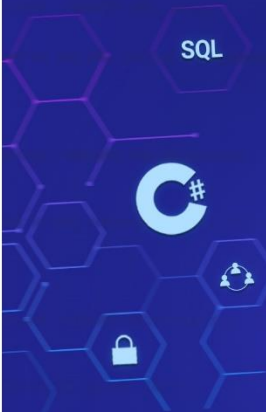
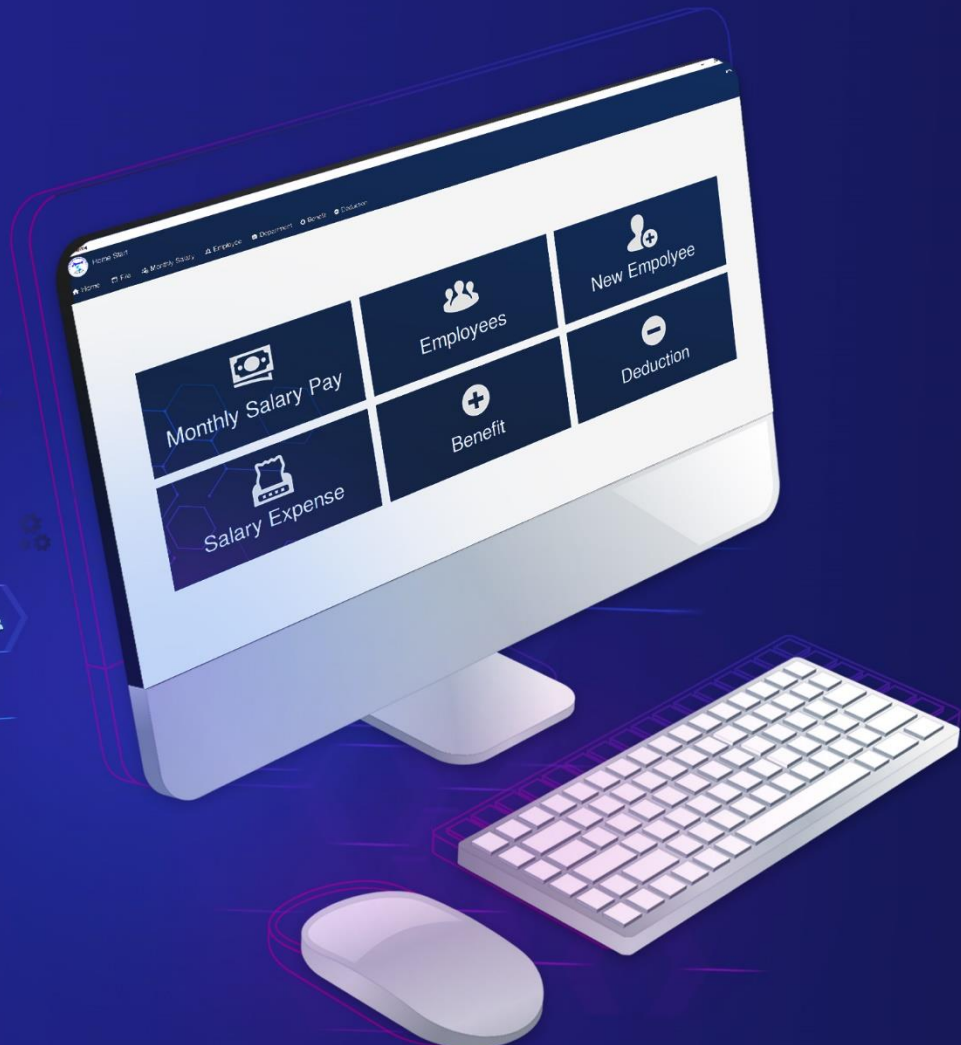




បណ្ណបណ្ណាលបច្ចេកវិទ្យាព័ត៌មានម៉ាស៊ីន  
Master Information Technology Center  
បណ្តុះបណ្តាលចំណេះជំនាញសម្រាប់បំពេញការងារនិងបង្កើតមុខរបរ

# OOP Database

Advanced C# + SQL Server + ADO.Net





## អារម្ភកថា

ការសរសេរកូដបង្កើតកម្មវិធីកុំព្យូទ័រដើម្បីបំពេញតម្រូវការជាក់ស្តែងរបស់មនុស្សកំពុងតែដើរតួយ៉ាងសំខាន់មួយនាយុគសម័យបច្ចេកវិទ្យានេះ។ កម្មវិធីគ្រប់គ្រងឆ្លើយតបព័ត៌មានលើកុំព្យូទ័រ ទូរសព្ទដៃ ដូចជាការកត់ត្រាទំនិញចេញចូល ត្រួតពិនិត្យព័ត៌មានក្នុងហាង គ្រប់គ្រងសណ្ឋាគារ ព័ត៌មានបុគ្គលិក ។ល។ សុទ្ធសឹងតែត្រូវបានចងក្រង រៀបចំ និងបង្កើតឡើងពីការសរសេរកូដកម្មវិធី។ ដើម្បីឆ្លើយតបតាមតម្រូវការនេះ មជ្ឈមណ្ឌលម៉ាស៊ីនអាយធី យើងខ្ញុំបានចងក្រងសៀវភៅ C Sharp Database Programming នេះឡើង ដោយប្រើប្រាស់ភាសាកូដ C Sharp Programming Language និងប្រព័ន្ធគ្រប់គ្រងទិន្នន័យ(RDBMS) Microsoft SQL Server ដើម្បីឲ្យអ្នកសិក្សាទាំងឡាយបានស្វែងយល់ពីដំណើរការនៃការកំណត់កម្មវិធីកុំព្យូទ័រ និងអាចបង្កើតកម្មវិធីកុំព្យូទ័រទៅតាមគំនិតប្រឌិតច្នៃតាមតម្រូវការផ្ទាល់ខ្លួនបាន។

ចំពោះគោលបំណងនៃសៀវភៅ C Sharp Database Programming នេះគឺចង់ឲ្យអ្នកសិក្សាទាំងឡាយបានយល់ជ្រាបពីផ្នែកសំខាន់ៗគឺផ្នែកភាសាកូដ និងផ្នែកគ្រប់គ្រងទិន្នន័យ។ យើងប្រើប្រាស់ភាសាកូដ C Sharp Programming Language ភ្ជាប់ជាមួយនឹងគម្រោង Object Oriented Programming (OOP) ដើម្បីកំណត់ជាធាតុនិងអន្តរកម្មរួមផ្សំគ្នាទៅជាកម្មវិធីកុំព្យូទ័រ។ ចំណែកឯ Microsoft SQL Server ត្រូវបានកំណត់ជាប្រព័ន្ធគ្រប់គ្រងទិន្នន័យ ដោយយើងនឹងសិក្សាពីរបៀបនៃការវិភាគទិន្នន័យ រហូតដល់ការរចនា Database ឲ្យចេញជាប្រភេទឡើង។ វគ្គសិក្សានេះផ្តោតសំខាន់ទៅលើចលនការនៃបម្រែបម្រួលទិន្នន័យពី Database ទៅ Programming ឬត្រឡប់មកវិញ នៅលើទម្រង់ WinForm Application (Desktop App) ដែលសម្រាប់អ្នកមានចំណេះដឹង Programming ជាលក្ខណៈមូលដ្ឋានគ្រឹះពីមុនមក។ សៀវភៅមួយក្បាលនេះគឺជាជំនួយបន្ថែមដល់អ្នកសិក្សាក្នុងការស្វែងយល់ពីទ្រឹស្តី វិធីសាស្ត្រមួយចំនួន និងអាចធ្វើការកត់ត្រាបន្ថែមពេលកំពុងសិក្សានិងអនុវត្ត។

ជាចុងក្រោយ យើងខ្ញុំសង្ឃឹមថា បងប្អូន សិស្សានុសិស្សទាំងអស់គ្នានឹងខិតខំប្រឹងប្រែងសិក្សាមេរៀនក្នុងសៀវភៅនេះដោយយកចិត្តទុកដាក់ និងចូលរួមអនុវត្តន៍លំហាត់ឲ្យបានច្រើន ទើបការសិក្សាអាចទទួលបានជោគជ័យ។ យើងខ្ញុំសូមអភ័យទោសផងដែររាល់ចំពោះកំហុសឆ្គងផ្សេងៗ ដែលកើតមានដោយអចេតនាក្នុងសៀវភៅនេះ។ សូមអរគុណ!!

រៀបរៀងដោយ

**រ៉ូន រីហ្វាន់**

បោះពុម្ពលើកទី១ឆ្នាំ២០២២





## មាតិកា

<b>ជំពូកទី១ ..Object Oriented C Sharp Programming</b>	<b>1</b>
<b>មេរៀនទី១  ស្វែងយល់ពី Class និង Object.....</b>	<b>1</b>
១. និយមន័យ .....	1
២. ការប្រើប្រាស់ Field នៃ Class .....	1
៣. Method នៃ Class .....	2
៣.១ ការប្រើប្រាស់ Method .....	2
៣.២ Method Overloading .....	4
៤. Constructor .....	6
៥. Encapsulation .....	9
៦. Property នៃ Class.....	11
៧. Object Initializer .....	13
<b>មេរៀនទី២  ទំនាក់ទំនងរវាង Classes (Associated Classes) .....</b>	<b>15</b>
១. ការបន្តសមាសភាព ( Inheritance ).....	15
១.១ និយមន័យ .....	15
១.២ ការប្រើប្រាស់ Inheritance .....	15
១.៣ Inheritance Constructor .....	18
១.៤ Upcast និង Downcast .....	20
១.៥ Boxing និង Unboxing .....	22
២. ការដាក់បញ្ចូល (Composition) .....	23
<b>មេរៀនទី៣  ស្វែងយល់ពី Polymorphism .....</b>	<b>25</b>
១. Method Overriding .....	25
២. Abstract Class.....	27
គម្រូ ៖ បង្កើតកម្មវិធីគ្រប់គ្រងបញ្ជីឈ្មោះដោយប្រើប្រាស់ Windows Form App.....	29
<b>ជំពូកទី២ ...Microsoft SQL Server Database Management System</b>	<b>31</b>
<b>មេរៀនទី១  ការ Query ទិន្នន័យពី Table តែមួយ .....</b>	<b>31</b>

១. រំសាយលើ Select Statement .....	31
២. ការប្រើប្រាស់ Select Top Statement.....	32
៣. ការកំណត់លក្ខខណ្ឌលើ Select Statement (Where Clause) .....	32
៣.១ Comparison Operator .....	32
៣.២ Logical Operator .....	33
៣.៣ IN Operator .....	34
៣.៤ Between Operator .....	34
៣.៥ Like Operator .....	35
៤. ការតម្រៀបទិន្នន័យលើ Select Statement (Order By Clause).....	35
<b>មេរៀនទី២ ការ Query ទិន្នន័យពី Tables ច្រើន .....</b>	<b>37</b>
១. ការប្រើប្រាស់ INNER JOIN.....	37
២. ការប្រើប្រាស់ OUTER JOIN.....	38
៣. ការប្រើប្រាស់ SELF JOIN .....	39
<b>មេរៀនទី៣ ការកែប្រែទិន្នន័យនៅលើ Table .....</b>	<b>41</b>
១. ការបញ្ចូលទិន្នន័យឲ្យ Table (Insert Statement) .....	41
១.១ ការ Insert តែមួយ Table .....	41
១.២ ការ Insert បែបច្រើនសម្ព័ន្ធ .....	41
២. ការលុបទិន្នន័យពី Table (Delete Statement) .....	42
៣. ការកែសម្រួលទិន្នន័យលើ Table (Update Statement).....	43
<b>មេរៀនទី៤ Aggregate Function.....</b>	<b>45</b>
១. ការរាប់ចំនួន Rows នៅក្នុងបណ្តុំ (COUNT).....	45
២. ការបូកសរុបតម្លៃនៅក្នុងបណ្តុំ (SUM) .....	45
៣. ការរកមធ្យមតម្លៃនៅក្នុងបណ្តុំ (AVG).....	45
៤. ការ Query ទិន្នន័យដោយដាក់ជាក្រុម (GROUP BY) .....	46
៥. ការកំណត់លក្ខខណ្ឌ Query ទិន្នន័យដោយដាក់ជាក្រុម (HAVING) .....	46
<b>មេរៀនទី៥ ការសិក្សាពី Complex Query .....</b>	<b>47</b>

១. ស្វែងយល់ពី Sub Query .....	47
២. ការកំណត់ Correlated Sub Query .....	48
៣. ការប្រើប្រាស់ Exists Operator .....	48
<b>មេរៀនទី៦ ការសិក្សាពី View.....</b>	<b>49</b>
១. និយមន័យ .....	49
២. ការកំណត់ View .....	49
៣. ការកែប្រែ View .....	50
៤. ការលុប View .....	50
<b>មេរៀនទី៧ ការសិក្សាពី Store Procedure.....</b>	<b>51</b>
១. និយមន័យ .....	51
២. ការកំណត់ Store Procedure.....	51
៣. ការកែប្រែ Store Procedure.....	53
៤. ការលុប Store Procedure .....	54
<b>មេរៀនទី៨ User Defined Function (UDF) .....</b>	<b>55</b>
១. និយមន័យ .....	55
២. ការកំណត់ User Defined Function .....	55
៣. ការកែប្រែ User Defined Function .....	56
៤. ការលុប User Defined Function .....	56
<b>ជំពូកទី៣...C Sharp Database Programming ជាមួយ ADO.NET</b>	<b>57</b>
<b>មេរៀនទី១ ADO.NET Architecture .....</b>	<b>57</b>
១. ADO.NET Provider.....	57
២. ADO.NET Data Set .....	60
<b>មេរៀនទី២ សំណង់កម្មវិធីដោយប្រើប្រាស់ Data Set.....</b>	<b>63</b>
១. ការសរសេរកូដបង្កើតកម្មវិធីជាមួយ Data Set.....	63
២. ការប្រើប្រាស់ Data Source Wizard បង្កើតកម្មវិធីជាមួយ Data Set.....	68
៣. ការកំណត់ Master and Detail Form.....	77

ប្រជុំបញ្ចប់វគ្គ គ្រប់គ្រងប្រាក់ខែបុគ្គលិក .....	78
១. ការរចនា Database .....	78
២. ការរចនា User Interface ដោយប្រើ Windows Form.....	79
៣. ការសរសេរកូដអន្តរកម្ម .....	80





## ជំពូកទី១

## Object Oriented C Sharp Programming

## មេរៀនទី១

## ស្វែងយល់ពី Class និង Object

## ១. និយមន័យ

Class គឺជាសំណង់កូដដែលអាចផ្ទុក ឬរក្សា Data (store in field) និង ការប្រព្រឹត្តរបស់កូដកម្ម វិធី (Define by Method)។

Class គឺជាប្រភេទទិន្នន័យកំណត់មួយដែលចងក្រងឡើងដោយអ្នកសរសេរកូដ។

ឧទាហរណ៍៖

```
public class Person
{
    // Field
    public string Name;

    // Method
    public void SetName()
    {
        // ...
    }
}
```

Object គឺជាលក្ខណសម្បត្តិទាំងឡាយណា(មានទាំង Field និង Method) ដែលកើតពីលំនាំគំរូ របស់ Class។ Object ត្រូវបានគេបង្កើតដោយប្រើប្រាស់ Keyword **new** ។

ឧទាហរណ៍៖

```
Person person = new Person();
or
var person = new Person();
```

## ២. ការប្រើប្រាស់ Field នៃ Class

Field គឺជា Data ឬបណ្តុំ Data ទាំងឡាយណាដែលពាក់ព័ន្ធនឹងគម្រូនៃ Class ណាមួយ

ឧទាហរណ៍៖ បង្កើត Class Item ដែលមាន

- Fields
  - name : string
  - description : string
  - quantity : int
  - price : double

```
public class Item
{
    public string name;
    public string description;
    public int quantity;
    public double price;
}
```

- បង្កើត Object នៃ Class Item នៅក្នុង Block Main() { ... }

```
static void Main(string[] args)
{
    Item phone = new Item();
    phone.name = "iPhone 13 Pro Max";
    phone.description = "Just Black, 256GB 6GB RAM";
    phone.quantity = 2;
    phone.price = 1359;

    Console.WriteLine(" == Item Information ==");
    Console.WriteLine("  Name: {0}", phone.name);
    Console.WriteLine("  Description: {0}", phone.description);
    Console.WriteLine("  Quantity: {0}", phone.quantity);
    Console.WriteLine("  Price: {0}", phone.price);
}
```

### លំហាត់ទី១.១៖ ចូរអនុវត្តន៍បង្កើត Class Phone ដែលមាន Fields

- brand : string
- model : string
- color : string
- ram : int
- rom : int

## ៣. Method នៃ Class

Method គឺជាប្លុកនៃកូដដែលមានតួនាទី មុខងារច្បាស់លាស់មួយដែលដំណើរការនៅពេលត្រូវបានគេហៅប្រើ។

### ៣.១ ការប្រើប្រាស់ Method

ឧទាហរណ៍ទី១៖ ថែម Method GetTotalPrice() ទៅលើ Class Item ដែលបានបង្កើតពីខាងលើ

```
public class Item
{
    public string name;
    public string description;
    public int quantity;
    public double price;

    public double GetTotalPrice()
    {
        return quantity * price;
    }
}
```

### ការហៅប្រើប្រាស់៖

```
static void Main(string[] args)
{
    Item watch = new Item();
    watch.name = "Samsung Galaxy Watch4";
    watch.description = "Mid Night Blue, 32GB 1.5GB RAM";
    watch.quantity = 3;
    watch.price = 199;
```



```

    Console.WriteLine(" Watch Total Price: {0:C}", watch.GetTotalPrice());
}

```

ឧទាហរណ៍ទី២៖ បន្ថែម Method ២ទៀតទៅឲ្យ Class item

- SetPrice(double amount)
- AddQuantity(int amount)

```

public class Item
{
    // Fields
    public string name;
    public string description;
    public int quantity;
    public double price;

    // Methods
    public double GetTotalPrice()
    {
        return quantity * price;
    }

    public void SetPrice(double amount)
    {
        this.price = amount;
    }

    public void AddQuantity(int amount)
    {
        this.quantity += amount;
    }
}

```

ការហៅប្រើប្រាស់៖

```

static void Main(string[] args)
{
    Item item = new Item();
    item.name = "Atomic Habits";
    item.description = "Self Development Book";
    item.quantity = 1;
    item.price = 15.9;

    Console.WriteLine(" Item Details");
    Console.WriteLine(" Name: {0}", item.name);
    Console.WriteLine(" Description: {0}", item.description);
    Console.WriteLine(" Quantity: {0}", item.quantity);
    Console.WriteLine(" Price: {0}", item.price);
    Console.WriteLine("-----\n");

    item.SetPrice(14.9);
    item.AddQuantity(3);
    Console.WriteLine(" Quantity: {0}", item.quantity);
    Console.WriteLine(" Price: {0}", item.price);
    Console.WriteLine("-----\n");
}

```

លំហាត់ទី១.២៖ ចូរអនុវត្តសរសេរកូដបន្ថែម Method ទៅឲ្យ Class Phone ខាងបានបង្កើតនៅក្នុងលំហាត់ទី១.១។ Method ដែលត្រូវថែម៖

- SetModel(string model) ផ្តល់តម្លៃទៅឲ្យ model នៃ Phone
- SetColor(string color) ផ្តល់តម្លៃទៅឲ្យ color នៃ Phone
- GetPhoneInfo() ចាប់តម្លៃដែលជាព័ត៌មានរបស់Phone ដោយផ្តល់តម្លៃត្រឡប់មកវិញជាString តម្លៃតម្រូវ៖ “Brand: brand, Model: model, Color: color, Price: price”

### ៣.២ Method Overloading

សញ្ញាណរបស់ Method មានដូចជា

- ចំនួន Parameters របស់វា
- ប្រភេទទិន្នន័យនៃ Parameters របស់វា
- និងការលំដាប់លំដោយនៃ Parameters របស់វា

នៅក្នុងClass មួយអាចមាន Method ដែលមានឈ្មោះដូចគ្នាច្រើន(Overloading) តែសញ្ញាណរបស់វាមិនអាចដូចគ្នាទេ។

ឧទាហរណ៍៖ បង្កើត Class Employee ដែលមាន

- Fields
  - name: string
  - phone: string
  - address: string
  - department: string
  - position: string
  - baseSalary: double
  - bonus: double
- Method
  - GetSalary() Return Base Salary + Bonus
  - GiveBonus() Add Default value(10) more to bonus
  - GiveBonus(bonus) Add more value to bonus via bonus parameter
  - GiveBonus(time) Multiply bonus by time parameter

```
public class Employee
{
    // Fields
    public string name;
    public string phone;
    public string address;
    public string department;
    public string position;
    public double baseSalary;
    public double bonus;

    //Methods
    public double GetSalary()
```





```

    {
        return this.baseSalary + this.bonus;
    }

    public void GiveBonus()
    {
        this.bonus += 10;
    }
    public void GiveBonus(double bonus)
    {
        this.bonus += bonus;
    }

    public void GiveBonus(int times)
    {
        if (this.bonus > 0)
            this.bonus = this.bonus * times;
        else
            GiveBonus();
    }
}

```

ការហៅប្រើប្រាស់៖

```

class Program
{
    static void Main(string[] args)
    {
        // declare Employees Array
        Employee[] employees = new Employee[5];
        employees[0] = new Employee();
        employees[0].name = "Bona";
        employees[0].phone = "0973743297";
        employees[0].address = "Battambang, Banan";
        employees[0].department = "Web Development";
        employees[0].position = "Instructor";
        employees[0].baseSalary = 200;
        employees[0].GiveBonus();

        employees[1] = new Employee();
        employees[1].name = "Chamreoun";
        employees[1].phone = "0963874914";
        employees[1].address = "Pursat Province";
        employees[1].department = "Web Development";
        employees[1].position = "Web Developer";
        employees[1].baseSalary = 200;
        employees[1].GiveBonus(15.0);

        // Give more bonus to employee index: 0
        employees[0].GiveBonus(2);
        int count = 0;
        foreach(var employee in employees)
        {
            if(employee != null)
            {
                Console.WriteLine("\n Employee: {0}", ++count);
                Console.WriteLine("-----");
                Console.WriteLine("      Name: {0}", employee.name);
                Console.WriteLine("      Phone: {0}", employee.phone);
                Console.WriteLine("      Base Salary: {0}", employee.baseSalary);
                Console.WriteLine("      Bonus: {0}", employee.bonus);
                Console.WriteLine("      Salary: {0}", employee.GetSalary());
            }
        }
    }
}

```

```

        Console.WriteLine("
        Console.WriteLine("
        Console.WriteLine("
    }
    }
    Console.ReadLine();
}

```

```

    Department: {0}", employee.department);
    Position: {0}", employee.position);
    Address: {0}", employee.address);

```

### លំហាត់ទី១.៣៖ ចូរអនុវត្តសរសេរកូដបង្កើត Class Rectangle ដែលមាន

Fields: width និង height ជា double

Method:

