# MÔN : LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
## Bài thực hành số 10.3 : Xây dựng class tổng quát hóa "Stack các tham khảo"

**I. Mục tiêu :**

- Giúp SV làm quen với việc xây dựng class tổng quát hóa có dùng tham khảo thuộc kiểu bất kỳ.

**II. Nội dung :**

- Viết code miêu tả class tổng quát hóa "Stack các tham khảo bất kỳ" cung cấp 2 tác vụ push() và pop() để cất/lấy lại từng tham khảo trên đỉnh stack.
- Viết chương trình nhỏ thử dùng class tổng quát hóa "Stack các tham khảo bất kỳ" để tạo tự động 2 class Stack các đối tượng nguyên và Stack các đối tượng thực, thử sử dụng 2 class cụ thể này.

**III. Chuẩn đầu ra :**

- Sinh viên nắm vững việc đặc tả class tổng quát hóa có dùng tham khảo thuộc kiểu bất kỳ.

**IV. Qui trình :**

1. Chạy VS .Net, chọn menu File.Open.Project để hiển thị cửa sổ duyệt file. Duyệt và tìm file *.sln quản lý Project "AnyStackApp" có sẵn từ bài thực hành 8.2 để mở lại Project này.
2. Ấn phải chuột vào phần tử gốc của cây Project trong cửa sổ Solution Explorer, chọn option Add.Class, đặt tên là RefStack.cs để tạo ra file đặc tả class tổng quát hóa RefStack. Khi cửa sổ hiển thị mã nguồn của class RefStack hiển thị, copy code đã viết trong class IntStack, dán code này vào thân của RefStack, rồi hiệu chỉnh lại như sau (phần màu đỏ là phần thay đổi so với class IntStack) :

```
//định nghĩa class tổng quát hóa : Stack các giá trị thuộc kiểu T
namespace AnyStackApp {
public class RefStack <T> where T : class {
private T[] data;   //danh sách các phần tử trong stack
private int top;    // chỉ số phần tử đỉnh stack
private int max;    // số lượng max hiện hành stack

// khai báo hằng miêu tả số lượng phần tử cần thêm mỗi lần thiếu stack
private int GROWBY = 4;

//hàm constrcutor
public RefStack () {
    top = 0;
    max =GROWBY;
    data = (T[])new T[max];
}

//hàm push phần tử vào đỉnh
public bool push(T newVal) {
T[] newdata;
  if (top==max) { //nếu đầy stack
      //xin cấp phát lại vùng nhớ lớn hơn GROWBY phần tử so với stack hiện hành
      try {
          newdata = (T[])new T[GROWBY+max];
       } catch (Exception e){
         //System.out.println("He thong het cho roi!!!");
         return false;
      }
```

```
            //đi chuyển stack hiện hành về stack mới
            for (int i = 0; i<max; i++)
               newdata[i] =data[i];
            //cập nhật lại stack mới, để hệ thống xóa stack cũ tự động
            data = newdata;
            max += GROWBY;
        }
        //chứa giá trị mới vào đỉnh stack
        data[top++] =newVal;
        return true;
    }

    //hàm pop 1 phần tử từ đỉnh stack
    public T pop() {
        if (top == 0) //nếu cạn stack  thì tạo Exception
            throw new Exception ("Cạn stack");
        else //trả về trị ở đỉnh stack
            return data[--top];
    }
    } //hết class ValueStack
    } //hết namespace AnyStackApp
```

3. Ấn phải chuột vào phần tử gốc của cây Project trong cửa sổ Solution Explorer, chọn option Add.Class, đặt tên là MyInt.cs để tạo ra file đặc tả class đối tượng mà mỗi đối tượng chứa được 1 số nguyên. Khi cửa sổ hiển thị mã nguồn của class MyInt hiển thị, viết code cho class như sau :

```
namespace AnyStackApp {
    class MyInt {
        private int m_value;
        //thuộc tính giao tiếp Value
        public int Value {
            get { return m_value; }
            set { m_value = value; }
        }
        //hàm contructor
        public MyInt(int val) { m_value = val; }
    }
}
```

