Удамшил Ба Бүрдэл Харьцаа, Бүрдэл Харьцааны Хориглолт (Лаборатори №8)

Н. Энхболд

ХШУИС-МКУТ, Програм хангамж, 20B1NUM0102

1. ОРШИЛ/УДИРТГАЛ

Удамшил болон бүрдэл харьцааны хэрэглээ, ялгааг ойлгож практикт хэрэгжүүлсэн ба бүрдэл харьцаа бүхий классуудыг тодорхойлохдоо “Constraint” буюу хориглолт гэх ойлголтыг ашигласан. Эдгээрийг объект хандлагат программчлалын түвшинд C++ хэл дээр хэрэгжүүлэв.

2. ЗОРИЛГО

Хүн гэх классаас шаталсан хэлбэрээр удамшуулан Эхнэр, Ажилчин, Хүүхэд гэх классуудыг шинээр тодорхойлсон ба Эхнэр, Хүүхэд, АжлынТухай, Салбар зэрэг классуудыг Ажилчин классын нэгэн хэсэг байхаар эдгээр зорилтыг тавин классуудыг тодорхойлов. Үүнд:

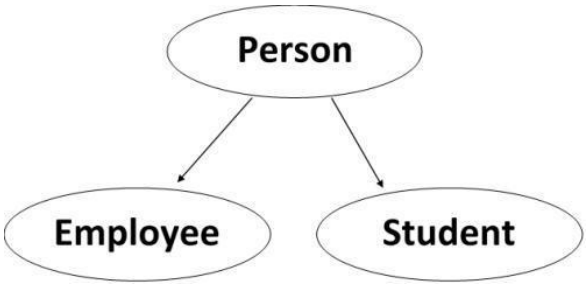
1. Удамшил ямар харьцаа үүсгэдэг вэ? Объект хандлагат программчлалд хэрхэн хэрэгжүүлдэг вэ?
2. Бүрдэл харьцаа гэж юу вэ? Объект хандлагат программчлалд хэрхэн хэрэгжүүлдэг вэ?
3. Бүрдэл харьцааны хориглолт (constraint) гэж юу вэ?

3. ОНОЛЫН СУДАЛГАА

3.1 Удамшлын харьцаа

Удамшил нь “тэр бол” буюу “is a” гэх хамаарлыг төлөөлдөг бөгөөд ямарваа нэгэн эх класс нь хүүхэд классынхаа ерөнхий шинж чанар нь болж өгдөг. Энэхүү лабораторийн ажлаас жишээ татаж үзвэл:

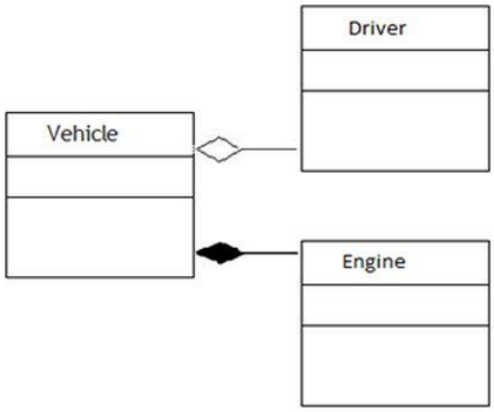
Хүн гэдэг классаас удамшин Эхнэр, Хүүхэд, Ажилчин гэх илүү тодорхой статус бүхий класс үүсч байгаа билээ. Үүнээс эхнэр бол хүн, хүүхэд бол хүн, ажилчин бол хүн гэх хамаарал бий болж байгаа юм.



Энэхүү диаграмаас бид Ажилчин бол Хүн, Сурагч бол Хүн гэх удамлын хамаарлыг хэлж чадах билээ.

3.2 Бүрдэл харьцаа

Бүрдэл харьцаа нь ямарваа нэгэн класс тодорхойлохын тулд тухайн класс өөр нэгэн класс бүхий объектыг гишүүн өгөгдөл хэлбэрээр агуулах байдлыг хэлнэ. Бүрдэл харьцаа нь “түүнийг агуулсан” буюу “has a” гэх хамаарлыг төлөөлдөг. Энэхүү лабораторийн ажлаар бид Ажилчин нь Эхнэр болон Хүүхэдтэй гэх хамаарлыг мөн гаргаж өгсөн билээ. Бүрдэл харьцаан дээр жишээ авбал:

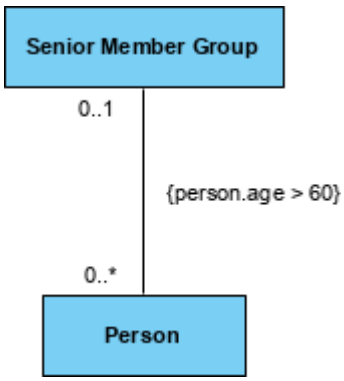


Энэхүү диаграмаас бид ТээврийнХэрэгсэл нь Жолооч болон Мотортой гэх хамаарлыг хэлж чадах билээ.

