# Класс хоорондын харьцаа
# (Лаборатори №11)

О. Ихбаяр

ХШУИС, Програм хангамж, 3-р түвшин, 17b1num2575@stud.num.edu.mn

## 1. ОРШИЛ

Класс хоорондын харьцаа гэж юу болох түүнчлэн түүний төрлүүдийнх нь талаар авч үзэх болно.

## 2. ЗОРИЛГО

Удамшлын харьцааг зөв тодорхойлон түүнийгээ эх кодон дээрээ хэрэглэж сурах зорилготой.

## 3. ОНОЛЫН СУДАЛГАА

Классууд нь 2 төрлийн харьцаа үсгэдэг.

### 3.1 Бүрдэл харьцаа

Классын шинж буюу гишүүн өгөгдөл нь өөр классынх байж болох ба нарийн бүтэцтэй бүрдмэл классыг үүсгэхэд ашигладаг. Бүрдэл харьцааны утга нь тийм юмтай байх. Жишээ нь: Комьпютер нь Ram – тай байна

```
Class ram {
        Int memory;
}
```

```
Class computer {
        Ram r1;
        Ram r2;
}
```

### 3.2 Удамшлын харьцаа

- Тэр бол тэр (is a) гэдэг харьцаа үүсгэнэ. Жишээ нь морь бол амьтан.
- (…-ны) төрлийнх буюу is a kind of харьцаа үүсгэнэ. Энэ нь сурагч бол хүн гэхдээ сүүн тэжээлтны төрөл.

Энэ харьцаа нь класс хооронд, объект хооронд байх “ерөнхийллөөс нарийсгал” холбоо юм.

# 4. ХЭРЭГЖҮҮЛЭЛТ

## 4.1 Бүрдэл харьцаа

```
//
class Employee : public Person{
        private :
                string companyID;
                string title;
                Date startDate;
                Spouse *spouse;
                vector<Child *> children;
                Division *division;
                vector<JobDescription *> jobs;
        ..................................................................
```

Тодорхойлолт: person классаас ажилтан классыг удамшуулаад түүндээ харгалзах гэр бүлийн мэдээлэл болон ажлын мэдээллийн харьцаануудыг тодорхойлж өгсөн.

```
name : bat    ssnum : ek1   age : 20
        companyID : num1   start date : 1/1/2011  title : title
        divition : div 1    jobs : {
                                    job 1 }
        Spouse : batmaa   spouse date : 3/7/2010     Children : {
                                                        bat1   4    tobot
                                                        bat2   3    barbie
                                                        bat3   6    lego }

name : dorj    ssnum : el2   age : 21
        companyID : num1   start date : 12/31/2012  title : title2
        divition : div 2    jobs : {
                                    job 2
                                    job 2.1 }
        Spouse : dorjmaa   spouse date : 11/20/2015     Children : {
                                                        dorj1   5    mega
                                                        dorj2   2    winks }
```

# 5. ДҮГНЭЛТ

Энэхүү лабораторын ажиллаар класс хоорондын харьцаа гэж юу болох түүнчлэн түүний төрлүүдийнх нь талаар тодорхой хэмжээний ойлголттой болсон.

# 6. АШИГЛАСАН МАТЕРИАЛ


1. Объект хандлагат технологийн C++ програмчлал, Ж.Пүрэв, 2008, Улаанбаатар.


