3.1 Удамшил ямар харьцаа үүсгэдэг вэ?

Объект хандлагат програмчлалд хэрхэн хэрэгжүүлдэг вэ? Inheritance буюу удамшил нь нэг классын бүх шинжийг удамшуулан, өөрийнх нь онцлог шинжүүдийг нэмэн үүссэн шинэ класс, эх класстайгаа үүсгэж буй холбоо хамаарлыг хэлнэ. Эдгээр классууд “is a” гэсэн үгээр холбогдоно.

3.2 Бүрдэл харьцаа гэж юу вэ? Объект хандлагат програмчлалд хэрхэн хэрэгжүүлдэг вэ?

Тухайн классын бүрдэл хэсэг болох класс тодорхойлоход бүрдэл харьцаатай байна. Классууд “has a” гэсэн үгээр холбогдоно. Composition (0..1 болон 1..1 холбоо хамааралтай, эхний классыг устгахад дараагийн класс дагаад устна) болон aggregation (0..1 болон 0..* холбоо хамааралтай, объект нь тусдаа үүсч болно.) гэсэн хоёр хэсэгт хуваагдана.

Бодлого:

main.cpp

Run

```cpp
#include <vector>
#include <iostream>
#include <string>
using namespace std;
class Date{
    public:
        int Year;
        int Month;
        int Day;
    Date(){
        Year = 0;
        Month = 0;
        Day = 0;
    }
    Date(int y, int m, int d){
        Year = y;
        Month = m;
        Day = d;
    }
};
class Person{
    protected:
        string Name = "Unnamed";
        string SSNum = "Numm";
        int Age;
    public:
        Person(string n, string s, int a){
            this -> Name = n;
            this -> SSNum = s;
            this -> Age = a;
```

```
        }
        string getName(){
            return this -> Name;
        }
        string getSSNum(){
            return this -> SSNum;
        }
        int getAge(){
            return this -> Age;
        }
        void setName(string n){
            this -> Name = n;
        }
        void setSSNum(string s){
            this -> SSNum = s;
        }
        void setAge(int a){
            this -> Age = a;
        }
};
class Division{
    protected:
        string DivisionName;
    public:
        Division(){
            DivisionName = "Division";
        }
        Division(string d){
            DivisionName = d;
        }
        string getDivisionName(){
            return this -> DivisionName;
        }
        void setDivisionName(string d){
            this -> DivisionName = d;
        }
};
class JobDescription{
    private:
        string Description;
    public:
        JobDescription(string d){
            Description = d;
        }
        string getDescription(){
            return this -> Description;
        }
        void setDescription(string d){
            this -> Description = d;
        }
};
class Spouse: public Person{
```

```
    protected:
        Date AnniversaryDate;
    public:
        Spouse(string n, string s, int a, Date ann) : Person(n, s, a){
            AnniversaryDate = ann;
        }
        Date getAnniversaryDate(){
            return this -> AnniversaryDate;
        }
        void setAnniversaryDate(Date a){
            this -> AnniversaryDate = a;
        }
        void printAnniversaryDate(){
            cout << AnniversaryDate.Year << "/";
            cout << AnniversaryDate.Month << "/";
            cout << AnniversaryDate.Day;
        }
};
class Child : public Person{
    protected:
        string FavoriteToy;
    public:
        Child(string n, string s, int a, string f) : Person(n, s, a){
            FavoriteToy = f;
        }
        string getFavoriteToy(){
            return this -> FavoriteToy;
        }
        void setFavoriteToy(string f){
            this -> FavoriteToy = f;
        }
};
class Employee : public Person{
    protected:
        string CompanyID;
        string Title;
        Date StartDate;
        int spo;
        Division division;
        vector < JobDescription* > jobDescription;
        Spouse* spouse;
        vector < Child* > children;
    public:
        Employee(string n, string s, int a, string id, string t, Date d, string div, string j) :
            Person(n, s, a){
            CompanyID = id;
            Title = t;
            StartDate = d;
            division.setDivisionName(div);
            JobDescription *Job = new JobDescription(j);
            jobDescription.push_back(Job);
        }
        void setDivision(Division &d){
            division = d;
        }
        void setJobDescription(JobDescription *job){
            jobDescription.push_back(job);
```

```
141         children.push_back(k);
142     }
143     void setSpouse(Spouse* s){
144         this -> spouse = s;
145     }
146     void setCompanyId(string &companyId) {
147         CompanyID = companyId;
148     }
149     void setTitle(string &title) {
150         Title = title;
151     }
152     void setStartDate(Date &startDate) {
153         StartDate = startDate;
154     }
155     string &getCompanyId(){
156         return CompanyID;
157     }
158     string &getTitle(){
159         return Title;
160     }
161     Date &getStartDate(){
162         return StartDate;
163     }
164     vector<Child *> &getChildren(){
165         return children;
166     }
167     Spouse *getSpouse(){
168         return spouse;
169     }
170     vector<JobDescription *> &getJd(){
171         return jobDescription;
172     }
173     Division &getDivision(){
174         return division;
175     }
176     void print(){
177         cout << "Name: " << Name << endl;
178         cout << "Social security number: " << SSNum << endl;
179         cout << "Age: " << Age << endl;
180         cout << "Company ID: " << CompanyID << endl;
181         cout << "Title: " << Title << endl;
182         cout << "Start date: " << StartDate.Year << " - " << StartDate.Month << " - " <<
                 StartDate.Day << endl;
183         cout << "Division: " << division.getDivisionName() << endl;
184         cout << "Job descriptions: ";
185         for (int i = 0; i < jobDescription.size(); i++) {
186             cout << jobDescription[i] -> getDescription() << " ";
187         }
188         cout << endl;
189         cout << "spouse?: ";
190         spouse -> printAnniversaryDate();
191         cout << endl << "Children's favourite toys: ";
192         for(int i = 0; i < children.size(); i++){
193             cout << children[i] -> getFavoriteToy() << ",";
194         }
195         cout << endl;
196     }
197 };
```

```cpp
int main(){
Division d1("Construction"),d2("engineer");
JobDescription j1("software"),j2("computer"),j3("information");
cout << endl;
Employee a("Amaraa", "Xn45t", 45, "112254", "chief", Date(2020, 01, 10), "ajilchin", "HTTP"
    );
a.setDivision(d1);
a.setJobDescription(&j2);
Spouse s1("Ganbaa", "qwert", 35, Date(2020, 02, 02));
a.setSpouse(&s1);
Child c1("Oyun", "73289", 4, "cube"), c2("Svren", "73921", 7, "chess");
a.setChild(&c1);
a.setChild(&c2);
a.print();
cout << endl;
cout << endl;
Employee b("Bat", "a12fg", 25, "76541", "se", Date(2029, 6, 7), "ajilchin", "FDP");
b.setDivision(d2);
b.setJobDescription(&j1);
b.setJobDescription(&j3);
b.print();
}
```

Үр дүн:

Output Clear

```
/tmp/24C1A6u4SX.o
Name: Amaraa
Social security number: Xn45t
Age: 45
Company ID: 112254
Title: chief
Start date: 2020 - 1 - 10
Division: Construction
Job descriptions: HTTP computer
spouse?: 2020/2/2
Children's favourite toys: cube,chess,


Name: Bat
Social security number: a12fg
Age: 25
Company ID: 76541
Title: se
Start date: 2029 - 6 - 7
Division: engineer
Job descriptions: FDP software information
Segmentation fault
```