3.3 Бүрдэл харьцааны хориглолт

Constraint буюу бүрдэл харьцааны хориглолт нь объект хандалтат програмчлалын түвшинд өгөгдлийн хязгаар гэх байдлаар ашиглагддаг бөгөөд тухайн өгөгдлийн доод болон дээд авч болох утгуудыг зааж өгсөн байдаг. Энэхүү лабораторийн ажлаар Ажилчин класст буй гишүүн объектуудыг 1 Ажилчинд дор хаяж болон хамгийн ихдээ хэд байхыг constraint

ашиглан хязгаарласан билээ.



Энэхүү диаграмаас бид хүмүүс дунд хамгийн ихдээ 1 л ахлах ажилтны групп байхыг зөвшөөрсөнг хэлж чадах билээ.

4. ХЭРЭГЖҮҮЛЭЛТ

Зургаар өгсөн диаграмын дагуу классуудыг байгуулж бүрдэл харьцаа болон удамшил гэх ойлголтуудыг хавсарган нэгэн програмд ашигласан.

Child болон JobDescription классуудын объектуудыг байгуулж Employee классын динамик санах ойн дараалалд push_back хийж өгөв.

Spouse болон Division классуудад зөвхөн setter функц хийж өгснөөр 1-ээс олон утга авахыг хязгаарлаж байгаа юм.

5. ДҮГНЭЛТ

Динамик хүснэгт буюу vector-ийн тусламжтайгаар олон объектыг нэгэн мөрөнд хадгалж болох бөгөөд аливаа хүснэгтийн багтаамж дүүрэх зэрэг асуудлуудаас зайлсхийх боломжтой юм. Мөн бүрдэл харьцаан дээр хориглолт буюу Constraint – ийг нэмж өгснөөр тухайн объект нэгэн объектэд хамгийн ихдээ хэд оршиж болохыг setter болон vector-ийн тусламжтайгаар программчлав.

6. АШИГЛАСАН МАТЕРИАЛ

1. Объект хандалтат програмчлал – Лекц 09
https://drive.google.com/file/d/1_AhR0fH2RVLExpzUcGCOHrp4p0km1uyB/view?usp=sharing

2. Объект хандалтат програмчлал – Лекц 10
https://drive.google.com/file/d/1Lj4dpOTX1VNvw1Q2a9wcYOygHMnD4YjK/view?usp=sharing

