 Удамшлын харьцаа болон бүрдэл харьцаа

(Лаборатори №9 )

Х.Очирсүх

Програм хангамжийн 3-р түвшний оюутан, 20B1NUM0429

# 1. ОРШИЛ

Удамшлын үүсгэх харьцаа болон бүрдэл харьцаа, түүний хориглолтын талаар дэлгэрэнгүй судлаж объект хандлагат програмчлалд хэрхэн хэрэгжүүлж болох талаар судласан. Объект хандлагат хэл болох С++ дээр удамшлын үүсгэх харьцаа болон бүрдэл харьцааны кодыг бичиж хэрэгжүүлэх.

# 2. ЗОРИЛГО

Удамшлын болон нийлмэл харилцааны хэрэглээ, ялгааг ойлгож, практикт нэвтрүүлж, нийлмэл харилцааны ангиллыг тодорхойлоход "хязгаарлалт" гэсэн ойлголтыг ашигласан. Эдгээрийг объект хандалтат програмчлалын түвшинд C++ хэл дээр хэрэгжүүлнэ. Зорилгодоо хүрэхийн тулд дараах зорилтуудыг дэвшүүлсэн.

1. Лекцийн материал болон лекцийн үзэж ойлгох
2. Шаардлагатай агуулгадаа бичиж тэмдэглэх
3. Лекцийн тэмдэглэлээ эмх цэгцтэй, ойлгомжтой байдлаар дүрслэх
4. Өөрийн үгээр баяжуулж, нэмэлт материал судлах
5. Объект хандлагат номноос шаардлагатай мэдээллүүдээ цуглуулах

# 3. ОНОЛЫН СУДАЛГАА

## 3.1 **Удамшил ямар харьцаа үүсгэдэг вэ? Объект хандлагат программчлалд хэрхэн хэрэгжүүлдэг вэ?**

Удамшил гэдэг нь нэг классаас өөр нэг класс үүсэхийг хэлнэ. Удамшсан класс буюу дэд класс нь эх классынхаа гишүүн өгөгдөл болон гишүүн функцыг бүгдийн өвлөж авдаг. Удамшил нь “тэр” буюу “is a” гэх хамаарлыг төлөөлдөг бөгөөд ямарваа нэгэн эх класс нь хүүхэд классынхаа ерөнхий шинж чанар нь болж өгдөг.

## 3.2 **Бүрдэл харьцаа гэж юу вэ? Объект хандлагат программчлалд хэрхэн хэрэгжүүлдэг вэ?**

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

## 3.4 **Aggregation харьцааны тухай ойлголтыг бич.**

Aggregation харьцаа гэдэг нь хүүхэд эхээс үл хамааран оршин тогтнох харилцааг илэрхийлдэг. Жишээ нь Car – passenger.

## 3.5 **Бүрдэл харьцааны хориглолт (constraint) гэж юу вэ?**

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.

Бүрдэл харьцаан хориглолт гэдэг нь классууд өөр бусад классын объектыг агуулахад хориглолт хийж байгааг хэлнэ. Жишээ нь:

1. 0…1 (Хамгийн багадаа 0 буюу байхгүй, хамгийн ихдээ 1 ширхэгийг агуулна)
2. 0…n (Хамгийн багадаа 0 буюу байхгүй, хамгийн ихдээ хэд ч байж болно)
3. 1…1 (Хамгийн багадаа 1, хамгийн их дээ 1 буюу ямар ч үед 1ийг агуулна)
4. 1…n (Хамгийн багадаа 1, хамгийн ихдээ хэд ч байж болно)

Дээрх бүрдэл харьцааны хориглолт нь ямар нэгэн классд хамгийн багадаа болон хамгийн ихдээ хэдэн өөр классын объект агуулагдаж болохы заана гэсэн үг.

4. ХЭРЭГЖҮҮЛЭЛТ

 Division болон JobDescription классуудын хэд хэдэн объект байгуул

Division div1("1st"), div2("2nd"), div3("3rd");

    JobDescription des1("manager"), des2("ceo"), des3("accountant"), des4("supervisor"), des5("supervisor");

Java implements:

Division div1 = new Division("1st");  
Division div2 = new Division("2st");  
Division div3 = new Division("3st");

Employee классын хэд хэдэн объект байгуулж тус бүрийн Division, JobDescription –ийг зааж өг.