# 7. ХАВСРАЛТ

```
#include <iostream>

#include <string>

#include "division.h"

#include "jobDescription.h"

#include "child.h"
```

```
#include "employee.h"
#include "spouse.h"
using namespace std;
int main() {
	Division div1("div 1"), div2;
	div2.setDivisionName("div 2");
	JobDescription job1("job 1"), job2( "job 2"), job21;
	job21.setDescription("job 2.1");
	Employee emp1("bat", "ek1", 20, "num1", "title", 1, 1, 2011), emp2(&div2, &job2);
	emp1.setDivision(&div1);
	emp1.addJob(&job1);
	emp2.setName("dorj");
	emp2.setSSNum("el2");
	emp2.setAge(21);
	emp2.setCompanyID("num1");
	emp2.setTitle("title2");
	emp2.setStartDate(12, 31, 2012);
	emp2.addJob(&job21);
	Spouse s1("batmaa", "ss1", 19 , 3, 7, 2010), s2("dorjmaa", "ss2", 20 , 11, 20, 2015);
	emp1.setSpouse(&s1);
	emp2.setSpouse(&s2);
	Child b1("bat1", "ekk1", 4, "tobot"), b2("bat2", "ekk2", 3, "barbie"), b3("bat3", "ekk3", 6,
"lego"),
		a1("dorj1", "ell1", 5, "mega"), a2("dorj2", "ell2", 2, "winks");
	emp1.addChild(&b1);
	emp1.addChild(&b2);
	emp1.addChild(&b3);
	emp2.addChild(&a1);
	emp2.addChild(&a2);
```

```
	emp1.print();
	emp2.print();
	return 0;
}
```

---

```
class Employee : public Person{
	private :
		string companyID;
		string title;
		Date startDate;

		Spouse *spouse;
		vector<Child *> children;
		Division *division;
		vector<JobDescription *> jobs;
	public :
		Employee();
		Employee(string pname,string pssnum, int page, string id, string s, int m, int d, int y);
		Employee(Division * div, JobDescription * job);
		~Employee();
		string getCompanyID();
		string getTitle();
		Date getStartDate();
		Division * getDivision();
		JobDescription ** getJobs();
		Spouse * getSpouse();
		Child** getChildren();
```

```
        void setCompanyID(string id);
        void setTitle(string s);
        void setStartDate(int m, int d, int y);
        void setDivision(Division * div);
        void addJob(JobDescription * job);
        void setSpouse(Spouse * sp);
        void addChild(Child * ch);
        void print();
};
Employee :: Employee(){
    companyID = " ";
    title = " ";
    Division div(" ");
    division = &div;
    JobDescription job(" ");
    jobs.push_back(&job);
    div.setEmployee(this);
    job.setEmployee(this);
    spouse = NULL;
    children.clear();
}
Employee :: Employee(string pname, string pssnum, int page, string id, string s, int m, int d, int y)
    : Person (pname, pssnum, page){
        companyID =id;
        title = s;
        startDate.month = m;
        startDate.day = d;
        startDate.year = y;
```

```
		Division divv(" ");
		divv.setEmployee(this);
		division = &divv;
		spouse = NULL;
		children.clear();
	}

Employee :: Employee(Division * div, JobDescription * job){
	companyID = " ";
	title = " ";
	div->setEmployee(this);
	job->setEmployee(this);
	division = div;
	jobs.push_back(job);
	spouse = NULL;
	children.clear();
}
Employee :: ~Employee(){
}
string Employee :: getCompanyID(){
	return companyID;
}
string Employee :: getTitle(){
	return title;
}
Date Employee :: getStartDate(){
	return startDate;
```

```
}
Division * Employee :: getDivision(){
	return division;
}
JobDescription ** Employee :: getJobs(){
	return jobs.data();
}
Spouse * Employee :: getSpouse(){
	return spouse;
}
Child** Employee :: getChildren(){
	return children.data();
}
void Employee :: setCompanyID(string id){
	companyID = id;
}
void Employee :: setTitle(string s){
	title = s;
}
void Employee :: setStartDate(int m, int d, int y){
	startDate.month = m;
	startDate.day = d;
	startDate.year = y;
}
void Employee :: setDivision(Division * div){
	div->setEmployee(this);
	division = div;
}
```

```
void Employee :: addJob(JobDescription * job){
        job->setEmployee(this);
        jobs.push_back(job);
}
void Employee :: setSpouse(Spouse * sp){
        sp->setEmployee(this);
        spouse = sp;
}
void Employee :: addChild(Child * ch){
        ch->setEmployee(this);
        children.push_back(ch);
}
void Employee :: print(){
        cout<< "name : " << this->getName() << "    ssnum : " << this->getSSNum() <<
            "   age : " << this->getAge() << "\n\t companyID : " << this->getCompanyID() <<
                "   start date : " ;
                 this->getStartDate().display1();
        cout  << "  title : " << this->getTitle() <<
                "\n\t divition : " << this->division->getDivisionName();
        cout << "    jobs : {" ;
        for(int i = 0 ; i < this->jobs.size(); i++) {
                cout << "\n\t\t\t\t\t";
                cout << this->jobs[i]->getDescription();
        }
        cout << " }   ";
        cout << "\n\t Spouse : " << this->spouse->getName() << "   spouse date : " ;
        this->spouse->getAnniversaryDate().display1();
        cout << "     Children : {";
```

```
	for(int i = 0 ; i < this->children.size(); i++) {
		cout << "\n\t\t\t\t\t\t\t\t\t";
		cout << this->children[i]->getName() << "   " << this->children[i]->getAge() << "
" << this->children[i]->getFavoriteToy();
	}
	cout << " }   ";
	cout << endl << endl;
}
```