- Initialize() មាននាទីកំណត់តម្លៃឲ្យ width = 10 និង height = 5
- Initialize(width, height) មាននាទីកំណត់តម្លៃឲ្យ width និង height តាម Parameters
- GetWidth() ចាប់តម្លៃ width ត្រឡប់មកវិញ(double)
- GetHeigth() ចាប់តម្លៃ height ត្រឡប់មកវិញ(double)
- GetArea() ចាប់តម្លៃក្រឡាផ្ទៃចតុកោណកែងត្រឡប់មកវិញ(double)
- GetPerimeter() ចាប់តម្លៃបរិមាត្រចតុកោណកែងត្រឡប់មកវិញ(double)

## ៤. Constructor

Constructor គឺជា Method ពិសេសមួយដែលត្រូវបានហៅនិងដំណើរការដោយស្វ័យប្រវត្តិ ខណៈការបង្កើត Object នៃ Class ណាមួយបានប្រព្រឹត្តឡើង។

Constructor ត្រូវតែមានឈ្មោះដូចគ្នាទៅនឹងឈ្មោះ Class ដែលវាស្ថិតនៅ។ Constructor មិនមាន Return Type ទេ សូម្បីតែ void ក៏មិនបានដែរ។

```

public class Customer
{
    // ...
    public Customer()
    {
        // ...
    }
}

```

ជាទូទៅ Constructor ត្រូវបានគេប្រើដើម្បី ផ្តល់តម្លៃដំបូងទៅ Object។ Constructor អាចគ្មាន ឬ មាន Parameters។

កាលណាគេមិនបានកំណត់បង្កើត Constructorនៅក្នុងClass ណាមួយ នោះ Default Constructor ដែលជា Parameter less Constructor ត្រូវបង្កើតឡើង។

ឧទាហរណ៍ទី១៖ បង្កើត Class Customer ដែលមាន៖

```
public class Customer
{
    public string name;
    public string phone;
    public string address;

    public Customer()
    {
        name = "General";
        phone = "N/A";
        address = "Battambang";
    }

    public void CustomerInfo()
    {
        Console.WriteLine(" --- Customer Information ---");
        Console.WriteLine("    Name: {0}", this.name);
        Console.WriteLine("    Phone: {0}", this.phone);
        Console.WriteLine("    Address: {0}", this.address);
    }
}

class Program
{
    static void Main(string[] args)
    {
        Customer customer = new Customer();
        customer.CustomerInfo();
    }
}
```

Constructor នៅក្នុង Class មួយអាចមានច្រើនជាងមួយ (Overloading) តែមិនអាចមានសញ្ញាណដូចគ្នាបានឡើយ។ សញ្ញាណនេះ សំដៅលើ ចំនួន ប្រភេទទិន្នន័យ និងលំដាប់លំដោយនៃ Parameters របស់ Constructor។

នៅក្នុង Class គេអាចហៅ Constructor មួយទៅ Constructor មួយទៀតដោយ ប្រើប្រាស់ `this()` keyword។

ឧទាហរណ៍២៖ ថែម Parameterize Constructor ទៅឲ្យ Customer Class

```
public class Customer
{
    public string name;
    public string phone;
    public string address;

    public Customer()
    {
        name = "General";
        phone = "N/A";
        address = "Battambang";
    }

    public Customer(string name)
    {
        this.name = name;
    }
}
```

```

public Customer(string name, string phone)
    :this(name)
{
    this.phone = phone;
}

public Customer(string name, string phone, string address)
    :this(name, phone)
{
    this.address = address;
}

public void CustomerInfo()
{
    Console.WriteLine(" --- Customer Information ---");
    Console.WriteLine("    Name: {0}", this.name);
    Console.WriteLine("    Phone: {0}", this.phone);
    Console.WriteLine("    Address: {0}", this.address);
}
}

class Program
{
    static void Main(string[] args)
    {
        List<Customer> customers = new List<Customer>();
        var bona = new Customer("Bona");
        bona.phone = "09742934";
        bona.address = "Battambang";
        customers.Add(bona);

        var sengly = new Customer("Sengly", "096374323", "Phnom Penh");
        customers.Add(sengly);

        var samnang = new Customer("Samnang", "012497958");
        samnang.address = "B. Meanchey";
        customers.Add(samnang);

        foreach(var customer in customers)
        {
            customer.CustomerInfo();
            Console.WriteLine();
        }
    }
}

```

### លំហាត់ទី១.៤៖ ចូរបង្កើត Class User មួយដែលមាន

Fields:

- Id : int
- FirstName: string
- LastName: string
- UserName: string
- Password: string

Constructor: បង្កើត Overloading Constructors ចំនួនបី

- User()
- User(Id, UserName, Password)
- User(Id, FirstName, LastName, UserName, Password)

Method: UserInfo()



## ៥. Encapsulation

Encapsulation គឺជាការចងក្រងកូដជាកញ្ចប់ឬជាក្រុម ដែលមាន Data និង Methods ពាក់ព័ន្ធ នឹង Object ណាមួយជាក់លាក់។ ជាទូទៅគេតែងតែលាក់ Data នៃ Class ដើម្បីការពារមិនឲ្យ Access ដោយផ្ទាល់ ដែលគេហៅវាថា Hidden Data។ គេអាច Access Data ទាំងអស់នោះបានដោយ ប្រយោលតាមរយៈ Methods ឬក៏ Properties។

Access Modifier គឺជាអ្នកកំណត់ដែន ឬសិទ្ធិនៃការចូលប្រើប្រាស់ Members របស់ Class ណាមួយ។ Access Modifier មានដូចជា៖

**public:** Members Class ឬ Class អាចចូលប្រើប្រាស់ ពីគ្រប់ទីកន្លែង

**private:** Members Class អាចប្រើប្រាស់បានតែក្នុង Class ណាមួយជាក់លាក់ប៉ុណ្ណោះ

ឧទាហរណ៍៖

```
public class Customer
{
    // Fields
    private string _name;
    private string _phone;
    private string _address;
    private int _point;

    public Customer(){}

    public Customer(string name, string phone)
    {
        _name = name;
        _phone = phone;
    }
    public Customer(string name, string phone, string address)
        :this(name, phone)
    {
        _address = address;
    }
    public void SetName(string name)
    {
        _name = name;
    }
    public string GetName()
    {
        return _name;
    }

    public void SetPhone(string phone)
    {
        _phone = phone;
    }

    public string GetPhone()
    {
        return _phone;
    }

    public void SetAddress(string address)
    {
        _address = address;
    }
}
```

```

public string GetAddress()
{
    return _address;
}

public void SetPoint(int point)
{
    _point = point;
}
public int GetPoint()
{
    return _point;
}
}
class Program
{
    static void Main(string[] args)
    {
        Customer customer = new Customer("Samnang", "012349234");
        customer.SetAddress("Banteay Meanchey");
        customer.SetPoint(100);

        Console.WriteLine(" Customer Details");
        Console.WriteLine("    Name: {0}", customer.GetName());
        Console.WriteLine("    Phone: {0}", customer.GetPhone());
        Console.WriteLine("    Address: {0}", customer.GetAddress());
        Console.WriteLine("    Point: {0}", customer.GetPoint());
        Console.WriteLine("    -----");

        customer.SetPoint(250);
        customer.SetPhone("0234297489");
        customer.SetName("Samnang Rongreoung");

        Console.WriteLine("\n -----");
        Console.WriteLine(" After Modified\n");
        Console.WriteLine("    Name: {0}", customer.GetName());
        Console.WriteLine("    Phone: {0}", customer.GetPhone());
        Console.WriteLine("    Address: {0}", customer.GetAddress());
        Console.WriteLine("    Point: {0}", customer.GetPoint());
        Console.WriteLine("    -----");
    }
}

```

### លំហាត់ទី១.៥៖ ចូរបង្កើត Class Person មួយដែលមាន

Fields:

- \_firstName: string (private)
- \_lastName: string (private)
- \_job: string (private)

Constructor: បង្កើត Overloading Constructors

Method:

- SetFirstName, GetFristName
- SetLastName, GetLastName
- GetFullName
- SetJob, GetJob
- PersonInfo()

## ៦. Property នៃ Class

Property គឺជាធាតុបូជាសមាជិករបស់ Class ដែលអនុញ្ញាតឱ្យគេ Access ទៅកាន់ Field របស់ Class បាន។

ជាទូទៅគេតែងតែប្រើប្រាស់ Property ជំនួសឱ្យ Get and Set Method នៅក្នុង C Sharp Programming។

Keyword:

- **get** ប្រើដើម្បីចាប់យកតម្លៃរបស់ Field
- **set** ប្រើដើម្បីផ្តល់តម្លៃទៅឱ្យ Field

Property មានបីប្រភេទគឺ៖

- Read and Write: Property ដែលមានទាំង get និង set

e.g.

```
private string _name;
public string Name
{
    get { return _name; }
    set { _name = value; }
}
```

- Read: Property ដែលមានតែ get

e.g.

```
private string _name;
public string Name
{
    get { return _name; }
}
```

- Write: Property ដែលមានតែ set

e.g.

```
private string _name;
public string Name
{
    set { _name = value; }
}
```

គេអាចបង្កើត Property ដោយមិនចាំបាច់មាន Field ក៏បានដែរ ដោយប្រើប្រាស់ Auto-implement Property ដោយ Program នឹងធ្វើការបង្កើត Field ពីខាងក្រោយដោយខ្លួនវា។ ដើម្បីបង្កើត គេកំណត់សរសេរ៖ `public string Name { get; set; }`

ជាទូទៅ Field តែងតែត្រូវបានគេ ប្រកាសជា private ហើយ Property ជា public.។

ឧទាហរណ៍៖

```
public class Student
{
    private string _id;
```

```

private string _firstName;
private string _lastName;
private string _gender;

private string Id { get { return _id; } }
public string FirstName
{
    get { return _firstName; }
    set { _firstName = value; }
}
public string LastName
{
    get { return _lastName; }
    set { _lastName = value; }
}

public string FullName
{
    get { return $"{_firstName} {_lastName}"; }
}
public string Gender
{
    get { return _gender; }
    set { _gender = value; }
}
public int Age { get; set; }

public Student(string id)
{
    _id = id;
}
public Student(string id, string firstName, string lastName)
{
    _id = id;
    _firstName = firstName;
    _lastName = lastName;
}
public string StudentInfo()
{
    string age = Age > 0 ? Age.ToString() : "N/A";

    string info = "";
    info += $"    Id: {_id}\n" +
        $"    Name: {FullName}\n" +
        $"    Gender: {_gender}\n" +
        $"    Age: {age}\n" +
        $"    ----- \n";
    return info;
}
}

class Program
{
    static void Main(string[] args)
    {
        List<Student> students = new List<Student>();

        Student student = new Student("0001", "Kiry", "Soun");
        student.Gender = "Male";
        student.Age = 18;
        students.Add(student);
    }
}

```



```

student = new Student("0002", "Savong", "Vy");
student.Gender = "Female";
students.Add(student);

student = new Student("0003");
student.FirstName = "Yura";
student.LastName = "Thum";
student.Gender = "Female";

students.Add(student);

Console.WriteLine(" --All Students Details ---");
foreach(var stu in students)
{
    Console.WriteLine(stu.StudentInfo());
}
}
}

```

### លំហាត់ទី១.៦៖ បង្កើត Class ឈ្មោះ Item ដែលមាន

1. Fields ជា private
  - \_id: int
  - \_name: string
  - \_quantity: int
  - \_cost: double
  - \_price: double
  - \_description: string
2. បង្កើត Property ដោយយោងទៅតាម Field នីមួយៗ
3. បង្កើត Constructor យ៉ាងតិចបំផុត៣
4. បង្កើត Method ដើម្បីចាប់តម្លៃត្រឡប់ពីតំបន់ ItemInfo() មកវិញជា String

### ៧. Object Initializer

Object Initializer គឺជាការផ្តល់តម្លៃទៅឲ្យ Members របស់ Object នៅកំឡុងពេលដែលគេបង្កើត Object នៃ Class ណាមួយ។

```

class Program
{
    static void Main(string[] args)
    {
        // Instanciate of Student Class
        var student = new Student("0007")
        {
            FirstName = "ReakSa",
            LastName = "Kheng",
            Age = 25,
            Gender = "Female"
        };
        //student Information via Property
        Console.WriteLine("Name: {0}", student.Name);
        Console.WriteLine("Gender: {0}", student.Gender);
        Console.WriteLine("Age: {0}", student.Age);
        Console.WriteLine();
    }
}

```

```
// Student Information via StudentInfo() Method
Console.WriteLine(student.StudentInfo());
    }
}
```

**លំហាត់ទី១.៦៖** ប្រើ Object Initializer ដើម្បីផ្តល់តម្លៃឱ្យ Properties ក្នុងការបង្កើត Object of Item Class ខាងលើ

កំណត់សម្គាល់ខ្លី៖



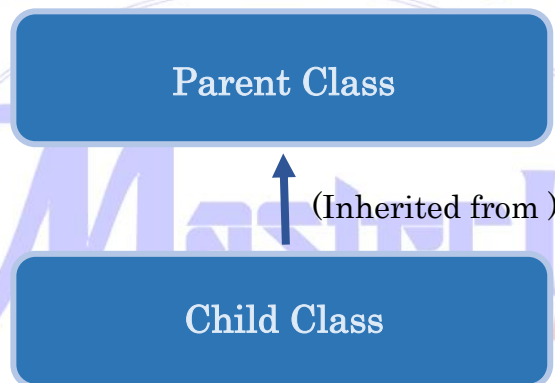
## មេរៀនទី២

## ទំនាក់ទំនងរវាង Classes (Associated Classes)

## ១. ការបន្តសមាសភាព (Inheritance)

## ១.១ និយមន័យ

Inheritance គឺជាគោលការណ៍សំខាន់ទី២នៅក្នុងគម្រោង OOP ។ ទម្រង់នេះ អនុញ្ញាតឱ្យគេប្រើប្រាស់ Code ដែលបានសរសេររួចហើយម្តងទៀត។ Inheritance គឺទំនាក់ទំនងរវាង Class ពីរដែលអនុញ្ញាតឱ្យ Class មួយអាចធ្វើការ Inherit Code ពី Class មួយទៀត។ Class ដែល Inherit ពីគេហៅថា Child Class ឬ Derived Class រីឯ Class ដែលត្រូវបានគេ Inherit ហៅថា Parent Class ឬ Base Class។



ដើម្បី Inherit គេប្រើសញ្ញា : (Colon ចុចពីរ) បន្ទាប់មក Parent Class

Access Modifier:

- **public** : Members Class ឬ Class អាចចូលប្រើប្រាស់ ពីគ្រប់ទីកន្លែង
- **private** : Members Class អាចប្រើប្រាស់បានតែក្នុង Class ណាមួយជាក់លាក់ប៉ុណ្ណោះ
- **protected** : Members Class អាចប្រើប្រាស់បានតែក្នុង Class និង Derived Class

## ១.២ ការប្រើប្រាស់ Inheritance

ឧទាហរណ៍ទី១៖ បង្កើត Class *Employee* ជា Child ដែលបាន Inherit ពី Class *Person* ជា Parent

Base Class or Parent Class (Person)

```

public class Person
{
    private string _firstName;
    private string _lastName;
    private string _gender;
    private string _phone;
    private string _address;

    public string FristName { get => _firstName; set => _firstName = value; }
    public string LastName { get => _lastName; set => _lastName = value; }
    public string FullName { get => _firstName + " " + _lastName; }
    public string Gender { get => _gender; set => _gender = value; }
    public string Phone { get => _phone; set => _phone = value; }
    public string Address { get => _address; set => _address = value; }
  
```

```

public string PersonInfo()
{
    return $"    Name: {FullName}\n" +
           $"    Gender: {_gender}\n" +
           $"    Phone: {_phone}\n" +
           $"    Address: {_address}";
}

```

### Derived Class or Child Class (Employee)

```

public class Employee : Person
{
    private string _department;
    private string _position;
    private double _salary;

    public string Department { get => _department; set => _department = value; }
    public string Position { get => _position; set => _position = value; }
    public double Salary { get => _salary; set => _salary = value; }

    public string EmployeeInfo()
    {
        return $"    {this.PersonInfo()}\n" +
               $"    Department: {this._department}\n" +
               $"    Position: {this._position}\n" +
               $"    Salary: {this._salary}";
    }
}

```

### Program Class មានផ្ទុក Main() Method

```

class Program
{
    static void Main(string[] args)
    {
        Employee employee = new Employee()
        {
            FristName = "Bona",
            LastName = "Kheng",
            Phone = "097342423",
            Gender = "Male",
            Address = "Battambang",
            Department = "Web Development",
            Position = "Web Instructor",
            Salary = 210
        };

        // As Person Information
        Console.WriteLine(" Employee As Person Information");
        Console.WriteLine (employee.PersonInfo());
        Console.WriteLine("-----\n");

        Console.WriteLine(" Employee As Employee Information");
        Console.WriteLine (employee.EmployeeInfo());

        Console.ReadLine();
    }
}

```



ឧទាហរណ៍ទី២៖ កំណត់ Child Class *Student* Inherited ពី Base Class *Person*

```
public class Student : Person
{
    private int _generation;
    private string _faculty;
    private string _subject;
    private string _room;

    public int Generation { get => _generation; set => _generation = value; }
    public string Faculty { get => _faculty; set => _faculty = value; }
    public string Subject { get => _subject; set => _subject = value; }
    public string Room { get => _room; set => _room = value; }

    public string StudentInfo()
    {
        return $"{this.PersonInfo()}\n" +
            $"    Faculty: {_faculty}\n" +
            $"    Subject: {_subject}\n" +
            $"    Generation: {_generation}\n" +
            $"    Class Room: {_room}";
    }

    public Student Copy()
    {
        Student student = new Student
        {
            FristName = this.FristName,
            LastName = this.LastName,
            Gender = this.Gender,
            Phone = this.Phone,
            Address = this.Address,
            Faculty = this.Faculty,
            Subject = this.Subject,
            Generation = this.Generation,
            Room = this.Room
        };

        return student;
    }
}
```

ការហៅប្រើប្រាស់ក្នុង Main() Method

```
static void Main(string[] args)
{
    var student = new Student
    {
        FristName = "Norngrey",
        LastName = "Ny",
        Phone = "095839234",
        Gender = "Female",
        Address = "K. Thom",
        Faculty = "Information Science",
        Subject = "Information Technology",
        Generation = 15,
        Room = "115"
    };

    var studentCopy = student.Copy();

    Console.WriteLine("    -> Student Raw Information");
}
```

```

Console.WriteLine(studentCopy.StudentInfo());
Console.WriteLine(" ----- \n");

Console.WriteLine(" -> Student Copy Information");
Console.WriteLine(studentCopy.StudentInfo());
}

```

**លំហាត់ទី២.១៖** ចូរអនុវត្តន៍កំណត់ Child Class នៃ Person ដែលមានឈ្មោះថា Teacher

1. Fields ជា private

- `_subjects: List<string>`

2. បង្កើត Property ដោយយោងទៅតាម Field

3. បង្កើត Method

- `TeacherInfo(): string` ផ្តល់តម្លៃត្រឡប់ជាព័ត៌មានគ្រូ
- `AddSubject(subject): void` បន្ថែមមុខវិជ្ជាគ្រូបង្រៀន
- `Copy(): Teacher` ចម្លង និងត្រឡប់តម្លៃនៃ Teacher Object

### ១.៣ Inheritance Constructor

Inheritance Constructor គឺជា Base Constructor ដែលត្រូវហៅមកប្រើដោយស្វ័យប្រវត្តិ

ឧទាហរណ៍: Child Class Object ត្រូវបានបង្កើត។ ជានិច្ចកាល Base Constructor តែងតែដំណើរមុន Child Constructor។

ឧទាហរណ៍៖ ការប្រើប្រាស់ Teacher ដែល Inherited ពី Person Class ដើម្បីឲ្យដឹងពីដំណើរការ Constructor

```

public class Teacher : Person
{
    private List<string> _subjects = new List<string>();

    public List<string> Subjects
    {
        get
        {
            return _subjects;
        }
    }

    public Teacher()
    {
        //Child Constructor always run after the Base one
        Console.WriteLine("Teacher Constructor");
    }

    public Teacher(string firstName, string lastName)
        :base(firstName, lastName)
    {
    }

    public Teacher(string firstName, string lastName, string gender, string phone)
        :base(firstName, lastName, gender)
    {
    }
}

```



```

{
    Phone = phone;
}
public void AddSubject(string subject) => _subjects.Add(subject);

public Teacher Copy()
{
    Teacher teacher = new Teacher
    {
        FristName = this.FristName,
        LastName = this.LastName,
        Gender = this.Gender,
        Phone = this.Phone,
        Address = this.Address,
        _subjects = this.Subjects
    };

    return teacher;
}

public string TeacherInfo()
{
    string str = $"{this.PersonInfo()}\n";
    if (_subjects.Count == 0)
        str += "    Subjects: N/A";
    else
        str += "    Subjects:\n";

    foreach (var sub in _subjects)
    {
        str += $"        - {sub}\n";
    }

    return str;
}
}

```

ការហៅប្រើប្រាស់ក្នុង Main() Method:

```

static void Main(string[] args)
{
    List<Teacher> teachers = new List<Teacher>();
    // at run time you will see the message from constructors
    Console.WriteLine(" -> This are the messages form Base and Child Constructors");
    Teacher teacher = new Teacher();
    teacher.FristName = "Lyneang";
    teacher.LastName = "Chan";
    teacher.Gender = "Female";
    teacher.Phone = "093842353";
    teacher.Address = "Oudormeanchery";
    teacher.AddSubject("C# Programming Instrcutor");
    teacher.AddSubject("Javascript Programming Instrcutor");
    teachers.Add(teacher);

    teacher = new Teacher("Sangha", "Sok", "Male", "0973423532")
    {
        Address = "Phnom Penh"
    };

    teacher.Subjects.Add("UX UI Design Instructor");
    teacher.Subjects.Add("Front End Developer");
    teacher.AddSubject("HTML & CSS Trainer");
}

```

```
teachers.Add(teacher);

foreach(var cher in teachers)
{
    Console.WriteLine( cher.TeacherInfo());
    Console.WriteLine("  -> -----");
}
}
```

## ១.៤ Upcast និង Downcast

### ក. និយមន័យ

Upcast គឺជាប្លែងប្រភេទទិន្នន័យពី Child Class ទៅ Parent Class ដោយមិនចាំបាច់បញ្ជាក់ពីឈ្មោះរបស់ Parent Class។

Downcast គឺជាការប្លែងប្រភេទទិន្នន័យពី Parent Class Object ដែលយោងតម្លៃទៅនឹង Child Class Object ទៅប្រភេទទិន្នន័យនៃ Child Class នោះ ដោយបញ្ជាក់ពីឈ្មោះរបស់ Child Class យ៉ាងច្បាស់លាស់។

### ខ. ការប្រើប្រាស់

ឧទាហរណ៍ទី១៖ ការប្រើប្រាស់ Upcasting នៅក្នុង Class Program

```
class Program
{
    static void Main(string[] args)
    {
        Teacher teacher = new Teacher();
        teacher.FristName = "Lyneang";
        teacher.LastName = "Chan";
        teacher.Gender = "Female";
        teacher.Phone = "093842353";
        teacher.Address = "Oudormeanchery";
        teacher.AddSubject("C# Programming Instrcutor");
        teacher.AddSubject("Javascript Programming Instrcutor");

        // Upcasting
        // Implicitly Cast teacher obj to person obj
        Person person = teacher;
        DisplayPerson(teacher);

    }
    static void DisplayPerson(Person person)
    {
        Console.WriteLine("  -> ... Person Information ...");
        Console.WriteLine(person.PersonInfo());
        Console.WriteLine("      .....");
    }
}
```

