Chapter-4 Observerパターン

6.1【ハンズオン】カスタムメモリアロケータを利用したObserverパターン

step-1 カスタムアロケーター用のメンバ変数を追加。

[ObserverList.h]

```
// step-1 カスタムアロケーター用のメンバ変数を追加。
Node m_nodeArray[MAX_NODE];          // ノードの配列
Node* m_unuserNodeArray[MAX_NODE];   // 未使用ノードの配列
int m_numUnuseNode = 0;                      // 未使
```

step-2 カスタムメモリアロケータ用のメモリ確保関数と開放関数を定義する

[ObserverList.h]

```
// step-2 カスタムメモリアロケータ用のメモリ確保関数と開放関数を定義する
// 新しいノードを確保。
Node* AllocNode();
// ノードを開放。
void FreeNode(Node* n);
```

step-3 カスタムアロケーター用のメンバを初期化する。

[ObserverList.cpp]

```
// step-3 カスタムアロケーター用のメンバを初期化する。
// 未使用ノードのポインタの配列を初期化する。
// 最初は全て未使用。
for (int i = 0; i < MAX_NODE; i++) {
    m_unuserNodeArray[i] = &m_nodeArray[i];
}
// 現在の未使用ノードの数は最大数。
m_numUnuseNode = MAX_NODE;
```

step-4 新しいノードを確保する処理を実装。

[ObserverList.cpp]

```
// step-4 新しいノードを確保する処理を実装。
if (m_numUnuseNode == 0) {
```

```
    // 未使用ノードがない。
    return nullptr;
  }
  // 未使用ノードから新しいノードをもらう。
  Node* allocNode = m_unuserNodeArray[m_numUnuseNode-1];
  allocNode->nextNode = nullptr;
  allocNode->value = nullptr;

  // 未使用ノードを１つ減らす
  m_numUnuseNode--;
  return allocNode;
```

step-5 ノードを解放する処理を実装。

[ObserverList.cpp]

```
// step-5 ノードを解放する処理を実装。
// ノードを未使用リストに返却する。
m_unuserNodeArray[m_numUnuseNode] = n;
// 未使用ノードの数を１つ増やす。
m_numUnuseNode++;
```

step-6 カスタムメモリアロケーターを使って新しいノードを確保する。

[ObserverList.cpp]

```
// step-6 カスタムメモリアロケーターを使って新しいノードを確保する。
Node* newNode = AllocNode();
```

step-7 カスタムメモリアロケーターに新しいノードを返却する。

[ObserverList.cpp]

```
// step-7 カスタムメモリアロケーターに新しいノードを返却する。
FreeNode(p);
```