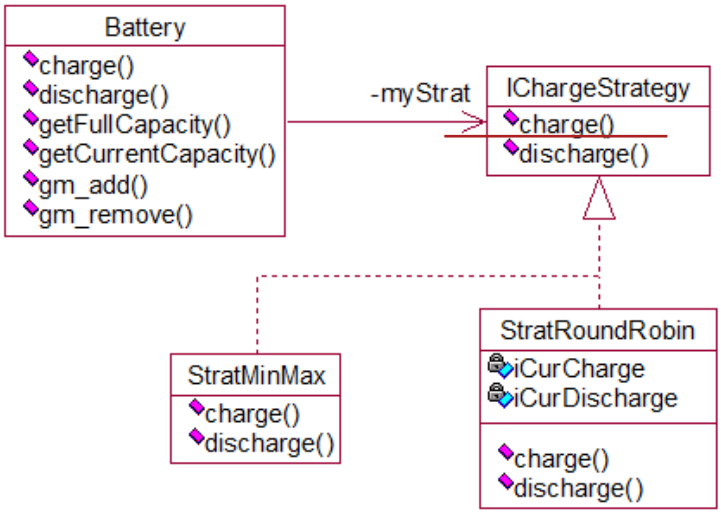
Nội dung 1:

Ta sẽ dùng mẫu thiết kế Strategy để cho phép client linh động chọn lựa giải thuật nạp/xả năng lượng với lược đồ class như sau :



Vai trò, vị trí của từng phần tử trong lược đồ trên như sau :

- IChargeStratery : cung cấp interface hợp nhất cho mọi đối tượng miêu tả giải thuật nạp/xả năng lượng cụ thể, interface hợp nhất này sẽ cung cấp 2 tác vụ : charge() và discharge().
- StratMinMax, StratRoundRobin… : mỗi class này sẽ hiện thực giải thuật nạp/xả cụ thể của mình.
- Battery : class sẽ linh động chọn và sử dụng giải thuật nạp/xả do các class trên miêu tả.

Sau đây là mã nguồn VC# để đặc tả các thành phần trên :

```
//đặc tả interface của mọi đối tượng miêu tả giải thuật nạp/xả năng lượng
//cho pin tích hợp
interface IChargeStrategy {
        //tác vụ nạp điện cho phần tử tích hợp theo 1 chiến lược xác định
        bool charge(List<IBattery> elems);
        //tác vụ thải điện khỏi phần tử tích hợp theo 1 chiến lược xác định
        bool discharge(List<IBattery> elems);
}
//class miêu tả giải thuật nạp/xả năng lượng theo dạng xoay vòng
class StratRoundRobin : IChargeStrategy {
        int aCharger = -1;  //thiết lập chỉ số battery đã nạp lần cuối
        int aDecharger = -1;  //thiết lập chỉ số battery đã thải lần cuối
```

```
        public bool charge(List<IBattery> elems)  {
            //hiệu chỉnh chỉ số battery cần nạp
            aCharger = (aCharger + 1) % elems.Count;
            return elems[aCharger].charge();
        }
        public bool discharge(List<IBattery> elems) {
            //hiệu chỉnh chỉ số battery cần thải
            aDecharger = (aDecharger + 1) % elems.Count;
            return elems[aDecharger].charge();
        }
}
//class miêu tả giải thuật nạp/xả năng lượng theo dạng Min-Max
class StratMinMax : IChargeStrategy {
        public bool charge(List<IBattery> elems) {
            int cellmin = 0;
            int capamin = Int32.MaxValue;
            //lặp tìm battery có công suất nhỏ nhất
            for (int c = 0; c < elems.Count; c++) {
                int capaCur = elems[c].getCurrentCapacity();
                if (capaCur < capamin) {
                    cellmin = c;
                    capamin = capaCur;
                }
            }
            //nạp 1u cho battery có công suất nhỏ nhất tìm được
            return elems[cellmin].charge();
        }
        public bool discharge(List<IBattery> elems) {
            int cellmax = 0;
            int capamax = Int32.MinValue;
            //lặp tìm battery có công suất lớn nhất
            for (int c = 0; c < elems.Count(); c++) {
                int capaCur = elems[c].getCurrentCapacity();
```