ឧទាហរណ៍ទី២៖ ការប្រើប្រាស់ Down casting នៅក្នុង Class Program

```
class Program
{
    static void Main(string[] args)
    {
        Person person = teacher;
```



```

        //DisplayPerson(teacher);

        // Down casting
        // Explicitly Cast person obj that is referenced to teacher obj into teacher o
bj
        Person bora = new Teacher("Bora", "Keo", "Male", "012343579");
        bora.Address = "Battambang";

        Teacher teacher_bora = (Teacher)bora;
        teacher_bora.Subjects.Add("Law");
        teacher_bora.Subjects.Add("Bank and Finance");

        DisplayTeacher((Teacher)bora);
    }

    static void DisplayTeacher(Teacher teacher)
    {
        Console.WriteLine("  -> ... Teacher Information ...");
        Console.WriteLine(teacher.TeacherInfo());
        Console.WriteLine("      .....");
    }
}

```

#### គ. as និង is Keyword

ការ Cast ពី Parent Object ដែលយោងតម្លៃ Child Object ណាមួយទៅ Child Class Type អាចនឹងប្រព្រឹត្តកំហុស ករណីការបញ្ជាក់ការបម្លែងមិនត្រឹមត្រូវ ។

```

static void Main(string[] args)
{
    Person bora = new Teacher("Bora", "Keo", "Male", "012343579");
    bora.Address = "Battambang";

    // Cast Exception Handling happen
    // person obj "bora" is reference to Teacher, but cast it into Student
    Student student = (Student)bora;
}

```

ដើម្បីការពារកំហុសគេអាចប្រើប្រាស់ **as** operator ដើម្បីបម្លែងវិញ

- ករណីតម្លៃយោងទៅ Child Class Object ដូចនឹង Child Class Type មានន័យថាការបម្លែងបានជោគជ័យ
- ករណីតម្លៃយោងទៅ Child Class Object មិនដូចនឹង Child Class Type តម្លៃដែលទទួលបានមកវិញគឺ Null

```

static void Main(string[] args)
{
    Person bora = new Teacher("Bora", "Keo", "Male", "012343579");
    bora.Address = "Battambang";

    // Legal cast
    var teacher = bora as Teacher;

    // Illegal cast => student obj = null
}

```



```

var student = bora as Student;
if(student == null)
{
    Console.WriteLine("Bora obj is a Teacher, Cannot cast as Student");
}
}

```

គេអាចប្រើ *is* keyword ដើម្បីពិនិត្យតម្លៃយោង Child Class Object និង Child Class Type

- ផ្តល់តម្លៃត្រឡប់មក True ករណីដូចគ្នា
- ផ្តល់តម្លៃត្រឡប់មក False ករណីមិនដូចគ្នា

```

static void Main(string[] args)
{
    Person bora = new Teacher("Bora", "Keo", "Male", "012343579");
    bora.Address = "Battambang";

    Student student;
    if(bora is Student)
    {
        student = bora as Student;
        DisplayPerson(student);
    }
    else
    {
        Console.WriteLine("bora obj is not a Student");
    }
}

```

### ១.៥ Boxing និង Unboxing

Boxing: គឺជាការបម្លែងប្រភេទទិន្នន័យដែលជាពណ៌ក Primary Type ឲ្យទៅជា Object Type

Unboxing: គឺជាការបម្លែងប្រភេទទិន្នន័យពី Object Type ឲ្យទៅជា Primary Type វិញ

Object គឺប្រភេទទិន្នន័យដែលធំជាងគេទាំងអស់នៅក្នុង C Sharp Programming ដោយសារវាជា Base Class នៃគ្រប់ប្រភេទ Type ទាំងអស់។

ប្រភេទទិន្នន័យនៅក្នុង C Sharp Programming ចែកចេញជាពីរពួក៖

- Primary Type មានដូចជា int, double, float, struct,... ដែលតម្លៃរបស់វាឯករាជ្យដាច់ចេញពីគ្នា
- Reference Type មានដូចជា string, class, array, ... ដែលតម្លៃរបស់វាយោងគ្នាទៅវិញទៅមក

```

//boxing
int age = 25;
object obj = age;

//unboxing
object data = 22.55;
double price = (double)data;

```

## ២. ការដាក់បញ្ចូល (Composition)

Composition គឺជាទំនាក់ទំនងរវាង Class មួយបែបដែលអនុញ្ញាតឱ្យគេដាក់បញ្ចូល Object នៃ Class Type មួយទុកជា Member ទៅក្នុង Class Type មួយផ្សេងទៀត។

ឧទាហរណ៍៖ កំណត់ Class Address រួចដាក់បញ្ចូលជា Object ទៅក្នុង Class Person ព្រមទាំងកែប្រែកូដនៅក្នុង PersonInfo() ដើម្បីកំណត់បង្ហាញ Address ឱ្យបានជាក់លាក់

```
public class Address
{
    public string No { get; set; }
    public string Village { get; set; }
    public string Commune { get; set; }
    public string DistrictOrCity { get; set; }
    public string ProvinceOrState { get; set; }
}

public class Person
{
    private string _firstName;
    private string _lastName;
    private string _gender;
    private string _phone;
    private Address _address = new Address();

    public string FristName { get => _firstName; set => _firstName = value; }
    public string LastName { get => _lastName; set => _lastName = value; }
    public string FullName { get => _firstName + " " + _lastName; }
    public string Gender { get => _gender; set => _gender = value; }
    public string Phone { get => _phone; set => _phone = value; }
    public Address Address { get => _address; set => _address = value; }

    public Person()
    {
        // Base Constructor Always run before the Child
        Console.WriteLine("Person Constructor");
    }

    public Person(string firstName, string lastName)
    {
        _firstName = firstName;
        _lastName = lastName;
    }

    public Person(string firstName, string lastName, string gender)
        :this(firstName, lastName)
    {
        _gender = gender;
    }

    public string PersonInfo()
    {
        return $"      Name: {FullName}\n" +
               $"      Gender: {_gender}\n" +
               $"      Phone: {_phone}\n" +
               $"      Address:\n" +
               $"          No: {_address.No}\n" +
               $"          Village: {_address.Village}\n" +
               $"          Commune: {_address.Commune}\n" +
               $"          District or City: {_address.DistrictOrCity}\n" +
               $"          Province or State: {_address.ProvinceOrState}";
    }
}
```

```
}  
}
```

## លំហាត់ទី២.២៖ ចូរអនុវត្តន៍កំណត់ Class Department

1. បង្កើត Property (public) ទៅឲ្យ Department
  - Id: string
  - Name: string
  - Description: string
2. ដាក់បញ្ចូល Department Object ជា Property របស់ Class Employee
3. កំណត់កែសម្រួល EmployeeInfo() Method ដោយបន្ថែមព័ត៌មានលំអិតរបស់ Department Object ត្រង់ចំណុចនេះយកតែ Name និង Description ។

កំណត់សម្គាល់ខ្លី៖





## មេរៀនទី៣

## ស្វ័យយល់ពី Polymorphism

Polymorphism គឺជាគោលការណ៍សំខាន់ទី៣នៅក្នុងទម្រង់ OOP និងជាពិសេសវាបំពេញបន្ថែមទៅលើគោលការណ៍ទី២គឺ Inheritance។ ជាទូទៅគេប្រើប្រាស់ Polymorphism ដើម្បីធ្វើឲ្យ Object មាន Type ខុសគ្នាតែយោងទៅលើ Base Class តែមួយ ឲ្យស្ថិតក្នុង Type ដូចគ្នា។

## 9. Method Overriding

**Virtual:** គឺជា Keyword ដែលកំណត់ Base Class Object Members ដូចជា Method, Property ... អាច Override នៅក្នុង Child Class Object

**Override:** គឺជា Keyword ដែលកំណត់ការកែសម្រួល (Overridden) virtual ឬ abstract Method, Property, ... របស់ Base Class Object នៅក្នុង Child Class Object។

ឧទាហរណ៍៖ ថែមក្នុងនៅក្នុង Person Class, Class Student និង Class Teacher ថែម virtual Method DisplayInfo(string separator) នៅក្នុង Person Class

```
public virtual string DisplayInfo(string separator)
{
    return $"{FullName}{separator}" +
           $"{_gender}{separator}" +
           $"{_phone}{separator}" +
           $"{_address.No}{separator}" +
           $"{_address.Village}{separator}" +
           $"{_address.Commune}{separator}" +
           $"{_address.DistrictOrCity}{separator}" +
           $"{_address.ProvinceOrState}";
}
```

ថែម Override Method DisplayInfo(string separator) នៅក្នុង Student Class

```
public override string DisplayInfo(string separator)
{
    return $"{base.DisplayInfo(separator)}{separator}" +
           $"{_faculty}{separator}" +
           $"{_subject}{separator}" +
           $"{_generation}{separator}" +
           $"{_room}";
}
```

ថែម Override Method DisplayInfo(string separator) នៅក្នុង Teacher Class

```
public override string DisplayInfo(string separator)
{
    string str = $"{base.DisplayInfo(separator)}{separator}";
    if (_subjects.Count > 0)
    {
        str += "[ ";
        foreach (var sub in _subjects)
        {
            str += $"{sub}, ";
        }
        str = str.Substring(0, str.Length - 2);
        str += " ]";
    }
    return str;
}
```

}

## ការហៅប្រើប្រាស់នៅក្នុង Method Main()

```

var person = new Student
{
    FristName = "Thiny",
    LastName = "Pheap",
    Gender = "Female",
    Address = new Address()
    {
        No = "#118 str.5",
        Village = "Dounteav",
        Commune = "Peam Ek",
        DistrictOrCity = "Ek Phnom",
        ProvinceOrState = "Battambang"
    },
    Phone = "0963342334",
    Faculty = "Sceince",
    Subject = "IT",
    Generation = 14,
    Room = "115"
};

var teacher1 = new Teacher("Bona", "Pheng", "Male", "097432434");
teacher1.Subjects.Add("Web Front End");
teacher1.Subjects.Add("UX UI Design");
teacher1.Subjects.Add("HTML CSS & JS");

var persons = new List<Person>();
persons.Add(person);
persons.Add(teacher1);

var teacher2 = new Teacher("Sengly", "Cheng", "Male", "0973432532")
{
    Address = new Address { No = "#1 Str.03", Village="Kdol", Commune="Kdol", DistrictOrCity="Battambang", ProvinceOrState = "Battambang"},
};
teacher2.AddSubject("Animation");
teacher2.AddSubject("Graphic Design");

persons.Add(teacher2);

Console.WriteLine("\n ---- Result List of Person Objs");
Console.WriteLine(" -----");
foreach(var obj in persons)
    Console.WriteLine(obj.DisplayInfo(", "));
Console.WriteLine();

```



**លំហាត់ទី៣.១៖ ចូរអនុវត្តន៍កំណត់**

Parent Class: Item ដែលមាន

1. Fields: private
  - `_name`, `_description`, `_quantity`, `_price`
2. បង្កើត Property (public) ដោយយោងតាមFields ខាងលើ
3. កំណត់ virtual method `DisplayInfo(string separator)`

Child Classes of Item :

1. 1<sup>st</sup> Child: Book { `_author:string`, `_publishDate:DateTime` }
2. 2<sup>nd</sup> Child: Phone { `_color:string`, `_capacity:string` }
3. បង្កើត Property (public) ដោយយោងតាមFields នៃ Class នីមួយៗ
4. កំណត់ override method `DisplayInfo(string separator)` ទៅឲ្យ Class នីមួយៗ

**២. Abstract Class**

Abstract Class គឺជាប្រភេទ Class ដែលមិនអាចយកទៅបង្កើតជា Object បាន។ ជាទូទៅ គេប្រើ Abstract Class សម្រាប់ធ្វើជា Base Class ឲ្យ Class ផ្សេងទៀត Inherit ប្រើប្រាស់។

នៅក្នុង Abstract Class ក៏អាចមាន Abstract Methods និង Abstract Properties។ Abstract Method ជា Method ទាំងឡាយណាដែលមាន `abstract` keyword ហើយមិនមានកូដក្នុង Method body។

ឧទាហរណ៍៖ បង្កើត Abstract Class Item និងមាន Abstract Method `GetDisplayInfo()`

```
public abstract class Item
{
    private string _name;
    private string _description;
    private int _quantity;
    private double _price;

    public Item()
    {
    }

    public Item(string name, string description, int quantity, double price)
    {
        Name = name;
        Description = description;
        Quantity = quantity;
        Price = price;
    }

    public string Name { get => _name; set => _name = value; }
    public string Description { get => _description; set => _description = value; }
    public int Quantity { get => _quantity; set => _quantity = value; }
    public double Price { get => _price; set => _price = value; }
```

```
// abstract method
public abstract string GetDisplayInfo();
}
```

បង្កើត Child Classes: Book និង Phone

```
public class Book : Item
{
    private string _author;
    private DateTime _publishDate;

    public string Author { get => _author; set => _author = value; }
    public DateTime PublishDate { get => _publishDate; set => _publishDate = value; }

    public Book()
    {
    }
    public Book(string name, string description, int quantity, double price, string author, DateTime publishDate)
        : base(name, description, quantity, price)
    {
        _author = author;
        _publishDate = publishDate;
    }

    public override string GetDisplayInfo()
    {
        return $"      Name: {this.Name}\n" +
               $"      Description: {Description}\n" +
               $"      Author: {Author}\n" +
               $"      PublishDate: {PublishDate}\n" +
               $"      Quantity: {Quantity}\n" +
               $"      Price: {Price}\n" +
               $"      Total Price: {Quantity * Price}";
    }
}
```

```
public class Phone : Item
{
    private string _color;
    private string _capacity;

    public Phone()
    {
    }

    public Phone(string name, string description, int quantity, double price, string color, string capacity)
        : base(name, description, quantity, price)
    {
        _color = color;
        _capacity = capacity;
    }

    public string Color { get => _color; set => _color = value; }
    public string Capacity { get => _capacity; set => _capacity = value; }

    public override string GetDisplayInfo()
    {
        return $"      Name: {Name}\n" +
               $"      Color: {Color}\n" +

```

```

    $" Capacity: {Capacity}\n" +
    $" Description: {Description}\n" +
    $" Quantity: {Quantity}\n" +
    $" Price: {Price}\n" +
    $" Total Price: {Price * Quantity}";
}
}

```

## លំហាត់ទី៣.២៖ ចូរអនុវត្តន៍កំណត់

1. Class Person ជា abstract Class
2. កំណត់ abstract method GetDisplayInfo() នៅក្នុង abstract class Person
3. កំណត់ override method GetDisplayInfo() នៅក្នុង Child Class: Student និង Teacher

**គម្រោង ៖ បង្កើតកម្មវិធីគ្រប់គ្រងបញ្ជីឈ្មោះដោយប្រើប្រាស់ Windows Form App**

The screenshot shows a Windows Form application titled 'MainForm'. It contains a table titled 'People List' with the following data:

ID	FirstName	LastName	Gender	Phone
1	Ryf	Ron	Male	0967342
2	Bona	Phen	Male	0973432
3	Sengna	Vy	Female	0123444
4	NaNa	Chea	Female	0153344
5	Sengly	Veang	Male	0988854

Below the table, it says 'People Amount: 5'. A modal dialog box titled 'AddOrModifyForm' is open, showing the details for 'Ryf Ron'. The dialog has fields for 'First Name', 'Last Name', 'Gender', and 'Phone', and 'Accept' and 'Cancel' buttons.

ចំពោះកម្មវិធីនេះគឺត្រូវការ Form ចំនួនពីរគឺ MainForm សម្រាប់បង្ហាញទិន្នន័យ និង AddOrModifyForm សម្រាប់បន្ថែមឬកែប្រែទិន្នន័យ។ គេប្រើ DataGridView ដើម្បីធ្វើការបង្ហាញទិន្នន័យ។ DataGridView គឺជា Control មួយដែលអាចឲ្យគេបង្ហាញទិន្នន័យបែបតារាង Table ហើយក៏អាចកែប្រែទិន្នន័យបានដែរ។

Properties របស់ DataGridView សំខាន់ៗ៖

- AlternatingRowsDefaultCellStyle: ដាក់ Style ឲ្យ row ដែលជាជួរលេខគូ
- RowsDefaultCellStyle: ដាក់ Style ឲ្យ Row ទូទៅ

- ColumnHeaderDefaultCellStyle: ដាក់ Style ឲ្យ Column
- EnableHeadersVisualStyle: បិទ ឬបើក Visual Style
- ColumnHeaderHeigthSizeMode: ទម្រង់ Column header height
- AutoSizeColumnsMode: ទម្រង់ Auto Size Column
- AutoSizeRowsMode: ទម្រង់ Auto Size Row

គេប្រើប្រាស់ Binding Source ដើម្បី Bind ទិន្នន័យពីប្រភពទៅ Control ដទៃដូចជា Data Grid View, Combo Box, List Box, ... ដែលត្រូវបានគេហៅថា Binding Source Control។ សមាសភាពសំខាន់ៗរបស់ Binding Source Object គឺ៖

- DataSource : get and set object to binding source object
- Current : get current item of data source
- Position : get or set current position list
- AddNew() : add new row to list
- RemoveCurrent() : remove current item in list
- CancelEdit() : cancel Current Edit Operation
- EndEdit() : save change to data source
- ResetCurrentItem : refresh current value

កំណត់សម្គាល់ខ្លះ៖



## ជំពូកទី២

## Microsoft SQL Server Database Management System

Microsoft SQL Server គឺជាប្រភេទ Relational Database Management System (RDBMS) របស់ក្រុមហ៊ុន Microsoft ដែលរចនាឡើងដើម្បីគ្រប់គ្រងនិងរក្សាទុកទិន្នន័យ។

Database គឺជាបណ្តុំទិន្នន័យ ឬព័ត៌មានដែលត្រូវបានគេរៀបចំចងក្រង ដើម្បីងាយក្នុងការចូលប្រើប្រាស់ ក៏ដូចជារក្សាទុកជាដើម។

Table ក្នុង Database គឺជា Object មួយដែលមានផ្ទុក Rows និង Columns ដែលជាបណ្តុំទិន្នន័យពាក់ព័ន្ធ និងរក្សាទុកទិន្នន័យក្នុងទម្រង់ Table format ។

A Column គឺជាបណ្តុំនៃទិន្នន័យ ឬតម្លៃដែលមានប្រភេទច្បាស់លាស់។ A Row គឺជាកន្សោមតម្លៃមួយធាតុនៅក្នុង Table មួយ ហើយគ្រប់ Row ទាំងអស់របស់ Table មួយគឺមានទ្រង់ទ្រាយដូចគ្នា។

	Code	ItemId	A Column Color	Capacity	InStock	UnitPrice
1	1	1	Silver	128GB 2GB RAM	10	250.00
2	2	1	Space Gray	128GB 2GB RAM	10	255.00
3	3	2	Silver	64GB 3GB RAM	30	280.00
A Row 4	4	2	Space Gray	256GB 3GB RAM	20	310.00
5	5	3	Space Gray	64GB 4GB RAM	30	340.00
6	6	3	Gold	256GB 4GB RAM	20	340.00
7	7	3	Gold	512GB 4GB RAM	20	360.00
8	8	4	Gold	128GB 4GB RAM	20	470.00
9	9	5	Black	64GB 4GB RAM	20	700.00
10	10	5	Green	128GB 4GB RAM	30	730.00

ដើម្បីមាន Database សម្រាប់សិក្សាលើជំពូកនេះ អ្នកត្រូវ Run Script CreateDatabase.sql ជាមុនសិន។

## មេរៀនទី១

## ការ Query ទិន្នន័យពី Table តែមួយ

## ១. ស្វែងយល់ពី Select Statement

Select Statement គឺជា Statement មួយរបស់ SQL ប្រើសម្រាប់ Query ទិន្នន័យចេញពី Table មួយឬច្រើន។

ឧទាហរណ៍ទី១៖

**SELECT \* FROM Items**

	ItemId	ItemName	CategoryId	BrandId
1	1	iPhone 8 Plus	1	1
2	2	iPhone X	1	1
3	3	iPhone XS	1	1

(40 rows)



ឧទាហរណ៍ទី២៖

```
SELECT FirstName, LastName, Phone
FROM Customers
```

	FirstName	LastName	Phone
1	General	Customer	0973927434
2	Si	Dalen	016878934
3	Cham	Chhara	092897943
4	Ka	Kakada	012873947
5	Kha	Sokhin	077432334
6	Veasna	SokLeap	098379427

**លំហាត់ទី១.១៖** ចូរអនុវត្តន៍សរសេរ SQL Select Statement ដើម្បី Query ពី Table Invoices នូវ Fields: InvoiceId, InvoiceNo, InvoiceDate, InvoiceTotal និង PaymentTotal

## ២. ការប្រើប្រាស់ Select Top Statement

Select Top Statement ប្រើដើម្បី Query Data ពី Table ក្នុងដែនមួយនៃការជ្រើសយក Top Records ដំបូងដែលបានកំណត់។

ឧទាហរណ៍៖ Query Top 5 Records

```
SELECT TOP (5) * FROM Items
```

	ItemId	ItemName	CategoryId	BrandId
1	1	iPhone 8 Plus	1	1
2	2	iPhone X	1	1
3	3	iPhone XS	1	1
4	4	iPhone 11 Pro	1	1
5	5	iPhone 12	1	1

**លំហាត់ទី១.២៖** ចូរអនុវត្តន៍សរសេរ SQL Select Statement ដើម្បី Query Top Records 10 ដំបូង ពី Table ItemProperties

## ៣. ការកំណត់លក្ខខណ្ឌលើ Select Statement (Where Clause)

Where Clause នៅក្នុង Select Statement ត្រូវបានគេប្រើដើម្បីកំណត់ជ្រើសរើស Records ដែលសមស្របទៅនឹងលក្ខខណ្ឌ។

### ៣.១ Comparison Operator

Comparison Operator គឺជាលក្ខខណ្ឌបែបប្រមាណវិធីប្រៀបធៀប

សញ្ញា	ការពិពណ៌នា
=	ស្មើនឹង

>	ធំជាង
<	តូចជាង
>=	ធំជាងឬស្មើ
<=	តូចជាងឬស្មើ
!=, <>	ខុសពី ឬផ្ទុយពី

ឧទាហរណ៍ទី១៖ Query Table Items ក្នុងលក្ខខណ្ឌ BrandId = 1

```
SELECT * FROM Items WHERE BrandId = 1
```

	ItemId	ItemName	CategoryId	BrandId
1	1	iPhone 8 Plus	1	1
2	2	iPhone X	1	1
3	3	iPhone XS	1	1

(13 rows)

ឧទាហរណ៍ទី២៖ Query Table Invoices ក្នុងលក្ខខណ្ឌ InvoiceTotal > 1000

```
SELECT *
FROM Invoices
WHERE InvoiceTotal > 1000
```

	InvoiceId	CustomerId	InvoiceNo	InvoiceDate	InvoiceTotal	PaymentTotal	InvoiceDueDate	PaymentDate	SaleId
1	6	2	1	2021-09-30 00:00:00.000	1970.00	0.00	2021-10-07 00:00:00.000	NULL	1
2	9	3	4	2021-09-30 00:00:00.000	1410.00	NULL	2021-10-07 00:00:00.000	NULL	2
3	13	1	3	2021-10-25 00:00:00.000	2000.00	2000.00	2021-10-25 00:00:00.000	2021-10-25 00:00:00.000	1

