LAB 9. ỦY NHIỆM HÀM VÀ SỰ KIỆN

THỜI LƯỢNG : 4 TIẾT

A. Mục tiêu

- Giúp sinh viên làm quen với lập trình hướng sự kiện (event) thông qua việc sử dụng ủy nhiệm hàm (delegate)
- Sau khi hoàn thành bài thực hành này, sinh viên cần:
  - Tạo được delegate phù hợp với yêu cầu bài toán
  - Tạo được thể hiện của delegate và tham chiếu tới các phương thức
  - Hiểu rõ và nắm vững cơ chế gọi phương thức thông qua delegate
  - Sử dụng được delegate với vai trò là tham số của phương thức
  - Định nghĩa được các sự kiện và delegate đại diện cho các phương thức xử lý sự kiện
  - Nắm vững cơ chế phát sinh, bắt và xử lý các sự kiện
  - Hiểu rõ cơ chế xử lý, truyền dữ liệu đi kèm sự kiện

B. Yêu cầu

- Sinh viên phải trả lời đầy đủ các câu hỏi (nếu có) hoặc chụp màn hình kết quả, ghi lại vào tập tin Word theo yêu cầu ghi trong phần hướng dẫn thực hành.
- Đặt tên tập tin Word theo dạng: Lab09_MSSV_HoVaTen.rar.
- Tạo thư mục, đặt tên là MSSV_Lab09 để lưu tất cả bài làm trong bài thực hành này.
- Sinh viên phải hoàn thành tối thiểu 2 bài tập bắt buộc.
- Phải tạo một dự án riêng cho mỗi phần hoặc mỗi bài tập.
- Cuối buổi thực hành, nén thư mục trên & nộp bài qua email: phucnguyen5555@gmail.com

C. Hướng dẫn thực hành

1. Định nghĩa delegate và sử dụng delegate để gọi hàm

Phần này hướng dẫn cách định nghĩa một delegate đơn giản và cách gọi phương thức thông qua thể hiện của delegate.

Bước 1. Tạo một thư mục, đặt tên theo dạng MSSV-Lab09. Ví dụ: 1210234-Lab09.
Bước 2. Tạo một dự án mới, đặt tên là Lab09_Bai01_DelegateDonGian
Bước 3. Trong file Program.cs, nhập đoạn mã sau để tạo delegate (nằm ngoài lớp Program):

```
// Định nghĩa một ủy nhiệm hàm đơn giản
// để thực hiện các thao tác trên chuỗi
public delegate void ThaoTac(string chuoi);
```

Bước 4. Trong lớp Program, định nghĩa các phương thức sau

```
// Định nghĩa các phương thức thực sự
// thao tác trên một chuỗi
static void XuatChuoi(string chuoi)
{
    Console.WriteLine(chuoi);
}

static void ChaoMung(string ten)
{
    Console.WriteLine("Chao ban {0}", ten);
}

static void ChuHoa(string chuoi)
{
    Console.WriteLine(chuoi.ToUpper());
}
```

Bước 5. Trong hàm Main, nhập đoạn mã sau:

```
// Khai báo một thể hiện của delegate ThaoTac
ThaoTac xuly;

// Trỏ biến xuly tới một hàm
// Hay cho biến xuly tham chiếu tới 1 hàm
xuly = new ThaoTac(XuatChuoi);

// Gọi hàm xuất chuỗi thông qua con trỏ hàm
xuly("Uy nhiem ham don gian");
```

Bước 6. Chạy chương trình và cho biết kết quả

Bước 7. Nhập tiếp đoạn mã sau

```
// Hoặc ngắn gọn hơn là
xuly = XuatChuoi;

// Gọi hàm xuất chuỗi thông qua con trỏ hàm
xuly("Goi ham lan thu 2");

Console.WriteLine("==================================");
```

Bước 8. Chạy chương trình và ghi nhận kết quả

Bước 9. Tiếp tục bổ sung đoạn mã sau để gọi nhiều phương thức chỉ với một lời gọi hàm

```
// Ta có thể gọi nhiều hàm cùng lúc thông
// qua con trỏ hàm bằng phép toán += như sau
xuly += ChaoMung;
xuly += ChuHoa;

// Gọi cả 3 hàm XuatChuoi, ChaoMung, ChuHoa
// bằng một lời gọi hàm duy nhất
xuly("Trinh Thanh Tai");
```

Bước 10. Chạy chương trình và ghi nhận kết quả.

Bước 11. Hãy mô tả (theo cách hiểu của bạn) cách chương trình xử lý khi gặp lệnh: xuly(" Trinh Thanh Tai")

Delegate cũng có thể gọi các phương thức thông qua một đối tượng, không nhất thiết phải là các phương thức tĩnh (static) như ở trên. Ngoài ra, có thể nhận kết quả trả về từ phương thức được gọi

Bước 12. Tạo một lớp mới, đặt tên là PhepToan (phép toán).

Bước 13. Bên ngoài lớp này, định nghĩa delegate TinhToan (tính toán) như sau:

```
// Định nghĩa một ủy nhiệm hàm (con trỏ hàm) đơn giản
// có hai tham số kiểu số nguyên. Nó có thể trỏ đến hay
// đại diện cho các hàm tính toán trên hai số nguyên.
public delegate int TinhToan(int x, int y);
```

Bước 14. Trong lớp PhepToan, định nghĩa các phương thức Cộng, Trừ, Nhân, Chia như sau

```
public class PhepToan
{
    // Những hàm dưới đây sẽ được gọi thông qua
    // ủy nhiệm hàm TinhToan. Vì vậy, chúng phải
    // được định nghĩa có dạng giống delegate
    // TinhToan ở trên
    public int Cong(int x, int y)
    {
        return x + y;
    }
    public int Tru(int x, int y)
    {
        return x - y;
    }
    public int Nhan(int x, int y)
    {
        return x * y;
    }
    public int Chia(int x, int y)
    {
        return x / y;
    }
}
```