4. Dời chuột về phần tử Program.cs trong cây Project trong cửa sổ Solution Explorer, ấn phải chuột trên nó để hiển thị menu lệnh, chọn lệnh “View Code” để hiển thị mã nguồn của chương trình (class Program), hiệu chỉnh hàm Main lại như sau (phần màu đỏ là phần thay đổi so với nội dung có sẵn):

```
namespace AnyStackApp {
    class Program  {
        static void Main(string[] args) {
            int i;
            //tạo class RefStack<MyInt> và đối tượng của class này để dùng
            RefStack<MyInt> si = new RefStack<MyInt> ();
            //push lần lượt 11 giá trị từ -5 tới 5
            for (i = -5; i <= 5; i++)  {
                if (!si.push(new MyInt(i))) {
                    Console.WriteLine("Khong push duoc nua!!!");
                    return;
```

```
            }
        }
        //pop các giá trị trong Stack ra và hiển thị để kiểm tra
        try {
            while (true) {
                MyInt ci = si.pop();
                Console.WriteLine("Tri vua pop ra la : " + ci.Value);
            }
        } catch (Exception e) {
            Console.Write("Hết stack. Ấn Enter để đóng cửa sổ");
            Console.Read();
        }
    } //hết hàm Main
  } //hết class Program
} //hết namespace
```

5. Chọn menu Debug.Start Debugging để dịch và chạy ứng dụng. Hãy xem thứ tự hiển thị các số nguyên và đánh giá chức năng của đối tượng RefStack<MyInt>.
6. Ấn phải chuột vào phần tử gốc của cây Project trong cửa sổ Solution Explorer, chọn option Add.Class, đặt tên là MyDouble.cs để tạo ra file đặc tả class đối tượng mà mỗi đối tượng chứa được 1 số thực. Khi cửa sổ hiển thị mã nguồn của class MyDouble hiển thị, viết code cho class như sau :

```
namespace AnyStackApp {
  class MyDouble {
    private double m_value;
    //thuộc tính giao tiếp Value
    public double Value {
      get { return m_value; }
      set { m_value = value; }
    }
    //hàm contructor
    public MyDouble(double val) { m_value = val; }
  }
}
```

7. Dời chuột về phần tử Program.cs trong cây Project trong cửa sổ Solution Explorer, ấn phải chuột trên nó để hiển thị menu lệnh, chọn lệnh “View Code” để hiển thị mã nguồn của chương trình (class Program), hiệu chỉnh hàm Main lại như sau (phần màu đỏ là phần thay đổi so với nội dung có sẵn):

```
namespace AnyStackApp {
  class Program  {
    static void Main(string[] args) {
      int i;
      //tạo class RefStack<MyDouble> và đối tượng của class này để dùng
      RefStack<MyDouble> si = new RefStack<MyDouble> ();
      //push lần lượt 10 giá trị thực từ 3.1416 tới 10*3.1416
      for (i = 1; i <= 10; i++)  {
        if (!si.push(new MyDouble(i*3.1416))) {
          Console.WriteLine("Khong push duoc nua!!!");
          return;
        }
      }
      //pop các giá trị trong Stack ra và hiển thị để kiểm tra
      try {
```

```
            while (true) {
                MyDouble ci = si.pop();
                Console.WriteLine("Tri vua pop ra la : " + ci);
            }
        } catch (Exception e) {
            Console.Write("Hết stack. Ấn Enter để đóng cửa sổ");
            Console.Read();
        }
    } //hết hàm Main
} //hết class Program
} //hết namespace
```

8. Chọn menu Debug.Start Debugging để dịch và chạy ứng dụng. Hãy xem thứ tự hiển thị các số thực và đánh giá chức năng của đối tượng RefStack<MyDouble>.