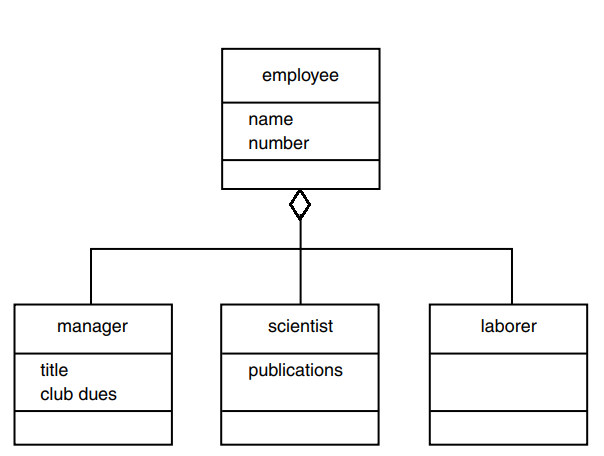
***Question*:-**

We have a big Company of Widgets and there are hundreds of person are working in the company. We want to have detailed record of persons to manage the database of the company and to want to add the information of the employee of this company.



***Solution:-***

In this example, **Aggregation** has been used to create new functionality to the class. Our example models a database of employees of a widget company. We’ve simplified the situation so that only three kinds of employees are represented.

Managers manage, scientists perform research to develop better widgets, and laborers operate the dangerous widget-stamping presses.

The database stores a name and an employee identification number for all employees, no matter what their category. However, for managers, it also stores their titles and golf club dues. For scientists, it stores the number of scholarly articles they have published. Laborers need no additional data beyond their names and numbers.

Our example program starts with a class employee. This class handles the employee’s last name and employee number. From this class three other classes are aggregated: manager, scientist, and laborer. The manager and scientist classes contain additional information about these categories of employee, and member functions to handle this information.

The main() part of the program declares three objects of different classes: one managers, a scientist, and a laborer. (Of course many more employees of each type could be defined, but the output would become rather large.) It then calls the **getdata()** member functions to obtain information about each employee, and the **display()** function to display this information.

***Code:-***

//Program starts from here,

#include<iostream>

using namespace std;

**class employee**

{

private:

char name[40]; //employee name

unsigned long number; //employee number

public:

void getdata()

{

cout<<" Enter Name:"<<endl;

cin>>name;

cout<<" Enter Number:";

cin>>number;

cout<<endl;

}

void display()

{

cout<<" Name: "<<name<<endl;

cout<<" Employee Number: "<<number<<endl;

}

};

////////////////////////////////////////

**class manager**

{

private:

char title[40]; //branch-manager,president , etc.

long dues; //golf club dues

employee emp; //object of employee(aggregation)

public:

void getdata()

{

emp.getdata();

cout<<" Enter title: ";

cin>>title;

cout<<" Enter golf club dues:";

cin>>dues;

cout<<endl;

}

void display()

{

cout<<"\*\*\*Manager Information\*\*\*"<<endl<<endl;

emp.display();

cout<<" Title: " <<title<<endl;

cout<<" Golf club dues: "<<dues<<endl<<endl;

}

};

////////////////////////////////////////////

**class scientist**

{

private:

int pubs; //number of publications

employee emp; //object of employee(aggregation)

public:

void getdata()

{

emp.getdata();

cout<<" Enter number of publications: ";

cin>>pubs;

}

void display()

{

cout<<"\*\*\*Scientist Information\*\*\*"<<endl<<endl;

emp.display();

cout<<" Number of Publications:"<<pubs<<endl<<endl;

}

};

/////////////////////////////////////////////////////////

**class laborer**

{

private:

employee emp; //object of employee(aggregation)

public:

void getdata()

{

emp.getdata();

//laborer class doesn't need additional info. other than name and no.