Bước 15. Trong hàm Main, nhập đoạn mã sau để tạo delegate và gọi phương thức vừa tạo

```
// Tạo một thể hiện của lớp PhepToan
PhepToan toan = new PhepToan();

// Phần sau minh họa delegate có trả về giá trị
// Tạo một thể hiện của delegate TinhToan
TinhToan phepTinh = toan.Cong;

// Gọi hàm tính tổng hai số thông qua delegate
int ketQua = phepTinh(5, 3);

// Xuất kết quả
Console.WriteLine(ketQua);
Console.WriteLine("==================================");
```

Bước 16. Chạy chương trình và ghi nhận kết quả

Bước 17. Nhập tiếp đoạn mã sau để gọi nhiều phương thức cùng lúc

```
// Tương tự, ta có thể gọi nhiều hàm cùng lúc
phepTinh += toan.Tru;
phepTinh += toan.Nhan;
phepTinh += toan.Chia;

// Gọi 4 hàm một lúc.
ketQua = phepTinh(5, 3);

// Xuất kết quả
Console.WriteLine(ketQua);
```

Bước 18. Chạy chương trình và ghi nhận kết quả

Bước 19. Dòng lệnh cuối cùng xuất ra giá trị gì? Tại sao có kết quả đó?

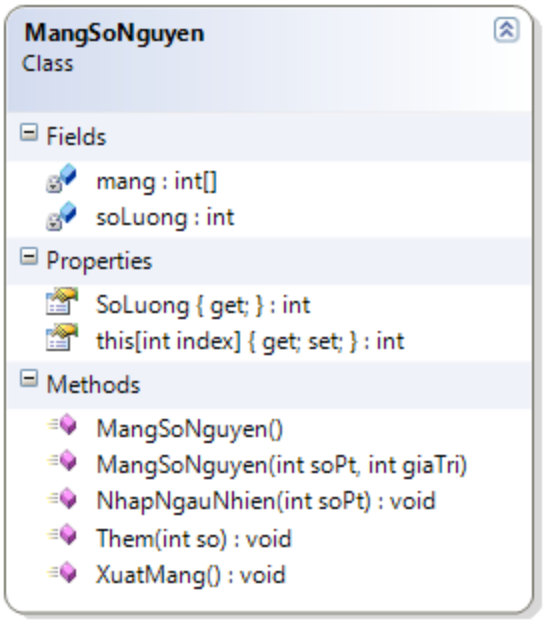
Bước 20. Tạo sơ đồ lớp, chụp hình, ghi lại kết quả

2. Sử dụng delegate làm tham số của phương thức

Phần này hướng dẫn cách truyền một delegate vào làm tham số của hàm. Cách hoạt động tương tự như con trỏ hàm trong C++.

Bước 1. Tạo một dự án Console Application mới, đặt tên là Lab09_Bai02_ThamSoLaDelegate

Bước 2. Tạo lớp mới, đặt tên là MangSoNguyen (mảng số nguyên). Cài đặt lớp này theo sơ đồ lớp như hình bên.



Bước 3. Trong hàm Main, nhập đoạn mã sau để

kiểm tra hoạt động của các phương thức vừa định nghĩa.

```
// Tạo một thể hiện của MangSoNguyen
MangSoNguyen mangNguyen = new MangSoNguyen();

// Gọi hàm nhập mảng
mangNguyen.NhapNgauNhien(100);

// Xuất mảng
mangNguyen.XuatMang();
Console.WriteLine("".PadRight(80, '='));
```

Bước 4. Chạy chương trình và ghi nhận kết quả

Bước 5. Trong file MangSoNguyen.cs, bên ngoài lớp MangSoNguyen, định nghĩa delegate sau:

```
// Định nghĩa delegate làm đại diện cho các
// hàm kiểm tra một phần tử hay số nguyên x
// có thỏa điều kiện nào đó hay không
public delegate bool KiemTraDieuKien(int x);
```

Bước 6. Trong lớp MangSoNguyen, định nghĩa phương thức tìm kiếm theo điều kiện như sau

```
// Hàm tìm các số nguyên theo điều kiện
public MangSoNguyen Tim(KiemTraDieuKien kiemTra)
{
    MangSoNguyen ketQua = new MangSoNguyen();

    for (int i = 0; i < soLuong; i++)
    {
        if (kiemTra(mang[i]))
            ketQua.Them(mang[i]);
    }

    return ketQua;
}
```

Bước 7. Trong lớp Program, định nghĩa hai phương thức (theo nguyên mẫu delegate) sau:

```
// Định nghĩa hàm kiểm tra điều kiện số nguyên tố
static bool LaSoNguyenTo(int x)
{
    if (x < 2) return false;

    for (int i = 2; i <= x / 2; i++)
    {
        if (x % i == 0)
            return false;
    }
    return true;
}
```

```
// Định nghĩa hàm kiểm tra điều kiện chia hết
// theo mẫu của delegate KiemTraDieuKien
static bool ChiaHetCho5(int x)
{
    return x % 5 == 0;
}
```

Bước 8. Trong hàm Main, bổ sung tiếp đoạn mã sau để gọi hàm tìm các số chia hết cho 5

```
// Tạo điều kiện tìm kiếm các số chia hết cho 5
KiemTraDieuKien dieuKien = ChiaHetCho5;

// Gọi hàm tìm các số chia hết cho 5
MangSoNguyen ketQua = mangNguyen.Tim(dieuKien);

// Xuất các số chia hết cho 5
ketQua.XuatMang();
Console.WriteLine("".PadRight(80, '='));
```

Bước 9. Chạy chương trình và ghi nhận kết quả

Bước 10. Tiếp tục bổ sung đoạn mã sau vào hàm Main để tìm các số nguyên tố

```
// Gọi hàm tìm các số nguyên tố
// Ở đây, ta truyền trực tiếp tên hàm làm đối số
ketQua = mangNguyen.Tim(LaSoNguyenTo);

// Xuất các số nguyên tố
ketQua.XuatMang();
```

Bước 11. Chạy chương trình và ghi nhận kết quả

Bước 12. Mô tả một cách vắn tắt hoạt động của toàn bộ chương trình.

3. Tạo và xử lý sự kiện đơn giản

Phần này hướng dẫn cách tạo, phát sinh, bắt và xử lý 1 sự kiện đơn giản qua 6 bước (chú ý xem thứ tự các bước trong chú thích). Ta sẽ sử dụng lại lớp MangSoNguyen trong bài trước để minh họa.

