G64OOS (Spring 2014)

Lab 06

Relationships

Peer-Olaf Siebers

The University of Nottingham
UNITED KINGDOM · CHINA · MALAYSIA

Motivation

- Get you prepared for the coursework :-)
  - Provide examples for implementing class relationships
  - Let you apply what you have learned in a small project

From UML to Code
Implementing Relationships

A good way to learn this is to look at design pattern implementations

Composition Example

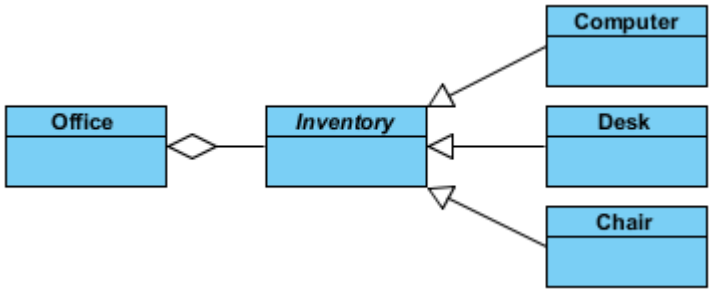
```
1   #include <iostream>
2   using namespace std;
3
4   class Office{
5   private:
6       int workSpaces;
7   public:
8       Office(int w);
9       int getWorkSpaces();
10  };
11
12  class OfficeBlock{
13  private:
14      Office office1,office2;
15      int houseNo;
16  public:
17      OfficeBlock(int hn);
18      void printOfficeBlockDetails();
19  };
20
21  int main(){
22      OfficeBlock myBase(10);
23      myBase.printOfficeBlockDetails();
24      return 0;
25  }
26
27  Office::Office(int w){workSpaces=w;}
28  int Office::getWorkSpaces(){return workSpaces;}
29  OfficeBlock::OfficeBlock(int hn):office1(4),office2(5){houseNo=hn;}
30  void OfficeBlock::printOfficeBlockDetails(){
31      cout<<"OfficeBlock "<<houseNo<<": office1 workspaces = "<<office1.getWorkSpaces()
32      <<", office2 workspaces = "<<office2.getWorkSpaces()<<endl;
33  }
```


```
OfficeBlock 10: office1 workspaces = 4, office2 workspaces = 5
```

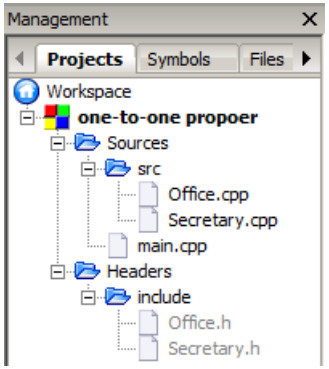
Aggregation Example

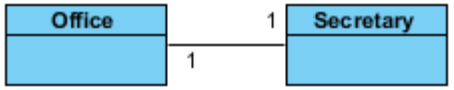
- Your job in the lab ...
  - Here is a code fragment to help you get started

```
1   #include <iostream>
2   using namespace std;
3
4   class Desk{
5   };
6
7   class Office{
8   private:
9       Desk* theDesk;
10  public:
11      void addDesk(Desk* desk){theDesk=desk;}
12      void removeDesk(){theDesk=NULL;}
13  };
14
15  int main(){
16      Office* office=new Office();
17      Desk* desk=new Desk();
18      office->addDesk(desk);
19      office->removeDesk();
20      return 0;
21  }
```



One-to-One Association





include\Office.h | src\Office.cpp | include\Secretary.h | src\Secretary.cpp | **main.cpp**

```
1   #include "Office.h"
2   #include "Secretary.h"
3   #include <iostream>
4   using namespace std;
5
6   int main(){
7       Office* facultyOffice=new Office(101);
8       Secretary* facultySecretary=new Secretary((char*)"facultySecretary"); // casting string to avoid warning
9       facultyOffice->addSecretary(facultySecretary);
10      cout<<"The "<<facultySecretary->getRole()<<" is in room "<<facultySecretary->getOffice()->getRoom()<<endl;
11      cout<<"Room "<<facultyOffice->getRoom()<<" is home to the "<<facultyOffice->getSecretary()->getRole()<<endl;
12      delete facultySecretary;
13      delete facultyOffice;
14      return 0;
15  }
16
```

```
The facultySecretary is in room 101
Room 101 is home to the facultySecretary
```

One-to-One Association

include\Office.h | src\Office.cpp | include\Secretary.h | src\Secretar

```
1  #ifndef OFFICE_H
2  #define OFFICE_H
3  // OFFICE.H
4  class Secretary; // forward declaration
5  class Office{
6  private:
7      int room;
8      Secretary* secretary;
9  public:
10     Office(int vRoom);
11     ~Office();
12     int getRoom();
13     Secretary* getSecretary();
14     void addSecretary(Secretary* vSecretary);
15 };
16 #endif
17
```

include\Office.h | src\Office.cpp | include\Secretary.h

```
1  #ifndef SECRETARY_H
2  #define SECRETARY_H
3  // SECRETARY.H
4  class Office; // forward declaration
5  class Secretary{
6  private:
7      char role[80];
8      Office* office;
9  public:
10     Secretary(char* vRole);
11     ~Secretary();
12     char* getRole();
13     void addOffice(Office* vRoom);
14     Office* getOffice();
15 };
16 #endif
```