Employee e1("Com1", "MCS", "2003-05-22", "Bold", "ri03252211", 18, des1, div1);

    Employee e2("Com2", "TavanBogd", "2001-01-01", "Bayr", "ri01210114", 20, des2, div2);

    Employee e3("Com2", "TavanBogd", "2000-05-11", "Tselmeg", "ri00251114", 21, des3, div3);

Java implements:

Employee emp1 = new Employee("Com1", "TavanBogd1", new Date(), "Bold1", "ri03252211", 18, job1, div1);  
Employee emp2 = new Employee("Com2", "TavanBogd2", new Date(), "Munguu", "ri01210114", 20, job2, div2);  
Employee emp3 = new Employee("Com3", "TavanBogd3", new Date(), "Hatka", "ri00251114", 21, job3, div3);

Employee классын объект тус бүрд Spouse, Child – уудыг тохируулж өг

e1.setSpouse(sp1);

e2.setSpouse(sp2);

    e3.setSpouse(sp3);

Java implements:

emp1.setSpouse(sp1);  
emp2.setSpouse(sp2);  
emp3.setSpouse(sp3);

# ДҮГНЭЛТ

Энэхүү лаборатори 9 хүрээнд удамшил харьцаа болон бүрдэл харьцаа, түүний хориглолтуудын ойлголтыг мэдэж авж, кодондоо хэрэгжүүлсэн. Удамшлын харьцаа ашигласнаар өмнө бичсэн кодыг эргүүлэн ашиглах механизм юм. Ингэснээр програмын алдааг бууруулж, программыг хурдан болгох, зардал бага давуу талыг үүсгэдэг. Бүрдэл харьцаа бид нарийн бүтэц бүхий класс бүрдүүлэхэд ашигладаг. 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 хандлагат технологийн С++ програмчлал, Ж.Пүрэв, 2008, Улаанбаатар.

2. Лекц – object relation, template function lecture teams

# ХАВСРАЛТ

Java implements:

Person class

package org.example;  
  
public class Person {  
  
 public String name;  
 public String ssnum;  
 public int age;  
  
 Person(){  
 name = "unnamed";  
 ssnum = "unnamed";  
 age = 0;  
 }  
 Person(String name, String ssnum, int age){  
 this.name = name;  
 this.ssnum = ssnum;  
 this.age = age;  
 }  
 String getName(){  
 return name;  
 }  
 String getSsnum(){  
 return ssnum;  
 }  
 int getAge(){  
 return age;  
 }  
 void setName(String name){  
 this.name = name;  
 }  
 void setSsnum(String ssnum){  
 this.ssnum = ssnum;  
 }  
 void getAge(int age){  
 this.age = age;  
 }  
}

Spouse class:

package org.example;  
  
import java.util.Date;  
  
public class Spouse extends Person{  
 private String anniversary;  
  
 Spouse(){  
 this.anniversary = "empty";  
 }  
 Spouse(String anniversary, String name, String ssnum, int age){  
  
 this.anniversary = anniversary;  
 this.name = name;  
 this.ssnum = ssnum;  
 this.age = age;  
 }  
  
 String getAnniversary(){  
 return anniversary;  
 }  
 void setAnniversary(String anniversary){  
 this.anniversary = anniversary;  
 }  
}

Child class:

package org.example;  
  
public class Child extends Person{  
  
 String favoriteToy;  
 Child() {  
  
 }  
 Child(String favoriteToy, String name, String ssnum, int age) {  
 this.favoriteToy = favoriteToy;  
 this.name = name;  
 this.age = age;  
 this.ssnum = ssnum;  
  
 }  
 String getFavoriteToy(){  
 return favoriteToy;  
 }  
 void setFavoriteToy(String favoriteToy){  
 this.favoriteToy = favoriteToy;  
 }  
}

Employee class:

package org.example;  
  
import java.text.DateFormat;  
import java.util.Date;  
import java.util.Vector;  
  
public class Employee extends Person{  
  
 private String CompanyID;  
 private String Title;  
 private Date StartDate;  
  
 Employee(){  
 CompanyID = "Unnamed";  
 Title = "Unnamed";  
 StartDate = new Date();  
 }  
 Employee(String CompanyID, String Title, Date StartDate, String name, String ssnum, int age, JobDescription job, Division division){  
 this.CompanyID = CompanyID;  
 this.Title = Title;  
 this.StartDate = StartDate;  
 this.name = name;  
 this.ssnum = ssnum;  
 this.age = age;  
 this.job.add(job);  
 this.division.setDivisionName(division.getDivisionName());  
  
