Tính đóng gói trong C#

Encapsulation (Tính đóng gói) được định nghĩa là "tiến trình đóng gói một hoặc nhiều mục bên trong một gói logic hoặc vật lý". Tính đóng gói, trong phương pháp lập trình hướng đối tượng, ngăn cản việc truy cập tới chi tiết của trình trình triển khai (Implementation Detail).

Tính trừu tượng và tính đóng gói là hai đặc điểm có liên quan với nhau trong lập trình hướng đối tượng. Tính trừu tượng cho phép tạo các thông tin liên quan có thể nhìn thấy và tính đóng gói cho lập trình viên khả năng *triển khai độ trừu tượng đã được kế thừa*.

Tính đóng gói được triển khai bởi sử dụng **Access Specifier**. Một Access Specifier định nghĩa phạm vi và tính nhìn thấy của một thành viên lớp. C# hỗ trợ các Access Specifier sau:

- Public
- Private
- Protected
- Internal
- Protected internal

Public Access Specifier trong C#

Public Access Specifier trong C# cho phép một lớp trưng bày các biến thành viên và các hàm thành viên của nó tới các hàm và đối tượng khác. Bất kỳ thành viên public nào trong C# có thể được truy cập từ bên ngoài lớp đó.

Ví dụ sau minh họa Public Access Specifier trong C#:

```
using System;

namespace RectangleApplication
{
   class Rectangle
   {
      //member variables
      public double length;
      public double width;


      public double GetArea()
```

```
      {
         return length * width;
      }
      public void Display()
      {
         Console.WriteLine("Length: {0}", length);
         Console.WriteLine("Width: {0}", width);
         Console.WriteLine("Area: {0}", GetArea());
      }
   }//end class Rectangle

   class ExecuteRectangle
   {
      static void Main(string[] args)
      {
         Rectangle r = new Rectangle();
         r.length = 4.5;
         r.width = 3.5;
         r.Display();
         Console.ReadLine();
      }
   }
}
```

Khi code trên được biên dịch và thực thi, nó sẽ cho kết quả:

```
Length: 4.5
Width: 3.5
Area: 15.75
```

Trong ví dụ, các biến thành viên length và width được khai báo là **public**, vì thế chúng có thể được truy cập từ hàm Main() bởi sử dụng một Instance (một sự thể hiện) của lớp Rectangle, tên là **r**.

Hàm thành viên *Display()* và *GetArea()* cũng có thể truy cập các biến này một cách trực tiếp mà không cần sử dụng bất kỳ instance nào của lớp.

Hàm thành viên *Display()* cũng được khai báo là **public**, vì thế nó cũng có thể được truy cập từ hàm **Main()** bởi sử dụng một Instance (một sự thể hiện) của lớp Rectangle, tên là **r**.

Private Access Specifier trong C#

Private Access Specifier trong C# cho phép một lớp ẩn các biến thành viên và các hàm thành viên của nó với các hàm và đối tượng khác. Chỉ có các hàm trong cùng lớp đó có thể truy cập tới các thành viên private. Ngay cả khi một Instance của một lớp cũng không thể truy cập các thành viên private của nó.

Ví dụ sau minh họa Private Access Specifier trong C#:

```
using System;
namespace RectangleApplication
{
   class Rectangle
   {
      //member variables
      private double length;
      private double width;

      public void Acceptdetails()
      {
         Console.WriteLine("Enter Length: ");
         length = Convert.ToDouble(Console.ReadLine());
         Console.WriteLine("Enter Width: ");
         width = Convert.ToDouble(Console.ReadLine());
      }
      public double GetArea()
      {
         return length * width;
      }
      public void Display()
      {
         Console.WriteLine("Length: {0}", length);
```

```
         Console.WriteLine("Width: {0}", width);
         Console.WriteLine("Area: {0}", GetArea());
      }
   }//end class Rectangle

   class ExecuteRectangle
   {
      static void Main(string[] args)
      {
         Rectangle r = new Rectangle();
         r.Acceptdetails();
         r.Display();
         Console.ReadLine();
      }
   }
}
```

Khi code trên được biên dịch và thực thi, nó sẽ cho kết quả:

```
Enter Length:
4.4
Enter Width:
3.3
Length: 4.4
Width: 3.3
Area: 14.52
```

Trong ví dụ, các biến thành viên length và width được khai báo **private**, vì thế chúng không thể được truy cập từ hàm **Main()**. Các hàm thành viên *AcceptDetails()* và *Display()* có thể truy cập các biến này. Khi các hàm thành viên *AcceptDetails()* và *Display()* được khai báo**public**, chúng có thể được truy cập từ hàm **Main()** bởi sử dụng một Instance (một sự thể hiện) của lớp Rectangle, tên là **r**.

Protected Access Specifier trong C#

Protected Access Specifier trong C# cho phép một lớp con truy cập các biến thành viên và các hàm thành viên của lớp cơ sở của nó. Cách này giúp triển khai tính kế thừa. Chúng ta sẽ thảo luận chi tiết về tính kế thừa trong chương sau đó.

Internal Access Specifier trong C#

Internal Access Specifier trong C# cho phép một lớp trưng bày các biến thành viên và các hàm thành viên của nó tới các hàm và đối tượng khác trong *Assembly* hiện tại. Nói cách khác, bất kỳ thành viên nào với Internal Access Specifier trong C# có thể được truy cập từ bất kỳ lớp hoặc phương thức được định nghĩa bên trong ứng dụng mà thành viên đó được định nghĩa.

Ví dụ sau minh họa Internal Access Specifier trong C#:

```
using System;
namespace RectangleApplication
{
   class Rectangle
   {
      //member variables
      internal double length;
      internal double width;

      double GetArea()
      {
         return length * width;
      }
      public void Display()
      {
         Console.WriteLine("Length: {0}", length);
         Console.WriteLine("Width: {0}", width);
         Console.WriteLine("Area: {0}", GetArea());
      }
   }//end class Rectangle
```

```
   class ExecuteRectangle
   {
      static void Main(string[] args)
      {
         Rectangle r = new Rectangle();
         r.length = 4.5;
         r.width = 3.5;
         r.Display();
         Console.ReadLine();
      }
   }
}
```

Khi code trên được biên dịch và thực thi, nó sẽ cho kết quả:

```
Length: 4.5
Width: 3.5
Area: 15.75
```

Trong ví dụ, bạn chú ý rằng hàm thành viên **GetArea()** không được khai báo với bất kỳ Access Specifier nào. Thì theo mặc định, Access Specifier của một thành viên lớp nếu chúng ta không khai báo là **private**.

Protected Internal Access Specifier trong C#

Protected Internal Access Specifier trong C# cho phép một lớp ẩn các biến thành viên và các hàm thành viên của nó với các hàm và đối tượng khác, ngoại trừ một lớp con bên trong cùng ứng dụng đó. Điều này cũng được sử dụng trong khi triển khai tính kế thừa trong C#.