(10 rows)

## ៣.២ Logical Operator

គេប្រើប្រាស់ Logical Operator ដើម្បីភ្ជាប់លក្ខខណ្ឌច្រើនជាងមួយជាមួយគ្នាដោយឈ្មោះ។

តំណាង	ពិពណ៌នា
AND	ឈ្នាប់និង
OR	ឈ្នាប់ឬ
NOT	ឈ្នាប់មិន

ឧទាហរណ៍ទី៣៖ Query Table Items ក្នុងលក្ខខណ្ឌ BrandId ជាApple និង Category ជាTablet

```
SELECT *
FROM Items
WHERE BrandId = 1 AND CategoryId = 2
```

	ItemId	ItemName	CategoryId	BrandId
1	9	iPad Pro 11(2020)	2	1
2	10	iPad Air 11(2020)	2	1
3	11	iPad Mini 11 (2019)	2	1

ឧទាហរណ៍ទី៤៖ Query Table Items ក្នុងលក្ខខណ្ឌ BrandId ជាApple ឬជាSamsung

```
SELECT *
FROM Items
WHERE BrandId = 1 OR BrandId = 2
```

	ItemId	ItemName	CategoryId	BrandId
1	1	iPhone 8 Plus	1	1
2	2	iPhone X	1	1
3	3	iPhone XS	1	1

(24 rows)

### ៣.៣ IN Operator

គេប្រើ IN Operator ដើម្បីយកតម្លៃជាក់លាក់មួយ ប្រៀបធៀបនឹងបណ្តុំតម្លៃថាតើវាត្រូវគ្នានឹងតម្លៃណាមួយឬទេ។

ឧទាហរណ៍ទី៥៖ Query Table Items ក្នុងលក្ខខណ្ឌ BrandId ជាមួយបញ្ជីតម្លៃ (1, 2, 3)

```
SELECT *
FROM Items
WHERE BrandId IN(1,2,3)
```

(30 rows)

លក្ខខណ្ឌផ្ទុយពីតម្លៃក្នុងបញ្ជី

```
SELECT *
FROM Items
WHERE BrandId NOT IN(1,2,3)
```

	ItemId	ItemName	CategoryId	BrandId
1	31	Vivo Y21	1	4
2	32	Vivo X70	1	4
3	33	Vivo X70 Pro	1	4

(10 rows)

### ៣.៤ Between Operator

គេប្រើ Between Operator ដើម្បីកំណត់លក្ខខណ្ឌចន្លោះរវាងតម្លៃពីរ

ឧទាហរណ៍ទី៦៖ Query Invoices លក្ខខណ្ឌចន្លោះរវាង 2021-9-1 ដល់ 2021-10-31

```
SELECT *
FROM Invoices
WHERE InvoiceDate BETWEEN '2021-9-1' AND '2021-10-31'
```

	InvoiceId	CustomerId	InvoiceNo	InvoiceDate	InvoiceTotal	PaymentTotal	InvoiceDueDate	PaymentDate	SaleId
1	1	1	1	2021-09-29 00:00:00.000	360.00	360.00	2021-09-29 00:00:00.000	2021-09-29 00:00:00.000	1
2	2	1	2	2021-09-29 00:00:00.000	590.00	590.00	2021-09-29 00:00:00.000	2021-09-29 00:00:00.000	1
3	3	1	3	2021-09-29 00:00:00.000	340.00	340.00	2021-09-29 00:00:00.000	2021-09-29 00:00:00.000	1

(13 rows)

## ៣.៥ Like Operator

គេប្រើ Like Operator ដើម្បីកំណត់លក្ខខណ្ឌលើ Character String ទៅនឹងលំនាំតម្លៃជាក់លាក់ណាមួយ។ សញ្ញា ‘ % ’ ត្រូវបានគេប្រើជាអក្សរជំនួស(សូន្យ ឬច្រើនតួ)

ឧទាហរណ៍ទី៧៖ Query Customer ក្នុងលក្ខខណ្ឌ FirstName តាមលំនាំតម្លៃខាងក្រោម

```
SELECT *
FROM Customers
WHERE FirstName LIKE 'K%'
```

	CustomerId	FirstName	LastName	Phone	Address
1	4	Ka	Kakada	012873947	Banteay Meanchey
2	5	Kha	Sokhin	077432334	Battambang

```
SELECT *
FROM Customers
WHERE FirstName LIKE '%e%'
```

	CustomerId	FirstName	LastName	Phone	Address
1	1	General	Customer	0973927434	Battambang
2	6	Veasna	SokLeap	098379427	SiemReap

លំហាត់ទី១.៣៖ ចូរអនុវត្ត Query Table ItemProperties Where UnitPrice តិចជាង 200 ឬច្រើនជាង 1000

លំហាត់ទី១.៤៖ ចូរអនុវត្ត Query Table Invoices Where InvoiceTotal ច្រើនជាង 1000

លំហាត់ទី១.៥៖ ចូរអនុវត្ត Query Table ItemProperties Where ItemId ដែលក្នុងបញ្ជីរ តម្លៃ (5,6,7)

## ៤. ការតម្រៀបទិន្នន័យលើ Select Statement (Order By Clause)

Order By Clause នៅក្នុង Select Statement ត្រូវបានគេប្រើដើម្បីតម្រៀបតម្លៃតាមអក្ខរក្រមពីទាបទៅខ្ពស់(ASC) ឬពីខ្ពស់មកទាប(DESC) ទៅលើ Field ឬ Column មួយឬច្រើន។

ឧទាហរណ៍ទី៨៖ Query Table Invoice ក្នុងលក្ខខណ្ឌកំណត់មួយនិងតម្រៀបដោយ InvoiceTotal តាមលំដាប់កើន (ASC)

```
SELECT *
FROM Invoices
WHERE PaymentTotal IS NULL OR PaymentTotal = 0
ORDER BY InvoiceTotal ASC
```

	InvoiceId	CustomerId	InvoiceNo	InvoiceDate	InvoiceTotal	PaymentTotal	InvoiceDueDate	PaymentDate	SaleId
1	10	4	5	2021-09-30 00:00:00.000	780.00	NULL	2021-10-10 00:00:00.000	NULL	2
2	9	3	4	2021-09-30 00:00:00.000	1410.00	NULL	2021-10-07 00:00:00.000	NULL	2
3	17	6	3	2021-11-01 00:00:00.000	1925.00	0.00	2021-11-04 00:00:00.000	NULL	3

(8 rows)



**លំហាត់ទី១.៦៖** ចូរអនុវត្ត Query Table Items Where BrandId ជា Samsung និងតម្រៀប  
ទិន្នន័យដោយ Field **ItemName** តាមលំដាប់ចុះ (DESC)

កំណត់សម្គាល់ខ្លី៖





## មេរៀនទី២

## ការ Query ទិន្នន័យពី Tables ច្រើន

ការ Join Table ត្រូវបានគេប្រើប្រាស់ដើម្បីដាក់បញ្ចូល Columns ពី Table ពីរប្រើនិច្ចលក្ខណៈទៅជាទម្រង់លទ្ធផលតែមួយ (Single Result Set)។

## ១. ការប្រើប្រាស់ INNER JOIN

សម្រាប់លទ្ធផលដែលនឹងត្រូវទទួលបានក្រោយការប្រើប្រាស់ INNER JOIN គឺមានតែ Rows ឬក៏ Records ដែលត្រូវនឹងលក្ខខណ្ឌ Join ដែលបានកំណត់តែប៉ុណ្ណោះ។

ជានិច្ចជាកាលគេតែងតែ Join Tables ដោយផ្អែកលើ Relationship រវាង Primary Key របស់ Table មួយទៅនឹង Foreign Key នៃ Table មួយទៀត។

ឧទាហរណ៍ទី១៖ Query Table Items INNER JOIN Brands ដោយជ្រើសរើស Columns ItemId, ItemName, BrandName

```
SELECT
    ItemId,
    ItemName,
    BrandName
FROM Items INNER JOIN Brands
    ON Items.BrandId = Brands.BrandId
```

	ItemId	ItemName	BrandName
1	1	iPhone 8 Plus	Apple
2	2	iPhone X	Apple
3	3	iPhone XS	Apple
4	4	iPhone 11 Pro	Apple

(40 rows)

ឧទាហរណ៍ទី២៖ Query Table Invoices INNER JOIN Sales ដោយប្រើប្រាស់ Table Alias Table Alias គឺជាឈ្មោះបណ្តោះអាសន្នតាងឲ្យ Table នៅក្នុង FROM Clause CONCAT Function គឺប្រើសម្រាប់ភ្ជាប់តម្លៃបញ្ចូលគ្នា

```
SELECT
    I.InvoiceId,
    I.CustomerId,
    I.InvoiceNo,
    I.InvoiceDate,
    I.InvoiceTotal,
    I.PaymentTotal,
    CONCAT (S.FirstName, ' ', S.LastName) AS Sale
FROM Invoices AS I JOIN Sales AS S
    ON I.SaleId = S.SaleId
```

	InvoiceId	CustomerId	InvoiceNo	InvoiceDate	InvoiceTotal	PaymentTotal	Sale
1	1	1	1	2021-09-29 00:00:00.000	360.00	360.00	Admin Store
2	2	1	2	2021-09-29 00:00:00.000	590.00	590.00	Admin Store
3	3	1	3	2021-09-29 00:00:00.000	340.00	340.00	Admin Store

(24 rows)

**លំហាត់ទី២.១៖** ចូរ Query Top 10 records ពី Table Items INNER JOIN ItemProperties

## ២. ការប្រើប្រាស់ OUTER JOIN

សម្រាប់លទ្ធផលដែលនឹងត្រូវទទួលបានក្រោយការប្រើប្រាស់ OUTER JOIN គឺមានទាំង Rows ឬក៏ Records ដែលត្រូវនឹងលក្ខខណ្ឌ Join និង Rows ដែលមិនត្រូវលក្ខខណ្ឌនៃ Table មួយឬពីរនោះ។ OUTER JOIN មានបីជម្រើសគឺ៖

1. LEFT JOIN: ទទួល Rows ទាំងអស់ពី Table ទី១ ឬខាងឆ្វេង
2. RIGHT JOIN: ទទួល Rows ទាំងអស់ពី Table ទី២ ឬខាងស្តាំ
3. FULL JOIN: ទទួល Rows ទាំងអស់ពី Table ទាំងទ្វេ

ឧទាហរណ៍៖ Query OUTER JOIN រវាង ItemProperties និង InvoiceItems

```
SELECT *
FROM ItemProperties AS Ip LEFT JOIN InvoiceItems AS It
ON Ip.Code = It.ItemCode
```

	Code	ItemId	Color	Capacity	InStock	UnitPrice	InvoiceId	ItemNo	ItemCode	Quantity	Price
1	1	1	Silver	128GB 2GB RAM	10	250.00	6	1	1	2	250.00
2	1	1	Silver	128GB 2GB RAM	10	250.00	17	3	1	1	250.00
3	2	1	Space Gray	128GB 2GB RAM	10	255.00	17	1	2	1	255.00

(76 rows)

ឧទាហរណ៍ទី២៖ Query OUTER JOIN រវាង Tables Invoices និង Sales

```
SELECT
    I.InvoiceId,
    I.InvoiceNo,
    I.InvoiceDate,
    I.InvoiceTotal,
    I.PaymentTotal,
    I.SaleId,
    S.FirstName,
    S.LastName,
    S.Phone
FROM Invoices AS I RIGHT JOIN Sales AS S
ON S.SaleId = I.SaleId
AND InvoiceDate >= '2021-12-1'
```

	InvoiceId	InvoiceNo	InvoiceDate	InvoiceTotal	PaymentTotal	SaleId	FirstName	LastName	Phone
1	23	2	2021-12-11 00:00:00.000	8700.00	0.00	1	Admin	Store	0973473223
2	24	1	2021-12-12 00:00:00.000	1998.00	1998.00	1	Admin	Store	0973473223
3	22	1	2021-12-11 00:00:00.000	700.00	700.00	2	Sok	Reaksa	012343987
4	25	2	2021-12-12 00:00:00.000	1998.00	1998.00	2	Sok	Reaksa	012343987
5	NULL	NULL	NULL	NULL	NULL	NULL	Na	Lyly	098734234
6	NULL	NULL	NULL	NULL	NULL	NULL	Si	DaDay	015987394



លំហាត់ទី២.២៖ ចូរ Query LEFT JOIN រវាង Table Categories និង Items  
ORDER BY Items.CategoryId

### ៣. ការប្រើប្រាស់ SELF JOIN

Self-Join គឺជាការJoin រវាង Table មួយ នឹងខ្លួនឯង។

ឧទាហរណ៍៖ Query Self Join នៃ Table Sales (INNER JOIN)

```
SELECT
    S.SaleId,
    s.FirstName,
    S.LastName,
    s.Phone,
    CONCAT(M.FirstName, ' ', M.LastName) AS Manager
FROM Sales S JOIN Sales M
    ON S.ManagerId = M.SaleId
```

	SaleId	FirstName	LastName	Phone	Manager
1	2	Sok	Reaksa	012343987	Admin Store
2	3	Na	Lyly	098734234	Admin Store
3	4	Si	DaDay	015987394	Admin Store

ការ Query Self Join នៃ Table Sales (LEFT JOIN)

IIF Function: ទទួលតម្លៃត្រឡប់ដោយផ្អែកលើ លក្ខខណ្ឌ Expression True ឬក៏ False

```
SELECT
    S.SaleId,
    s.FirstName,
    S.LastName,
    s.Phone,
    IIF(M.SaleId IS NULL, 'N/A', CONCAT(M.FirstName, ' ',
M.LastName)) AS Manager
FROM Sales S LEFT JOIN Sales M
    ON S.ManagerId = M.SaleId
```

	SaleId	FirstName	LastName	Phone	Manager
1	1	Admin	Store	0973473223	N/A
2	2	Sok	Reaksa	012343987	Admin Store
3	3	Na	Lyly	098734234	Admin Store
4	4	Si	DaDay	015987394	Admin Store

កំណត់សម្គាល់ខ្លី៖



## មេរៀនទី៣

## ការកែប្រែទិន្នន័យនៅលើ Table

## ១. ការបញ្ចូលទិន្នន័យទៅ Table (Insert Statement)

Insert Statement ត្រូវបានគេប្រើប្រាស់ដើម្បីបញ្ចូលទិន្នន័យទៅក្នុង Table។

## ១.១ ការ Insert តែមួយ Table

ឧទាហរណ៍៖ Insert Entry Data ទៅឲ្យ Table Items

```
INSERT Items VALUES('Samsung Galaxy Note 20', 1, 2)
```

Query ពី Table Items ក្រោយ Insert រួច

	ItemId	ItemName	CategoryId	BrandId
39	39	Sony Xperia 10 III Lite	1	5
40	40	Sony Xperia Pro	1	5
41	41	Samsung Galaxy Note 20	1	2

(41 rows)

លំហាត់ទី៣.១៖ Insert Entry Data ទៅឲ្យ Table Brands ចំនួនពីរ Rows

## ១.២ ការ Insert មេបេឡាសម្ព័ន្ធ

ឧទាហរណ៍៖ Insert Entry Data ទៅឲ្យ Invoices និង InvoiceItems

```
-- Invoices
INSERT Invoices (CustomerId, InvoiceNo, InvoiceDate)
VALUES(1, 1, '2021-12-30')

-- @@IDENTITY = Current Generated Identity
-- InvoiceItems
INSERT InvoiceItems VALUES
(@@IDENTITY, 1, 45, 1, 330),
(@@IDENTITY, 2, 50, 1, 100)
```

Invoices Table

	InvoiceId	CustomerId	InvoiceNo	InvoiceDate	InvoiceTotal	PaymentTotal
23	24	1	1	2021-12-12 00:00:00.000	1998.00	1998.00
24	25	1	2	2021-12-12 00:00:00.000	1998.00	1998.00
25	26	1	1	2021-12-30 00:00:00.000	NULL	NULL

InvoiceItems Table

	InvoiceId	ItemNo	ItemCode	Quantity	Price
47	24	1	58	2	999.00
48	25	1	59	2	999.00
49	26	1	45	1	330.00
50	26	2	50	1	100.00



## ២. ការលុបទិន្នន័យពី Table (Delete Statement)

Delete Statement ត្រូវបានគេប្រើដើម្បីលុប Row ពីក្នុង Table។

ឧទាហរណ៍ទី១៖ ការលុប Row ដែលមាន BrandId = 7 ចេញពី Table Brands

Query Table Brands មុនពេលលុប

	BrandId	BrandName
4	4	Vivo
5	5	Sony
6	6	LG
7	7	Google

(7 rows)

គេប្រើ WHERE Clause នៅក្នុង DELETE Statement ដើម្បីបញ្ជាក់ពីជួរណាមួយត្រូវលុប ពុំដូច្នោះទេ Rows ទាំងអស់នៅក្នុង Table នោះនឹងត្រូវបានលុប

```
DELETE Brands
WHERE BrandId = 7
```

Query Table Brands ក្រោយពេលលុប

	BrandId	BrandName
3	3	Oppo
4	4	Vivo
5	5	Sony
6	6	LG

(6 rows)

ឧទាហរណ៍ទី២៖ ការលុប Row ចេញពី Table Invoices

```
DELETE Invoices
WHERE InvoiceId = 26
```

មិនអាចលុបបាន មាន ERROR កើតឡើង ដោយសារ Invoices ដែលមាន InvoiceId = 26 មាន ទំនាក់ទំនងជាមួយ Table InvoiceItems

The DELETE statement conflicted with the REFERENCE constraint "FK\_InvoiceIt\_Invoi\_37A5467C". The conflict occurred in database "PhoneStore", table "dbo.InvoiceItems", column 'InvoiceId'.

ដូច្នេះ ដើម្បីលុបបានគេត្រូវលុប Rows នៅក្នុង InvoiceItems ដែលមាន InvoiceId = 26 សិន សឹមលុប Invoices តាមក្រោយ

```
-- First Delete Related Rows in InvoiceItems Depend On InvoiceId
DELETE InvoiceItems
WHERE InvoiceId = 26
```



```
-- Second Delete Rows in Invoices Depend On Same InvoiceId as
InvoiceItems' InvoiceId
DELETE Invoices
WHERE InvoiceId = 26
```

### លំហាត់ទី៣.២៖

Insert Entry Data ទៅឱ្យ Table Brands ដោយឱ្យ BrandName = Google (បើមិនទាន់មាន)

Insert 2 rows ឱ្យ Items

1. Row1 ('Pixel 3', 1, Google BrandId)
2. Row2 ('Pixel 3XL', 1, Google BrandId)

Insert rows ឱ្យ ItemProperties

1. Row1 (Pixel 3XL ItemId, 'White', '64GB 4GB RAM', 10, 140)
2. Row2 (Pixel 3XL ItemId, 'Black', '64GB 4GB RAM', 20, 150)

ចូរលុប Items Rows ដែលមាន ItemId ដូចនឹង Pixel 3 និង Pixel 3XL

### ៣. ការកែសម្រួលទិន្នន័យលើ Table (Update Statement)

Update Statement ត្រូវបានគេប្រើប្រាស់ដើម្បី កែសម្រួល Data Entry ឬក៏ Row នៅក្នុង Table។

គេប្រើ WHERE Clause នៅក្នុង UPDATE Statement ដើម្បីបញ្ជាក់ពីជួរណាមួយត្រូវកែប្រែ ពុំដូច្នោះទេ Rows ទាំងអស់នៅក្នុង Table នោះនឹងត្រូវបានកែទៅជាតម្លៃថ្មី

ឧទាហរណ៍៖ Update UnitPrice, InStock នៃ Table ItemProperties Where Code = 1

Query Table ItemProperties មុនUpdate

	Code	ItemId	Color	Capacity	InStock	UnitPrice
1	1	1	Silver	128GB 2GB RAM	10	250.00
2	2	1	Space Gray	128GB 2GB RAM	10	255.00

```
UPDATE ItemProperties
```

```
SET UnitPrice = 245,
    InStock = InStock + 5
```

```
WHERE Code = 1
```

Query Table ItemProperties ក្រោយUpdate

	Code	ItemId	Color	Capacity	InStock	UnitPrice
1	1	1	Silver	128GB 2GB RAM	15	245.00
2	2	1	Space Gray	128GB 2GB RAM	10	255.00

លំហាត់ទី៣.៣៖ ចូរ Update Phone, Address ពី Table Customers WHERE  
CustomerId = 6

កំណត់សម្គាល់ខ្លី៖



## មេរៀនទី៤

## Aggregate Function

Aggregate Function គឺជា Built In Function របស់ Ms SQL Server និងដែលត្រូវបានគេប្រើប្រាស់ដើម្បីសរុបរួមឬសង្ខេប Expression របស់ Table ។

Aggregate Function មានដូចជា៖ COUNT(), SUM(), AVG(), MIN(), MAX(), ...