 }  
  
 Vector<Spouse> spouse = new Vector<Spouse>();  
 Vector<Child> child = new Vector<Child>();  
 Vector<JobDescription> job = new Vector<JobDescription>();  
 Division division = new Division();  
  
 void addJob(JobDescription j){  
 this.job.add(j);  
 }  
 void addChild(Child son){  
 this.child.add(son);  
 }  
  
 void setSpouse( Spouse love){  
 if(spouse.size() == 1)  
 System.*out*.println("Hostoi hvn bna shvvdee ");  
 else  
 spouse.add(love);  
 }  
  
 void setDivision(Division a){  
 division.setDivisionName(a.getDivisionName());  
 }  
  
 // get  
 String getCompanyID(){  
 return CompanyID;  
 }  
 String getTitle(){  
 return Title;  
 }  
 Date getStartDate(){  
 return StartDate;  
 }  
 // set  
 void setCompanyID(String CompanyID){  
 this.CompanyID = CompanyID;  
 }  
 void setTitle(String Title){  
 this.Title = Title;  
 }  
 void setStartDate(Date StartDate) {  
 this.StartDate = StartDate;  
 }  
  
 void print(){  
 System.*out*.println(getName());  
 System.*out*.println(getSsnum());  
 System.*out*.println(getAge());  
 System.*out*.println(getCompanyID());  
 System.*out*.println(getTitle());  
 System.*out*.println(getStartDate());  
 //System.out.println(spouse.elementAt(0));  
 System.*out*.println(division.getDivisionName());  
 }  
  
  
}

Division class:

package org.example;  
  
public class Division {  
 private String DivisionName;  
 Division(){  
 DivisionName = "unnamed";  
 }  
 Division(String DivisionName){  
 this.DivisionName =DivisionName;  
 }  
 String getDivisionName(){  
 return DivisionName;  
 }  
 void setDivisionName(String DivisionName){  
 this.DivisionName = DivisionName;  
 }  
  
}

JobDescription class

package org.example;  
  
import javax.swing.\*;  
  
public class JobDescription {  
 private String Description;  
 JobDescription(){  
 Description = "Unnamed";  
 }  
 JobDescription(String Description){  
 this.Description = Description;  
 }  
 String getDescription(){  
 return Description;  
 }  
 void setDescription(String Description){  
 this.Description = Description;  
 }  
}

Main class

package org.example;  
  
  
import java.text.DateFormat;  
import java.time.LocalDate;  
import java.util.Date;  
  
public class Main {  
 public static void main(String[] args) {  
 Person person = new Person();  
 Division div1 = new Division("1st");  
 Division div2 = new Division("2st");  
 Division div3 = new Division("3st");  
  
 JobDescription job1 = new JobDescription("manager");  
 JobDescription job2 = new JobDescription("ceo");  
 JobDescription job3 = new JobDescription("accountant");  
 JobDescription job4 = new JobDescription("supervisor");  
 JobDescription job5 = new JobDescription("oojoo");  
  
 Employee emp1 = new Employee("Com1", "TavanBogd1", new Date(), "Bold1", "ri03252211", 18, job1, div1);  
 Employee emp2 = new Employee("Com2", "TavanBogd2", new Date(), "Munguu", "ri01210114", 20, job2, div2);  
 Employee emp3 = new Employee("Com3", "TavanBogd3", new Date(), "Hatka", "ri00251114", 21, job3, div3);  
  
  
 Spouse sp1 = new Spouse("2000-01-01", "Munguu", "ri01210112", 21);  
 Spouse sp2 = new Spouse("1999-10-04", "Zaya", "fb99210112", 21);  
 Spouse sp3 = new Spouse("1999-10-04", "Zaya", "fb99210112", 21);  
 emp1.setSpouse(sp1);  
 emp2.setSpouse(sp2);  
 emp3.setSpouse(sp3);  
  
 Child c11 = new Child("goku", "Bat", "ri12312312", 10);  
 Child c12 = new Child("naruto", "Dorj", "af12312312", 9);  
 Child c21 = new Child("levi", "Tsetseg", "gg11112222", 5);  
  
  
 emp1.addChild(c11);  
 emp2.addChild(c12);  
 emp3.addChild(c21);  
  
 emp1.print();  
 emp2.print();  
 emp3.print();  
  