Bước 1. Tạo dự án Console Application mới, đặt tên là Lab09_Bai03_SuKienDonGian

Bước 2. Tạo một lớp mới, đặt tên là MangSoNguyen.

Bước 3. Sao chép nội dung của lớp MangSoNguyen từ Lab09_Bai02 sang lớp vừa tạo.

Bước 4. Định nghĩa delegate sau ở đầu (bên ngoài) lớp MangSoNguyen

```
// 1. Tạo delegate đại diện cho hàm xử lý sự kiện
public delegate void XuLyThem(int phanTu);
```

Bước 5. Bên trong lớp MangSoNguyen, định nghĩa sự kiện ThemThanhCong như sau

```
// Lớp biểu diễn một mảng số nguyên
public class MangSoNguyen
{
    // 2. Tạo sự kiện bằng từ khóa event
    public event XuLyThem ThemThanhCong;
```

Bước 6. Trong lớp MangSoNguyen, định nghĩa phương thức phát sinh ra sự kiện

```
// 3. Tạo hàm phát sinh ra sự kiện
//    Thông thường, hàm này có mức truy xuất
//    protected và sử dụng thêm từ khóa virtual
//    để các lớp con có thể ghi đè lên.
protected virtual void PhatSinhSuKienThemThanhCong(int phanTu)
{
    if (ThemThanhCong != null)
        ThemThanhCong(phanTu);
}
```

Bước 7. Định nghĩa lại phương thức Them để phát sinh ra sự kiện khi thêm thành công

```
// Định nghĩa hàm thêm một số nguyên vào cuối mảng
public void Them(int so)
{
    mang[soLuong] = so;
    soLuong++;

    // 4. Gọi hàm phát sinh sự kiện
    PhatSinhSuKienThemThanhCong(so);
}
```

Bước 8. Trong lớp Program, định nghĩa phương thức sau để xử lý sự kiện

```
// 5. Tạo hàm xử lý sự kiện được phát sinh
static void XuatPhanTu(int phanTu)
{
    Console.Write("Phan tu moi : {0}\r\n", phanTu);
}
```

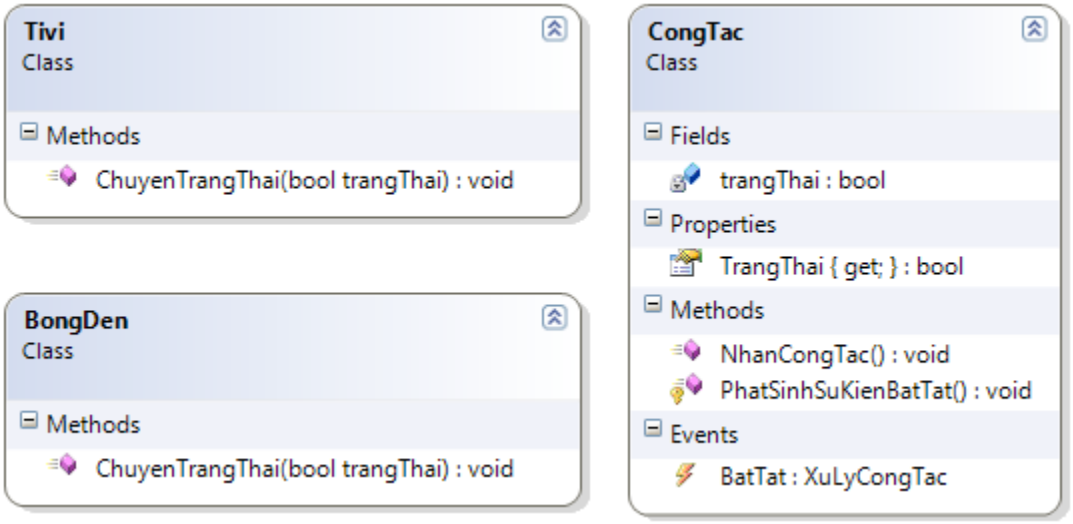
Bước 9. Trong hàm Main, bổ sung đoạn mã sau:

```
// Tạo thể hiện của lớp MangSoNguyen
MangSoNguyen mangNguyen = new MangSoNguyen();

// 6. Đăng ký hàm xử lý sự kiện
mangNguyen.ThemThanhCong += new XuLyThem(XuatPhanTu);

// hoặc là: mangNguyen.ThemThanhCong += XuatPhanTu;

// Gọi hàm nhập mảng
mangNguyen.NhapNgauNhien(100);
```

Bước 10. Chạy chương trình và ghi nhận kết quả.
Bước 11. Giải thích cơ chế hoạt động của chương trình trên.

4. Bắt và xử lý sự kiện (cách 1)

Phần này hướng dẫn cách tạo, phát sinh, bắt và xử lý sự kiện liên quan tới nhiều đối tượng. Cụ thể, trong ví dụ này, ta có các lớp CongTac (công tắc), Tivi (Ti-vi), BongDen (bóng đèn). Khi công tắc bật hay tắt thì bóng đèn và tivi cũng sẽ chuyển trạng thái tương ứng.



Bước 1. Tạo dự án Console Application mới, đặt tên là Lab09_Bai04_SuKienPhucTap
Bước 2. Tạo và định nghĩa hai lớp lớp Tivi, BongDen như sau

```
public class Tivi
{
    public void ChuyenTrangThai(bool trangThai)
    {
        if (trangThai)
            Console.WriteLine("Tivi dang bat");
        else
            Console.WriteLine("Tivi da tat");
    }
}
public class BongDen
{
    public void ChuyenTrangThai(bool trangThai)
    {
        if (trangThai)
            Console.WriteLine("Bong den dang sang");
        else
            Console.WriteLine("Bong den da tat");
    }
}
```

Bước 3. Tạo lớp CongTac.