## ១. ការរាប់ចំនួន Rows នៅក្នុងបង្គោល (COUNT)

Count() Function ត្រូវបានគេប្រើដើម្បី រាប់ចំនួន Rows ដែលមាននៅក្នុង Set ឬក្រុមទាំង Rows ដែលមានផ្ទុកតម្លៃ NULL ។

ឧទាហរណ៍៖ Query to COUNT Customer ទាំងអស់ដែលមានក្នុង Store

```
SELECT COUNT(*) AS Amount_Customer FROM Customers
```

	Amount_Customer
1	6

លំហាត់ទី៤.១៖ ចូររាប់ចំនួនសរុប Invoices ចាប់ពីថ្ងៃ InvoiceDate: '2021-11-01'

## ២. ការបូកសរុបតម្លៃនៅក្នុងបង្គោល (SUM)

SUM() Function ត្រូវបានគេប្រើប្រាស់ដើម្បីបូកសរុបតម្លៃទាំងអស់ Non NULL ដែលស្ថិតក្នុង Set ។

ឧទាហរណ៍៖ Query to SUM Total Price of InvoiceItems Where InvoiceId = 1

```
SELECT
    SUM(IT.Price * IT.Quantity) AS Sum_of_Invoice
FROM InvoiceItems IT
WHERE InvoiceId = 1
```

	Sum_of_Invoice
1	360.00

លំហាត់ទី៤.២៖ ចូរបូកសរុបចំនួនឧបករណ៍ InStock នៃ ItemProperties Where ItemId = -

## ៣. ការរកមធ្យមតម្លៃនៅក្នុងបង្គោល (AVG)

AVG() Function ត្រូវបានគេប្រើប្រាស់ដើម្បីសរុបមធ្យមតម្លៃ Non NULL ដែលស្ថិតក្នុង Set ។

ឧទាហរណ៍៖ Query to Average the InvoiceTotal នៃ Invoices នៅរវាង ឆ្នាំ 2021 ខែ 11

```
SELECT AVG(InvoiceTotal) Average_In_202111
FROM Invoices
WHERE InvoiceDate BETWEEN '2021-11-1' AND '2021-11-30'
```

	Average_In_202111
1	1929.285714



#### ៤. ការQueryទិន្នន័យដោយដាក់ជាក្រុម ( GROUP BY)

GROUP BY Clause ត្រូវបានគេប្រើប្រាស់សម្រាប់រៀបចំការ Query ដើម្បីទទួលបាន Rows បែបជាក្រុម។ គេអាចកំណត់ក្រុមបានដោយត្រូវបញ្ជាក់ឈ្មោះColumn នៅក្នុងGROUP BY Clause ឧទាហរណ៍៖ Query InvoiceId, SUM Total Price ពី Table InvoiceItems ដោយ GROUP BY ទៅលើ Column InvoiceId

```
SELECT
    InvoiceId,
    SUM(Price * Quantity) AS InvoiceTotal
FROM InvoiceItems
GROUP BY InvoiceId
ORDER BY InvoiceId DESC
```

	InvoiceId	InvoiceTotal
1	25	1998.00
2	24	1998.00
3	23	8700.00

(23 rows)

លំហាត់ទី៤.៣៖ ចូរQuery ពី Table Invoices នូវ InvoiceDate, Count of Invoice, Sum total of Invoice ដោយ GROUP BY InvoiceDate និង ORDER BY InvoiceDate DESC

#### ៥. ការកំណត់លក្ខខណ្ឌ Queryទិន្នន័យដោយដាក់ជាក្រុម ( HAVING)

HAVING Clause ត្រូវបានប្រើប្រាស់ជាញាក់ញាប់ជាមួយនឹង GROUP BY Clause ដើម្បីកំណត់លក្ខខណ្ឌទៅលើ GROUP BY Clause ឬ Aggregate Function។

ឧទាហរណ៍៖ Query Table Invoices JOIN Customer នូវ COUNT of Invoices and SUM of InvoiceTotal សម្រាប់ Customer (LastName) នីមួយៗ ដែលមិនមែនជា “General Customer” ហើយដែលមាន SUM of InvoiceTotal ច្រើនជាង 2000

```
SELECT
    C.LastName AS Customer,
    COUNT(*) AS Amount,
    SUM(I.InvoiceTotal) AS Total
FROM Invoices I JOIN Customers C
    ON I.CustomerId = C.CustomerId
WHERE C.LastName <> 'Customer'
GROUP BY C.LastName
HAVING SUM(I.InvoiceTotal) > 2000
```

លំហាត់ទី៤.៤៖ ចូរQuery table ដើម្បីរកចំនួនទំនិញដែលបានលក់ចេញ នៃ Item ប្រភេទខុសៗគ្នា ដោយជ្រើស ItemCode, ItemName, Color, Capacity, SUM of Quntity ហើយកំណត់យកតែទំនិញណាដែលមានចំនួនលក់ចេញចាប់ពី ៥ ឡើងទៅ



## មេរៀនទី៥

## ការសិក្សាពី Complex Query

## ១. ស្វែងយល់ពី Sub Query

Sub Query គឺជា Select Statement ដែលស្ថិតក្នុង SQL Statement ដទៃទៀត។

ឧទាហរណ៍ទី១៖ Query Table Items ដែលមាន Category ជា Tablet

```
SELECT * FROM Items
WHERE CategoryId =
(
    SELECT CategoryId
    FROM Categories
    WHERE CategoryName = 'Tablet'
)
```

	ItemId	ItemName	CategoryId	BrandId
1	9	iPad Pro 11(2020)	2	1
2	10	iPad Air 11(2020)	2	1
3	11	iPad Mini 11 (2019)	2	1
4	22	Samsung Galaxy Tab A7	2	2
5	23	Samsung Galaxy Tab S7	2	2

លំហាត់ទី៥.១៖ Query Table Items ដែលមាន Brand ជា Oppo (Use Sub Query to Query BrandId ពី Table Brand)

ឧទាហរណ៍ទី២៖ Query Table Customers ដែលមាន InvoiceDate ស្មើនឹង 2021-11-01

```
SELECT * FROM Customers
WHERE CustomerId IN
(
    SELECT CustomerId FROM Invoices
    WHERE InvoiceDate = '2021-11-1'
)
```

	CustomerId	FirstName	LastName	Phone	Address
1	2	Si	Dalen	016878934	SiemReap
2	5	Kha	Sokhin	077432334	Battambang
3	6	Veasna	SokLeap	098379427	SiemReap

លំហាត់ទី៥.២៖ Query Table Brands ដែលមាន Count of Items តិចជាង១០

(ចូរប្រើប្រាស់ Sub Query Table Items ដើម្បីរក BrandId List ដែលមាន Count of Items តិចជាង១០)

## ២. ការកំណត់ Correlated Sub Query

Correlated Sub Query គឺជា Sub Query ដែលពឹងផ្អែកលើតម្លៃរបស់ Outer Query។ មានន័យថា Correlated Sub Query មិនអាច Execute ដោយឯកឯងដូច Sub Query ធម្មតាបានឡើយ។

ឧទាហរណ៍៖ Query [ InvoiceId, InvoiceDate, InvoiceTotal, SaleId ] ពី Table Invoices ដែលមាន InvoiceTotal ធំជាង មធ្យមនៃ InvoiceTotal របស់ Sale (SaleId) នីមួយៗ

```
SELECT
    InvoiceId,
    InvoiceDate,
    InvoiceTotal,
    SaleId
FROM Invoices I
WHERE InvoiceTotal >
(
    SELECT AVG(InvoiceTotal)
    FROM Invoices WHERE SaleId = I.SaleId
)
```

	InvoiceId	InvoiceDate	InvoiceTotal	SaleId
1	6	2021-09-30 00:00:00.000	1970.00	1
2	13	2021-10-25 00:00:00.000	2000.00	1
3	23	2021-12-11 00:00:00.000	8700.00	1

(9 rows)

**លំហាត់ទី៥.៣៖** Query [ Code, ItemName, Color, Capacity, UnitPrice ] ពី Table Items JOIN ItemProperties ដែលមាន UnitPrice ច្រើនជាង មធ្យមនៃ UnitPrice របស់ Item (ItemId) នីមួយៗ

## ៣. ការប្រើប្រាស់ Exists Operator

Exists Operator ត្រូវបានគេប្រើដើម្បីពិនិត្យ Sub Query ថា តើមាន Row ត្រឡប់មកឬអត់។ ឧទាហរណ៍៖ Query Table ដើម្បីរក Sale ដែលមិនមាន Invoices នៅកាលវេលាណាមួយ

```
SELECT *
FROM Sales S
WHERE NOT EXISTS(
    SELECT * FROM Invoices
    WHERE InvoiceDate = '2021-11-1'
    AND SaleId = S.SaleId
)
```

(2 rows)

**លំហាត់ទី៥.៤៖** Query Table រក ItemProperties ដែលបានលក់ចេញ

## មេរៀនទី៦

## ការសិក្សាពី View

## ១. និយមន័យ

View គឺជា Object មួយរបស់ Database ដែលមានផ្ទុក SELECT Statement ។ View ត្រូវបានគេចាត់ទុកជា Table និមិត្តមួយ ដែលមិនមានផ្ទុកទិន្នន័យដោយផ្ទាល់ តែវាយោងតម្លៃតាម Base Table របស់វា។ View អាចផ្ទុក SELECT Statement ស្ទើរគ្រប់បែប តាំង Simple Query, Join Query, Sub Query, Summary Query(Aggregation)។

## ២. ការកំណត់ View

ដើម្បីបង្កើត View គេប្រើ៖ CREATE VIEW Keyword  
ឧទាហរណ៍ទី១៖ បង្កើត View ដោយយោងពី Table តែមួយ

```
CREATE VIEW VSaleInfo
AS
    SELECT
        CONCAT(S.FirstName, ' ', S.LastName) AS SaleName,
        S.Phone
    FROM Sales S
```

Query ពី View VSaleInfo

```
SELECT * FROM VSaleInfo
```

	FullName	Phone
1	Admin Store	0973473223
2	Sok Reaksa	012343987
3	Na Lyly	098734234
4	Si DaDay	015987394

លំហាត់ទី៦.១៖ បង្កើត View VCustomerInfo ដោយយោងពី Table Customers ដែលមាន Columns: CustomerName, Phone និង Address

View អាចបំពាក់ SELECT Statement ដែលមានលក្ខណៈសំបូរ បង្កឱ្យមានភាពងាយស្រួលសម្រាប់ Application Programming ក្នុងការ Query ទិន្នន័យប្រើប្រាស់។

ឧទាហរណ៍ទី២៖ បង្កើត View ដោយយោងពី Joined Table

```
CREATE VIEW ItemList
AS
    SELECT
        IP.Code,
        I.ItemName,
        IP.Color,
        IP.Capacity,
        IP.InStock,
        IP.UnitPrice
    FROM Items I JOIN ItemProperties IP
    ON I.ItemId = IP.ItemId
```

Query View ItemList

```
SELECT
    I.Code,
    I.ItemName,
    I.Color,
    I.Capacity,
    I.UnitPrice
FROM ItemList I
WHERE I.UnitPrice < 300
```

	Code	ItemName	Color	Capacity	UnitPrice
1	1	iPhone 8 Plus	Silver	128GB 2GB RAM	245.00
2	2	iPhone 8 Plus	Space Gray	128GB 2GB RAM	255.00
3	3	iPhone X	Silver	64GB 3GB RAM	280.00

(25 rows)

**លំហាត់ទី២.២៖** បង្កើត View SaleShortList ដើម្បីរក Sale ដែលមាន Invoices

### ៣. ការកែប្រែ View

ដើម្បីកែសម្រួល View របស់ Database គេប្រើប្រាស់ Keyword ALTER VIEW ។

ឧទាហរណ៍៖ កែសម្រួល View SaleShortList

```
ALTER VIEW SaleShortList
AS
    SELECT * FROM Sales
    WHERE SaleId IN (
        SELECT SaleId FROM Invoices
        GROUP BY SaleId
        HAVING COUNT(*) > 5
    )
```

**លំហាត់ទី២.៣៖** កែសម្រួល View VItemList ដោយកំណត់ Query Join Table (Items, ItemProperties) WHERE InStock > 15

### ៤. ការលុប View

ដើម្បីលុប View របស់ Database គេប្រើប្រាស់ Keyword DROP VIEW ។

ឧទាហរណ៍៖ ការលុប View SaleShortList

```
DROP VIEW SaleShortList
```

## មេរៀនទី៧

## ការសិក្សាពី Store Procedure

## ១. និយមន័យ

Store Procedure គឺជា Object របស់ Database ដែលអាច Execute ឬក៏ Run និងមានផ្ទុក SQL Statement មួយឬច្រើន។ គេអាចហៅ Store Procedure ជាខ្លីបានថា *sproc* ឬ *procedure*។ គុណសម្បត្តិរបស់ Store Procedure គឺការបង្កភាពងាយស្រួលដល់ Application Programmer ដោយគាត់អាចហៅប្រើ Store Procedure បាន ដោយមិនបាច់យល់ពីរចនាសម្ព័ន្ធ Database ឬក៏ការសរសេរ SQL Statement។

## ២. ការកំណត់ Store Procedure

ដើម្បីបង្កើត Store Procedureបាន គេប្រើប្រាស់ Keyword CREATE PROC។ ជាទូទៅគេតែងដាក់ឈ្មោះឱ្យ Store Procedure ដោយផ្ដើមដោយ “sp”។

ឧទាហរណ៍ទី១៖ បង្កើត Simple Store Procedure ដើម្បី Query Customers Table

```
CREATE PROC spCustomerInfo
AS
```

```
SELECT * FROM Customers
```

ដើម្បី Run ឬក៏ Call Store Procedure គេប្រើប្រាស់ Keyword EXEC

```
EXEC spCustomerInfo
```

	CustomerId	FirstName	LastName	Phone	Address
1	1	General	Customer	0973927434	Battambang
2	2	Si	Dalen	016878934	SiemReap
3	3	Cham	Chhara	092897943	Pusat
4	4	Ka	Kakada	012873947	Banteay Meanchey
5	5	Kha	Sokhin	077432334	Battambang
6	6	Veasna	SokLeap	098379427	SiemReap

លំហាត់ទី៧.១៖ បង្កើត Store Procedure ឈ្មោះ spGetItemList ដែលកើតចេញការ

Joined Table Items និង ItemProperties។

Fields: [Code, ItemId, ItemName, Color, Capacity, InStock, UnitPrice]

Store Procedure អាចមាន Parameters។ ឈ្មោះរបស់ Parameter ត្រូវផ្ដើមដោយ (@) បន្ទាប់មក SQL Server Data Type។ គេចែក Parameter ជាពីរគឺ Input និង Output Parameter។ Input ប្រើសម្រាប់ Pass តម្លៃទៅឱ្យ Store Procedure ហើយ Output ប្រើសម្រាប់ត្រឡប់តម្លៃមកវិញ នៅពេលគេ EXEC Store Procedure។

ឧទាហរណ៍ទី២៖ បង្កើត Store Procedure ដែលមាន Parameter ដើម្បី Query Invoices តាម CustomerId

```
CREATE PROC spGetInvoicesByCustomer @customerId Int
AS
```

```
SELECT * FROM Invoices
```



**WHERE** CustomerId = @customerId

Execute *spGetInvoicesByCustomer* ដោយ Pass CustomerId = 2

**EXEC** spGetInvoicesByCustomer 2

	InvoiceId	CustomerId	InvoiceNo	InvoiceDate	InvoiceTotal	PaymentTotal	InvoiceDueDate	PaymentDate	SaleId
1	6	2	1	2021-09-30 00:00:00.000	1970.00	0.00	2021-10-07 00:00:00.000	NULL	1
2	14	2	1	2021-11-01 00:00:00.000	3680.00	0.00	2021-11-08 00:00:00.000	NULL	4

ឧទាហរណ៍ទី៣៖ បង្កើត Store Procedure មាន Parameter Input និង Output ដើម្បីរកផលបូក InvoiceTotal ក្នុងកាលបរិច្ឆេទជាក់លាក់មួយ

```
CREATE PROC spInvoiceTotal @date Date, @invoiceTotal Decimal OUTPUT
AS
    SELECT @invoiceTotal = SUM(InvoiceTotal)
    FROM Invoices
    WHERE InvoiceDate = @date
```

ការ Call Store Procedure ដែលមាន Output parameter

```
DECLARE @InvoiceTotal AS Decimal;
EXEC spInvoiceTotal '2021-11-01', @InvoiceTotal OUTPUT

SELECT @InvoiceTotal AS InvoiceTotal
```

	InvoiceTotal
1	7585

ឧទាហរណ៍ទី៤៖ បង្កើត Procedure ដើម្បី Insert Entry ឲ្យ Table Items

```
CREATE PROC spInsertItem
    @itemName VARCHAR(50),
    @categoryId INT,
    @brandId INT
AS
BEGIN
    INSERT Items VALUES (@itemName, @categoryId, @brandId)
END
```

ការ Execute Procedure

```
EXEC spInsertItem 'iPhone 13 Pro', 1, 1

DECLARE @brandId INT = (SELECT BrandId FROM Brands WHERE
BrandName='LG')
EXEC spInsertItem 'LG V50 ThinQ', 1, @brandId
```

**លំហាត់ទី៧.២៖** បង្កើត Store Procedure ឈ្មោះ spInsertItemProperty ដើម្បី insert Entry ឲ្យ Table ItemProperties និងមាន output parameter ដើម្បីត្រឡប់តម្លៃ Code ចុងក្រោយនៃ ItemProperty ក្រោយការហៅ procedure

ឧទាហរណ៍ទី៥៖ បង្កើត Procedure សម្រាប់ Insert Invoice និង InvoiceItems

TypeInvoiceItem គឺជា User Defined Type បង្កើតឡើងក្នុងគោលបំណងរក្សាទុកទិន្នន័យមុននឹងបញ្ជូនទៅ Table InvoiceItems

```
CREATE TYPE typeInvoiceItem AS TABLE
(
    ItemNo INT,
    ItemCode INT,
    Quantity INT,
    Price Decimal
);
```

```
CREATE PROC spInsertInvoice
    @customerId Int,
    @invoiceNo Int,
    @invoiceDate Date,
    @dueDay Int,
    @saleId Int,
    @invoiceItems TypeInvoiceItem READONLY
AS
BEGIN
    DECLARE @invoiceDueDate AS DATE = DATEADD(DD, @dueDay,
@invoiceDate);
    DECLARE @invoiceTotal AS DECIMAL = (SELECT SUM(Quantity * Price)
FROM @invoiceItems)

    -- Invoice
    INSERT Invoices VALUES (@customerId, @invoiceNo, @invoiceDate,
@invoiceTotal, 0, @invoiceDueDate, NULL, @saleId)

    DECLARE @lastInvoiceId INT = @@IDENTITY
    -- Invoice Items
    -- Copy Data from @invoiceItems to InvoiceItems Table
    INSERT InvoiceItems SELECT @lastInvoiceId, * FROM @invoiceItems
END
```

ការហៅ Procedure spInsertInvoice

```
DECLARE @invoiceItems AS TypeInvoiceItem;
INSERT @invoiceItems VALUES(1, 62, 1, 2500);
INSERT @invoiceItems VALUES(2, 61, 1, 380);

EXEC spInsertInvoice 1, 1, '2021-12-13', 0, 2, @invoiceItems
```

### ៣. ការកែប្រែ Store Procedure

ដើម្បីកែសម្រួល Store Procedure ឡើងវិញ គេប្រើប្រាស់ Keyword ALTER PROC។

ឧទាហរណ៍៖ ការកែសម្រួល spInsertItem Procedure ដោយកំណត់

```
ALTER PROC spInsertItem
    @itemName VARCHAR(50),
```

```
@categoryId INT,
@brandId INT,
@itemId INT OUTPUT
AS
BEGIN
    INSERT Items VALUES (@itemName, @categoryId, @brandId)
    SET @itemId = @@IDENTITY
END
```

#### ៤. ការលុប Store Procedure

ដើម្បីលុប Store Procedure ដែលមានក្នុង Database គេប្រើប្រាស់ Keyword DROP PROC។

ឧទាហរណ៍៖ ការលុប spInsertItem Procedure ពី Database

```
DROP PROC spInsertItem
```

កំណត់សម្គាល់ខ្លះ៖



## មេរៀនទី៨

## User Defined Function (UDF)

## ១. និយមន័យ

User Define Function (UDF) ឬហៅដោយខ្លីថា Function គឺជា Object មានផ្ទុក SQL Statements ហើយដែលអាច Execute ឬ Run បាន។ ជាធម្មតា ការដាក់ឈ្មោះរបស់ Function ត្រូវផ្ដើមដោយ *fn* ។ Function ត្រូវបានគេប្រើដើម្បី Return តម្លៃ ខណៈហៅប្រើ។ Function អាច Return តម្លៃជា Scalar Value ជាតម្លៃតែមួយ ឬក៏ជា Table Value ជាតម្លៃចេញជា Table។ Function មិនអាច Run Action Query ដូចជា INSERT, UPDATE, DELETE ទៅលើ Database ដូច Procedure បានឡើយ។ UDF អាចមាន Input Parameter មួយ ឬច្រើន តែមិនមាន Output Parameter ទេ។

## ២. ការកំណត់ User Defined Function

គេអាចកំណត់បង្កើត UDF ដោយប្រើប្រាស់ Keyword CREATE FUNCTION ឧទាហរណ៍ទី១៖ បង្កើត Function ដើម្បី ត្រឡប់តម្លៃឈ្មោះ Sale តាមរយៈ SaleId

```
CREATE FUNCTION fnGetSaleName(@saleId INT)
RETURNS VARCHAR(50)
AS
BEGIN

    DECLARE @saleName VARCHAR(50);
    SELECT @saleName = CONCAT(FirstName, ' ', LastName) FROM Sales
    WHERE SaleId = @saleId

    RETURN @saleName

END
```

គេអាចហៅឬ Execute UDF ដោយផ្ដើមពីឈ្មោះ Schema បន្ទាប់មក ឈ្មោះរបស់ UDF

```
SELECT DBO.fnGetSaleName(2)
```

(No column name)	
1	Sok Reaksa

លំហាត់ទី៨.១៖ បង្កើត UDF fnGetItemName មាន Parameter @itemId ដើម្បី Return តម្លៃ ItemName

ឧទាហរណ៍ទី២៖ បង្កើត Function fnCountInvoice ដើម្បី Return តម្លៃបែប Aggregation ដោយ រាប់ Invoices តាម Date និង Sale

```
CREATE FUNCTION fnCountInvoices(@date Date, @saleId Int)
RETURNS INT
AS
BEGIN
```

```

DECLARE @count AS INT;
SELECT @count = COUNT(*) FROM Invoices
WHERE InvoiceDate = @date AND SaleId = @saleId

RETURN @count;

```

END

ការហៅ fnCountInvoice

```
SELECT dbo.fnCountInvoices('2021-11-1', 4) AS Invoice_Amount
```

	Invoice_Amount
1	1

**លំហាត់ទី៨.២៖** បង្កើត UDF fnCheckInStock (@code) ដើម្បីដឹងថាតើ Product ដែលមាន @code មាននៅសល់ប៉ុន្មានក្នុង Stock

### ៣. ការកែប្រែ User Defined Function

ដើម្បីកែប្រែ User Defined Function គេប្រើប្រាស់ Keyword ALTER FUNCTION

ឧទាហរណ៍៖ កែសម្រួល fnCountInvoices ដោយជំនួសតែ Parameter មួយគឺ @date ហើយ Return តម្លៃចំនួន Invoices ដែលមាន InvoiceDate ដូចនឹង @date

```

ALTER FUNCTION fnCountInvoices(@date Date)
RETURNS INT
AS
BEGIN

```

```

    DECLARE @count AS INT;
    SELECT @count = COUNT(*) FROM Invoices
    WHERE InvoiceDate = @date

    RETURN @count;

```

END

```
SELECT dbo.fnCountInvoices('2021-11-1') AS Invoice_Amount
```

### ៤. ការលុប User Defined Function

ដើម្បីលុប User Defined Function គេប្រើប្រាស់ Keyword DROP FUNCTION

ឧទាហរណ៍៖ ការលុប UDF fnCountInvoices

```
DROP FUNCTION fnCountInvoices
```



## ជំពូកទី៣

## C Sharp Database Programming ជាមួយ ADO.NET

## មេរៀនទី១

## ADO.NET Architecture

ADO.NET មានពាក្យពេញថា *ActiveX Data Objects .NET* គឺជាកញ្ចប់កូដកម្មវិធីរបស់ .NET Framework ដែលអាចឱ្យគេ Access Data ពី Database បាន។ ADO.NET មានផ្ទុក Classes ពាក់ព័ន្ធសម្រាប់ឱ្យគេអាចបង្កើតកម្មវិធី Database Application ជាមួយនឹងភាសា C# ក៏ដូចជាភាសាផ្សេងទៀតរបស់ .NET ។

## ១. ADO.NET Provider

ADO.NET Providers ផ្តល់នូវ Classes ដែលប្រើសម្រាប់ភ្ជាប់ (Connecting) និងធ្វើការផ្ទាល់ជាមួយ Database។ Classes គោលសំខាន់ៗសម្រាប់បង្កើត Objects មានបួនប្រភេទខុសគ្នា។