7. ХАВСРАЈТ

```
#include <iostream>
#include <string.h>
#include <vector>

using namespace std;

class Person{
	private:
		char *name;
		char *ssnum;
		int age;
	public:
		Person(){
			name = new char[strlen("unnamed") + 1];
			strcpy(name, "unnamed");
			ssnum = new char[strlen("unknown") + 1];
			strcpy(ssnum, "unknown");
			age = 0;
		}
		Person(char *ner, char* num, int nas){
			name = new char[strlen(ner) + 1];
			strcpy(name, ner);
			ssnum = new char[strlen(num) + 1];
			strcpy(ssnum, num);
			age = nas;
		}
		char *getName(){
```

```
		return name;
	}
	char *getSSNum(){
		return ssnum;
	}
	int getAge(){
		return age;
	}
	void setName(char *ner){
		delete name;
		name = new char[strlen(ner) + 1];
		strcpy(name, ner);
	}
	void setSSNum(char *num){
		delete ssnum;
		ssnum = new char[strlen(num) + 1];
		strcpy(ssnum, num);
	}
	void setAge(int nas){
		age = nas;
	}
	~Person(){
		delete name;
		delete ssnum;
	}
};

class Spouse : public Person{
	private:
		char *anniversaryDate;
	public:
		Spouse() : Person(){
```

```
                anniversaryDate = new char[strlen("unknown") + 1];
                strcpy(anniversaryDate, "unknown");
        }
        Spouse(char *date, char *ner, char* num, int nas) : Person(ner, num, nas){
                anniversaryDate = new char[strlen(date) + 1];
                strcpy(anniversaryDate, date);
        }
        char* getAnniversaryDate(){
                return anniversaryDate;
        }
        void setAnniversaryDate(char *date){
                delete anniversaryDate;
                anniversaryDate = new char[strlen(date) + 1];
                strcpy(anniversaryDate, date);
        }
        ~Spouse(){
                delete anniversaryDate;
        }
};

class Child : public Person{
        private:
                char *favoriteToy;
        public:
                Child() : Person(){
                        favoriteToy = new char[strlen("unknown") + 1];
                        strcpy(favoriteToy, "unknown");
                }
                Child(char *toy, char *ner, char* num, int nas) : Person(ner, num, nas){
                        favoriteToy = new char[strlen(toy) + 1];
                        strcpy(favoriteToy, toy);
                }
```

```
		char* getFavoriteToy(){
			return favoriteToy;
		}
		void setFavoriteToy(char *toy){
			delete favoriteToy;
			favoriteToy = new char[strlen(toy) + 1];
			strcpy(favoriteToy, toy);
		}
		~Child(){
			delete favoriteToy;
		}
};

class Division{
	private:
		char *divisionName;

	public:
		Division(){
			divisionName = new char[strlen("unknown") + 1];
			strcpy(divisionName, "unknown");
		}
		Division(char *name){
			divisionName = new char[strlen(name) + 1];
			strcpy(divisionName, name);
		}
		char* getDivisionName(){
			return divisionName;
		}
		void setDivisionName(char* name){
			delete divisionName;
			divisionName = new char[strlen(name) + 1];
```

```
                strcpy(divisionName, name);
        }
        ~Division(){
                delete divisionName;
        }
};

class JobDescription{
        private:
                char *description;
        public:
                JobDescription(){
                        description = new char[strlen("unknown") + 1];
                        strcpy(description, "unknown");
                }
                JobDescription(char *desc){
                        description = new char[strlen(desc) + 1];
                        strcpy(description, desc);
                }
                char* getDescription(){
                        return description;
                }
                void setDescription(char *desc){
                        description = new char[strlen(desc) + 1];
                        strcpy(description, desc);
                }
                ~JobDescription(){
                        delete description;
                }
};

class Employee : public Person{
```

```
	private:
		Spouse spouse;
		vector<Child> child;
		vector<JobDescription> job;
		Division div;
		char *companyID;
		char *title;
		char *startDate;
	public:
		Employee() : Person(){
			companyID = new char[strlen("unknown") + 1];
			strcpy(companyID, "unknown");
			title = new char[strlen("unknown") + 1];
			strcpy(title, "unknown");
			startDate = new char[strlen("unknown") + 1];
			strcpy(startDate, "unknown");
			cout << "*Ajilchin nemegdsen*" << endl;
		}
		Employee(char *id, char *garchig, char*date, char *ner, char* num, int
nas) : Person(ner, num, nas){
			companyID = new char[strlen(id) + 1];
			strcpy(companyID, id);
			title = new char[strlen(garchig) + 1];
			strcpy(title, garchig);
			startDate = new char[strlen(date) + 1];
			strcpy(startDate, date);
		}
		void setDesc(JobDescription desc){
			job.push_back(desc);
		}
		void setDiv(Division d){
			div.setDivisionName(d.getDivisionName());
```

```
}

void setChild(vector<Child> hvvhed){
	child = hvvhed;
}
int childCount(){
	return child.size();
}
void setSpouse(Spouse ehner){
	spouse.setName(ehner.getName());
	spouse.setSSNum(ehner.getSSNum());
	spouse.setAge(ehner.getAge());
}
char* getCompanyID(){
	return companyID;
}
char* getTitle(){
	return title;
}
char* getStartDate(){
	return startDate;
}
void setCompanyID(char *id){
	delete companyID;
	companyID = new char[strlen(id) + 1];
	strcpy(companyID, id);
}
void setTitle(char *garchig){
	delete title;
	title = new char[strlen(garchig) + 1];
	strcpy(title, garchig);
}
```

```
        void setStartDate(char* start){
                delete startDate;
                startDate = new char[strlen(start) + 1];
                strcpy(startDate, start);
        }
        ~Employee(){
                delete companyID;
                delete title;
                delete startDate;
                cout << endl;
        }
        void print(){
                cout << "Name: " << getName() << endl;
                cout << "SSNum: " << getSSNum() << endl;
                cout << "Age: " << getAge() << endl;
                cout << "CompanyID: " << companyID << endl;
                cout << "Title: " << title << endl;
                cout << "StartDate: " << startDate << endl;
                cout << "Spouse: " << spouse.getName() << endl;
                cout << "Child: " << childCount() << endl;
                cout << "Division" << div.getDivisionName() << endl;
        }
};

int main(){
        Division div1("1st"), div2("2nd");
        JobDescription des1("manager"), des2("ceo"), des3("accountant"),
des4("supervisor");

        Employee e1("Com1", "MCS", "2003-05-22", "Bold", "ri03252211", 18);
        Employee e2("Com2", "TavanBogd", "2001-01-01", "Bayr", "ri01210114", 20);
        Employee e3("Com2", "TavanBogd", "2000-05-11", "Tselmeg", "ri00251114", 21);
```

```
e1.setDesc(des1);
e1.setDesc(des2);
e1.setDiv(div1);
e2.setDesc(des3);
e2.setDesc(des4);
e2.setDiv(div2);
e3.setDesc(des3);
e3.setDiv(div2);

Spouse sp1("2000-01-01", "Dulmaa", "ri01210112", 21);
Spouse sp2("1999-10-04", "Zaya", "fb99210112", 21);

e1.setSpouse(sp1);
e2.setSpouse(sp2);

vector<Child> c1;
vector<Child> c2;
Child c11("robot", "Bat", "ri12312312", 10);
Child c12("buu", "Dorj", "af12312312", 9);
Child c21("barbie", "Tsetseg", "gg11112222", 5);

c1.push_back(c11);
c1.push_back(c12);
c2.push_back(c21);

e1.setChild(c1);
e2.setChild(c2);

e1.print();
e2.print();
e3.print();
return 0;
```

```
}
```