Bước 4. Định nghĩa lớp CongTac và delegate XyLyCongTac như sau:

```
// 1. Tạo delegate xử lý sự kiện bật công tắc
// sender là đối tượng nguồn phát ra sự kiện
// e là tham số mang dữ liệu kèm theo sự kiện
public delegate void XuLyCongTac(object sender, EventArgs e);

// Lớp biểu diễn công tắc nguồn
public class CongTac
{
    // 2. Tạo sự kiện
    public event XuLyCongTac BatTat;

    // Khai báo biến lưu trạng thái công tắc
    private bool trangThai = false;

    // Định nghĩa sự kiện
    public bool TrangThai
    {
        get { return trangThai; }
    }

    public void NhanCongTac()
    {
        // Chuyển trạng thái
        trangThai = !trangThai;

        // 4. Gọi hàm phát sinh sự kiện
        PhatSinhSuKienBatTat();
    }

    // 3. Tạo hàm phát sinh ra sự kiện
    protected virtual void PhatSinhSuKienBatTat()
    {
        if (BatTat != null)
        {
            EventArgs ea = new EventArgs();

            // Công tắc chính là đối tượng nguồn
            // phát ra sự kiện nên sender = this
            BatTat(this, ea);
        }
    }
}
```

Bước 5. Trong lớp Program, khai báo hai biến sau

```
// Tạo thể hiện của lớp Tivi và BongDen
static Tivi samsung = new Tivi();
static BongDen rangDong = new BongDen();
```

Bước 6. Tiếp tục định nghĩa phương thức xử lý sự kiện như sau

```
// 5. Tạo hàm xử lý sự kiện
static void XuLySuKienBatCauGiao(object sender, EventArgs e)
{
    CongTac cauGiao = sender as CongTac;

    samsung.ChuyenTrangThai(cauGiao.TrangThai);
    rangDong.ChuyenTrangThai(cauGiao.TrangThai);
}
```

Bước 7. Trong hàm Main, nhập đoạn mã sau để kiểm tra hoạt động của chương trình

```
// Tạo thể hiện của lớp CongTac
CongTac cauGiao = new CongTac();

// 6. Đăng ký hàm xử lý sự kiện
cauGiao.BatTat += XuLySuKienBatCauGiao;

// Gọi hàm bật công tắc lần 1.
cauGiao.NhanCongTac();

Console.WriteLine("".PadRight(80, '='));

// Gọi hàm bật công tắc lần 2.
cauGiao.NhanCongTac();
```

Bước 8. Chạy chương trình và cho biết kết quả.

Bước 9. Tiếp tục bổ sung đoạn mã sau

```
// Hủy bỏ hàm xử lý sự kiện
cauGiao.BatTat -= XuLySuKienBatCauGiao;

Console.WriteLine("".PadRight(80, '='));

// Gọi hàm bật công tắc lần 3.
cauGiao.NhanCongTac();
```

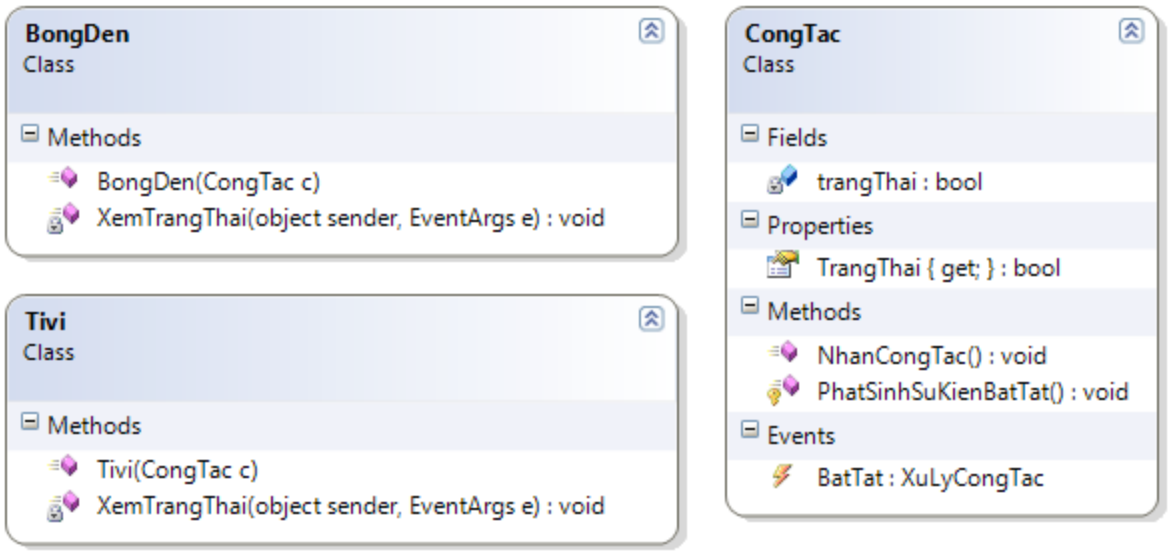
Bước 10. Chạy chương trình và ghi nhận kết quả. So sánh với kết quả ở bước 8.

Bước 11. Tại sao lần gọi phương thức thứ 3 không xuất ra thông báo gì cả?

5. Bắt và xử lý sự kiện (cách 2)

Cũng với bài toán Công tắc – Bóng đèn ở trên nhưng phần này hướng dẫn cách bắt sự kiện và tạo các phương thức xử lý sự kiện khác nhau ứng với các đối tượng khác nhau. Các lớp cần thiết được cho trong sơ đồ dưới đây:

Bước 1. Tạo dự án Console Application mới, đặt tên là Lab09_Bai05_CongTacBongDen



Bước 2. Định nghĩa lớp CongTac và delegate XuLyCongTac như sau

```
// 1. Tạo delegate xử lý sự kiện bật công tắc
// sender là đối tượng nguồn phát ra sự kiện
// e là tham số mang dữ liệu kèm theo sự kiện
public delegate void XuLyCongTac(object sender, EventArgs e);

// Lớp biểu diễn công tắc nguồn
public class CongTac
{
    // 2. Tạo sự kiện
    public event XuLyCongTac BatTat;

    // Khai báo biến lưu trạng thái công tắc
    private bool trangThai = false;

    // Định nghĩa sự kiện
    public bool TrangThai
    {
        get { return trangThai; }
    }

    public void NhanCongTac()
    {
        // Chuyển trạng thái
        trangThai = !trangThai;

        // 4. Gọi hàm phát sinh sự kiện
        PhatSinhSuKienBatTat();
    }

    // 3. Tạo hàm phát sinh ra sự kiện
    protected virtual void PhatSinhSuKienBatTat()
    {
```

```
            if (BatTat != null)
            {
                EventArgs ea = new EventArgs();

                // Công tắc chính là đối tượng nguồn
                // phát ra sự kiện nên sender = this
                BatTat(this, ea);
            }
        }
    }
```