One-to-One Association

include\Office.h | **src\Office.cpp** | include\Secretary.h | src\Secretary.cpp

```cpp
#include "Office.h"
#include "Secretary.h"
// OFFICE.CPP
Office::Office(int vRoom){
    room=vRoom;
}
Office::~Office(){
}
int Office::getRoom(){
    return room;
}
Secretary* Office::getSecretary(){
    return secretary;
}
void Office::addSecretary(Secretary* vSecretary){
    secretary=vSecretary;
    secretary->addOffice(this);
}
```

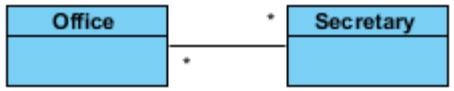
include\Office.h | src\Office.cpp | include\Secretary.h | **src\Secretary.cpp**

```cpp
#include "Office.h"
#include "Secretary.h"
#include <cstring>
// SECRETARY.CPP
Secretary::Secretary(char* vRole){
    strcpy(role,vRole);
}
Secretary::~Secretary(){
}
char* Secretary::getRole(){
    return role;
}
Office* Secretary::getOffice(){
    return office;
}
void Secretary::addOffice(Office* vRoom){
    office=vRoom;
}
```

Many-to-Many Association

include\Office.h | src\Office.cpp | include\Secretary.h | src\Secretary.cpp | main.cpp

```
1  #include <iostream>
2  #include "Office.h"
3  #include "Secretary.h"
4  using namespace std;
5  // TEST.CPP
6  int main(){
7      Office* office101=new Office(101);
8      Office* office102=new Office(102);
9      Secretary* secretary1=new Secretary((char*)"secretary1");
10     Secretary* secretary2=new Secretary((char*)"secretary2");
11     office101->addSecretary(secretary1);
12     office101->addSecretary(secretary2);
13     office102->addSecretary(secretary2);
14 //     secretary2->addOffice(office102);
15     cout<<secretary1->getOfficesToString()<<endl;
16     cout<<secretary2->getOfficesToString()<<endl;
17     cout<<office101->getSecretariesToString()<<endl;
18     cout<<office102->getSecretariesToString()<<endl;
19     delete office101;
20     delete office102;
21     delete secretary1;
22     delete secretary2;
23     return 0;
24 }
```




```
secretary1: 101
secretary2: 101 102
101: secretary1 secretary2
102: secretary2
```

Many-to-Many Association

include\Office.h | src\Office.cpp | include\Secretary.h | src\Secretar

```
1  #ifndef OFFICE_H
2  #define OFFICE_H
3  #include <iostream>
4  #include <vector>
5  using namespace std;
6  // OFFICE.H
7  class Secretary;
8  class Office{
9  private:
10     int room;
11     vector<Secretary> secretaryList;
12 public:
13     Office(int vRoom);
14     ~Office();
15     int getRoom();
16     void addSecretary(Secretary* vSecretary);
17     vector<Secretary> getSecretaries();
18     string getSecretariesToString();
19 };
20 #endif
```

include\Office.h | src\Office.cpp | include\Secretary.h

```
1  #ifndef SECRETARY_H
2  #define SECRETARY_H
3  #include <iostream>
4  #include <vector>
5  using namespace std;
6  // SECRETARY.H
7  class Secretary{
8  private:
9      char role[80];
10     vector<Office> officeList;
11 public:
12     Secretary(char* vRole);
13     ~Secretary();
14     char* getRole();
15     void addOffice(Office* vRoom);
16     vector<Office> getOffices();
17     string getOfficesToString();
18 };
19 #endif
```

Many-to-Many Association

include\Office.h | **src\Office.cpp** | include\Secretary.h | src\Secretary.cpp

```cpp
#include "Office.h"
#include "Secretary.h"
#include <sstream>
#include <cstring>
// OFFICE.CPP
Office::Office(int vRoom){
    room=vRoom;
}
Office::~Office(){
}
int Office::getRoom(){
    return room;
}
vector<Secretary> Office::getSecretaries(){
    return secretaryList;
}
void Office::addSecretary(Secretary* vSecretary){
    secretaryList.push_back(*vSecretary);
    vSecretary->addOffice(this);
}
string Office::getSecretariesToString(){
    stringstream ss;
    ss<<getRoom()<<": ";
    for(unsigned int i=0; i<secretaryList.size();i++){
        ss<<secretaryList[i].getRole()<<" ";
    }
    return ss.str();
}
```

include\Office.h | src\Office.cpp | include\Secretary.h | **src\Secretary.cpp**

```cpp
#include "Office.h"
#include "Secretary.h"
#include <sstream>
#include <cstring>
// SECRETARY.CPP
Secretary::Secretary(char* vRole){
    strcpy(role,vRole);
}
Secretary::~Secretary(){
}
char* Secretary::getRole(){
    return role;
}
vector<Office> Secretary::getOffices(){
    return officeList;
}
void Secretary::addOffice(Office* vRoom){
    officeList.push_back(*vRoom);
//    vRoom->addSecretary(this);
}
string Secretary::getOfficesToString(){
    stringstream ss;
    ss<<getRole()<<": ";
    for(unsigned int i=0; i<officeList.size();i++){
        ss<<officeList[i].getRoom()<<" ";
    }
    return ss.str();
}
```

Apply what you have learned

- Check out the lab manual for further instructions ...