  
  
  
  
  
 System.*out*.println(person.getName());  
 }  
}

C++ implements:

1. #include <iostream>
2. #include <string.h>
3. #include <vector>
4. using namespace std;
5. class Person{
6. private:
7. char \*name;
8. char \*ssnum;
9. int age;
10. public:
11. Person(){
12. name = new char[strlen("unnamed") + 1];
13. strcpy(name, "unnamed");
14. ssnum = new char[strlen("unknown") + 1];
15. strcpy(ssnum, "unknown");
16. age = 0;
17. }
18. Person(char \*ner, char\* num, int nas){
19. name = new char[strlen(ner) + 1];
20. strcpy(name, ner);
21. ssnum = new char[strlen(num) + 1];
22. strcpy(ssnum, num);
23. age = nas;
24. }
25. char \*getName(){
26. return name;
27. }
28. char \*getSSNum(){
29. return ssnum;
30. }
31. int getAge(){
32. return age;
33. }
34. void setName(char \*ner){
35. delete name;
36. name = new char[strlen(ner) + 1];
37. strcpy(name, ner);
38. }
39. void setSSNum(char \*num){
40. delete ssnum;
41. ssnum = new char[strlen(num) + 1];
42. strcpy(ssnum, num);
43. }
44. void setAge(int nas){
45. age = nas;
46. }
47. ~Person(){
48. delete name;
49. delete ssnum;
50. }
51. };
52. class Spouse : public Person{
53. private:
54. char \*anniversaryDate;
55. public:
56. Spouse() : Person(){
57. anniversaryDate = new char[strlen("unknown") + 1];
58. strcpy(anniversaryDate, "unknown");
59. }
60. Spouse(char \*date, char \*ner, char\* num, int nas) : Person(ner, num, nas){
61. anniversaryDate = new char[strlen(date) + 1];
62. strcpy(anniversaryDate, date);
63. }
64. char\* getAnniversaryDate(){
65. return anniversaryDate;
66. }
67. void setAnniversaryDate(char \*date){
68. delete anniversaryDate;
69. anniversaryDate = new char[strlen(date) + 1];
70. strcpy(anniversaryDate, date);
71. }
72. ~Spouse(){
73. delete anniversaryDate;
74. }
75. };
76. class Child : public Person{
77. private:
78. char \*favoriteToy;
79. public:
80. Child() : Person(){
81. favoriteToy = new char[strlen("unknown") + 1];
82. strcpy(favoriteToy, "unknown");
83. }
84. Child(char \*toy, char \*ner, char\* num, int nas) : Person(ner, num, nas){
85. favoriteToy = new char[strlen(toy) + 1];
86. strcpy(favoriteToy, toy);
87. }
88. char\* getFavoriteToy(){
89. return favoriteToy;
90. }
91. void setFavoriteToy(char \*toy){
92. delete favoriteToy;
93. favoriteToy = new char[strlen(toy) + 1];
94. strcpy(favoriteToy, toy);
95. }
96. ~Child(){
97. delete favoriteToy;
98. }
99. };
100. class Division{
101. private:
102. char \*divisionName;
103. public:
104. Division(){
105. divisionName = new char[strlen("unknown") + 1];
106. strcpy(divisionName, "unknown");
107. }
108. Division(char \*name){
109. divisionName = new char[strlen(name) + 1];
110. strcpy(divisionName, name);
111. }
112. char\* getDivisionName(){
113. return divisionName;
114. }
115. void setDivisionName(char\* name){
116. delete divisionName;
117. divisionName = new char[strlen(name) + 1];
118. strcpy(divisionName, name);
119. }
120. ~Division(){
121. delete divisionName;
122. }
123. };
124. class JobDescription{
125. private:
126. char \*description;
127. public:
128. JobDescription(){
129. description = new char[strlen("unknown") + 1];
130. strcpy(description, "unknown");
131. }
132. JobDescription(char \*desc){
133. description = new char[strlen(desc) + 1];
134. strcpy(description, desc);
135. }
136. char\* getDescription(){
137. return description;
138. }
139. void setDescription(char \*desc){
140. description = new char[strlen(desc) + 1];
141. strcpy(description, desc);
142. }
143. ~JobDescription(){
144. delete description;
145. }
146. };
147. class Employee : public Person{
148. private:
149. vector<Spouse> spouse;
