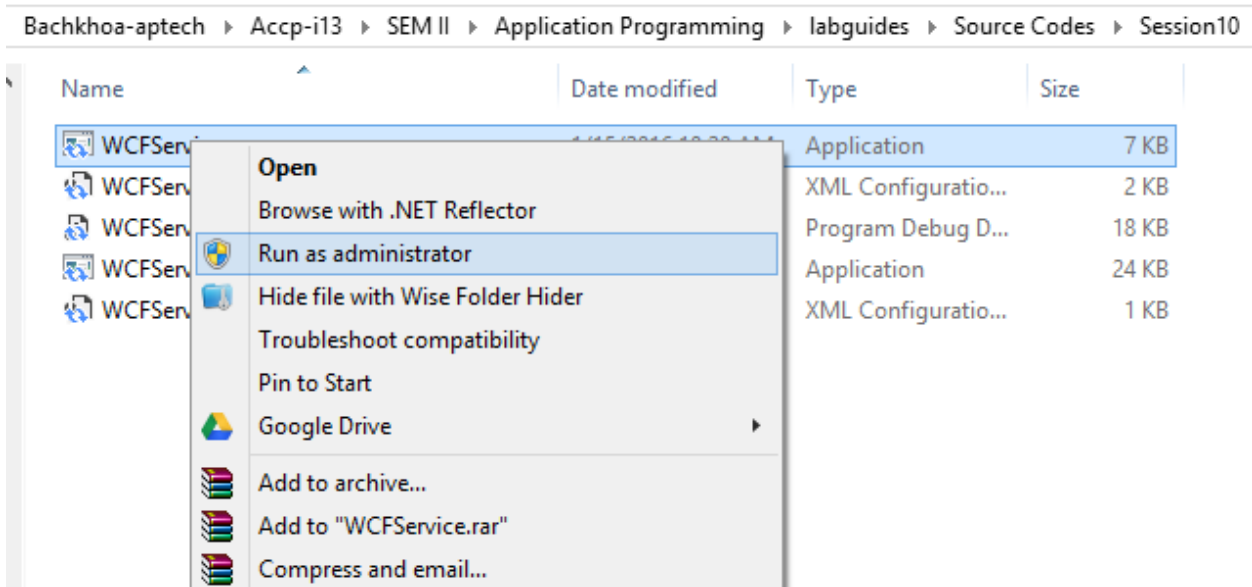
Bước 4: Tạo lớp ProductService.cs thực thi từ giao diện IProductService theo code gợi ý sau:

```
//Định nghĩa lớp dịch vụ thực thi từ giao diện
public class ProductService:IProductService
{
    //Khai báo danh sách sản phẩm
    List<Product> products = new List<Product>();
    public ProductService()
    {
        //Khởi tạo danh sách sản phẩm
        products.Add(new Product() { Id = 1, Name = "Galaxy Tab
5",Price=1300000 });
        products.Add(new Product() { Id = 2, Name = "Sony Z5", Price =
1800000 });
        products.Add(new Product() { Id = 3, Name = "LG V10", Price =
1600000 });
        products.Add(new Product() { Id = 4, Name = "Samsung S4", Price =
1100000 });
    }
    //Thực thi phương thức lấy danh sách products
    public List<Product> GetProducts()
    {
        return products;
    }
    //Thực thi phương thức lấy product theo Id
    public Product GetProduct(int productId)
    {
        return products.FirstOrDefault(p => p.Id == productId);
    }
}
```