---

```
class Child : public Person {
	private:
		string favoriteToy;
		Employee *emp;
	public:
		Child();
		Child(string pname, string pssnum, int page, string favToy);
		~Child();
		string getFavoriteToy();
		Employee * getEmployee();
		void setFavoriteToy(string favToy);
		void setEmployee(Employee * e);
};
Child :: Child(){
	favoriteToy = " ";
}
Child :: Child(string pname, string pssnum, int page, string favToy)
		: Person(pname, pssnum, page){
			favoriteToy = favToy;
		}
```

```
Child :: ~Child(){
}
string Child :: getFavoriteToy(){
        return favoriteToy;
}
Employee * Child :: getEmployee(){
        return emp;
}
void Child :: setFavoriteToy (string favToy){
        favoriteToy = favToy;
}
void Child :: setEmployee(Employee * e){
        emp = e;
}
```

---

```
class Date
{
public:
   int month;
   int day;
   int year;
   Date();
   Date(int month,int day,int year);
   void display1();
   void display2();
   void increment();
   Date &operator=(const Date &T);
};
```

```
Date::Date()
{
   month = 1;//default month value
   day = 1;//default day value
   year = 2000;//default year value
}
//postcondition: a Date with a month, day and year has been created
//precondition: Date will check if any of the conditions have been violated
Date::Date(int Month,int Day,int Year)
{
   if((Month < 1||Month > 12)||(Day < 1||Day > 31)||(Year < 1900||Year > 2020))
   {
      std::cout<<"Invalid"<<std::endl;
   }
   else
   {
      month = Month;
      day = Day;
      year = Year;
   }
}
//postcondition: Date checked that the code does not violate any of the parameters
//precondition: Day will have been incremented by 1
void Date::increment()
{
   //month += 1;
   //assert(month >= 1 && month <= 12);
   day += 1;
```

```
   assert(day >= 1 && day <= 31);
   if(month == 2 && day == 28 || day == 29)
   {
    if(year % 4 || year % 400)
     {
          std::cout<<"Thats a Leap Year"<<std::endl;
          //month += 1;
          day += 1 ;
          //year++;
          assert(day >= 1 && day <= 31);
          assert(month >= 1 && month <= 12);
     }
    }
}
//postcondition: Day has been incremented by 1
void Date::display1()
{
   std::cout<<month<<'/'<<day<<'/'<<year;
}
//postcondition: Date has been displayed in number format
void Date::display2()
{
   string Month;
   switch(month)
   {
      case 1:
          Month="January";
          break;
```

```
case 2:
   Month="February";
   break;
case 3:
   Month="March";
   break;
case 4:
   Month="April";
   break;
case 5:
   Month="May";
   break;
case 6:
   Month="June";
   break;
case 7:
   Month="July";
   break;
case 8:
   Month="August";
   break;
case 9:
   Month="September";
   break;
case 10:
   Month="October";
   break;
case 11:
```

```
            Month="November";
            break;
        case 12:
            Month="December";
            break;
    }
    std::cout<<Month<<'/'<<day<<'/'<<year<<std::endl;
}
Date &Date::operator=(const Date &T) {
    month = T.month;
    day = T.day;
    year = T.year;
    return *this;
}
```