Object	ការប្រើប្រាស់
Connection	បញ្ជាក់ការភ្ជាប់ទៅកាន់ Database
Command	បញ្ជូនការលើ SQL statement ឬ Store Procedure ក្នុងការExecute នឹង Database
Data Reader	ទទួល Data បែប read-only ពីក្នុង Database
Data Adapter	ផ្តល់ជាផ្នែករវាង Command ជាមួយ Connection Object និង Dataset Object

ADO.NET Provider មានច្រើនប្រភេទដូចជា SqlClient, OLE DB, ODBC, MySQL ... ជាដើម ដោយអាស្រ័យទៅលើប្រភេទ Database។ សម្រាប់ Microsoft SQL Server គឺគេប្រើ SqlClient ។ Classes គោលទាំងបួនខាងមានឈ្មោះខាងដើម យោងតាមប្រភេទ Provider។

SqlClient Provider, Classes គោលមានឈ្មោះខាងដើមថា “*Sql*” ៖

- SqlConnection
- SqlCommand
- SqlDataReader
- SqlDataAdapter

ដើម្បីប្រើប្រាស់ គេត្រូវតែចែង using statement ទៅ namespace ដែលមានផ្ទុកនូវ Classes ទាំងនោះ។ ឧទាហរណ៍៖ using System.Data.SqlClient;

## SqlConnection Class

Property	ការពិពណ៌នា
ConnectionString	ផ្ទុកព័ត៌មានដែលអាចភ្ជាប់ទៅកាន់ SQL Server Database មានដូចជា Server Name, Database Name និង Login Information
Method	ការពិពណ៌នា
Open	បើក Connection
Close	បិទ Connection

ឧទាហរណ៍៖ ការភ្ជាប់ទៅកាន់ Database

```
SqlConnection connection = new SqlConnection();
connection.ConnectionString = @"Data Source=localhost\SqLExpress;
Initial Catalog=PhoneStore;Integrated Security=True";
```

## SqlCommand Class

Property	ការពិពណ៌នា
Connection	SqlConnection Object ដែលភ្ជាប់ទៅ Database
CommandText	Text ជា SQL Statement ឬ Store Procedure
CommandType	Enumerator សម្រាប់កំណត់ប្រភេទ Command ជា SQL Statement (Text) ឬក៏ឈ្មោះរបស់ store procedure (StoredProcedure)
Parameters	បណ្តុំនៃ Parameters នៃ Command Obejct
Method	ការពិពណ៌នា
ExecuteReader	Exectute Query និងត្រឡប់តម្លៃជា SqlDataReader Object
ExecuteNonQuery	Execute Command និងត្រឡប់តម្លៃជា Integer តំណាងឲ្យចំនួនជួរដែលប្រព្រឹត្តទៅ
ExecuteScalar	Execute Query មួយ និងត្រឡប់តម្លៃរបស់ Column និង Row ដំបូងនៃ Query

ឧទាហរណ៍៖ ការប្រើ SqlCommand ដើម្បី Query Customers Table

```
SqlCommand command = new SqlCommand();
command.Connection = connection;
command.CommandText = "SELECT * FROM Customers";
command.CommandType = CommandType.Text;

// Open Connection
connection.Open();
```

```
SqlDataReader dataReader = command.ExecuteReader();

// Close Connection
connection.Close();
```

### SqlDataReader Class

Property	ការពិពណ៌នា
<i>Indexer</i>	ចូលទាញទិន្នន័យ Column ដោយ Index ឬឈ្មោះ Column ក្នុង Row បច្ចុប្បន្ន
FieldCount	ចំនួន Columns ក្នុង Row បច្ចុប្បន្ន
Method	ការពិពណ៌នា
Read	អាន Row បន្ទាប់ និងត្រឡប់តម្លៃ True បើមាន Rows ទៀត តែបើគ្មាន ត្រឡប់តម្លៃ False
Close	បិទ Data Reader

ឧទាហរណ៍៖ ការប្រើប្រាស់ Data Reader

```
connection.Open();

SqlDataReader dataReader = command.ExecuteReader();
while (dataReader.Read())
{
    Console.WriteLine($" CustomerId: {dataReader["CustomerId"]}");
};
    Console.WriteLine($" FirstName: {dataReader["FirstName"]}");
    Console.WriteLine($" LastName: {dataReader["LastName"]}");
    Console.WriteLine($" Phone: {dataReader["Phone"]}");
    Console.WriteLine("-----");
}

// Close Connection
connection.Close();
```

## SqlDataAdapter Class

Property	ការពិពណ៌នា
SelectCommand	SqlCommand Object តាងឱ្យ Select Statement ឬ Procedure សម្រាប់ Query Database
DeleteCommand	SqlCommand Object តាងឱ្យ Delete Statement ឬ Procedure សម្រាប់ Delete Row ពី Database
InsertCommand	SqlCommand Object តាងឱ្យ Insert Statement ឬ Procedure សម្រាប់ថែម Row ទៅ Database
UpdateCommand	SqlCommand Object តាងឱ្យ Update Statement ឬ Procedure សម្រាប់កែសម្រួល Row ក្នុង Database
Method	ការពិពណ៌នា
Fill	Execute SelectCommand Property និងផ្ទុកលទ្ធផលក្នុង Data Set Object
Update	Execute DelectCommand, InsertCommand និង UpdateCommand Properties រាល់ row ដែលបាន លុប ថែម និង កែប្រែ ក្នុង Data Set Object

ឧទាហរណ៍៖ ការប្រើប្រាស់ SQL Data Adapter សម្រាប់ Query Data ទៅ DataSet

```

SqlDataAdapter adapter = new SqlDataAdapter();
adapter.SelectCommand = new SqlCommand();
adapter.SelectCommand.CommandText = "SELECT CustomerId, FirstName
, LastName, Phone, Address FROM Customers";
adapter.SelectCommand.Connection = connection;

// Delete Command
// Insert Command
// Update Command

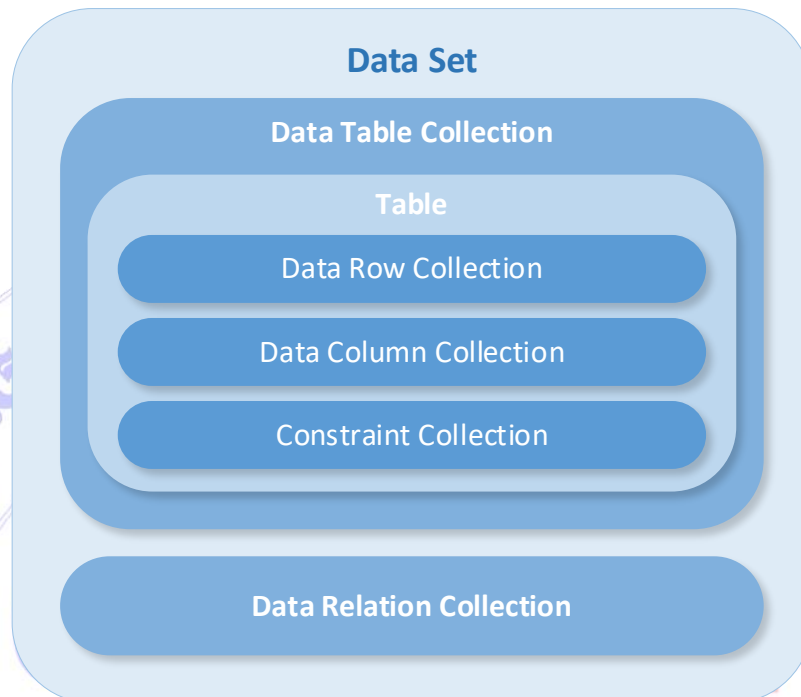
DataSet dataSet = new DataSet();
adapter.Fill(dataSet);
    
```

## ២. ADO.NET Data Set

Data Set គឺជា Class បង្កើត Object ឯករាជ្យមួយដែលគេហៅថា Disconnected Data Object សម្រាប់ធ្វើការជាមួយនឹង Data ស្ថិតក្នុង Database។ Data Set object មានផ្ទុកបណ្តុំនៃ Data Table និង Data Relation object។ Data Table Object មានផ្ទុកបណ្តុំ Data Column



Data Row និង Constraint object ។ Data Relation object សម្រាប់កំណត់ពីទំនាក់ទំនង Table ក្នុង Data Set ។



រូបភាពបង្ហាញពី រចនាសម្ព័ន្ធរបស់ ADO.NET Data Set

## DataSet Class

Property	ការពិពណ៌នា
DataSetName	ឈ្មោះរបស់ Data Set
Tables	បណ្តុំនៃ Data Table objects
Relations	បណ្តុំនៃ Data Relation objects

## DataTable Class

Property	ការពិពណ៌នា
TableName	ឈ្មោះរបស់ Table
Columns	បណ្តុំនៃ Data Column objects
Rows	បណ្តុំនៃ Data Row objects
Constraints	បណ្តុំនៃ Constraint objects
Method	ការពិពណ៌នា
NewRow	បង្កើត Row មួយថ្មីក្នុង Table



**DataColumn Class**

Property	ការពិពណ៌នា
ColumnName	ឈ្មោះរបស់ Column
AllowDBNull	សម្រាប់កំណត់ថាតើ Columns មានតម្លៃ Null បានអត់
AutoIncrement	សម្រាប់កំណត់ថាតើមាន auto-increment នៅលើ Column ឬទេ

**DataRow Class**

Property	ការពិពណ៌នា
<i>Indexer</i>	ចូលប្រើប្រាស់ Column ជាក់លាក់ក្នុង Row
Method	ការពិពណ៌នា
Delete	លុប Row
IsNull	ពិនិត្យថាតើ Column នៃ Row មានផ្ទុកតម្លៃ Null ឬអត់

## មេរៀនទី២

## សំណង់កម្មវិធីដោយប្រើប្រាស់ Data Set

កាលណាគេសង់កម្មវិធីដោយប្រើប្រាស់ Data Set Object គេបែងចែក Application ជាពីរផ្នែក គឺ ផ្នែកទី១ Presentation layer ដែលមាន User Interface ក៏អាចមានដែរ Data Set Class Object ។ ផ្នែកទី២ Database layer ដែលមាន database និង database classes object សម្រាប់ទទួលទិន្នន័យ និងកែប្រែទិន្នន័យពី database

## ១. ការសរសេរកូដបង្កើតកម្មវិធីជាមួយ Data Set

ឧទាហរណ៍ទី១៖ បង្កើតកម្មវិធី CustomersControl (Windows Form App) សម្រាប់ធ្វើការបង្ហាញរាល់ Customers ទាំងអស់មកបង្ហាញលើ Form ។

កម្មវិធីបង្ហាញតារាងដូចខាងក្រោម៖

	CustomerId	First Name	Last Name	Phone	Address
▶	1	General	Customer	0973927434	Battambang
	2	Si	Dalen	016878934	SiemReap
	3	Cham	Chhara	092897943	Pusat
	4	Ka	Kakada	012873947	Banteay Meanc...
	5	Kha	Sokhin	077432334	Battambang
	6	Veasna	SokLeap	098379427	SiemReap

ចំពោះកម្មវិធីនេះគឺគេប្រើ DataGridView ដើម្បីធ្វើការបង្ហាញទិន្នន័យ។

```
public partial class frmCustomersControl : Form
{
    string connectionString = @"Data Source=localhost\SqLExpress;
    Initial Catalog=PhoneStore; Integrated Security=True;";
    SqlConnection connection;
    SqlDataAdapter dataAdapter;
    DataSet dataSet;

    public frmCustomersControl()
    {
        InitializeComponent();

        connection = new SqlConnection(connectionString);
    }
}
```

```
SqlCommand selectCommand = new SqlCommand("SELECT * FROM Customers", connection);

dataAdapter = new SqlDataAdapter();
dataAdapter.SelectCommand = selectCommand;

dataSet = new DataSet();
dataAdapter.Fill(dataSet, "Customers");

dataGridView1.DataSource = dataSet.Tables["Customers"];
}
}
```

**លំហាត់ទី ២.១៖** បង្កើតកម្មវិធី ItemsControl ដោយប្រើប្រាស់ Windows Form App ដើម្បីបង្ហាញទិន្នន័យ Items ជាតារាង Table មកលើ DataGridView Control

**ឧទាហរណ៍ទី២៖** សង់កម្មវិធី ItemsControl ក្នុងទម្រង់ CRUD ពី Database ដោយប្រើប្រាស់ Data Set object



The screenshot shows a Windows Form application titled "Form2". The form has a title bar with standard Windows window controls. Below the title bar, there is a search bar with a "Search" button. The main area of the form contains a table with 4 columns: ID, Item Name, Brand, and Category. The table has 12 rows of data. To the left of the table, there are input fields for ID, Item Name, Brand, and Category, along with buttons for "Cancel", "New", "Delete", and "Save".

ID	Item Name	Brand	Category
1	iPhone 8 Plus	Samsung	Tablet
2	iPhone X	Apple	Mobile Phone
3	iPhone XS	Apple	Mobile Phone
4	iPhone 11 Pro	Apple	Mobile Phone
5	iPhone 12	Apple	Mobile Phone
6	iPhone 12 Pro	Apple	Mobile Phone
7	iPhone 12 Pro Max	Apple	Mobile Phone
8	iPhone 13	Apple	Mobile Phone
9	iPad Pro 11(2020)	Apple	Tablet
10	iPad Air 11(2020)	Apple	Tablet
11	iPad Mini 11 (2019)	Apple	Tablet
12	iWatch Series 6	Apple	Watch

## Code ក្នុង Form Class

- ដំណាក់កំណត់ផ្នែកពាក់ព័ន្ធ

```
public partial class Form2 : Form
{
    // Define Revelent Object
    DataSet dataSet = new DataSet();
    DataSet dataSetBackup = new DataSet();
    SqlConnection connection;
    SqlDataAdapter itemAdapter = new SqlDataAdapter();
```

```
SqlDataAdapter brandAdapter = new SqlDataAdapter();
SqlDataAdapter categoryAdapter = new SqlDataAdapter();

BindingSource itemBindingSource = new BindingSource();
BindingSource brandBindingSource = new BindingSource();
BindingSource categoryBindingSource = new BindingSource();

List<SqlCommand> cmdSelects = new List<SqlCommand>();
```

- កំណត់ក្នុង Form Contructor

```
public Form2()
{
    InitializeComponent();

    // Initialize Connection
    connection = new SqlConnection(@"Data Source=localhost\SqLExpress; Initial Catalog=PhoneStore; Integrated Security=True;");

    // Define SelectCommand List
    cmdSelects.Add(new SqlCommand(@"SELECT * FROM VItemList", connection));
    SqlCommand cmd = new SqlCommand("spFindItems", connection);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.Add("@Find", SqlDbType.VarChar, 50);
    cmd.Parameters[0].Direction = ParameterDirection.Input;
    cmd.Parameters[0].SourceVersion = DataRowVersion.Current;
    cmdSelects.Add(cmd);

    // define item Adapter Properties: Select, Update, Insert and Delete Command
    itemAdapter.SelectCommand = cmdSelects[0];

    itemAdapter.UpdateCommand = new SqlCommand(@"UPDATE Items SET ItemName=@ItemName,
BrandId=@BrandId, CategoryId=@CategoryId WHERE ItemId=@ItemId;
SELECT * FROM VItemList WHERE ItemId
=@ItemId", connection);
    itemAdapter.UpdateCommand.Parameters.Add("@ItemName", SqlDbType.VarChar, 50, "ItemName");
    itemAdapter.UpdateCommand.Parameters.Add("@BrandId", SqlDbType.Int, 0, "BrandId");
    ;
    itemAdapter.UpdateCommand.Parameters.Add("@CategoryId", SqlDbType.Int, 0, "CategoryId");
    itemAdapter.UpdateCommand.Parameters.Add("@ItemId", SqlDbType.Int, 0, "ItemId");

    itemAdapter.InsertCommand = new SqlCommand(@"INSERT Items (ItemName, BrandID, CategoryId) VALUES(@ItemName, @BrandId, @CategoryId);
SELECT * FROM VItemList WHERE ItemId
=@@IDENTITY", connection);
    itemAdapter.InsertCommand.Parameters.Add("@ItemName", SqlDbType.VarChar, 50, "ItemName");
    itemAdapter.InsertCommand.Parameters.Add("@BrandId", SqlDbType.Int, 0, "BrandId");
    ;
    itemAdapter.InsertCommand.Parameters.Add("@CategoryId", SqlDbType.Int, 0, "CategoryId");

    itemAdapter.DeleteCommand = new SqlCommand(@"DELETE Items WHERE ItemId=@ItemId", connection);
    itemAdapter.DeleteCommand.Parameters.Add("@ItemId", SqlDbType.Int, 0, "ItemId");

    // Mapping Name from Table to Items
    itemAdapter.TableMappings.Add("Table", "Items");
    itemAdapter.Fill(dataSet);
```

```
// Define Brand Adapter
brandAdapter.SelectCommand = new SqlCommand("SELECT * FROM Brands", connection);
brandAdapter.TableMappings.Add("Table", "Brands");
brandAdapter.Fill(dataSet);

// Define Category Adapter
categoryAdapter.SelectCommand = new SqlCommand("SELECT * FROM Categories", connection);
categoryAdapter.TableMappings.Add("Table", "Categories");
categoryAdapter.Fill(dataSet);

// Binding data set, each table to a specific DataSource of BindingSource
itemBindingSource.DataSource = dataSet;
itemBindingSource.DataMember = "Items";

brandBindingSource.DataSource = dataSet;
brandBindingSource.DataMember = "Brands";

categoryBindingSource.DataSource = dataSet;
categoryBindingSource.DataMember = "Categories";

// Backup Data Set object
dataSetBackup = dataSet.Copy();

}
```

- ប្រើប្រាស់ Row ថ្មី

```
private void btnNew_Click(object sender, EventArgs e)
{
    itemBindingSource.AddNew();
}
```

- ក្នុងក្នុង Form\_Load

```
private void Form2_Load(object sender, EventArgs e)
{
    // apply data to Data Grid View object
    dataGridView1.DataSource = itemBindingSource;

    // binding Control Properties to binding source object
    txtId.DataBindings.Add(new Binding("Text", itemBindingSource, "ItemId"));
    txtItemName.DataBindings.Add(new Binding("Text", itemBindingSource, "ItemName"));

    // fill Source to ComboBox
    cbBrand.DataSource = brandBindingSource;
    cbBrand.DisplayMember = "BrandName";
    cbBrand.ValueMember = "BrandId";

    cbCategory.DataSource = categoryBindingSource;
    cbCategory.DisplayMember = "CategoryName";
    cbCategory.ValueMember = "CategoryId";

    // binding Combo Box Property ("SelectedValue") to BindingSource object
    cbBrand.DataBindings.Add(new Binding("SelectedValue", itemBindingSource, "BrandId"));
    cbCategory.DataBindings.Add(new Binding("SelectedValue", itemBindingSource, "CategoryId"));
}
```

- រក្សាទុក Data ក្នុង Data Set និង DataSource





```
private void btnSave_Click(object sender, EventArgs e)
{
    itemBindingSource.EndEdit();
    itemAdapter.Update(dataSet, "Items");

    dataSetBackup.Tables["Items"].Merge(dataSet.Tables["Items"]);
}
```

- លុប Current Row

```
private void btnDelete_Click(object sender, EventArgs e)
{
    itemBindingSource.RemoveCurrent();
}
```

- ត្រឡប់ទៅ Data ដើមមុនពេល Save

```
private void btnCancel_Click(object sender, EventArgs e)
{
    dataSet.Tables["Items"].RejectChanges();
}
```

- ស្វែងរក Data

```
private void btnSearch_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txtSearch.Text))
    {
        var table1 = dataSet.Tables["Items"];
        var table2 = dataSetBackup.Tables["Items"];
        if (table1.Rows.Count != table2.Rows.Count)
        {
            dataSet.Merge(dataSetBackup);
        }
    }
    else
    {
        var table = dataSet.Tables["Items"];
        SearchItems(table, txtSearch.Text);
    }
}
```

- Method SearchItems from Database(Data Source)

```
public void SearchItems(DataTable table, string find)
{
    cmdSelects[1].Parameters["@Find"].Value = find;
    itemAdapter.SelectCommand = cmdSelects[1];
    table.Clear();
    itemAdapter.Fill(table);
    Console.WriteLine();
}
```

- គ្រប់គ្រងលើការបញ្ជូន Search Button ខណៈចុច Enter Key លើ Search TextBox

```
private void txtSearch_KeyPress(object sender, KeyPressEventArgs e)
{
    if(e.KeyChar == (char)Keys.Enter)
    {
        e.Handled = true;
        btnSearch.PerformClick();
    }
}
```

```
}
    }
}
```

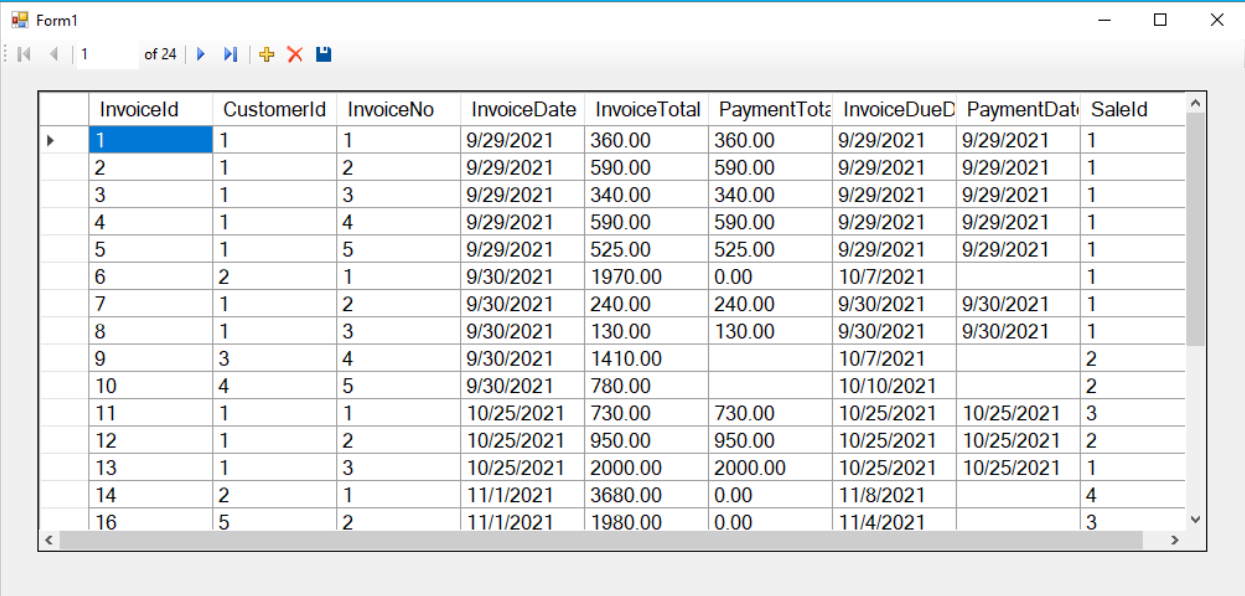
Binding Source Control ត្រូវបានគេប្រើដើម្បី Binding Data Source ទៅ Control ដទៃ។  
Data Set Object Method

- Copy() : make a copy of data set
- Merge() : merge other data set to current data set

**លំហាត់ទី២.២៖** សង់កម្មវិធី CustomersControl ជាទម្រង់ CRUD ដោយប្រើប្រាស់  
Data Set Object និង Data Binding Source Object