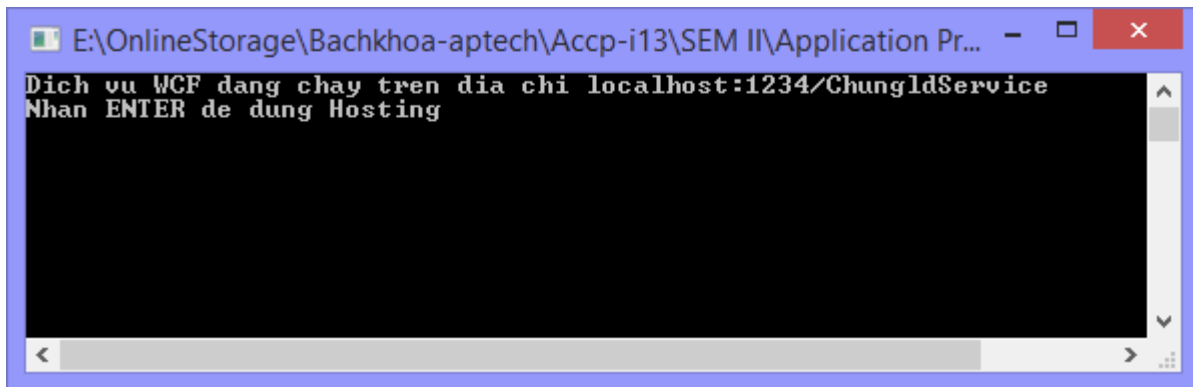
Bước 5: Mở tệp App.config trong project vào cấu hình theo mẫu sau:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>
  <!-- Cấu hình dịch vụ -->
  <system.serviceModel>
    <services>
      <service name="WCFSvc.ProductService"
        behaviorConfiguration="productServiceBehavior">
        <host>
          <baseAddresses>
            <add baseAddress="http://localhost:1234/ChungIdService"/>
          </baseAddresses>
        </host>
        <endpoint address=""
          binding="basicHttpBinding"
          contract="WCFSvc.IProductService" />
        <endpoint address="mex"
          binding="mexHttpBinding"
          contract="IMetadataExchange" />
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior name="productServiceBehavior">
          <serviceMetadata httpGetEnabled="true"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
  <!--Kết thúc phần cấu hình-->
</configuration>
```

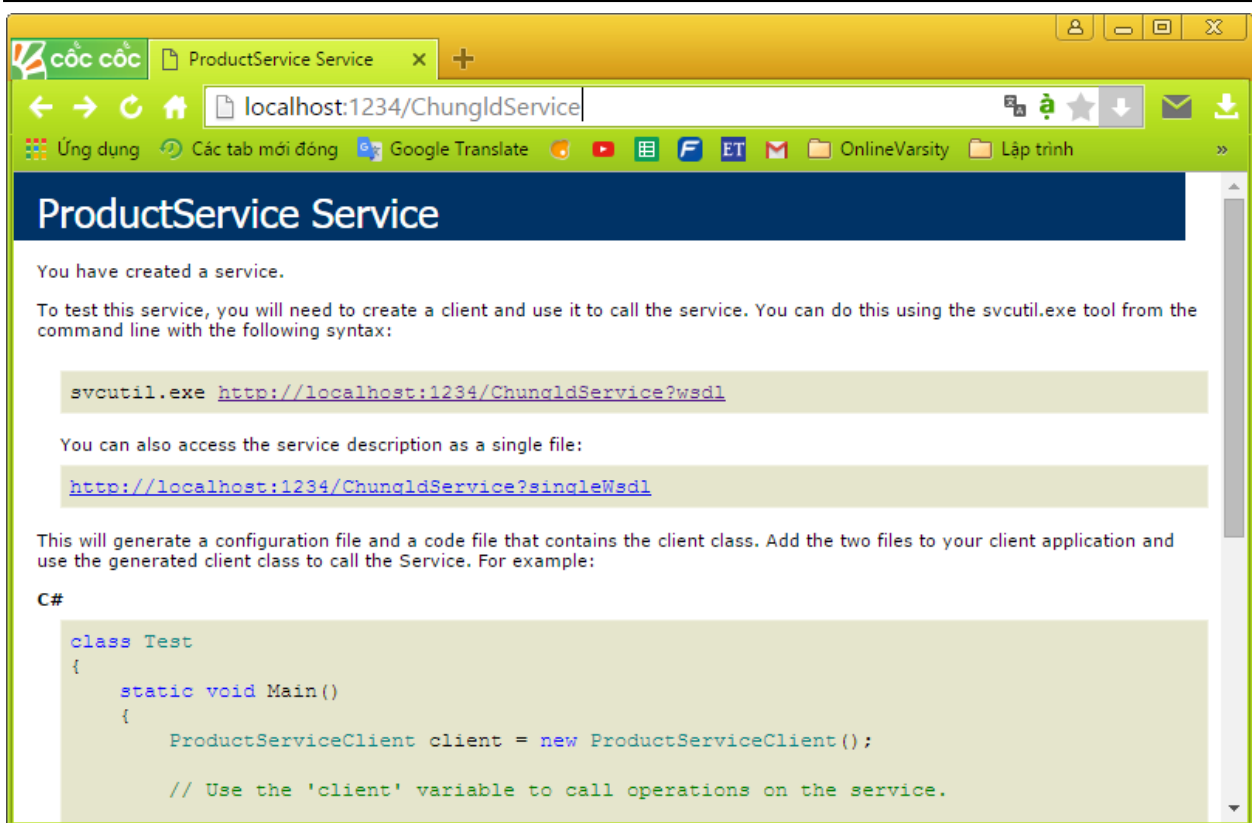
BƯỚC 6: Build project -> mở thư mục ...Session10\WCFSERVICE\bin\Debug ra để khởi động dịch vụ nhé:



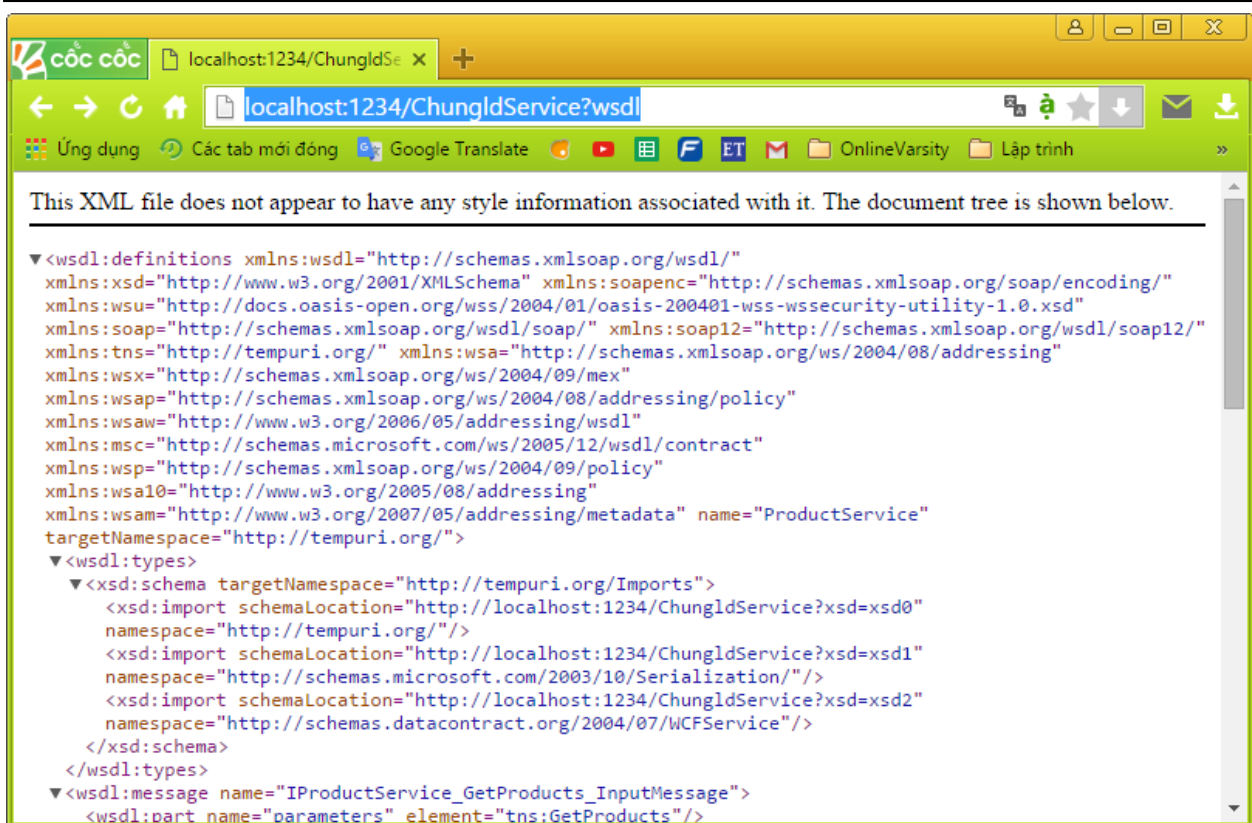
Dịch vụ đã chạy (các bạn nhớ đừng đóng cửa sổ này nha)



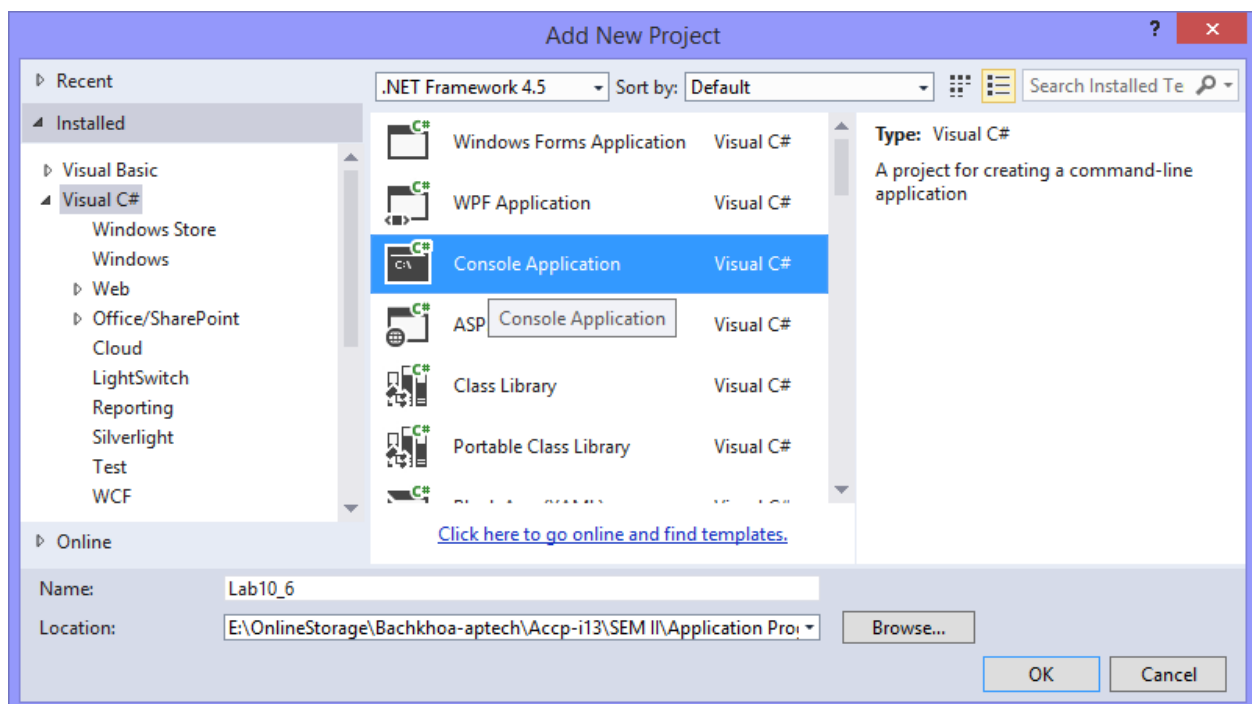
BƯỚC 7: Mở cửa sổ trình duyệt vào gõ vào địa chỉ trên các bạn sẽ thấy



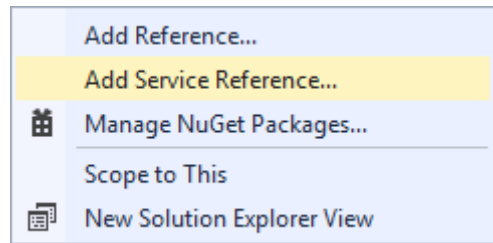
Kích vào link đầu tiên sẽ nhìn thấy tài liệu WSDL được sinh ra (nhớ copy url trên trình duyệt để lát đưa vào client dùng nhé – copy ngay đi kẻo quên)



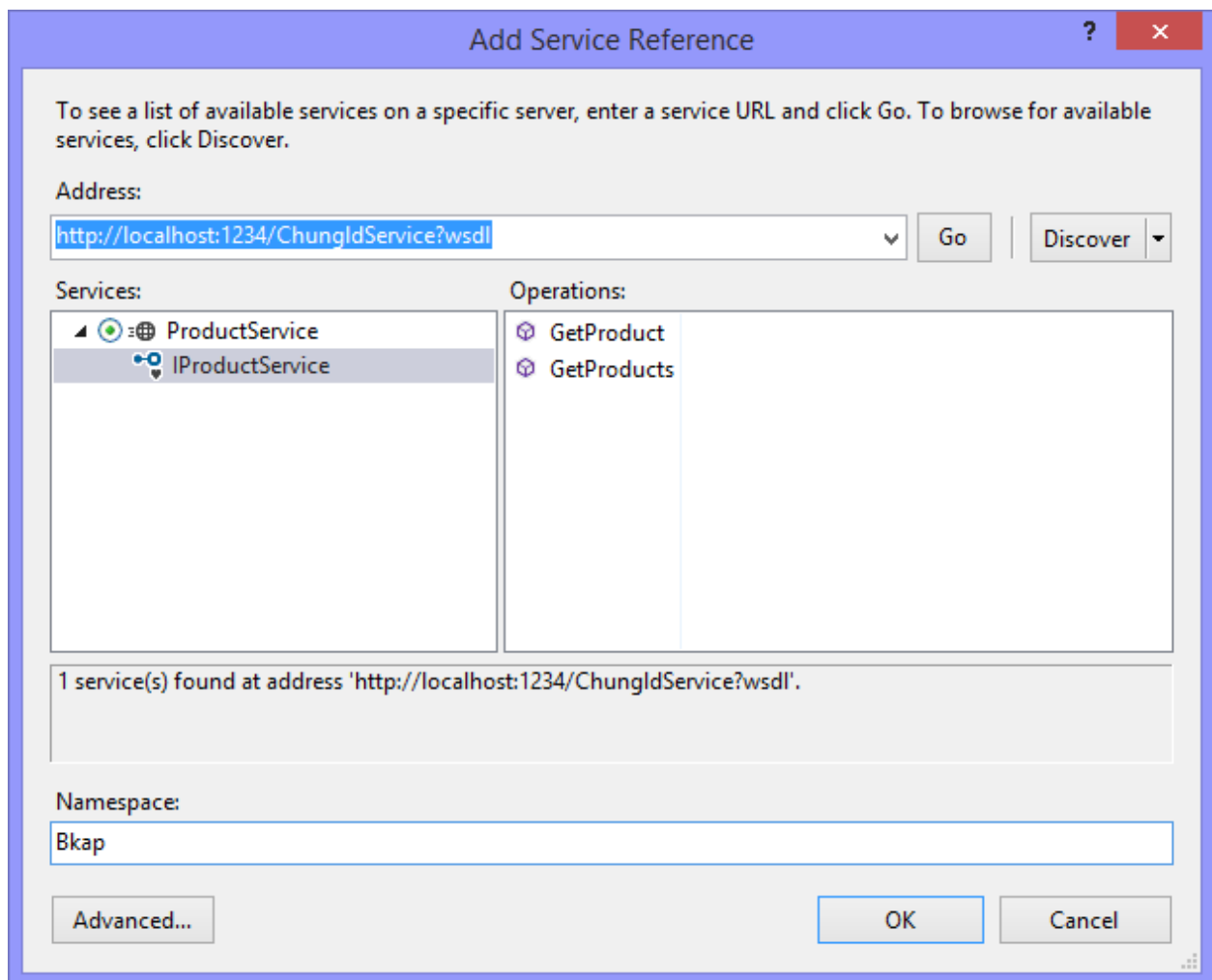
BƯỚC 8: Thêm một project mới “Lab10_6” vào solution “Session10” để gọi dịch vụ trên



BƯỚC 9: Kích chuột phải vào mục References chọn Add Service Reference...



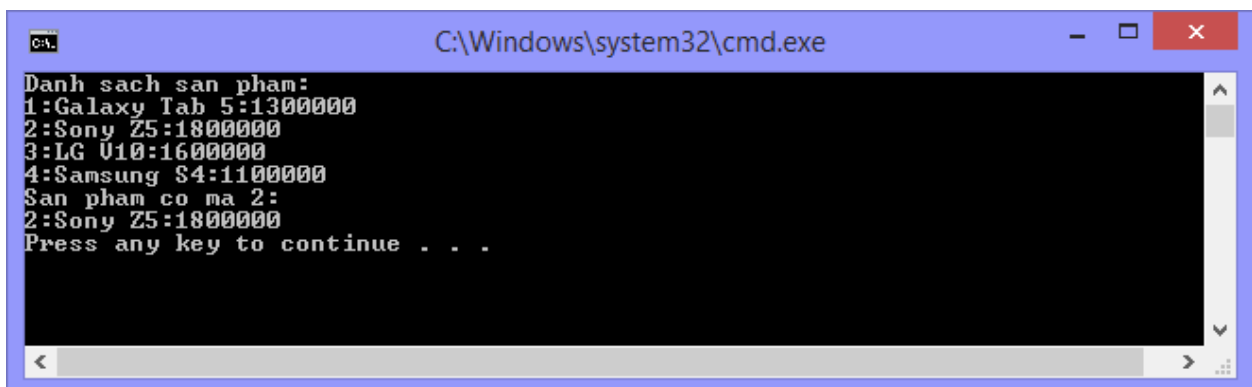
Paste url vừa copy ở trên vào ô Address và click “Go” -> nhập tên Bkap vào ô Namespace -> OK.



BƯỚC 10: Mở tệp Program.cs và code theo gợi ý sau:

```
class Program
{
    static void Main(string[] args)
    {
        //Tạo đối tượng Client
        Bkap.ProductServiceClient client = new Bkap.ProductServiceClient();
        //lấy danh sách sản phẩm
        Bkap.Product[] products = client.GetProducts();
        //in kết quả
        Console.WriteLine("Danh sach san pham:");
        foreach (var p in products)
        {
            Console.WriteLine(p.Id + ":" + p.Name + ":" + p.Price);
        }
        //lấy sản phẩm có mã 2
        Bkap.Product p2 = client.GetProduct(2);
        //in kết quả
        Console.WriteLine("San pham co ma 2:");
        Console.WriteLine(p2.Id + ":" + p2.Name + ":" + p2.Price);
    }
}
```

Bước 11: Ctrl+F5 chạy và xem kết quả:



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed as follows:

```
Danh sach san pham:
1:Galaxy Tab 5:1300000
2:Sony Z5:1800000
3:LG U10:1600000
4:Samsung S4:1100000
San pham co ma 2:
2:Sony Z5:1800000
Press any key to continue . . .
```

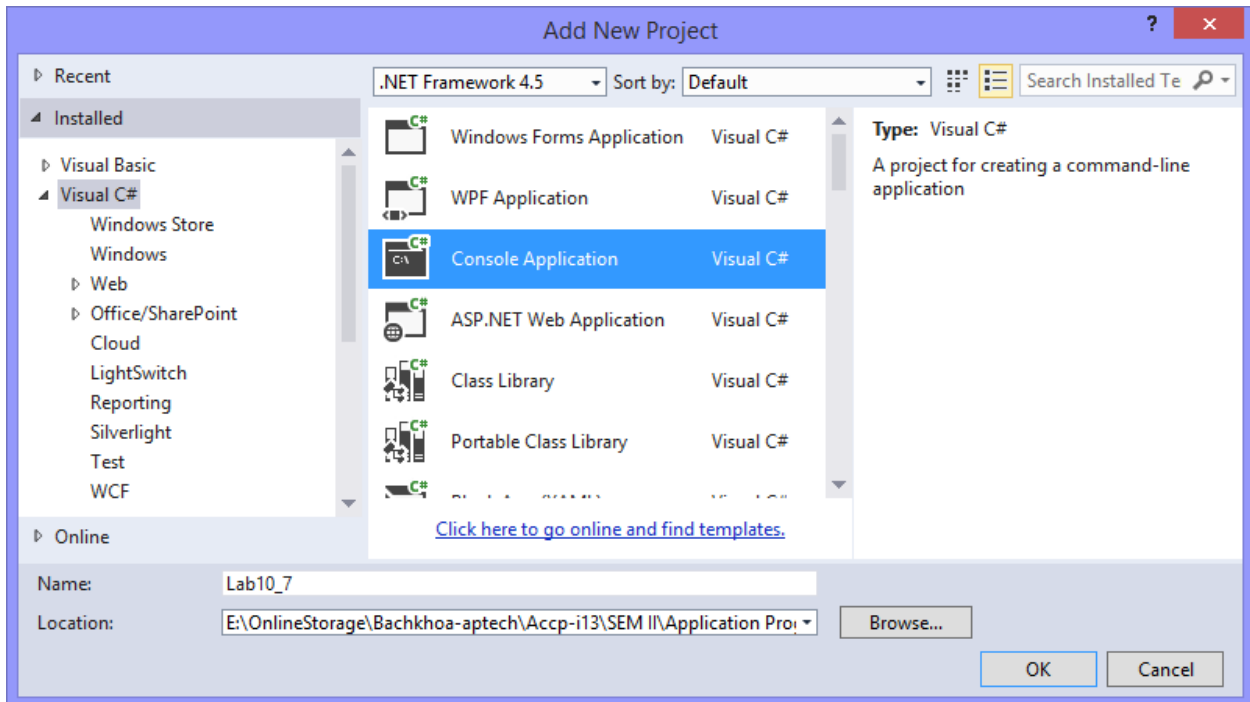

Application Programming in C#



Bài 10.7

Viết chương trình C# minh họa việc chạy đa luồng:

BƯỚC 1: Kích chuột phải vào Solution “Session10” chọn Add -> New Project -> nhập tên.



BƯỚC 2: Mở Program.cs và code theo gợi ý sau:

```
class Program
{
    /// <summary>
    /// Chương trình minh họa 2 thread chạy cùng nhau
    /// </summary>
    /// <param name="args"></param>
    static void Main(string[] args)
    {
        //Tạo một thread mới và gọi phương thức Print trong lớp Bkap
        Thread th = new Thread(new ThreadStart(Bkap.Print));
        //chạy thread
        th.Start();
    }
}
```