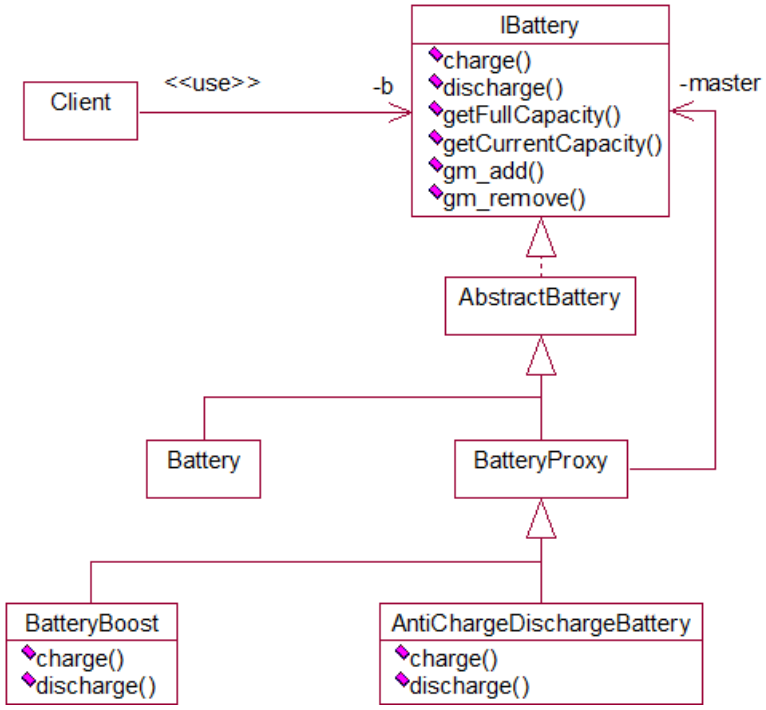
```
            if (capaCur > capamax) {
                cellmax = c;
                capamax = capaCur;
            }
        }
        //thải 1u của battery có công suất lớn nhất tìm được
        return elems[cellmax].discharge();
    }
}
//hoàn chỉnh class đặc tả đối tượng pin tích hợp
class Battery : AbstractBattery  {
    //danh sách các battery thành phần
    private List<IBattery> elems = new List<IBattery>();
    //chiến lược nạp/thải năng lượng
    private IChargeStrategy myStrat;
    //tác vụ khởi tạo battery tích hợp có nCells, mỗi cell có công suất capCells
    //dùng chiến lược nạp/thải được qui định bởi bycycles
    public Battery(int nCells, int capCells, bool bycycles) {
        for (int i = 0; i < nCells; i++) {
            elems.Add(new Cell(capCells));
        }
        if (bycycles)
            myStrat = new StratRoundRobin();
        else
            myStrat = new StratMinMax();
    }
    //tác vụ thêm battery b vào battery hiện hành
    public override void gm_add(IBattery b)  {
        elems.Add(b);
    }
    //tác vụ bớt battery b ra khỏi battery hiện hành
    public override void gm_remove(IBattery b)  {
        elems.Remove(b);
```

```
}
//tác vụ nạp 1u vào battery
public override bool charge() {
    return myStrat.charge(elems);
}
//tác vụ thải 1u khỏi battery
public override bool discharge() {
    return myStrat.discharge(elems);
}
//tác vụ tham khảo công suất hiện hành của battery
public override int getCurrentCapacity()  {
    int sum = 0;
    foreach (IBattery c in elems) {
        sum += c.getCurrentCapacity();
    }
    return sum;
}
//tác vụ tham khảo công suất max của battery
public override int getFullCapacity() {
    int sum = 0;
    foreach (IBattery c in elems) {
        sum += c.getFullCapacity();
    }
    return sum;
}
//tác vụ giải mã trạng thái battery thành chuỗi
public override String ToString()  {
    String buf = "[";
    foreach (IBattery c in elems) {
        buf += c + ", ";
    }
    return buf.Substring(0, buf.Length - 2) + "]";
}
```

}

2. Nội dung 2:

Ta sẽ dùng mẫu thiết kế Proxy để hoặc tăng cường khả năng nạp pin theo kỹ thuật boost hoặc kiểm soát việc nạp/xả khi pin đầy/cạn với lược đồ class như sau :



Ngoài các thành phần đã biết như Client, IBattery, AbstractBattery, Battery, trong lược đồ class trên ta còn dùng các phần tử như sau :

- BatteryProxy : class chứa các thông tin dùng chung cho các Proxy, thí dụ như tham khảo đến đối tượng cần kiểm soát, tăng cường chức năng.
- BatteryBoost, AntiChargeDischargeBattery… : mỗi class này sẽ miêu tả hoạt động Proxy cụ thể nào đó trên đối tượng gốc Battery.