## ២. ការប្រើប្រាស់ Data Source Wizard បង្កើតកម្មវិធីជាមួយ Data Set

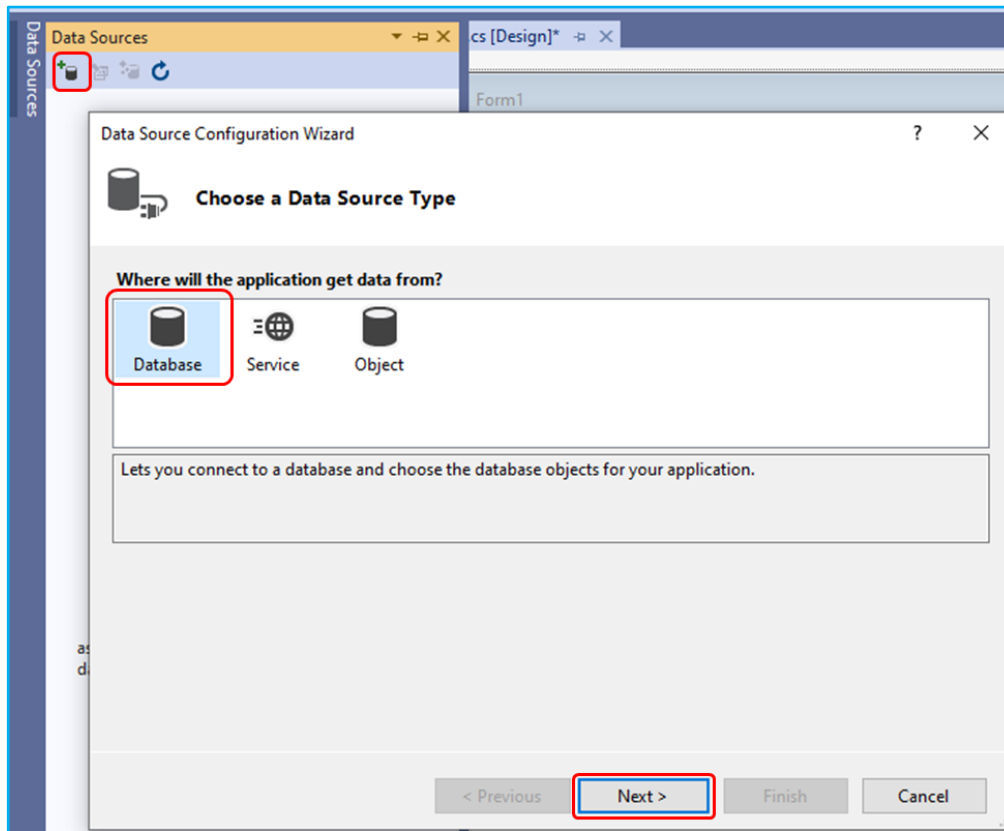
ឧទាហរណ៍ទី១៖ សង់កម្មវិធីបង្ហាញ Invoices ឈ្មោះថា InvoiceView ជាមួយ Data Source  
Data Set



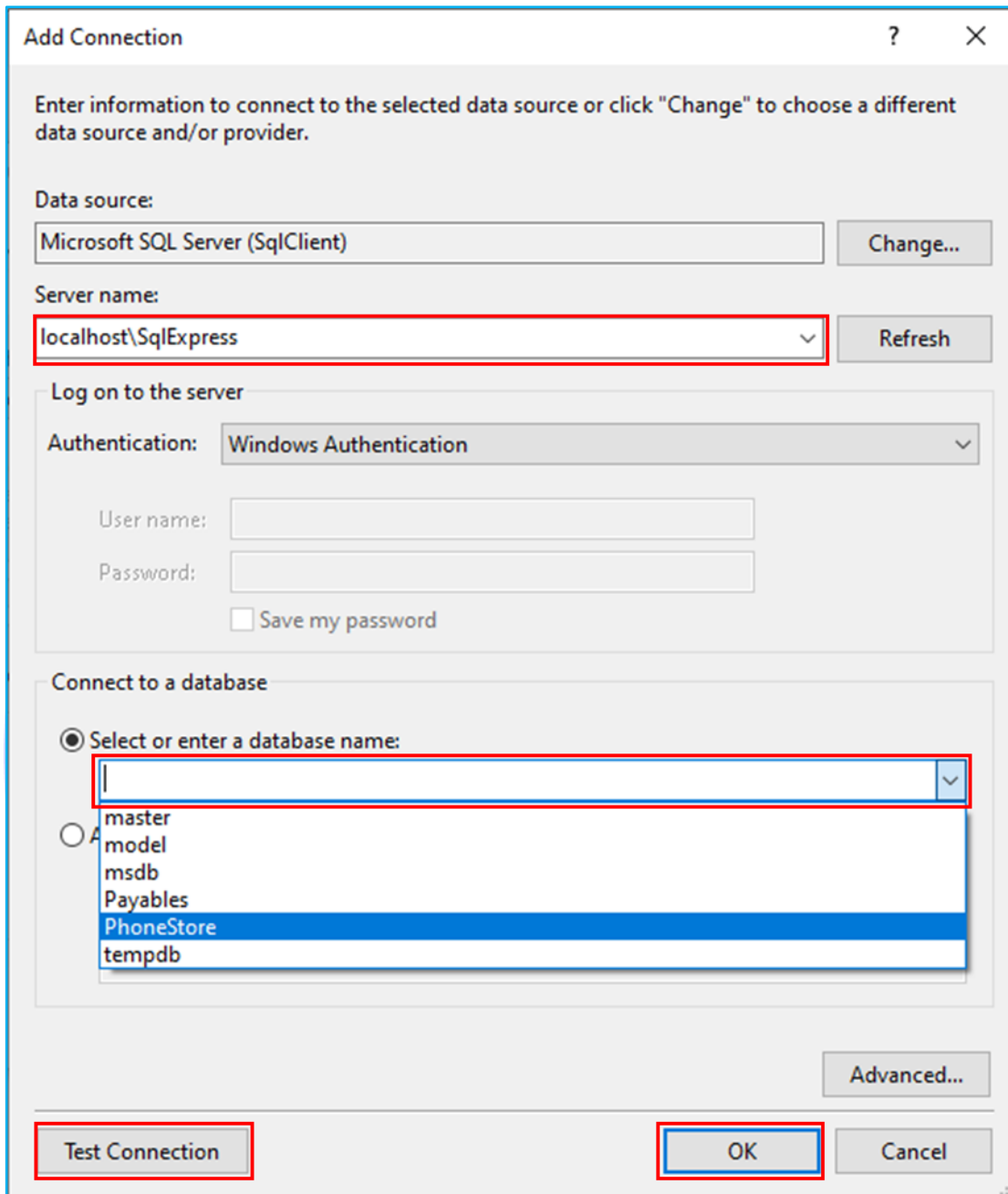
	InvoiceId	CustomerId	InvoiceNo	InvoiceDate	InvoiceTotal	PaymentTotal	InvoiceDueDate	PaymentDate	SaleId
▶	1	1	1	9/29/2021	360.00	360.00	9/29/2021	9/29/2021	1
	2	1	2	9/29/2021	590.00	590.00	9/29/2021	9/29/2021	1
	3	1	3	9/29/2021	340.00	340.00	9/29/2021	9/29/2021	1
	4	1	4	9/29/2021	590.00	590.00	9/29/2021	9/29/2021	1
	5	1	5	9/29/2021	525.00	525.00	9/29/2021	9/29/2021	1
	6	2	1	9/30/2021	1970.00	0.00	10/7/2021		1
	7	1	2	9/30/2021	240.00	240.00	9/30/2021	9/30/2021	1
	8	1	3	9/30/2021	130.00	130.00	9/30/2021	9/30/2021	1
	9	3	4	9/30/2021	1410.00		10/7/2021		2
	10	4	5	9/30/2021	780.00		10/10/2021		2
	11	1	1	10/25/2021	730.00	730.00	10/25/2021	10/25/2021	3
	12	1	2	10/25/2021	950.00	950.00	10/25/2021	10/25/2021	2
	13	1	3	10/25/2021	2000.00	2000.00	10/25/2021	10/25/2021	1
	14	2	1	11/1/2021	3680.00	0.00	11/8/2021		4
	16	5	2	11/1/2021	1980.00	0.00	11/4/2021		3

ការកំណត់ Data source Data set តាម Data source Wizard

Data Source -> Add New Data source -> Database -> Next -> Dataset -> Next -> New Connection->



បំពេញ Server name (ឧ. Localhost\SqlExpress), ជ្រើស database name ចុច Test connection ដើម្បីបញ្ជាក់ពីស្ថានភាពការភ្ជាប់ database និងចុច Ok ដើម្បីកំណត់យក Connection



**Add Connection**

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient) Change...

Server name: localhost\SqlExpress Refresh

Log on to the server

Authentication: Windows Authentication

User name:

Password:

☐ Save my password

Connect to a database

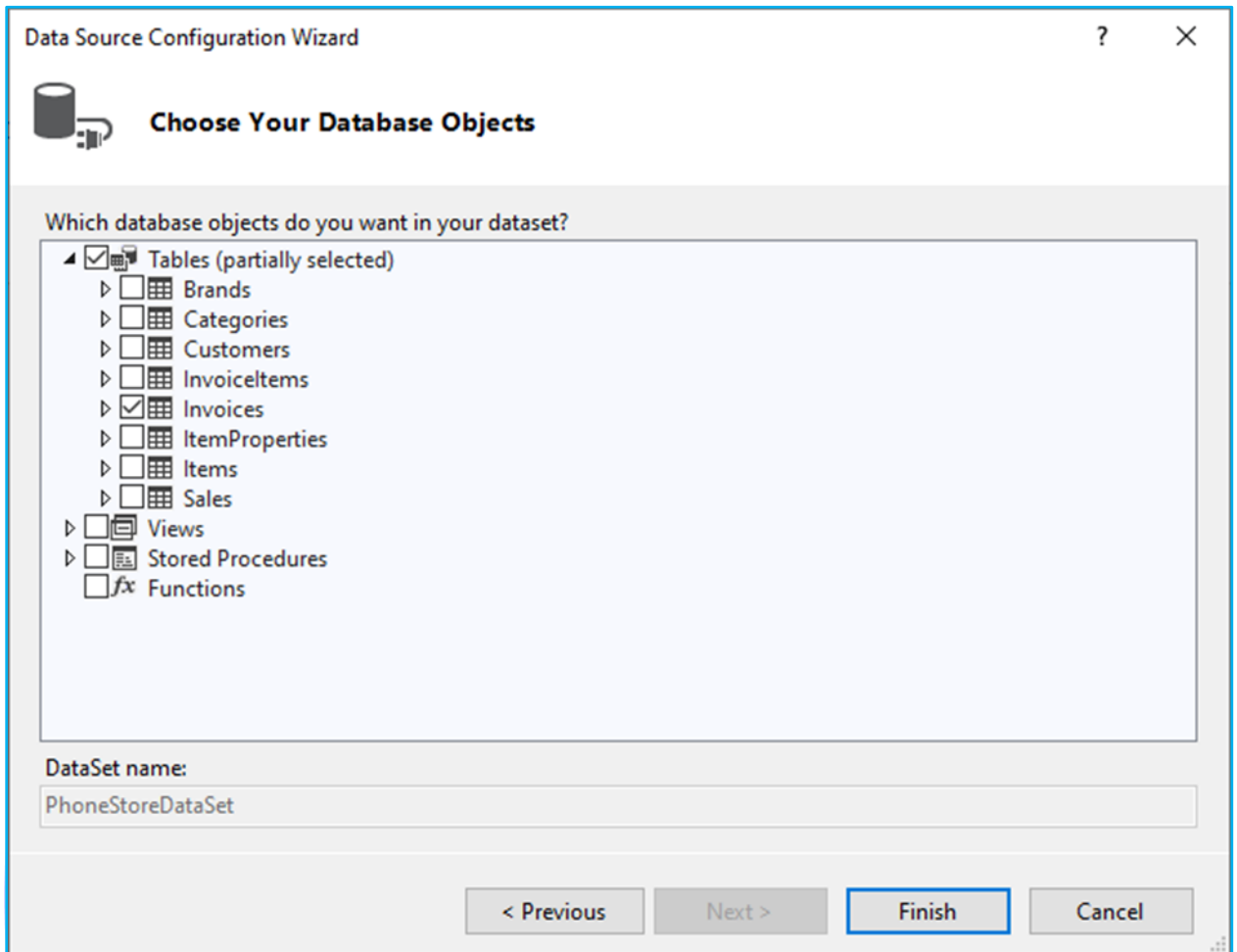
☒ Select or enter a database name:

☐ Advanced...

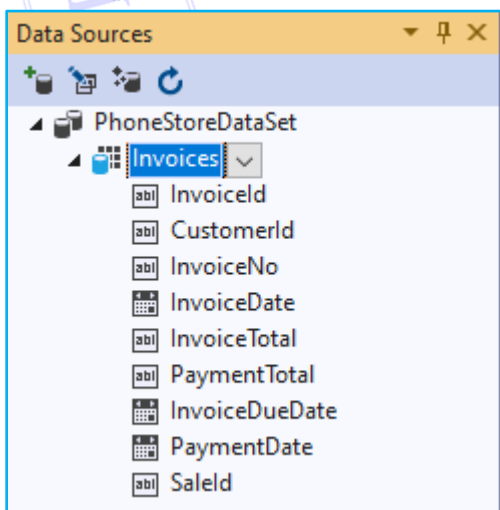
master  
model  
msdb  
Payables  
PhoneStore  
tempdb

Test Connection OK Cancel

ចុច Next -> Next -> ជ្រើសរើស Database Object -> Finish



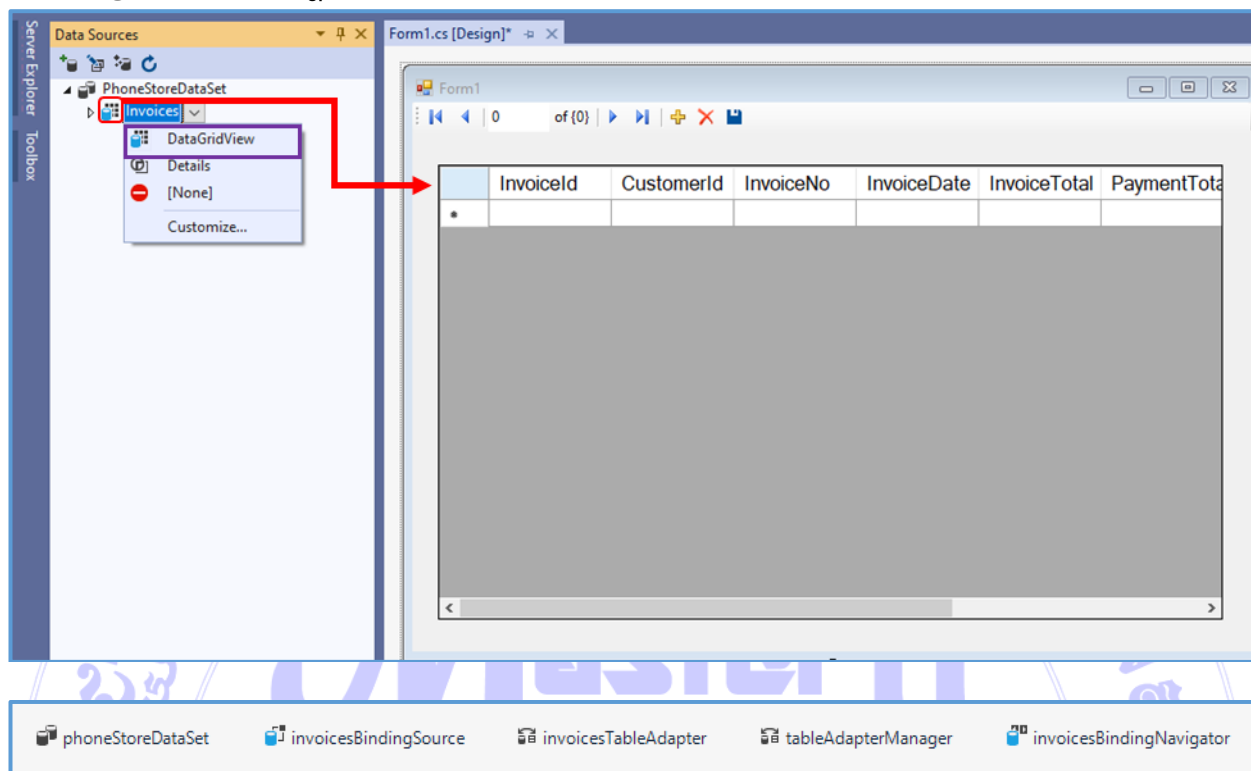
គេនឹងទទួលបាន Data Source Data Set ដែលមាន Items ផ្សេងៗ





អូស Item ក្នុង Data source Data set ចូលមកក្នុង Form Design

គេអាចជ្រើសរើសបែបបង្ហាញជា Data Grid View ឬ Details



Property	ការពិពណ៌នា
BindingNavigator	Toolbar សម្រាប់ Navigate, Add, Delete, Save
BindingSource object	ភ្ជាប់រវាង Data source ទៅនឹង Control របស់ Form
DataSet object	ក្លែបតាំងពី tables, views, procedures និង function
TableAdapter object	ផ្ទុក data adapter, connection និង command សម្រាប់អានទិន្នន័យពី និងសរសេរទិន្នន័យទៅឲ្យ table ណាមួយក្នុង database
TableAdapterManager object	សម្រាប់សរសេរទិន្នន័យលើ Tables ទៅ Databaseដោយយោងទៅតាម Relation រវាង Tables។

កូដក្នុង Form Class

```
private void invoicesBindingNavigatorSaveItem_Click(object sender, EventArgs e)
{
    this.Validate();
    this.invoicesBindingSource.EndEdit();
    this.tableAdapterManager.UpdateAll(this.phoneStoreDataSet);
}

private void Form1_Load(object sender, EventArgs e)
{
    // TODO: This line of code loads data into the 'phoneStoreDataSet.Invoices' table
    . You can move, or remove it, as needed.
```

```
this.invoicesTableAdapter.Fill(this.phoneStoreDataSet.Invoices);  
}
```

**លំហាត់ទី២.៣៖** ចូរបន្ថែម InvoicesView ខាងលើ ដោយធ្វើការកំណត់បន្ថែមបង្ហាញ ឈ្មោះ របស់ Customer(First + Last Name) ព្រមទាំងរបស់ Sale(First + Last Name) មកលើ Data Grid View Control

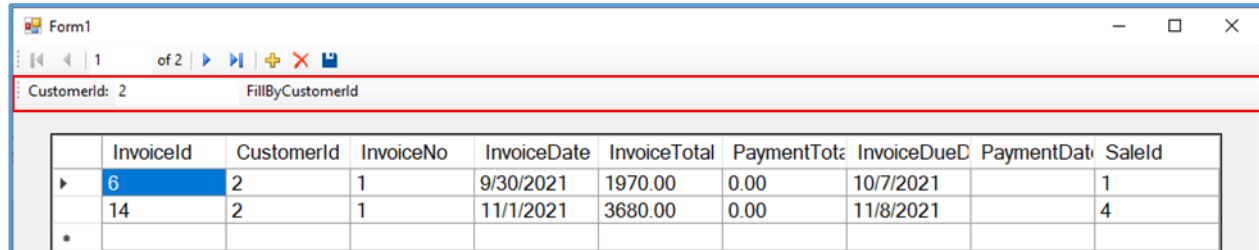
**លំហាត់ទី២.៤៖** សង់កម្មវិធី ItemsView ដោយបង្ហាញ Item list មកលើ DataGridView

តាមរយៈការអូស item ពីData Source data set item

កំណត់សម្គាល់ខ្ញុំ



## ឧទាហរណ៍ទី២៖ ការប្រើប្រាស់ Query Parameters



InvoiceId	CustomerId	InvoiceNo	InvoiceDate	InvoiceTotal	PaymentTotal	InvoiceDueDate	PaymentDate	SaleId
6	2	1	9/30/2021	1970.00	0.00	10/7/2021		1
14	2	1	11/1/2021	3680.00	0.00	11/8/2021		4

### ការកំណត់ Query Parameter

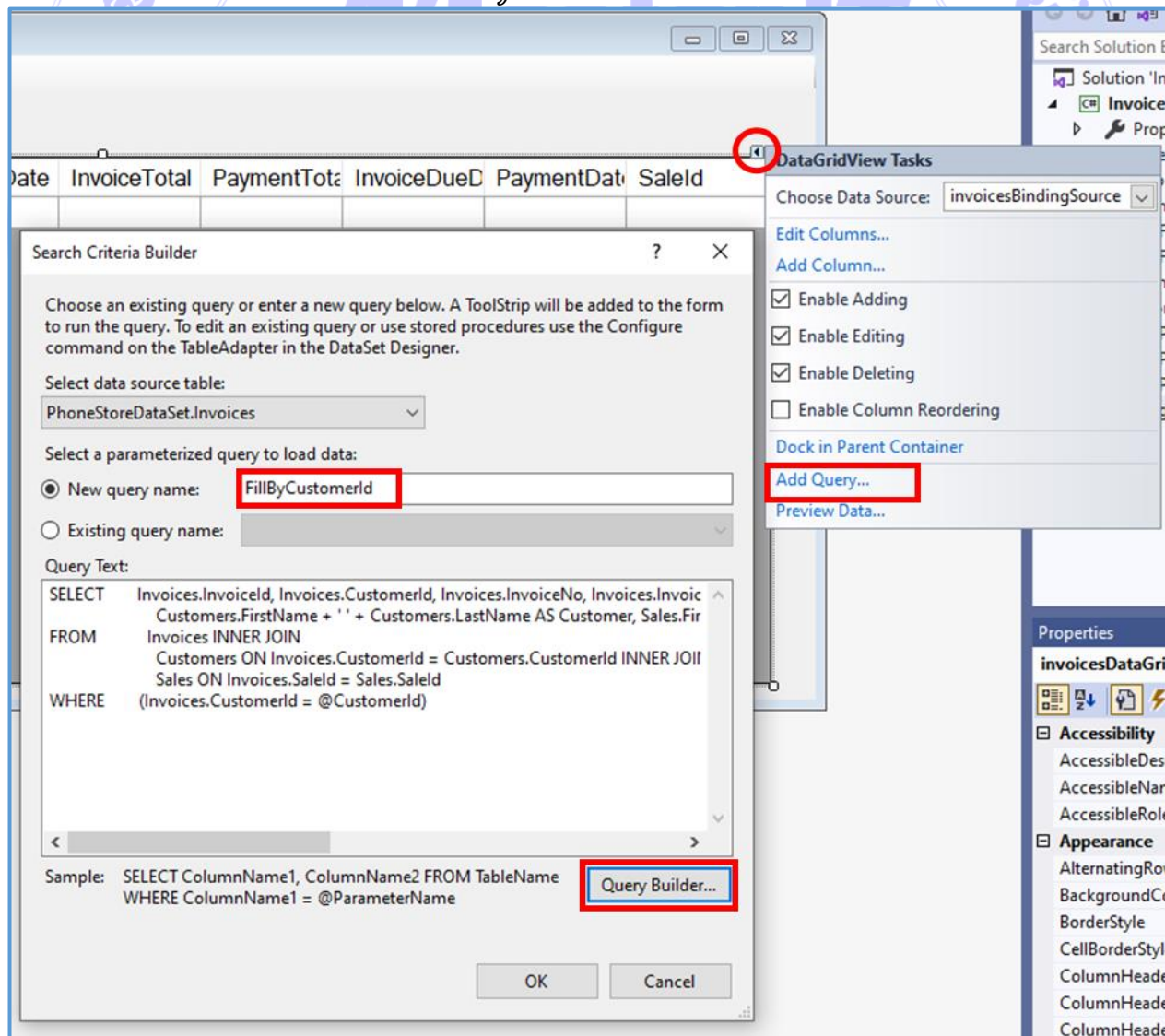
ចុចលើ smart tag របស់ Control មួយណាក៏បាន -> Add Query ->

ដាក់ឈ្មោះឲ្យ Query

កំណត់ឈ្មោះ Parameter ដោយផ្ដើម “@”

គេអាចប្រើប្រាស់ Query Builder ដើម្បីរៀបចំ Query

ក្រោយចុច Ok ToolStrip Control នឹងត្រូវបានបង្កើតដោយស្វ័យប្រវត្ត



Search Criteria Builder

Choose an existing query or enter a new query below. A ToolStrip will be added to the form to run the query. To edit an existing query or use stored procedures use the Configure command on the TableAdapter in the DataSet Designer.