```
//lệnh lặp này sẽ chạy trong Thread Main và in ra dòng Thread Main  
mãi mãi  
while (true)  
    Console.WriteLine(" Thread Main ");  
}  
}  
//Định nghĩa lớp Bkap  
class Bkap  
{  
    //phương thức tĩnh Print sẽ in ra từ Thread Bkap mãi mãi  
    public static void Print()  
    {  
        while(true)  
            Console.WriteLine(" Thread Bkap ");  
    }  
}
```

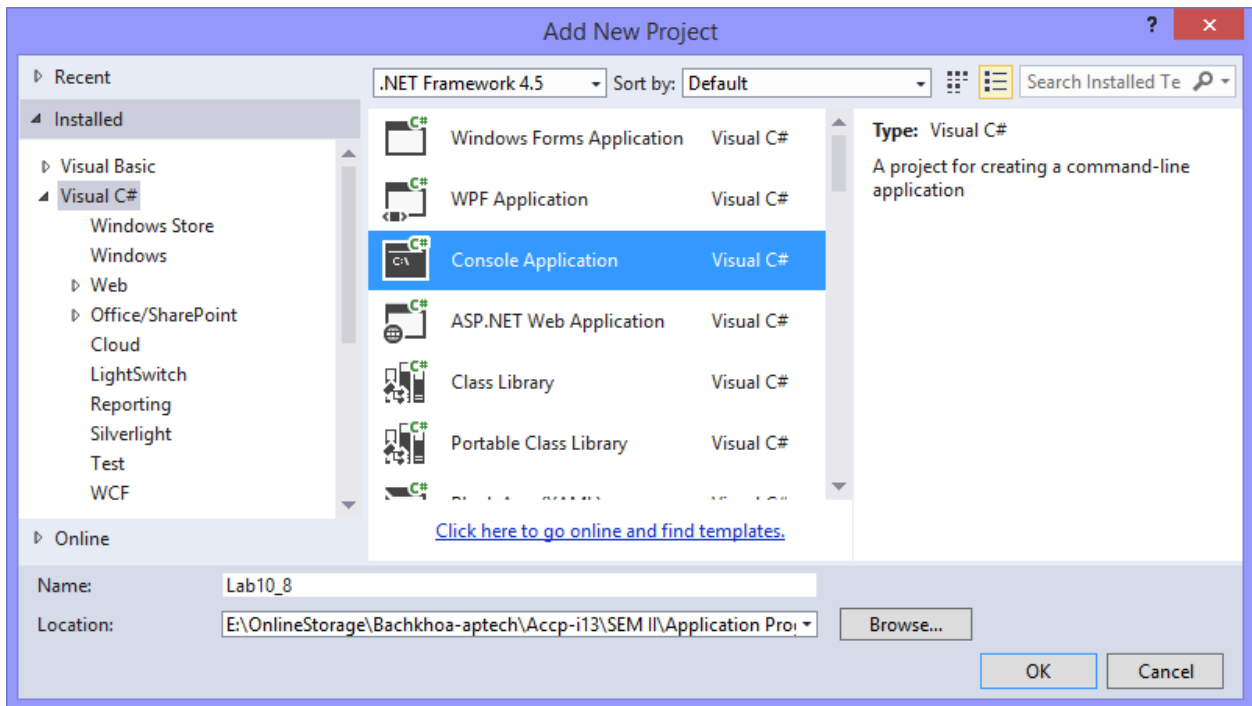
BƯỚC 3: Ctrl+F5 để chạy và kiểm tra kết quả



Bài 10.8

Viết chương trình C# minh họa việc chạy đa nhiệm:

BƯỚC 1: Kích chuột phải vào Solution “Session10” chọn Add -> New Project -> nhập tên.

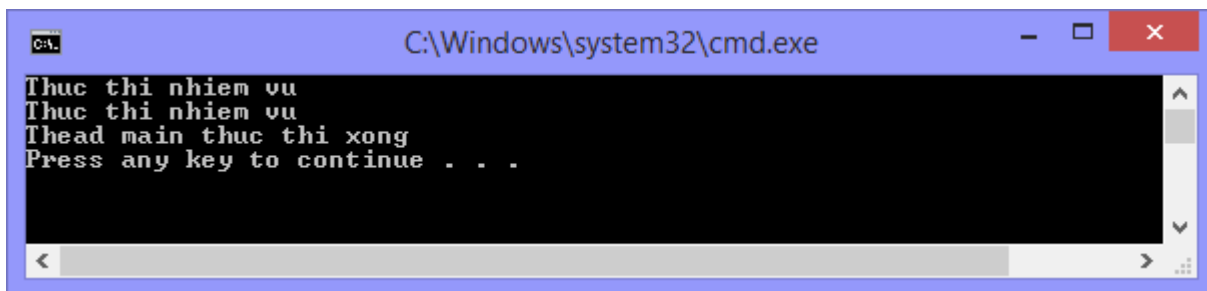


BƯỚC 2: Mở Program.cs và code theo gợi ý sau:

```
/// <summary>
/// Chương trình minh họa việc chạy đa nhiệm
/// </summary>
class Program
{
    //định nghĩa phương thức print
    static void Print()
    {
        Console.WriteLine("Thuc thi nhien vu");
    }
    static void Main(string[] args)
    {
```

```
//tạo nhiệm vụ 1
Task t1 = new Task(new Action(Print));
//chạy
t1.Start();
//tạo nhiệm vụ 2 và chạy luôn
Task t2 = Task.Run(() => Print());
//đợi kết thúc
t1.Wait();
t2.Wait();
//thông báo luồng main
Console.WriteLine("Thread main thuc thi xong");
}
}
```

BƯỚC 3: Ctrl+F5 để chạy và kiểm tra kết quả



The screenshot shows a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe". The window has a blue title bar and a black background. The output text is as follows:

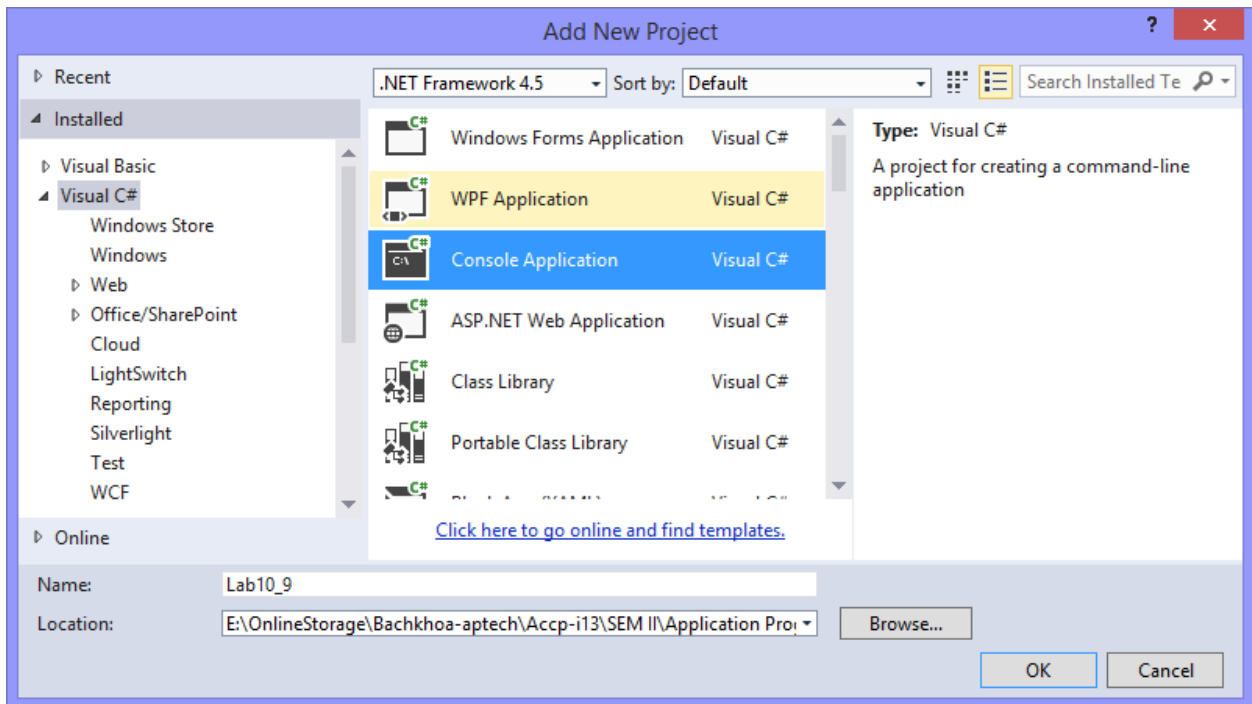
```
Thuc thi nhiem vu
Thuc thi nhiem vu
Thread main thuc thi xong
Press any key to continue . . .
```

The text is displayed in a monospaced font. The first two lines are "Thuc thi nhiem vu", the third line is "Thread main thuc thi xong", and the fourth line is "Press any key to continue . . .". The window has a scrollbar on the right side.

Bài 10.9

Viết chương trình C# minh họa so sánh việc sử dụng vòng lặp thông thường và lặp song song:

BƯỚC 1: Kích chuột phải vào Solution “Session10” chọn Add -> New Project -> nhập tên.

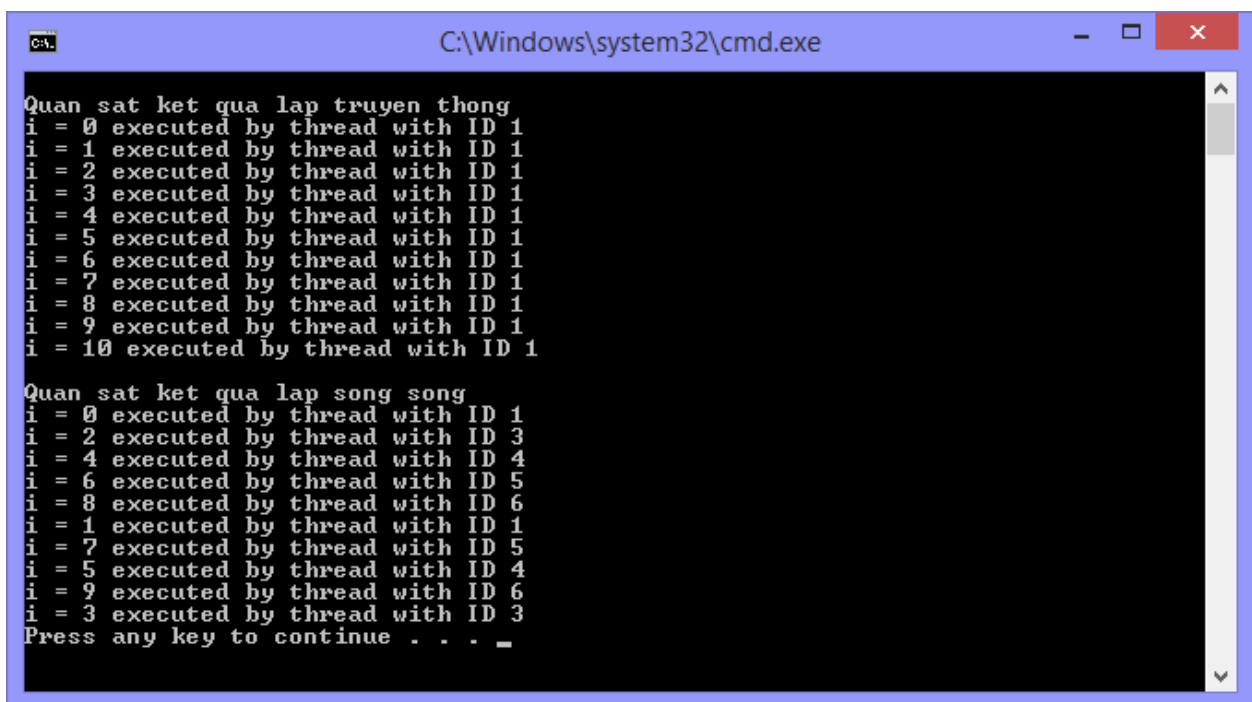


BƯỚC 2: Mở Program.cs và code theo gợi ý sau:

```
class Program
{
    /// <summary>
    /// Chương trình minh họa so sánh việc lặp truyền thống và lặp song song
    /// </summary>
    /// <param name="args"></param>
    static void Main(string[] args)
    {
        Console.WriteLine("\nQuan sat ket qua lap truyen thong");
        for (int i = 0; i <= 10; i++)
        {
```

```
        Console.WriteLine("i = {0} executed by thread with ID {1}",  
            i, Thread.CurrentThread.ManagedThreadId);  
        Thread.Sleep(200);  
    }  
    Console.WriteLine("\nQuan sat ket qua lap song song");  
    Parallel.For(0, 10, i =>  
    {  
        Console.WriteLine("i = {0} executed by thread with ID {1}",  
            i, Thread.CurrentThread.ManagedThreadId);  
        Thread.Sleep(200);  
    });  
}
```

BƯỚC 3: Ctrl+F5 để chạy và kiểm tra kết quả

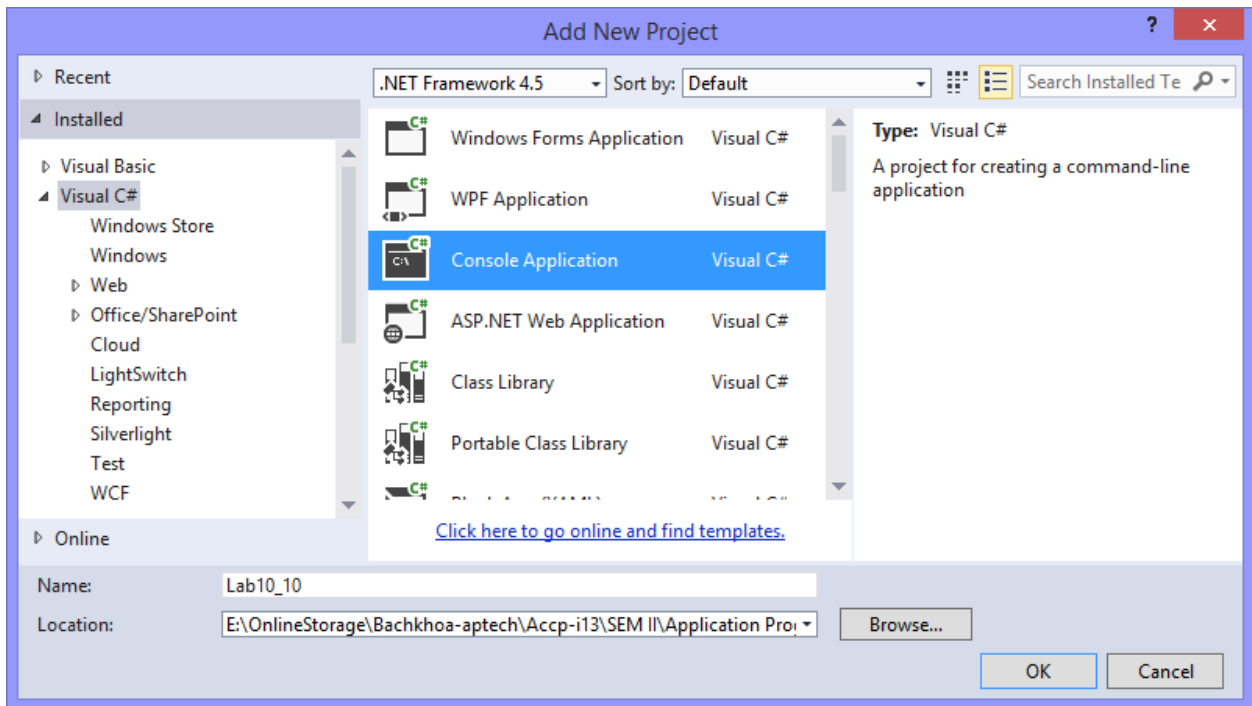


```
C:\Windows\system32\cmd.exe  
Quan sat ket qua lap truyen thong  
i = 0 executed by thread with ID 1  
i = 1 executed by thread with ID 1  
i = 2 executed by thread with ID 1  
i = 3 executed by thread with ID 1  
i = 4 executed by thread with ID 1  
i = 5 executed by thread with ID 1  
i = 6 executed by thread with ID 1  
i = 7 executed by thread with ID 1  
i = 8 executed by thread with ID 1  
i = 9 executed by thread with ID 1  
i = 10 executed by thread with ID 1  
Quan sat ket qua lap song song  
i = 0 executed by thread with ID 1  
i = 2 executed by thread with ID 3  
i = 4 executed by thread with ID 4  
i = 6 executed by thread with ID 5  
i = 8 executed by thread with ID 6  
i = 1 executed by thread with ID 1  
i = 7 executed by thread with ID 5  
i = 5 executed by thread with ID 4  
i = 9 executed by thread with ID 6  
i = 3 executed by thread with ID 3  
Press any key to continue . . . _
```

Bài 10.10

Viết chương trình C# minh họa việc sử dụng LINQ để truy vấn song song

BƯỚC 1: Kích chuột phải vào Solution “Session10” chọn Add -> New Project -> nhập tên.



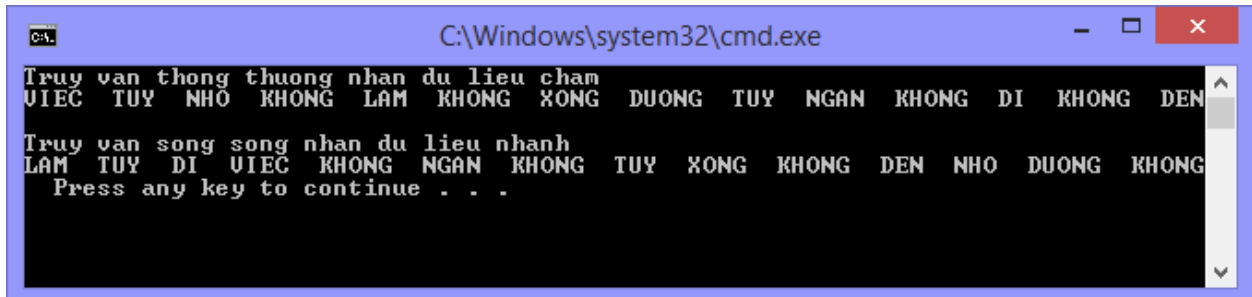
BƯỚC 2: Mở Program.cs và code theo gợi ý sau:

```
class Program
{
    /// <summary>
    /// Chương trình minh họa việc sử dụng LINQ để thực hiện truy vấn song
    /// song
    /// </summary>
    /// <param name="args"></param>
    static void Main(string[] args)
    {
        string[] words = { "viec", "tuy", "nho", "khong", "lam", "khong",
            "xong", "duong", "tuy", "ngan", "khong", "di", "khong", "den" };
        Console.WriteLine("Truy van thong thuong nhan du lieu cham");
    }
}
```



```
IEnumerable<string> resultNormal = from w in words select
w.ToUpper();
foreach (var r in resultNormal)
{
    Console.Write(r + " ");
}
Console.WriteLine("\nTruy van song song nhan du lieu nhanh");
IEnumerable<string> resultParallel = from wp in words.AsParallel()
select wp.ToUpper();
foreach (var r in resultParallel)
{
    Console.Write(r + " ");
}
}
```

BƯỚC 3: Ctrl+F5 để chạy và kiểm tra kết quả

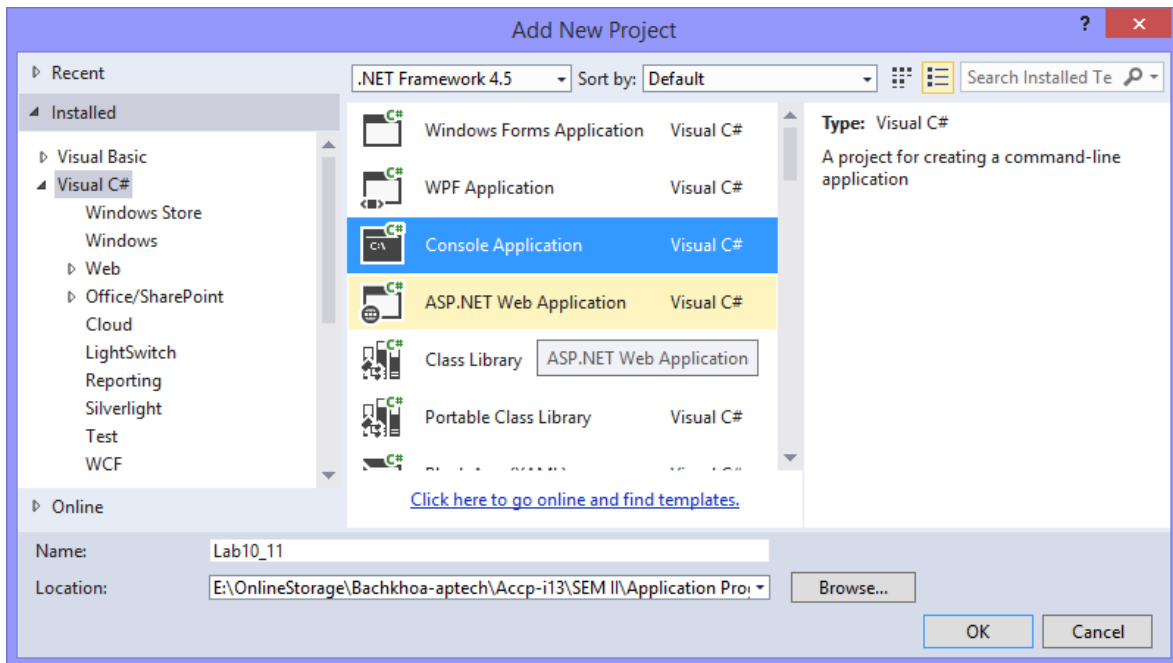


```
C:\Windows\system32\cmd.exe
Truy van thong thuong nhan du lieu cham
UIEC TUY NHỎ KHÔNG LAM KHÔNG XONG DUONG TUY NGAN KHÔNG DI KHÔNG DEN
Truy van song song nhan du lieu nhanh
LAM TUY DI UIEC KHÔNG NGAN KHÔNG TUY XONG KHÔNG DEN NHỎ DUONG KHÔNG
Press any key to continue . . .
```

Bài 10.11

Viết chương trình C# minh họa việc thao tác đồng thời trên Collection

BƯỚC 1: Kích chuột phải vào Solution “Session10” chọn Add -> New Project -> nhập tên.

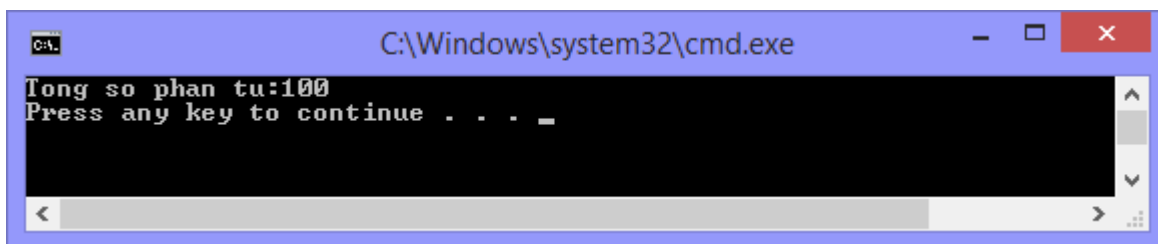


BƯỚC 2: Mở Program.cs và code theo gợi ý sau:

```
class Program
{
    //khai báo một dictionary cho phép thêm dữ liệu đồng thời
    static ConcurrentDictionary<string, int> dic = new
    ConcurrentDictionary<string, int>();
    //tạo phương thức add dữ liệu vào dic
    public static void AddToDictionary()
    {
        //duyệt từ 1-100
        for (int i = 1; i <= 100; i++)
        {
            //add vào dic, nếu phần tử nào có rồi thì nó từ bỏ qua
            dic.TryAdd(i.ToString(), i);
        }
    }
}
```

```
    }  
}  
static void Main(string[] args)  
{  
    //tạo thread 1 và thực thì AddTodictionary  
    Thread th1 = new Thread(new ThreadStart(AddTodictionary));  
    //tạo thread 2 và thực thì AddTodictionary  
    Thread th2 = new Thread(new ThreadStart(AddTodictionary));  
    //chạy 2 thread đồng thời  
    th1.Start();  
    th2.Start();  
    //chờ th1 và th2 hoàn thành  
    th1.Join();  
    th2.Join();  
    //in ra tổng  
    Console.WriteLine("Tong so phan tu:" + dic.Count);  
}  
}
```

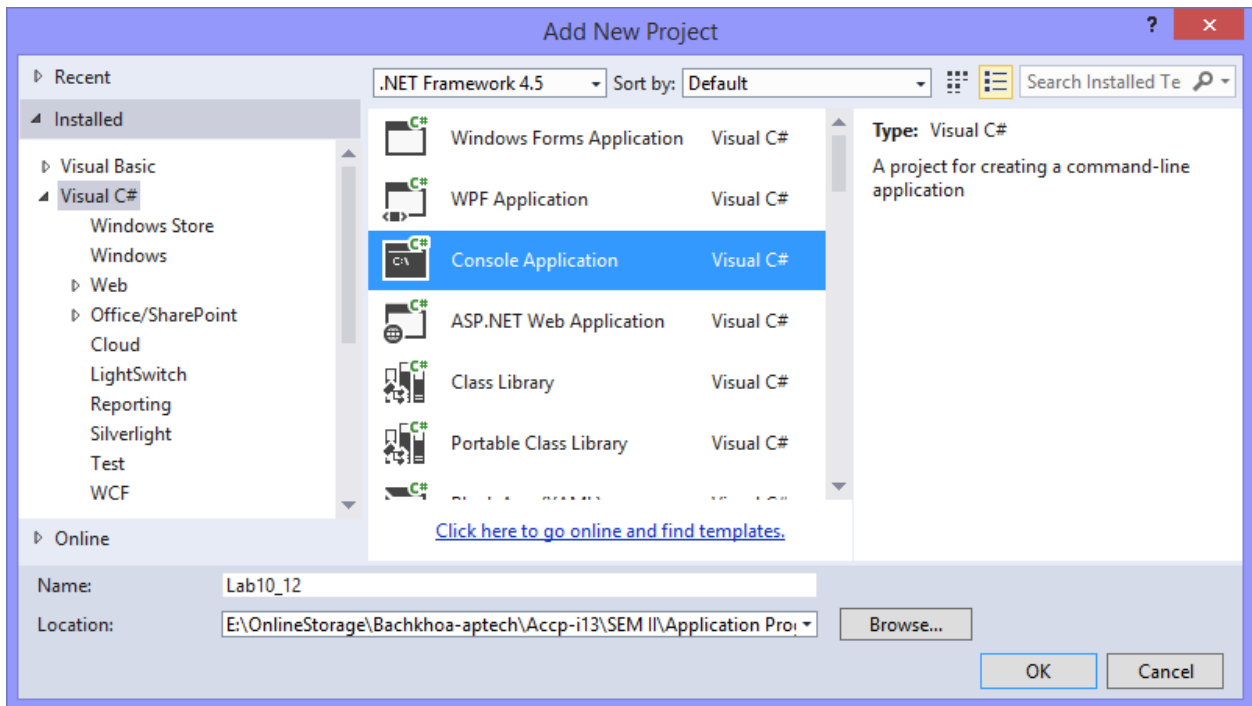
BƯỚC 3: Ctrl+F5 để chạy và kiểm tra kết quả



Bài 10.12

Viết chương trình C# minh họa việc tạo và sử dụng phương thức bất đồng bộ

BƯỚC 1: Kích chuột phải vào Solution “Session10” chọn Add -> New Project -> nhập tên.

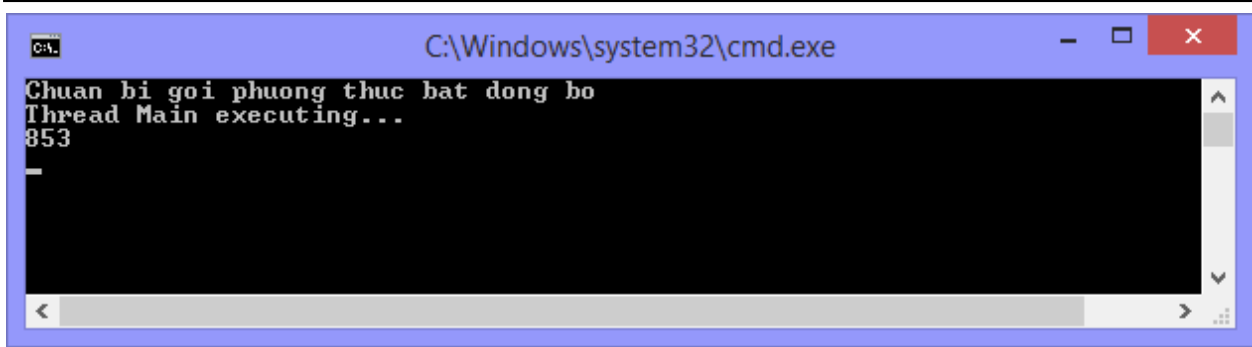


BƯỚC 2: Mở Program.cs và code theo gợi ý sau:

```
class Program
{
    //Định nghĩa phương thức xử lý bất đồng bộ
    static async void PerformComplexTaskAsync()
    {
        Console.WriteLine("Chuan bi goi phuong thuc bat dong bo");
        int result = await new ComplexTask().AnalyzeData();
        Console.WriteLine(result.ToString());
    }
    static void Main(string[] args)
    {
        PerformComplexTaskAsync();
    }
}
```

```
        Console.WriteLine("Thread Main executing...");
        Console.Read();
    }
}
//tạo 1 lớp xử lý công việc phức tạp
class ComplexTask
{
    //Định nghĩa phương thức phân tích dữ liệu và lấy kết quả
    public Task<int> AnalyzeData()
    {
        //khởi tạo đối tượng nhiệm và nhận kết quả là kiểu nguyên
        Task<int> task = new Task<int>(GetResult);
        //bắt đầu thực thi
        task.Start();
        //trả về kết quả
        return task;
    }
    //Định nghĩa phương thức lấy kết quả
    public int GetResult()
    {
        //Giả định công việc thực hiện hết 5s
        Thread.Sleep(5000);
        //trả về số ngẫu nhiên từ 1-1000
        return new Random().Next(1, 1000);
    }
}
```

Bước 3: Ctrl+F5 để chạy và kiểm tra kết quả

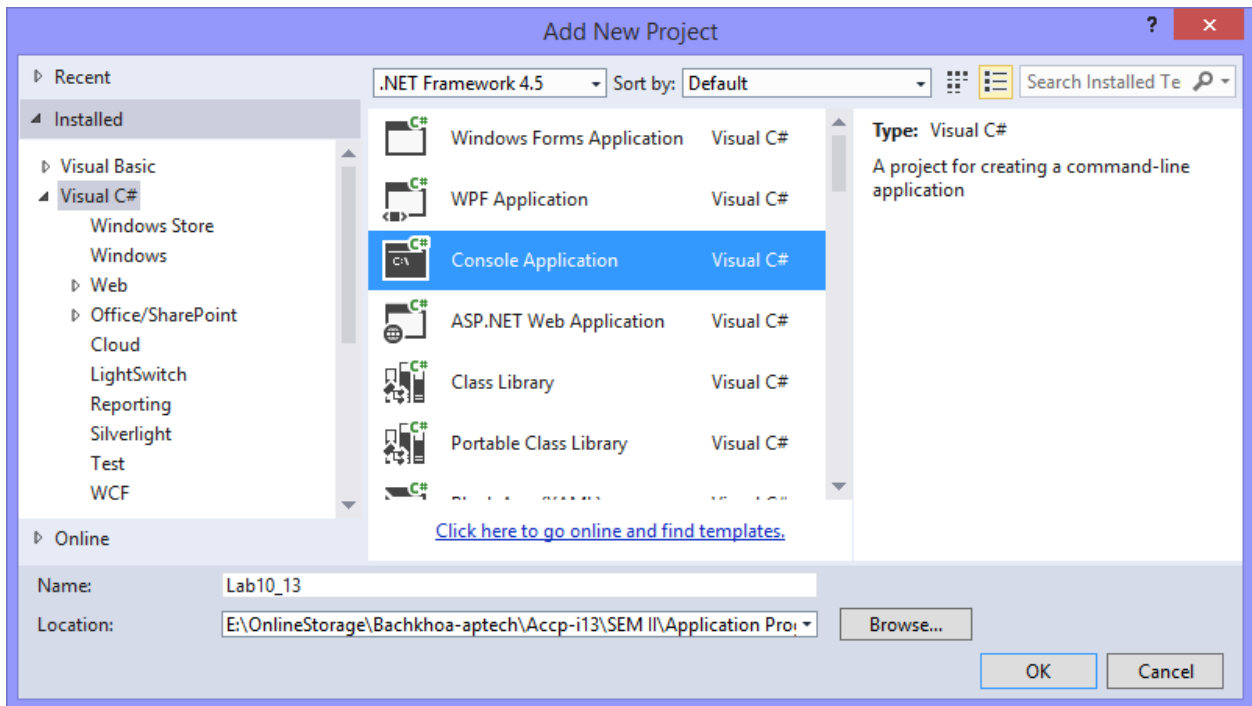


```
C:\Windows\system32\cmd.exe
Chuan bi goi phuong thuc bat dong bo
Thread Main executing...
853
```

Bài 10.13

Viết chương trình C# minh họa việc tạo và sử dụng phương thức động(dynamic)

BƯỚC 1: Kích chuột phải vào Solution “Session10” chọn Add -> New Project -> nhập tên.

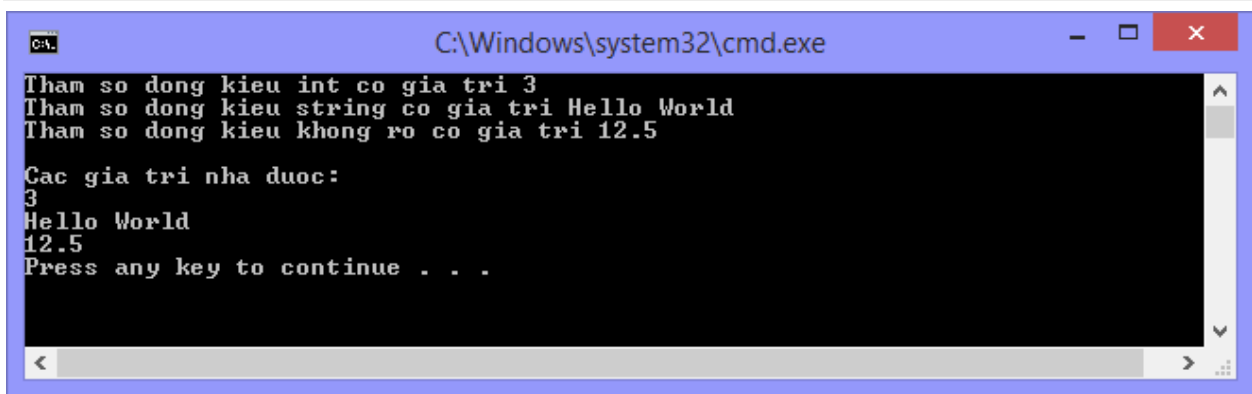


BƯỚC 2: Mở Program.cs và code theo gợi ý sau:

```
class Program
{
    /// <summary>
    /// Chương trình minh họa phương thức động
    /// </summary>
    /// <param name="param"></param>
    /// <returns></returns>
    static dynamic DynamicMethod(dynamic param)
    {
        if (param is int)
        {
```