Bước 3. Định nghĩa hai lớp BongDen và Tivi.

```
public class Tivi
{
    public Tivi(CongTac c)
    {
        // đăng ký hàm xử lý riêng khi công tắc bật
        c.BatTat += ChuyenTrangThai;
    }

    // Định nghĩa hàm xử lý sự kiện bật công tắc tivi
    private void ChuyenTrangThai(object sender, EventArgs e)
    {
        CongTac cauGiao = sender as CongTac;

        if (cauGiao.TrangThai)
            Console.WriteLine("Tivi dang bat");
        else
            Console.WriteLine("Tivi da tat");
    }
}

public class BongDen
{
    public BongDen(CongTac c)
    {
        // đăng ký hàm xử lý riêng khi công tắc bật
        c.BatTat += ChuyenTrangThai;
    }

    // Định nghĩa hàm xử lý sự kiện bật công tắc đèn
    private void ChuyenTrangThai(object sender, EventArgs e)
    {
        CongTac cauGiao = sender as CongTac;

        if (cauGiao.TrangThai)
            Console.WriteLine("Bong den dang sang");
        else
            Console.WriteLine("Bong den da tat");
    }
}
```

Bước 4. Trong hàm Main, nhập đoạn mã sau

```
// Tạo thể hiện của các lớp CongTac
CongTac cauGiao = new CongTac();

// Tạo thể hiện của các lớp Tivi và BongDen
Tivi samsung = new Tivi(cauGiao);
BongDen rangDong = new BongDen(cauGiao);

// Gọi hàm bật công tắc lần 1.
cauGiao.NhanCongTac();

Console.WriteLine("".PadRight(80, '='));

// Gọi hàm bật công tắc lần 2.
cauGiao.NhanCongTac();

Console.WriteLine("".PadRight(80, '='));

Console.ReadKey();
```

Bước 5. Chạy chương trình và ghi nhận kết quả.

Bước 6. So sánh với chương trình ở phần trước và đưa ra nhận xét.

6. Gửi thông điệp đi kèm sự kiện

Trong hai phần trước, trạng thái của Tivi và Bóng đèn được lấy trực tiếp từ trạng thái của công tắc. Ta lấy nó từ tham số sender. Tuy nhiên, trong nhiều trường hợp, thông tin đi kèm theo sự kiện có thể là kết quả của một tính toán nào đó, không có sẵn trong đối tượng phát ra sự kiện. Giải pháp là lưu trữ những thông tin này trong tham số thứ hai (e). Muốn vậy, ta phải tạo ra một lớp mới để chứa những thông tin này và dùng lớp này làm kiểu của tham số e. Phần này cũng hướng dẫn cách sử dụng delegate EventHandler đã có sẵn thay vì phải tạo delegate như các phần trước.

Bước 1. Tạo dự án Console Application mới, đặt tên là Lab09_Bai06_SuDungEventHandler

Bước 2. Định nghĩa lớp StatusEventArgs như sau

```
// Định nghĩa lớp mang thông tin đi kèm sự kiện
// Lớp này sẽ lưu trạng thái của công tắc
public class StatusEventArgs : EventArgs
{
    // Khai báo biến lưu trạng thái công tắc
    private bool trangThai = false;

    // Định nghĩa sự kiện
    public bool TrangThai
    {
        get { return trangThai; }
    }
```

```
        // Định nghĩa phương thức khởi tạo
        public StatusEventArgs(bool trangThai)
        {
            this.trangThai = trangThai;
        }
    }
```

Bước 3. Định nghĩa lớp CongTac như sau

```
// Lớp biểu diễn công tắc nguồn
public class CongTac
{
    // 2. Tạo sự kiện
    public event EventHandler BatTat;

    // Khai báo biến lưu trạng thái công tắc
    private bool trangThai = false;

    // Định nghĩa hàm nhấn (bật) công tắc
    public void NhanCongTac()
    {
        // Chuyển trạng thái
        trangThai = !trangThai;

        // 4. Gọi hàm phát sinh sự kiện
        PhatSinhSuKienBatTat();
    }

    // 3. Tạo hàm phát sinh ra sự kiện
    protected virtual void PhatSinhSuKienBatTat()
    {
        if (BatTat != null)
        {
            EventArgs ea = new StatusEventArgs(trangThai);

            // Công tắc chính là đối tượng nguồn
            // phát ra sự kiện nên sender = this
            BatTat(this, ea);
        }
    }
}
```

Bước 4. Cài đặt các lớp Tivi và BongDen

```
public class Tivi
{
    public Tivi(CongTac c)
    {
        // đăng ký hàm xử lý riêng khi công tắc bật
        c.BatTat += XemTrangThai;
    }
```

```
    // Định nghĩa hàm xử lý sự kiện bật công tắc tivi
    private void XemTrangThai(object sender, EventArgs e)
    {
        StatusEventArgs sea = (StatusEventArgs) e;

        if (sea.TrangThai)
            Console.WriteLine("Tivi dang bat");
        else
            Console.WriteLine("Tivi da tat");
    }
}

public class BongDen
{
    public BongDen(CongTac c)
    {
        // đăng ký hàm xử lý riêng khi công tắc bật
        c.BatTat += XemTrangThai;
    }

    // Định nghĩa hàm xử lý sự kiện bật công tắc đèn
    private void XemTrangThai(object sender, EventArgs e)
    {
        StatusEventArgs sea = (StatusEventArgs)e;

        if (sea.TrangThai)
            Console.WriteLine("Bong den dang sang");
        else
            Console.WriteLine("Bong den da tat");
    }
}
```