Sau đây là mã nguồn VC# để đặc tả các thành phần trên :

```
//đặc tả class chứa các thông tin chung của các proxy
class BatteryProxy : AbstractBattery {
    //tham khảo đến đối tượng gốc cần proxy
    private IBattery master;
    //tác vụ khởi tạo đối tượng BatteryProxy
    public BatteryProxy(IBattery master) {
      this.master = master;
    }
```

```
        //tác vụ tham khảo công suất max của battery
        public override int getFullCapacity() {
            return master.getFullCapacity();
        }
        //tác vụ tham khảo công suất hiện hành của battery
        public override int getCurrentCapacity()  {
            return master.getCurrentCapacity();
        }
        //tác vụ thải 1u khỏi battery
        public override bool discharge() {
            return master.discharge();
        }
        //tác vụ nạp 1u vào battery
        public override bool charge() {
            return master.charge();
        }
       //tác vụ giải mã trạng thái battery thành chuỗi
        public override String ToString()
        {
            return master.ToString();
        }
}
//đặc tả class BatteryBoost chứa năng lượng dạng nén gấp đôi
class BatteryBoost : BatteryProxy  {
        bool doIt = true;
        bool isEmpty = false;
        public BatteryBoost(IBattery master) : base(master) {
        }
        //tác vụ thải 1u khỏi battery
        public override bool discharge() {
            if (isEmpty) return false;
            if (doIt) {
                isEmpty = base.discharge();
```

```
            }
            doIt = !doIt;
            return isEmpty;
        }
        //tác vụ nạp 1u vào battery
        public override bool charge()  {
            isEmpty = false;
            return base.charge();
        }
    }

    //đặc tả class kiểm soát nạp/xả khi pin đầy/cạn
    class AntiChargeDisChargeBattery : BatteryProxy {
        public AntiChargeDisChargeBattery(IBattery master) : base(master) { }
        //tác vụ nạp 1u vào battery
        public override bool charge()  {
            //kiểm tra nạp quá công suất
            if (getCurrentCapacity() < getFullCapacity())
                return base.charge();
            else
                return false; //báo lỗi chứ không nạp
        }
        //tác vụ thải 1u khỏi battery
        public override bool discharge() {
            //kiểm tra việc thải khi hết năng lượng
            if (getCurrentCapacity() > 0)
                return base.discharge();
            else
                return false; //báo lỗi chứ không xả
        }
    }
```

3. Chương trình test :

Code của chương trình nhỏ để dùng và kiểm tra pin tích hợp Battery có dùng các

chiến lược nạp/xả không, các loại pin Proxy như BatteryBoost, AntiChargeDisChargeBattery có thực hiện việc tăng cường, kiểm soát pin gốc hay không như sau :

```
static void Main(string[] args) {
    //thử tạo 1 Cell có 2u
    IBattery b1 = new Cell(2);
    //hiển thị trình trạng pin hiện hành
    Console.WriteLine("DL max của b1 = " + b1.getFullCapacity()
            + ", DL hiện hành = " + b1.ToString());
    //thử nạp thêm 1u và hiển thị kết quả để kiểm tra
    b1.charge();
    Console.WriteLine("Dung lượng của b1 sau khi nạp thêm 1u = "
            + b1.ToString());
    //thử xả 4u và hiển thị kết quả để kiểm tra
    b1.discharge(); b1.discharge(); b1.discharge(); b1.discharge();
    Console.WriteLine("Dung lượng của b1 sau khi xả 4u = " + b1.ToString());
    //thử tạo pin tích hợp
    b1 = new Cell(10);
    IBattery b2 = new Battery(3, 5, false);        //dùng chiến lược nạp/xả MinMax
    b2.gm_add(b1); b2.gm_add(b2);
    Console.WriteLine("Trạng thái của b2 = " + b2.ToString());
    //thử xả 2u và hiển thị kết quả để kiểm tra
    b2.discharge(); b2.discharge();
    Console.WriteLine("b2 sau khi xả 2u = " + b2.ToString());
    //thử nạp 2u và hiển thị kết quả để kiểm tra
    b2.charge(); b2.charge();
    Console.WriteLine("b2 sau khi nạp 2u = " + b2.ToString());

    IBattery b3 = new Battery(2, 10, true);        //dùng chiến lược nạp/xả
    RoundRobin
    b3.gm_add(b1); b3.gm_add(b2);
    Console.WriteLine("Trạng thái của b3 = " + b3.ToString());
    //thử xả 2u và hiển thị kết quả để kiểm tra
    b3.discharge(); b3.discharge();
    Console.WriteLine("b3 sau khi xả 2u = " + b3.ToString());
    //thử nạp 2u và hiển thị kết quả để kiểm tra
    b3.charge(); b3.charge();
    Console.WriteLine("b3 sau khi nạp 2u = " + b3.ToString());

   //kiểm tra pin BatteryBoost
    b2 = new BatteryBoost(b2);
    …

   //kiểm tra pin AntiChargeDisChargeBattery
```

```
        b3 = new BatteryBoost(b3);
        …
        Console.Read();
    }
```