```
        Console.WriteLine("Tham so dong kieu int co gia tri {0}",  
            param);  
        return param;  
    }  
    else if (param is string)  
    {  
        Console.WriteLine("Tham so dong kieu string co gia tri {0}",  
            param);  
        return param;  
    }  
    else  
    {  
        Console.WriteLine("Tham so dong kieu khong ro co gia tri  
{0}", param);  
        return param;  
    }  
}  
  
static void Main(string[] args)  
{  
    dynamic dynaVar1 = DynamicMethod(3);  
    dynamic dynaVar2 = DynamicMethod("Hello World");  
    dynamic dynaVar3 = DynamicMethod(12.5);  
    Console.WriteLine("\nCac gia tri nha duoc:\n{0} \n{1} \n{2}",  
        dynaVar1, dynaVar2, dynaVar3);  
}  
}
```

Bước 3: Ctrl+F5 để chạy và kiểm tra kết quả



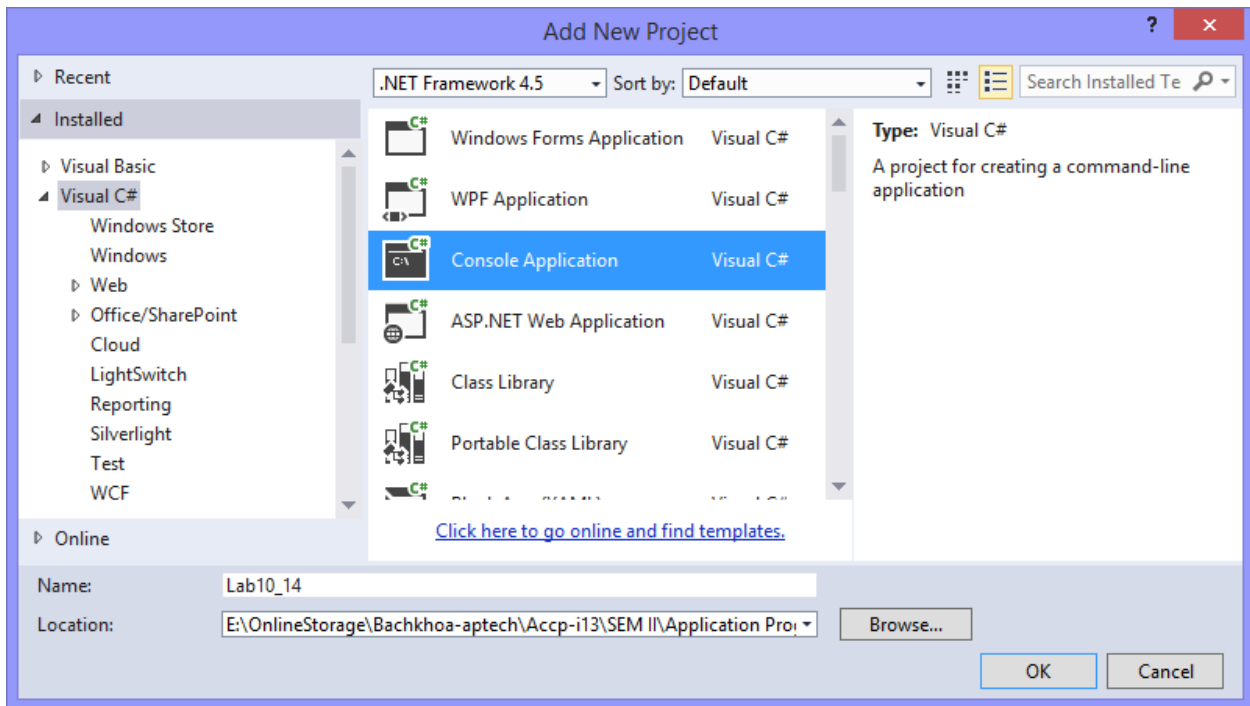
```
C:\Windows\system32\cmd.exe
Tham so dong kieu int co gia tri 3
Tham so dong kieu string co gia tri Hello World
Tham so dong kieu khong ro co gia tri 12.5

Cac gia tri nha duoc:
3
Hello World
12.5
Press any key to continue . . .
```

Bài 10.14

Viết chương trình C# minh họa việc mã hóa đối xứng sử dụng giải thuật RijndaelManaged

BƯỚC 1: Kích chuột phải vào Solution “Session10” chọn Add -> New Project -> nhập tên.



BƯỚC 2: Mở Program.cs và code theo gợi ý sau:

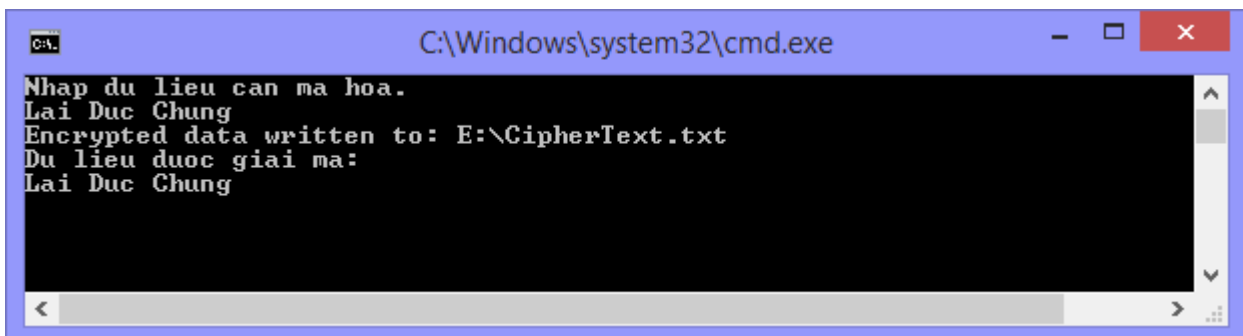
```
/// <summary>
/// Chương trình minh họa việc mã hóa đối xứng sử dụng giải thuật
RijndaelManaged
/// </summary>
class Program
{
    //Phương thức mã hóa với chuỗi đầu vào và đối tượng giải thuật
    static void EncryptData(String plainText, RijndaelManaged algo)
    {
        //chuyển thành mảng byte
        byte[] plainDataArray = ASCIIEncoding.ASCII.GetBytes(plainText);
        //khởi tạo đối tượng mã hóa
```

```
ICryptoTransform transform = algo.CreateEncryptor();
//mở tệp tin để ghi dữ liệu mã hóa vào
using (var fileStream = new FileStream("E:\\CipherText.txt",
    FileMode.OpenOrCreate, FileAccess.Write))
{
    //tạo stream tham chiếu tới tệp tin
    using (var cryptoStream = new CryptoStream(fileStream,
        transform, CryptoStreamMode.Write))
    {
        //viết dữ liệu từ mảng byte vào stream
        cryptoStream.Write(plainDataArray, 0,
            plainDataArray.GetLength(0));
        Console.WriteLine("Encrypted data written to:
            E:\\CipherText.txt");
    }
}

//Phương thức giải mã dữ liệu với đầu vào là đối tượng giải thuật
static void DecryptData(RijndaelManaged algo)
{
    //khởi tạo đối tượng mã hóa
    ICryptoTransform transform = algo.CreateDecryptor();
    //mở tệp tin để đọc nội dung
    using (var fileStream = new FileStream("E:\\CipherText.txt",
        FileMode.Open, FileAccess.Read))
    {
        //giải mã và đọc nội dung ra stream
        using (CryptoStream cryptoStream = new
            CryptoStream(fileStream, transform, CryptoStreamMode.Read))
        {
            //khởi tạo bộ đọc dữ liệu từ stream
            using (var streamReader = new
                StreamReader(cryptoStream))
            {
                //đọc dữ liệu từ stream
            }
        }
    }
}
```

```
        string decryptedData = streamReader.ReadToEnd();  
        //in kết quả giải mã  
        Console.WriteLine("Du lieu duoc giai ma: \n{0}",  
            decryptedData);  
    }  
}  
}  
}  
static void Main(string[] args)  
{  
    //Khởi tạo đối tượng giải thuật  
    RijndaelManaged symAlgo = new RijndaelManaged();  
    Console.WriteLine("Nhap du lieu can ma hoa.");  
    string dataToEncrypt = Console.ReadLine();  
    //mã hóa  
    EncryptData(dataToEncrypt, symAlgo);  
    //giải mã  
    DecryptData(symAlgo);  
    Console.Read();  
}  
}
```

BƯỚC 3: Ctrl+F5 để chạy và kiểm tra kết quả



```
C:\Windows\system32\cmd.exe  
Nhap du lieu can ma hoa.  
Lai Duc Chung  
Encrypted data written to: E:\CipherText.txt  
Du lieu duoc giai ma:  
Lai Duc Chung
```

Phần II Bài tập tự làm

Quá nhiều bài tập minh họa nên phần này tạm thời không đưa ra.

HẾT