マルチキャスト

デリゲートの合成

デリゲートが関数ポインタのように、メソッドの位置をあらわすものということは理解できたでしょう
そして、これは一般にコールバックの登録などに用いることができます

デリゲート型は、さらにこれまでの言語にはない特殊な性質があります
じつは、あるデリゲート型は他のデリゲート型と加算演算子で結合できるのです
この機能をマルチキャストデリゲートと呼びます

結合されたデリゲートは、配列のように複数のデリゲートがパックされています
このデリゲートのメソッドを実行すれば、登録されている全てのデリゲートが実行されます

```
class Test {
 delegate void KittyCallback();
 static void Main() {
  KittyCallback kitty = new KittyCallback(Kitty);
  kitty += new KittyCallback(Temp);
  kitty();
 }
 static void Kitty() {
  System.Console.WriteLine("Kitty on your lap");
 }
 static void Temp() {
  System.Console.WriteLine("Silver Line");
 }
}
```

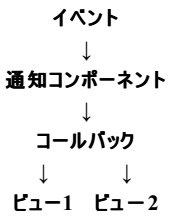
このプログラムのデリゲート型変数 kitty は最初、Kitty() メソッドを格納しています
そして、さらにこの変数に Temp() メソッドを持つデリゲート型を加算しています
これを実行すると、次のような結果になりました

Kitty on your lap
Silver Line

これを見てわかるように、マルチキャストデリゲートは一度に登録されている
全てのデリゲートを、格納順にシーケンシャルに実行します

なぜこのような機能が必要なのか、最初は戸惑うかもしれません
しかし、高度なオブジェクト指向プログラマなどは、すぐにこの意味を理解できるでしょう

例えば、GUI システムのイベント呼び出し機能でよく行われるのですが
あるイベントが、他の多くのグラフィックコンポーネントに影響するという処理を実現する時
多くのコンポーネントが同時に、コールバックの登録をすることになります
MVCアーキテクチャなどを勉強すれば、この概念をより具体的に理解できます



実際にはもう少し複雑ですが、GUI システムのイベント呼び出しの遷移図です
何らかのコンポーネントがイベントを受けると、登録されているコールバックを呼び出します
登録されるメソッドは複数でも問題がないため(むしろ、複数の登録が望まれる)
コンポーネントは、このようなモデリングを実現しなければならないのです

代表的なのは、画像処理ソフトの解像度選択スライダーです
スライダーを動かせば、プレビュー画像の解像度を変更するこのコンポーネントは
スライダーを動かすというイベントで、スライダーの移動やプレビュー画像の大きさ
それに関連する様々なコンポーネントのビューを同時に変更する必要があります

Java の場合、コンポーネントが登録されるコールバックメソッドを受け取り
メソッドで動的に配列を処理して扱う必要がありました
この作業は、少なくとも初心者の方には難しく感じてしまうでしょう

デリゲートを用いれば、このような面倒は一気に解決します
単純にコールバックするメソッドをマルチキャストデリゲートでパックするだけで
後は、簡単に呼び出すことができるからです

しかし、コンポーネントの状況が変更すれば
イベントの通知の必要性がなくなり、コールバックを解除したくなるかもしれません
このような場合に応じて、マルチキャストデリゲートは
特定のデリゲートを解除することができるように設計されています

マルチキャストデリゲートから特定のデリゲートを解除するには
解除したいデリゲートをマルチキャストデリゲートから減算します
この処理は、非常に直感的で理解しやすいと思います

```
delegate void KittyCallback();

class Kitty {
 private string str;
 public Kitty(string str) {
  this.str = str;
 }
 public static implicit operator KittyCallback(Kitty obj) {
  return new KittyCallback(obj.Write);
 }
 public void Write() {
  System.Console.WriteLine(str);
 }
}

class Test {
 static void Main() {
  KittyCallback[] obj = new KittyCallback[] {
   new Kitty("Kitty on your lap") ,
   new Kitty("Selver Gene") ,
   new Kitty("Tokyo mew mew")
  };

  KittyCallback kitty = null;
  foreach(KittyCallback tmp in obj)
   kitty += tmp;
  kitty();

  System.Console.WriteLine("¥n----減算後----");
  kitty -= obj[1];
  kitty();
 }
}
```

このプログラムでは、マルチキャストデリゲートの kitty を作成し
作成したデリゲートを foreach ですべて kitty に加算します

その後、この kitty からデリゲート obj1を減算しています
これによって obj1 は kitty から外されます
このプログラムを実行すると、次のような結果になります

Kitty on your lap
Selver Gene
Tokyo mew mew

----減算後----
Kitty on your lap
Tokyo mew mew

この結果を見れば、意図どおりにデリゲートが解除されていることが確認できます

前のページへ　　戻る　　次のページへ