Select data source table: PhoneStoreDataSet.Invoices

Select a parameterized query to load data:

☒ New query name: FillByCustomerId

☐ Existing query name:

Query Text:

```
SELECT Invoices.InvoiceId, Invoices.CustomerId, Invoices.InvoiceNo, Invoices.InvoiceTotal, Invoices.InvoiceDueDate, Invoices.PaymentTotal, Invoices.PaymentDate, Invoices.SaleId
FROM Invoices INNER JOIN Customers ON Invoices.CustomerId = Customers.CustomerId INNER JOIN Sales ON Invoices.SaleId = Sales.SaleId
WHERE (Invoices.CustomerId = @CustomerId)
```

Sample: SELECT ColumnName1, ColumnName2 FROM TableName WHERE ColumnName1 = @ParameterName

Query Builder...

DataGridView Tasks

Choose Data Source: invoicesBindingSource

Edit Columns...

Add Column...

☒ Enable Adding

☒ Enable Editing

☒ Enable Deleting

☐ Enable Column Reordering

Dock in Parent Container

Add Query...

Preview Data...

ឧទាហរណ៍ទី៣៖ កំណត់ ItemsView ដោយបន្ថែមទម្រង់បង្ហាញ បែប Details

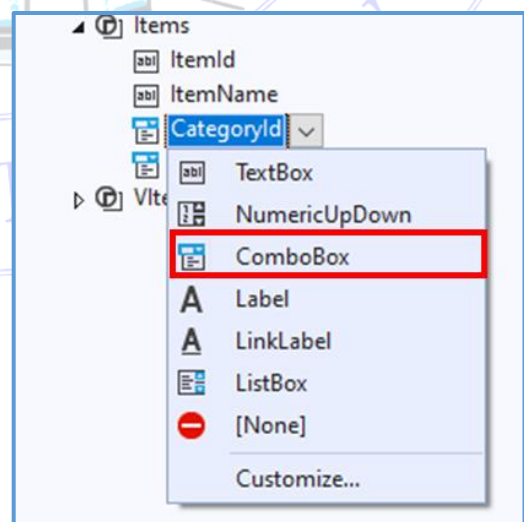
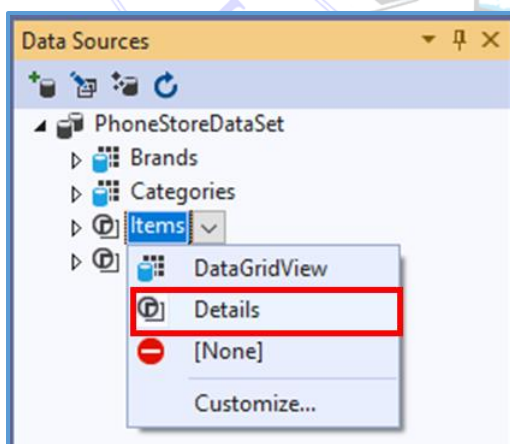
Form1

Item Id: 7 Category Id: Mobile Phone

Item Name: iPhone 12 Pro M Brand Id: Apple

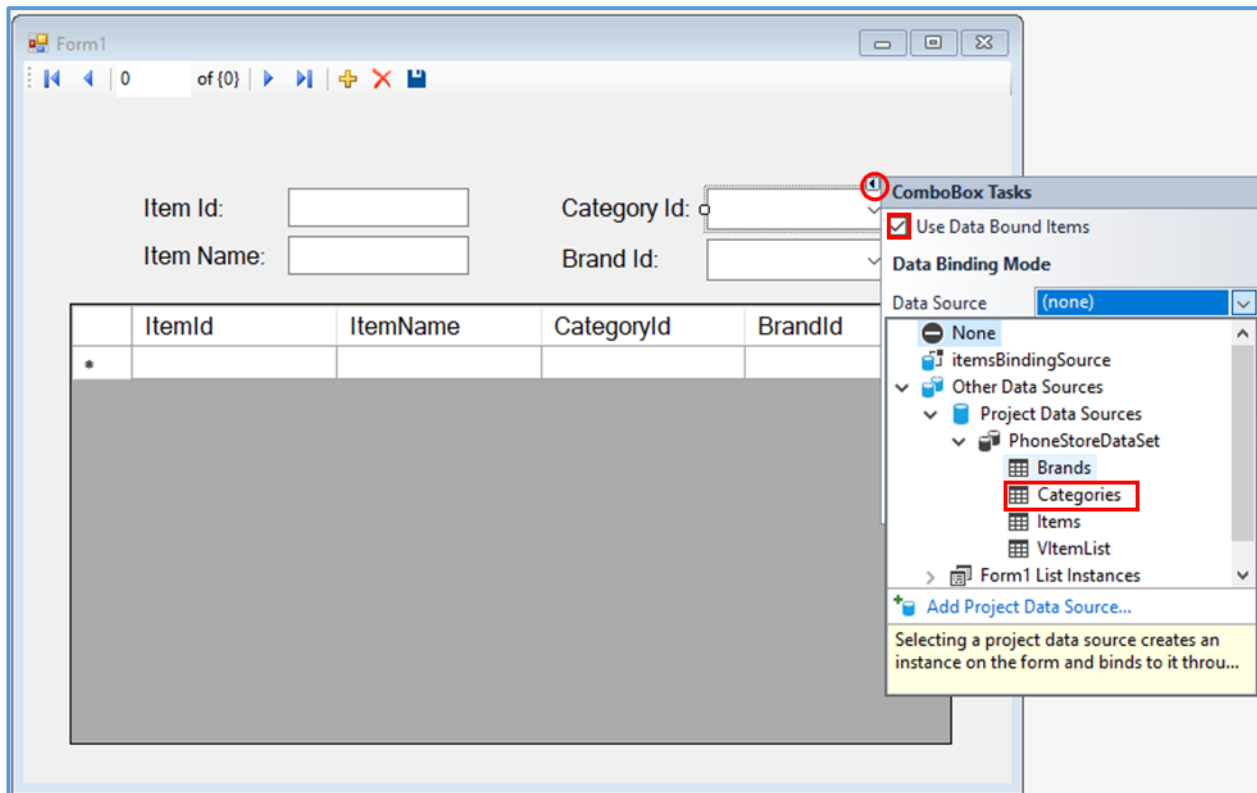
ItemId	ItemName	CategoryId	BrandId
1	iPhone 8 Plus	2	2
2	iPhone X	1	1
3	iPhone XS	1	1
4	iPhone 11 Pro	1	1
5	iPhone 12	1	1
6	iPhone 12 Pro	1	1
7	iPhone 12 Pro Max	1	1
8	iPhone 13	1	1
9	iPad Pro 11(20...	2	1
10	iPad Air 11(2020)	2	1
11	iPad Mini 11 (2...	2	1
12	iWatch Series 6	3	1

- ពន្លាត Items
- កំណត់ CategoryId និង BrandId ជាទម្រង់ Combo Box
- អូស Items ពី Data Set មកក្នុង Form Design



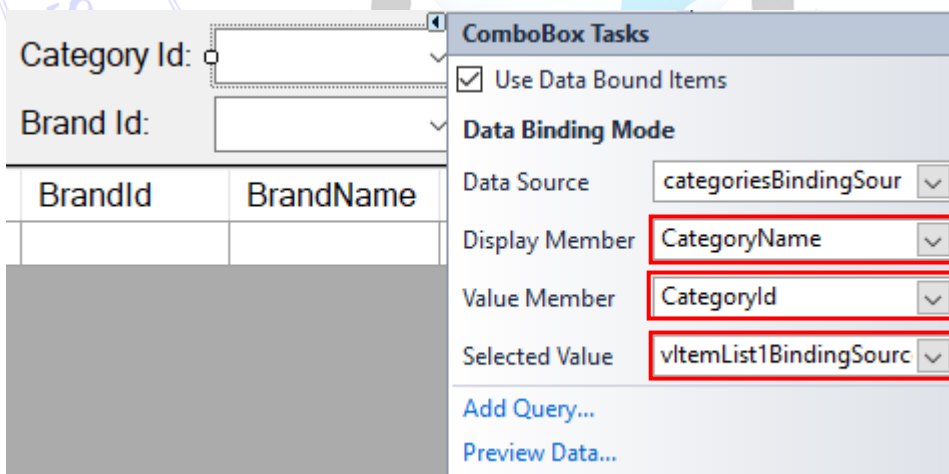
## នៅក្នុង Form Design

- Bind Brands and Categories in Data Set to Brand and Category Combo Box



## Data Binding Mode

- Data Source
- Display Member
- Value Member
- Selected Value





### ៣. ការកំណត់ Master and Detail Form

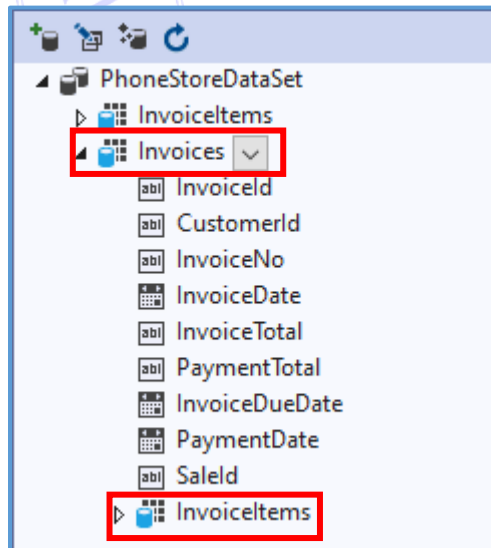
ឧទាហរណ៍៖ ការកំណត់ Master (Invoices) និង Details(InvoiceItems) ក្នុង Form ដោយប្រើប្រាស់ Data Grid View Control

InvoiceId	CustomerId	InvoiceNo	InvoiceDate	InvoiceTotal	PaymentTotal
1	1	1	9/29/2021	360.00	360.00
2	1	2	9/29/2021	590.00	590.00
3	1	3	9/29/2021	340.00	340.00
4	1	4	9/29/2021	590.00	590.00
5	1	5	9/29/2021	525.00	525.00
6	2	1	9/30/2021	3390.00	0.00
7	1	2	9/30/2021	240.00	240.00
8	1	3	9/30/2021	130.00	130.00
9	3	4	9/30/2021	2730.00	
10	4	5	9/30/2021	3600.00	
11	1	1	10/25/2021	730.00	730.00
12	1	2	10/25/2021	650.00	650.00

ItemNo	ItemCode	Quantity	Price	Sub Total
1	47	3	\$130.00	390.00
2	48	3	\$130.00	390.00
3	49	3	\$150.00	450.00
4	51	2	\$150.00	300.00
5	52	1	\$500.00	500.00
6	55	2	\$350.00	700.00

Total: **\$2,730.00**

- ក្រោយមាន Phone Store Data Set -> ពិន្ទុត Invoices -> អូស Invoices ចូល Form ដើម្បីបង្កើតជា Master View ។
- InvoiceItems ជា sub item របស់ Invoices ដោយសារ items ទាំងពីរមានទំនាក់ទំនងគ្នា។
- អូស Sub item: InvoiceItems នៃ Invoices ចូល Form



## ប្រូជិកបញ្ចប់វគ្គ

## គ្រប់គ្រងប្រាក់ខែបុគ្គលិក

គោលបំណងនៃប្រូជិកបញ្ចប់វគ្គនេះគឺចង់ឲ្យបងប្អូនទាំងអស់គ្នាចេះកំណត់កម្មវិធីគ្រប់គ្រងប្រាក់ខែបុគ្គលិក ដែលធ្វើជាគម្រោងនៃការកំណត់កម្មវិធីផ្សេងៗជាក់ស្តែងតាមការគួរ។ ដំបូងឡើយយើងត្រូវរៀបចំផ្នែករចនា Database ផ្នែករចនា User Interface និងចុងក្រោយផ្នែកសរសេរកូដភ្ជាប់ផ្នែកទាំងពីរនេះ។

### ១. ការរចនា Database

#### 1. ប្រភេទទិន្នន័យនៅក្នុង Ms SQL Server

ប្រភេទទិន្នន័យក្នុង MS SQL Server ត្រូវបានចែកជាបួនផ្នែក៖

- 1) String រក្សាទុកទិន្នន័យជាតួអក្សរ លេខ និងនិមិត្តសញ្ញា
- 2) Numeric រក្សាទុកទិន្នន័យជាលេខ លេខគត់ ឬលេខទស្សភាគ
- 3) Temporal (Date/Time) រក្សាទុកទិន្នន័យជាកាលបរិច្ឆេទ និងពេល
- 4) Other រក្សាទុកទិន្នន័យបែបអត្ថបទ ជារូបភាព ឬ XML ជាដើម

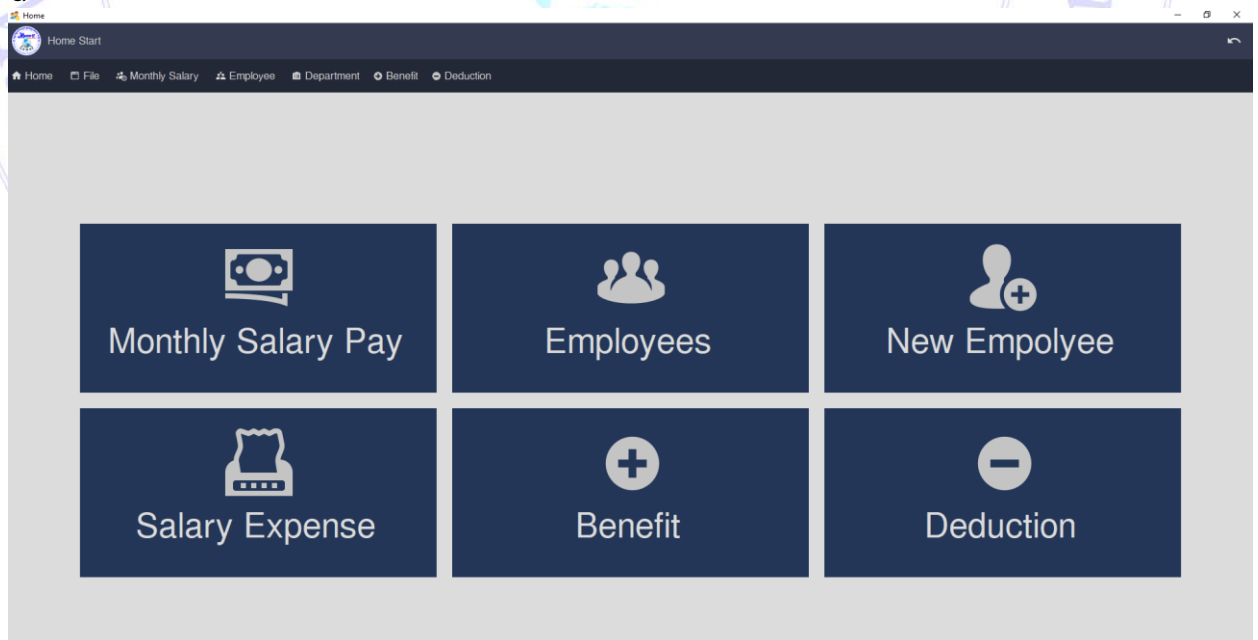
ប្រភេទ	Bytes	ការពិពណ៌នា
bigint	8	រក្សាទុកចំនួនគត់ចន្លោះពី $-2^{63}$ ដល់ $2^{63}-1$
int	4	រក្សាទុកចំនួនគត់ចន្លោះពី $-2^{31}$ ដល់ $2^{31}-1$
smallint	2	រក្សាទុកចំនួនគត់ចន្លោះពី $-2^{15}$ ដល់ $2^{15}-1$
tinyint	1	រក្សាទុកចំនួនគត់ចន្លោះពី 0 ដល់ 255
bit	1	រក្សាទុកចំនួនគត់ត្រឹមត្រូវ 0 ឬ 1
decimal[(p,[s])]	5-17	រក្សាទុកចំនួនទស្សភាគដែលមានចំនួនតួលេខជាក់លាក់(p) ព្រមទាំងបញ្ជាក់ពីចំនួនតួក្រោយកៀស(s)។ p: មានតម្លៃចន្លោះពី 1-38, តម្លៃដើមគឺ 18 s: មានតម្លៃចន្លោះពី 0-p, តម្លៃដើមគឺ 0
numeric[(p,[s])]	5-17	ដូចគ្នានឹង Decimal ដែរ
money	8	រក្សាទុកជាទឹកលុយដែលមានក្រោយកៀស៤ខ្ទង់ នៅចន្លោះតម្លៃ -922,337,203,685.5808 ដល់ 922,337,203,685.5807 [ decimal(19,4) ]
smallmoney	4	រក្សាទុកជាទឹកលុយដែលមានក្រោយកៀស៤ខ្ទង់ នៅចន្លោះតម្លៃ -214,748.3648 ដល់ 214,748.3647 [ decimal(10,4) ]
float	4, 8	រក្សាទុកចំនួនទស្សភាគចន្លោះពី $-1.79 \times 10^{308}$ ដល់ $1.79 \times 10^{308}$
real	4	រក្សាទុកចំនួនទស្សភាគចន្លោះពី $-3.4 \times 10^{38}$ ដល់ $3.4 \times 10^{38}$
char[(n)]	n	រក្សាទុកតម្លៃជាអក្សរដែលមានចំនួនតួចំនួន n តំណាងឲ្យចំនួនតួ

varchar(n)	n	រក្សាតម្លៃជាអក្សរដែលមានចំនួនតួមិនថេរ n តំណាងឲ្យចំនួនតួធំបំផុត
nchar(n)	2n	រក្សាតម្លៃជាអក្សរបែបអន្តរជាតិ (Unicode) ដែលមានចំនួនតួថេរ n តំណាងឲ្យចំនួនតួ
nvarchar(n)	2n	រក្សាតម្លៃជាអក្សរបែបអន្តរជាតិ (Unicode) ដែលមានចំនួនតួមិនថេរ n តំណាងឲ្យចំនួនតួធំបំផុត
datetime	8	រក្សាទិន្នន័យជាកាលនិងវេលាចន្លោះពី 1-1-1753 to 31-12-9999
smalldatetime	4	រក្សាទិន្នន័យជាកាលនិងវេលាចន្លោះពី 1-1-1900 to 31-12-2079
date	3	រក្សាទិន្នន័យតែកាលចន្លោះពី 1-1-0001 to 31-12-9999
time	3,5	រក្សាទិន្នន័យតែវេលាចន្លោះពីម៉ោង 00:00:00.0000000 ដល់ 23:59:59.9999999
datetime2	6,8	រក្សាទិន្នន័យតែកាលចន្លោះពី 1-1-0001 to 31-12-9999 និងវេលាចន្លោះពីម៉ោង 00:00:00.0000000 ដល់ 23:59:59.9999999

2. Data Modeling
3. Constraint
4. Normalization
5. Index

## ២. ការរចនា User Interface ដោយប្រើ Windows Form

ផ្ទាំង Dashborad



Form ព័ត៌មានបុគ្គលិកទូទៅ

EmployeeId	LastName	FirstName	Gender	Date of Birth	Joined Date	Contract Period	Department	Base Salary	Active	Edit	Detail
1	Bona	Kaka	Female	1/11/1997	6/30/2021	2	Accounting	200	<input type="checkbox"/>		
2	ChamrouenTo	PHP	Female	1/1/1997	6/29/2021	3	IT	300	<input checked="" type="checkbox"/>		
4	ប៊ុណ	សាលាផែន	Male	6/30/2021	6/30/2021	2	HR	300	<input type="checkbox"/>		
7	Kak	ប៊ុនី	Male	1/1/1999	6/30/2021	2	Accounting	300	<input checked="" type="checkbox"/>		
8	Theary	Kakada	Female	1/3/1999	7/1/2021	2	Administration	200	<input checked="" type="checkbox"/>		
9	ThearithY	XY	Male	1/1/1992	7/6/2021	2	Administration	500	<input checked="" type="checkbox"/>		
10	YaLin	Tiny	Female	8/17/1989	7/13/2021	2	Accounting	250	<input checked="" type="checkbox"/>		
11	Ury	Thay	Female	2/1/2000	7/12/2021	2	Administration	180	<input checked="" type="checkbox"/>		
12	ធី	ស៊ី	Female	7/13/2021	7/13/2021	2	Media	200	<input checked="" type="checkbox"/>		
15	BoTa	Lala	Female	11/25/1999	7/16/2021	2	IT	250	<input checked="" type="checkbox"/>		
16	Mala	Savandy	Male	7/16/2021	7/16/2021	2	Media	300	<input checked="" type="checkbox"/>		
17	May	Charly	Male	7/4/1999	7/16/2021	2	Accounting	350	<input checked="" type="checkbox"/>		
18	May	Chaitha	Male	7/16/2021	7/16/2021	3	Accounting	240	<input type="checkbox"/>		
19	Yira	YaTy	Male	7/16/2021	7/16/2021	3	Media	250	<input checked="" type="checkbox"/>		
20	Uthina	Serey	Male	7/16/2021	7/16/2021	3	Accounting	220	<input checked="" type="checkbox"/>		
21	ALy	Sa	Female	7/16/2021	7/16/2021	2	Accounting	222	<input checked="" type="checkbox"/>		
22	Thiya	Kaka	Male	2/8/2000	7/19/2021	3	IT	250	<input type="checkbox"/>		
1024	Bun	Bopha	Female	2/1/2000	7/21/2021	2	Online learning	200	<input checked="" type="checkbox"/>		

## Form បើកប្រាក់ខែបុគ្គលិក

Monthly Salary Pay

Salary Expense

Employee Salary Pay

EmployeeId

19

EmployeeName

Yira YaTy

Department

Media

BaseSalary

\$250.00

Salary

\$295.00

Choose Benefit

Outstanding Staff

Insert

Monthly Bonus

10

×

Annual Bonus

45

×

Choose Deduction

Uniform

Insert

Late

10

×

Note

សូមកុំភ្លេចឱ្យបុគ្គលិកទាំងអស់

Submit

Cancel

## ៣. ការសរសេរកូដអន្តរកម្ម

---



---



---



---