---

```
class Division {
        private:
                string divisionName;
                Employee * emp;
        public:
                Division();
                Division(string s);
                ~Division();
                Employee * getEmployee();
                string getDivisionName();
                void setEmployee(Employee * e);
                void setDivisionName(string s);
};
Division :: Division(){
```

```
	divisionName = " ";
}
Division :: Division(string s){
	divisionName = s;
}
Division :: ~Division(){
}
string Division :: getDivisionName(){
	return divisionName;
}
Employee * Division :: getEmployee(){
	return emp;
}
void Division :: setDivisionName(string s){
	divisionName = s;
}
void Division :: setEmployee(Employee * e){
	emp = e;
}
```

---

```
class JobDescription {
	private:
		string description;
		Employee * emp;
	public:
		JobDescription();
		JobDescription(string s);
		~JobDescription();
```

```
        Employee * getEmployee();
        string getDescription();
        void setEmployee(Employee * e);
        void setDescription(string s);
};
JobDescription :: JobDescription(){
    description = " ";
}
JobDescription :: JobDescription(string s){
    description = s;
}
JobDescription :: ~JobDescription(){
}
string JobDescription :: getDescription(){
    return description;
}
Employee * JobDescription :: getEmployee(){
    return emp;
}
void JobDescription :: setDescription(string s){
    description = s;
}
void JobDescription :: setEmployee(Employee * e){
    emp = e;
}
```

---

```
class Person {
    private :
        string name;
```

```
        string ssnum;
        int age;
    public:
        Person();
        Person(string pname, string pssnum, int page);
        ~Person();
        string getName();
        string getSSNum();
        int getAge();
        void setName(string pname);
        void setSSNum(string pssnum);
        void setAge(int page);
};
Person :: Person() {
    name = " ";
    ssnum = " ";
    age = 0;
}
Person :: Person(string pname, string pssnum, int page) {
    name = pname;
    ssnum = pssnum;
    age = page;
}
Person :: ~Person(){
}
string Person :: getName() {
    return name;
}
```

```
string Person :: getSSNum(){
        return ssnum;
}
int Person :: getAge(){
        return age;
}
void Person :: setName(string pname){
        name = pname;
}
void Person :: setSSNum(string pssnum){
        ssnum = pssnum;
}
void Person :: setAge(int page){
        age = page;
}
```

---

```
class Spouse : public Person{
        private:
                Employee * emp;
                Date anniversaryDate;
        public:
                Spouse();
                Spouse(string pname, string pssnum, int page, int m, int d, int y);
                ~Spouse();
                Employee * getEmployee();
                Date getAnniversaryDate();
                void setEmployee(Employee * e);
                void setAnniversarDate(int m, int d, int y);
```

```
};


Spouse :: Spouse(){
}
Spouse :: Spouse(string pname, string pssnum, int page, int m , int d , int y)
 : Person (pname, pssnum, page){
        anniversaryDate = Date(m , d, y);
}
Spouse :: ~Spouse() {
}
Date Spouse :: getAnniversaryDate(){
        return anniversaryDate;
}
Employee * Spouse :: getEmployee(){
        return emp;
}
void Spouse :: setAnniversarDate(int m, int d, int y){
        anniversaryDate.month = m;
        anniversaryDate.day = d;
        anniversaryDate.year = y;
}
void Spouse :: setEmployee(Employee * e){
        emp = e;
}
```