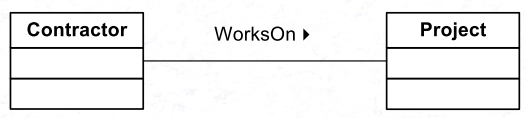
**Implementation of Association**



**Contractor.h**

#ifndef \_CONTRACTOR\_H\_

#define \_CONTRACTOR\_H\_

class Project; // forward delcaration

#include <iostream.h>

#include "Project.h"

class Contractor {

private:

char\* name;

Project\* project;

double rate;

double time;

public:

Contractor(char\* n);

void setProject(Project\* p);

void updateTime(double t);

void billProject();

friend ostream& operator<<(ostream& outstr, Contractor& cont);

};

#endif

**Contractor.cpp**

#include "Contractor.h"

Contractor::Contractor(char\* n)

{

name = n; rate = 100; time = 0;

}

void Contractor::setProject(Project\* p)

{

project = p;

}

void Contractor::updateTime(double t)

{

time += t;

}

void Contractor::billProject()

{

project->logExpenses(rate \* time);

}

ostream& operator<<(ostream& outstr, Contractor& cont)

{ outstr << cont.name << " worked " << cont.time << " hours at " << cont.rate << " per hour";

return outstr;

}

**Project.h**

#ifndef \_PROJECT\_H\_

#define \_PROJECT\_H\_

class Contractor; // forward declaration

#include <iostream.h>

#include "Contractor.h"

class Project {

private:

char\* name;

double expenses;

Contractor\* contractor;

public:

Project(char\* n);

void setContractor(Contractor\* c);

void logExpenses(double amount);

void authorizeTime(double time);

friend ostream& operator<<(ostream& outstr, Project& proj);

};

#endif

**Project.cpp**

#include "Project.h"

Project::Project(char\* n)

{

name = n; expenses = 0;

}

void Project::setContractor(Contractor\* c)

{

contractor = c;

}

void Project::logExpenses(double amount)

{

expenses += amount;

}

void Project::authorizeTime(double time)

{

contractor->updateTime(time);

}

ostream& operator<<(ostream& outstr, Project& proj)

{ outstr << "Project: " << proj.name << " has " << proj.expenses << " in expenses";

return outstr;

}

**Association.cpp**

#include <iostream.h>

#include "Contractor.h"

#include "Project.h"

int main(void)

{

//instantiate objects

Contractor cont("Fred");

Project proj("Compiler");

// build association relationship

cont.setProject(&proj);

proj.setContractor(&cont);

proj.authorizeTime(50);

cont.billProject();

cout << proj << endl;

cout << cont << endl;

return 0;

}