Bước 5. Trong hàm Main, nhập đoạn mã sau

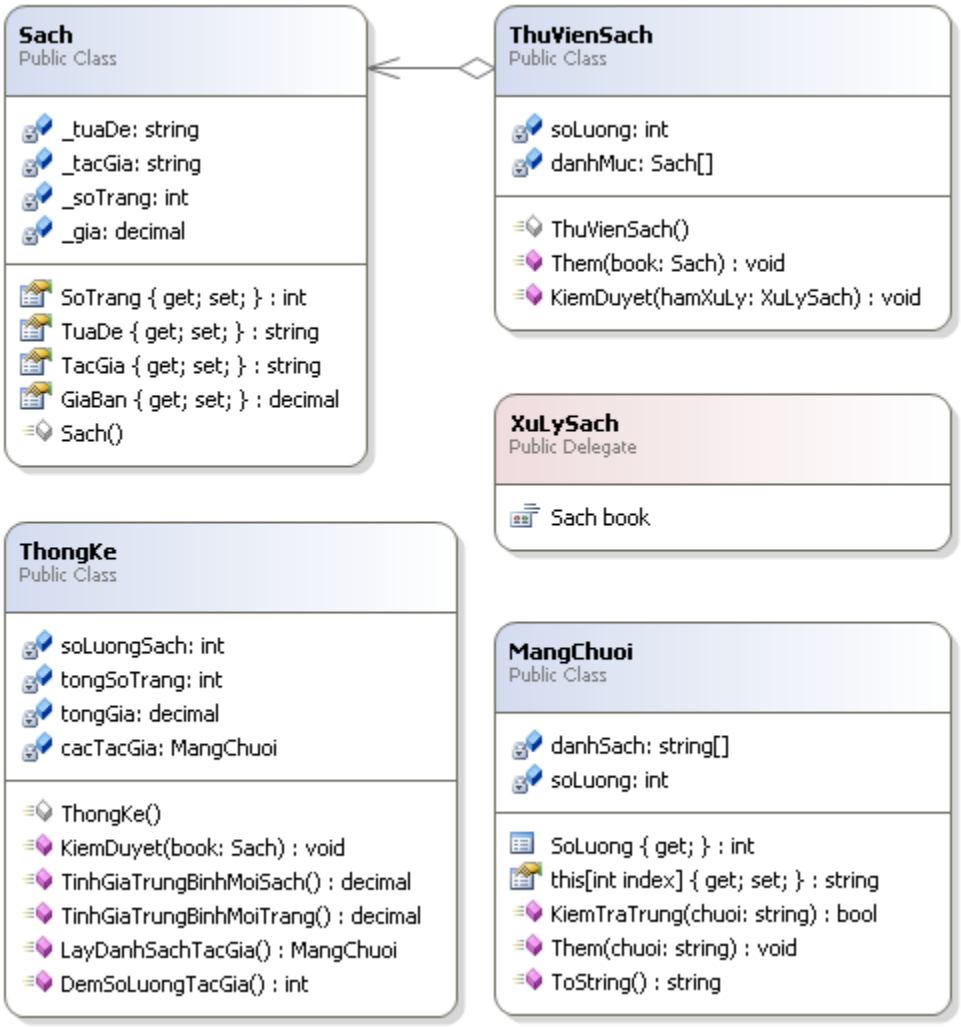
```
// Tạo thể hiện của các lớp CongTac
CongTac cauGiao = new CongTac();

// Tạo thể hiện của các lớp Tivi và BongDen
Tivi samsung = new Tivi(cauGiao);
BongDen rangDong = new BongDen(cauGiao);

// Gọi hàm bật công tắc lần 1.
cauGiao.NhanCongTac();

Console.WriteLine("".PadRight(80, '='));

// Gọi hàm bật công tắc lần 2.
cauGiao.NhanCongTac();

Console.WriteLine("".PadRight(80, '='));

Console.ReadKey();
```

Bước 6. Chạy chương trình và ghi nhận kết quả.

Bước 7. So sánh với chương trình Lab09_Bai05 và đưa ra những điểm giống-khác nhau.

D. Bài tập bắt buộc

1. Viết chương trình thống kê sách trong thư viện theo mô tả dưới đây



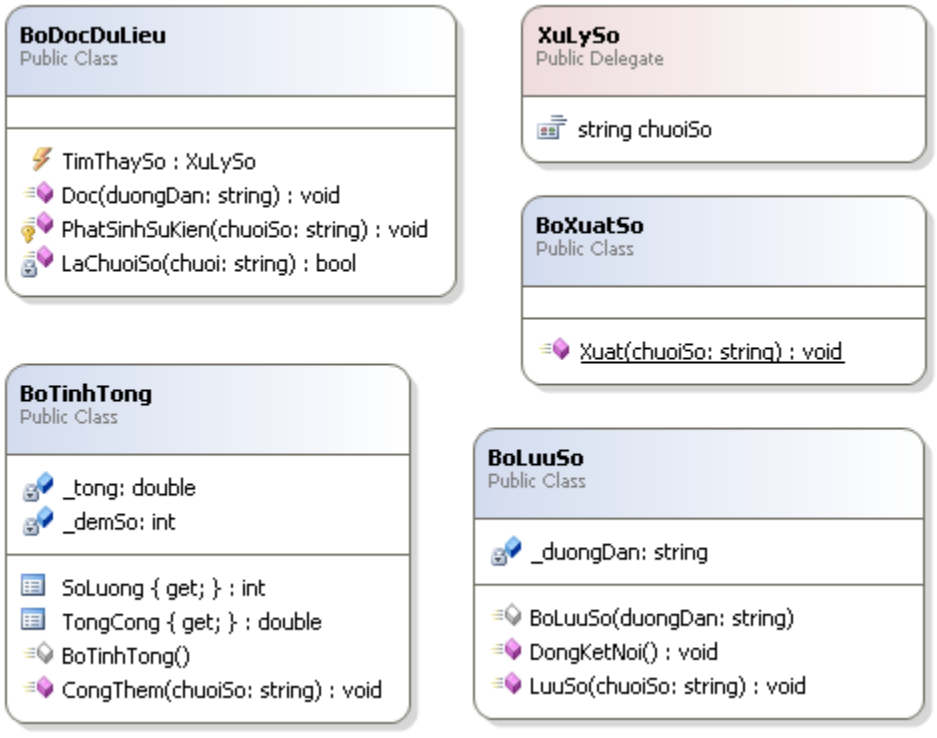
Trong đó:

- delegate void XuLySach(book) đại diện cho 1 hàm xử lý nào đó trên một cuốn sách
- Phương thức void KiemDuyet(hamXuLy) trong lớp ThuVienSach sẽ duyệt qua từng cuốn sách và gọi hàm hamXuLy trên từng cuốn sách đó.
- Phương thức bool KiemTraTrung(chuoi) trong lớp MangChuoi dùng để kiểm tra chuỗi đã có trong mảng hay chưa.
- Phương thức KiemDuyet(book) trong lớp ThongKe sẽ thực hiện các thống kê theo yêu cầu.

Thực hiện các yêu cầu sau:

a. Cài đặt các lớp theo sơ đồ trên
b. Cài đặt phương thức KiemDuyet(book) trong lớp ThongKe để thực hiện việc đếm số lượng sách, đếm tổng số trang sách, tính tổng giá bán và danh sách các tác giả (nếu trùng tên thì chỉ lấy 1).
c. Trong hàm Main, tạo một thể hiện của lớp ThuVienSach. Thêm 10 cuốn sách khác nhau vào thư viện sách.
d. Gọi hàm để thực hiện việc thống kê theo các yêu cầu ở câu b.
e. Xuất danh sách tên tác giả của các cuốn sách
f. Xuất số lượng tác giả
g. Xuất giá bán trung bình của mỗi cuốn sách
h. Xuất giá trung bình của mỗi trang sách
i. Tính số lượng trang trung bình trên mỗi cuốn sách.
j. Sửa đổi phương thức KiemDuyet ở câu b và cài đặt thêm các hàm để cho biết giá sách lớn nhất, giá sách rẻ nhất.

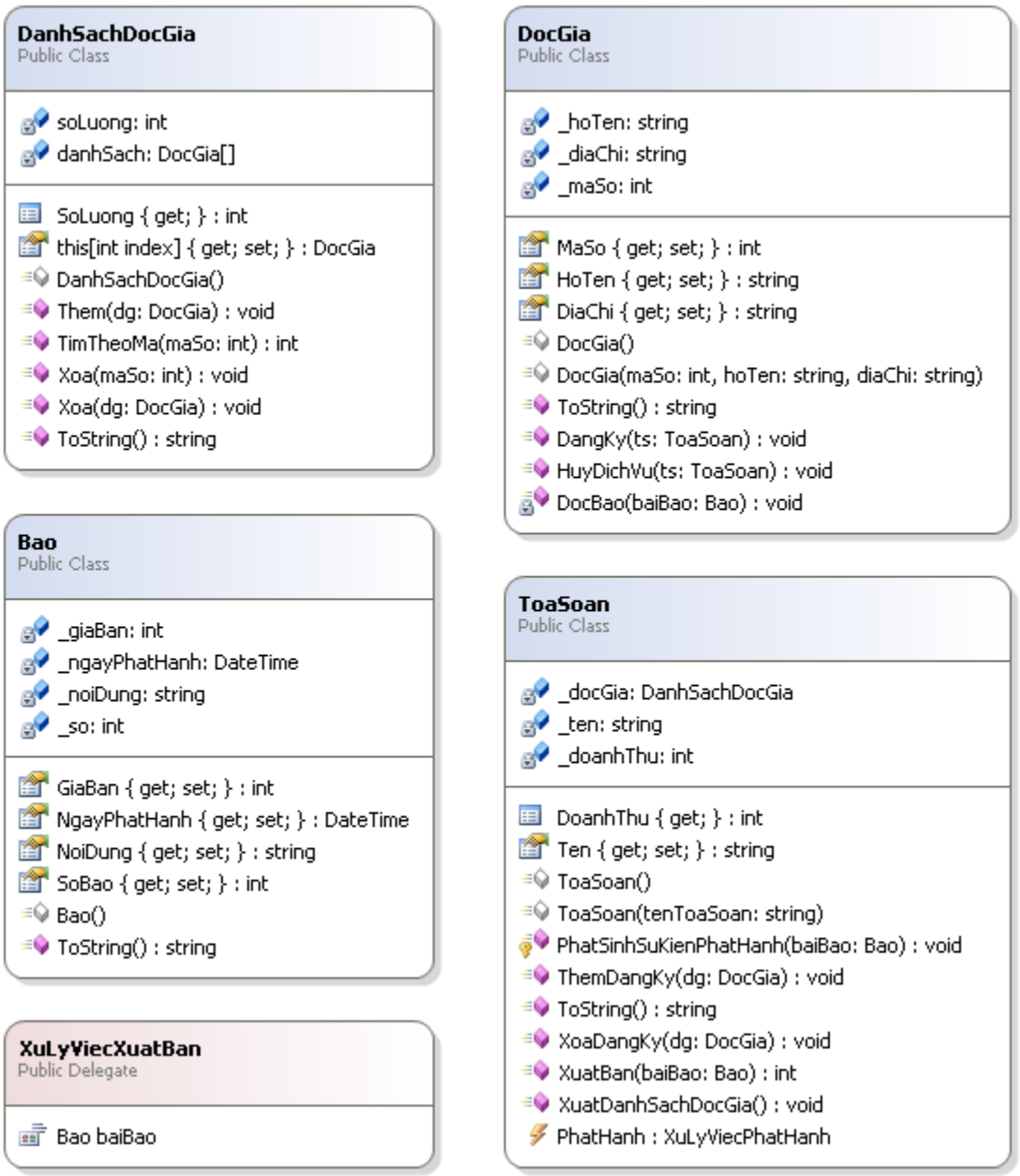
2. Viết chương trình đọc (từng từ) một tập tin văn bản. Nếu mỗi từ là một chuỗi số (có thể chuyển thành một con số) thì xuất ra màn hình, ghi xuống file, tính tổng và đếm các số tìm được.