}

void display()

{

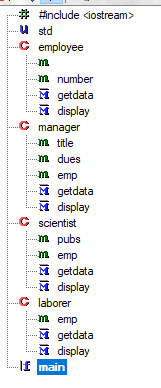
cout<<"\*\*\*Laborer Information\*\*\*"<<endl;

emp.display();

cout<<endl<<endl;

}

};

/////////////////////////////////////////////////////////////

**int main()**

{

manager m1;

scientist s1;

laborer l1;

cout<<"Enter Data for Manager:"<<endl;

m1.getdata();

m1.display();

cout<<"--------------------------------"<<endl;

cout<<"Enter Data for Scientist:"<<endl;

s1.getdata();

s1.display();

cout<<"--------------------------------"<<endl;

cout<<"Enter Data for laborer:"<<endl;

l1.getdata();

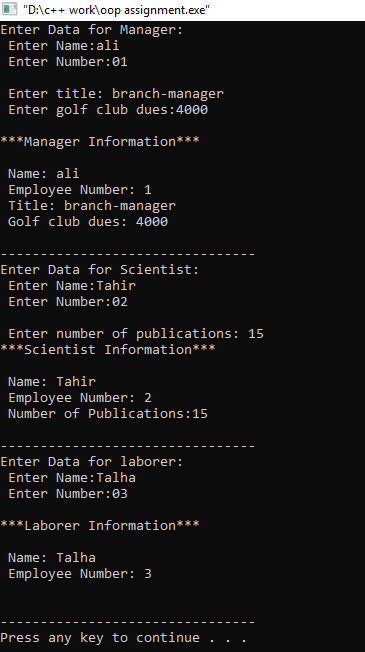
l1.display();

cout<<"--------------------------------"<<endl;

return 0;

}

**Output:-**



***Alternate Approach:-***

We can also used Inheritance to solve this problem. In inheritance we will used employee as a base class and then we will drived: manager, scientist and laborer class.

In employee class we will use name and emp. No. as a protected data number. And two member function getdata() and display() function to get the input and output as well as.

Then we derived the manager class from base class Employee to extended the functionality of employee class. And similarly we do the same with derived scientist class to add the additional information to the employee class. and the same with the derived laborer class. In every derived class we use the scope resolution operator with the getdata() function and with the display() function to access the employee class member function too. And their output will remain same.

**Code:-**

#include<iostream>

using namespace std;

class employee

{

protected:

char name[40]; //employee name

unsigned long number; //employee number

public:

void getdata()

{

cout<<" Enter Name:";

cin>>name;

cout<<" Enter Number:";

cin>>number;

cout<<endl;

}

void display()

{

cout<<" Name: "<<name<<endl;

cout<<" Employee Number: "<<number<<endl;

}

};

////////////////////////////////////////

class manager:public employee //derived class

{

protected:

char title[40]; //branch-manager,president , etc.

long dues; //golf club dues

public:

void getdata()

{

employee::getdata();

cout<<" Enter title: ";

cin>>title;

cout<<" Enter golf club dues:";

cin>>dues;

cout<<endl;

}

void display()

{

cout<<"\*\*\*Manager Information\*\*\*"<<endl<<endl;

employee::display();

cout<<" Title: " <<title<<endl;

cout<<" Golf club dues: "<<dues<<endl<<endl;

}

};

////////////////////////////////////////////

class scientist:public employee

{

protected:

int pubs; //number of publications

public:

void getdata()

{

employee::getdata();

cout<<" Enter number of publications: ";

cin>>pubs;

}

void display()

{

cout<<"\*\*\*Scientist Information\*\*\*"<<endl<<endl;

employee::display();

cout<<" Number of Publications:"<<pubs<<endl<<endl;

} };

/////////////////////////////////////////////////////////

class laborer:public employee

{

//laborer class doesn't need additional info. other than name and no.

};

int main()

{

manager m1;

scientist s1;

laborer l1;

cout<<"Enter Data for Manager:"<<endl;

m1.getdata();

m1.display();

cout<<"--------------------------------"<<endl;

cout<<"Enter Data for Scientist:"<<endl;

s1.getdata();

s1.display();

cout<<"--------------------------------"<<endl;

cout<<"Enter Data for laborer:"<<endl;

l1.getdata();

l1.display();

cout<<"--------------------------------"<<endl;

return 0;

}

***Result:-***

* **Aggregation:** create new functionality by taking other classes and combining them into a new class. While **inheritance:** extend the functionality of a class by creating a subclass. Override superclass members in the subclasses to provide new functionality.
* Inheritance breaks encapsulation. By inheriting from a class you're coupling child class with number of potential implementation details of the parent.
* There is no conflict between methods/properties names, which might occur with inheritance.
* Inheritance establishes "IS-A" relationship between classes and composition/aggregation is "HAS-A" relationship.
* There is no ambiguity issue in the aggregation approach.