150. vector<Child> child;
151. vector<JobDescription> job;
152. Division div;
153. char \*companyID;
154. char \*title;
155. char \*startDate;
156. public:
157. Employee() : Person(){
158. companyID = new char[strlen("unknown") + 1];
159. strcpy(companyID, "unknown");
160. title = new char[strlen("unknown") + 1];
161. strcpy(title, "unknown");
162. startDate = new char[strlen("unknown") + 1];
163. strcpy(startDate, "unknown");
164. cout << "\*Ajilchin nemegdsen\*" << endl;
165. }
166. Employee(char \*id, char \*garchig, char\*date, char \*ner, char\* num, int nas, JobDescription desc, Division d) : Person(ner, num, nas){
167. companyID = new char[strlen(id) + 1];
168. strcpy(companyID, id);
169. title = new char[strlen(garchig) + 1];
170. strcpy(title, garchig);
171. startDate = new char[strlen(date) + 1];
172. strcpy(startDate, date);
173. job.push\_back(desc);
174. div.setDivisionName(d.getDivisionName());
175. cout << "\*Ajilchin nemegdsen\*" << endl;
176. }
177. void addDesc(JobDescription desc){
178. this->job.push\_back(desc);
179. }
180. void setDiv(Division d){
181. div.setDivisionName(d.getDivisionName());
182. }
183. void addChild(Child hvvhed){
184. this->child.push\_back(hvvhed);
185. }
186. int childCount(){
187. return child.size();
188. }
189. void setSpouse(Spouse ehner){
190. if(spouse.size() == 1)
191. cout << "Ehnertei bn\n";
192. else
193. spouse.push\_back(ehner);
194. }
195. char\* getCompanyID(){
196. return companyID;
197. }
198. char\* getTitle(){
199. return title;
200. }
201. char\* getStartDate(){
202. return startDate;
203. }
204. void setCompanyID(char \*id){
205. delete companyID;
206. companyID = new char[strlen(id) + 1];
207. strcpy(companyID, id);
208. }
209. void setTitle(char \*garchig){
210. delete title;
211. title = new char[strlen(garchig) + 1];
212. strcpy(title, garchig);
213. }
214. void setStartDate(char\* start){
215. delete startDate;
216. startDate = new char[strlen(start) + 1];
217. strcpy(startDate, start);
218. }
219. ~Employee(){
220. delete companyID;
221. delete title;
222. delete startDate;
223. cout << endl;
224. }
225. void print(){
226. cout << "Ajiltan: " <<getName() << "-ii medeelel: \n" <<endl;
227. cout << "SSNum: " << getSSNum() << endl;
228. cout << "Age: " << getAge() << endl;
229. cout << "CompanyID: " << companyID << endl;
230. cout << "Title: " << title << endl;
231. cout << "StartDate: " << startDate << endl;
232. cout << "Spouse: " << spouse[1].getName() << endl;
233. cout << "Child: " << childCount() << endl;
234. cout << "Division" << div.getDivisionName() << endl;
235. }
236. };
237. int main(){
238. Division div1("1st"), div2("2nd"), div3("3rd");
239. JobDescription des1("manager"), des2("ceo"), des3("accountant"), des4("supervisor"), des5("supervisor");
241. Employee e1("Com1", "TavanBogd", "2002-09-22", "Bold", "ri03252211", 18, des1, div1);
242. Employee e2("Com2", "TavanBogd", "2001-01-01", "Bayr", "ri01210114", 20, des2, div2);
243. Employee e3("Com2", "TavanBogd", "2000-05-11", "Tselmeg", "ri00251114", 21, des3, div3);
245. e1.addDesc(des4);
246. e2.addDesc(des5);
248. Spouse sp1("2000-01-01", "Dulmaa", "ri01210112", 21);
249. Spouse sp2("1999-10-04", "Zaya", "fb99210112", 21);
250. Spouse sp3("1999-10-04", "Zaya", "fb99210112", 21);
251. e1.setSpouse(sp1);
252. e2.setSpouse(sp2);
253. e3.setSpouse(sp3);
255. Child c11("goku", "Bat", "ri12312312", 10);
256. Child c12("naruto", "Dorj", "af12312312", 9);
257. Child c21("levi", "Tsetseg", "gg11112222", 5);

260. e1.addChild(c11);
261. e1.addChild(c12);
262. e2.addChild(c21);
264. e1.print();
265. e2.print();
266. e3.print();
267. return 0;
268. }