Sơ đồ lớp gợi ý

3. Viết chương trình quản lý việc đăng ký và phát hành báo theo mô tả sau

- Mỗi tòa soạn báo (ToaSoan) có một tên.
- Hàng tuần, các tòa soạn sẽ xuất bản một kỳ báo (Bao) nhưng không cố định ngày phát hành
- Độc giả có thể đăng ký với các tòa soạn để nhận báo mỗi khi có kỳ báo mới. Họ cũng có thể hủy dịch vụ này bất kỳ lúc nào họ muốn. Những độc giả đã đăng ký sẽ được tòa soạn phân phát báo tới tận nhà.
- Tòa soạn cần phải lưu trữ lại danh sách các độc giả đăng ký. Khi độc giả hủy dịch vụ thì xóa độc giả đó ra khỏi danh sách.
- Mỗi đợt xuất bản, tòa soạn cần tính doanh thu của đợt đó và cập nhật lại tổng doanh thu.



Sơ đồ lớp gợi ý

Thực hiện các yêu cầu sau:

a. Phương thức DocBao trong lớp DocGia sẽ xuất ra màn hình: tên đọc giả và nội dung bài báo
b. Tạo 3 thể hiện của lớp ToaSoan ứng với 3 tòa soạn báo khác nhau
c. Tạo 5 thể hiện của lớp DocGia ứng với 5 độc giả khác nhau
d. Thực hiện việc đăng ký (tùy ý) mua báo cho 5 độc giả từ 3 tòa soạn đã tạo ở trên
e. Xuất danh sách độc giả của mỗi tòa soạn.
f. Mỗi tòa soạn xuất bản ít nhất 3 bài báo (khác nhau hoàn toàn). Mỗi lần xuất bản, xuất ra tên tòa soạn, ngày xuất bản và doanh thu của đợt xuất bản đó.
g. Cho biết tổng doanh thu của mỗi tòa soạn
h. Với 5 độc giả ở trên, hủy 5 đăng ký dịch vụ bất kỳ đã đăng ký trước đó.
i. Mỗi tòa soạn xuất bản tiếp một bài báo.
j. Chuyển sang sử dụng delegate EventHandler để định nghĩa và xử lý sự kiện.

E. Bài tập làm thêm

1. Viết chương trình quản lý việc kiểm duyệt các bài báo khoa học tại một hội thảo theo mô tả sau
   - Hội thảo kêu gọi các cá nhân hoặc tổ chức trên toàn thế giới tham dự với tư cách là người gửi bài báo khoa học hoặc người kiểm duyệt nội dung bài báo.
   - Ban tổ chức hội thảo sẽ đề xuất trước các chủ đề hay các chuyên mục.
   - Các thành viên tham gia hội thảo phải đăng ký trước. Họ cần cung cấp tên, email, nơi làm việc để tiện cho việc liên lạc.
   - Đối với những thành viên chỉ tham gia với tư cách là người kiểm duyệt nội dung thì cần cung cấp thêm các chủ đề nằm trong chuyên môn để ban tổ chức có thể gửi bài có nội dung phù hợp.
   - Đối với những thành viên gửi bài thì cần cung cấp thêm bài báo và chủ đề mà nội dung bài báo hướng đến. Mỗi người có thể gửi nhiều bài báo khác nhau.
   - Mỗi bài báo cần lưu trữ tựa đề, nội dung, chủ đề, thông tin các tác giả và thông tin về người chịu trách nhiệm chính.
   - Mỗi khi có một bài báo được gửi đến hội thảo, bài báo đó sẽ được đánh mã số. Sau đó, hệ thống sẽ tự động chuyển bài báo đó đến cho 3 người kiểm duyệt có đúng chuyên môn để xét duyệt bài báo đó.
   - Hệ thống phải đảm bảo phân phát bài báo đến người kiểm duyệt (cùng chuyên môn) một cách đồng đều. Nghĩa là không để một người kiểm duyệt quá nhiều, người kia lại quá ít.
   - Người kiểm duyệt sau khi xem xét xong một bài báo nào đó sẽ gửi lại phản hồi (gồm những đánh giá, góp ý, kết luận: đạt/không đạt) kèm theo mã số của bài báo.
   - Những góp ý ứng với mỗi bài báo sẽ được hệ thống tự động chuyển tiếp đến cho tác giả.
   - Những bài báo có ít nhất 2 kết luận là đạt thì sẽ được thông báo là chấp nhận đăng ở hội thảo.

2. Viết chương trình giả lập một vụ khủng bố bằng bom hẹn giờ theo mô tả sau:
    - Một kẻ khủng bố lên kế hoạch đặt 10 quả bom hẹn giờ tại một trung tâm thương mại lớn.
    - Những quả bom này được kết nối thành một chuỗi. Mỗi quả bom có một số hiệu và sẽ được kết nối với đúng một quả bom khác.
    - Quả bom đầu tiên sẽ được kích nổ bởi một đồng hồ hẹn giờ. Thời gian được thiết lập là N + M giây. Khi một quả bom này nổ thì quả bom nối tới cũng sẽ phát nổ.
    - Quá trình khủng bố được chia làm hai giai đoạn. Giai đoạn thứ nhất, đồng hồ sẽ đếm lùi N giây. Hết N giây này, toàn bộ hệ thống cửa ra vào sẽ bị khóa lại (tự động). Đồng thời, toàn bộ các Tivi trong trung tâm thương mại sẽ phát một đoạn băng thông báo lẫn đe dọa. Thông báo này cho biết lực lượng an ninh có M giây để tìm và phá ngòi nổ cũng như dừng đồng hồ hẹn giờ.
    - Giai đoạn hai là nếu qua M giây này mà vẫn chưa phá được bom thì quả bom đầu tiên sẽ phát nổ. Và do đó, lần lượt 9 quả gom còn lại cũng sẽ phát nổ theo.

Lưu ý:

- Các giá trị M, N và thông báo đe dọa được thiết lập khi khởi tạo đồng hồ hẹn giờ.
- Để chương trình dừng 1 giây, dùng lệnh System.Threading.Thread.Sleep(1000);