# Классыг java/c++ аль нэгэн дээр байгуулан харьцаа тодорхойлох лабораторийн ажил (Лаборатори №9)

М.Солонго

ХШУИС, МКУТ-ийн оюутан, 20b1num2104@stud.num.edu.mn

# 1. ОРШИЛ

Классуудыг java дээр байгуулж employee класст 0..1, 0..n эсвэл 1..1, 1..n харьцаагаар зургийн дагуу програмчлахын тулд шаардлагатай онолын мэдлэг aggregation, composition харьцаа болон бүрдэл харьцааны хориглолтыг судлан кодын хамт хавсаргав.

# 2. ЗОРИЛГО

Aggregation, composition харьцааг ойлгон таниж хэрэглэж сурахын тулд дараах зорилтуудыг тавьж ажилласан. Үүнд:

1. Зурагт харуулсан класс диаграмын дагуу классуудыг байгуулах
2. Division JobDescription классуудын хэд хэдэн объект байгуулах
3. Employee классын хэд хэдэн объект байгуулан тус бүрд Division JobDescription-ийг зааж өгөх
4. Employee классын объект тус бүрд spouse, child тохируулж өгөх
5. Тохируулсны дараагаар employee классын объект тус бүрийг хэвлэж харуулах

# 3. ОНОЛЫН СУДАЛГАА

## 3.1 Composition (бүрдэл) харьцаа гэж юу вэ?

Олон объект нийлж нэг том түвэгтэй юм хийхдээ ашиглах ба гишүүн өгөгдөл нь класс төрөлтэй байж болно. Ингэснээрээ эх класс устахад бүрдүүлж байсан объектууд дагаад устдаг. 1..n харьцаа үүсгэдэг.

## 3.2 Aggregation харьцаа

Олон объект нийлж нэг объект үүсгэхийг хэлэх ба бүрдэл харьцаанаас ялгагдах нэг онцлог нь объект устахад бүрдүүлж байсан объект устах шаардлагагүй. Энэ нь тэр гишүүн класс нь тухайн объектод байсан ч болно, байхгүй байсан ч болно гэдэг утгаараа 0..1, 1..1, 0..n, 1..n зэрэг холбоо үүсгэнэ.

# 4. ХЭРЭГЖҮҮЛЭЛТ

## 4.1. Удамшил ямар харьцаа үүсгэдэг? Объект хандлагат програмчлалд хэрхэн хэрэгжүүлдэг?

Удамшил гэдэг нь өөр классын бүх шинж эсвэл зарим шинжийг удамшиж буй класс өвлөж авдаг тул эх класс, удамшсан класс гэсэн харьцаа үүсгэдэг.

Жишээ: Class employee::public person{

…

};

## 4.2. Бүрдэл харьцааг объект хандлагат програмчлалд хэрхэн хэрэгжүүлдэг?

Class division{

date startDate;

};

Class Employee{

Division division;

Employee(division){

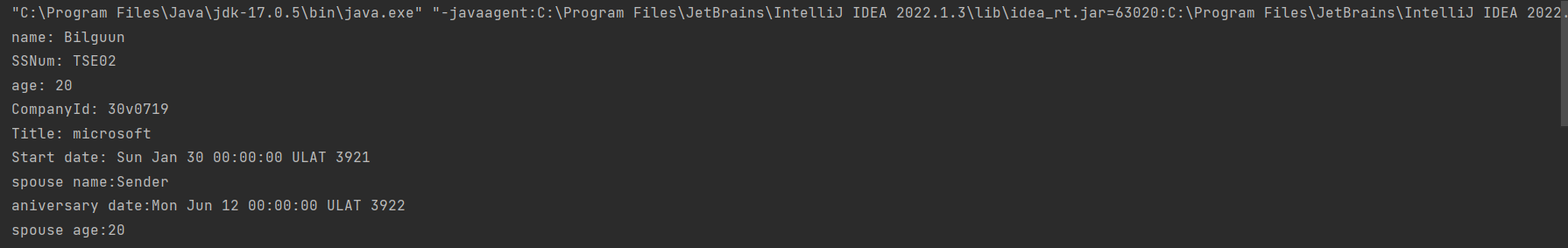
strcpy(This->division,division);

}

};

## 4.3. Бүрдэл харьцааны хориглолт?

Бүрдэл харьцаа буюу 1..n харьцаа агуулсан класст 0..n харьцаа бичихийг хориглоно. Жишээлбэл логикийн хувьд машин үйлвэрлэгчид машинд 1 мотор угсарч байж хэрэглээнд гаргана. Үүнийг машинд мотор байсан ч болно байхгүй байсан ч болно эсвэл 3 4 мотортой байж болно гэдгээр тодорхойлж болохгүй.





# 5. ДҮГНЭЛТ

Класс хоорондын харьцааг удамшлын харьцаагаар буюу эх класс, удамшсан классаар тодорхойлж болох ба эсвэл composition, aggregation харьцаагаар удамшаагүй 2 классыг холбож ашиглаж болох ба тэдгээрийн хэрэгцээ шаардлагыг ойлгож авлаа.

## 6. АШИГЛАСАН МАТЕРИАЛ

1. Объект хандлагат технологийн С++ програмчлал, Ж.Пүрэв, 2008, Улаанбаатар.
2. https://nummta.sharepoint.com/:p:/r/sites/grp-oop-fall-2022461/Shared%20Documents/General/lectures/11.02%20Object%20relation%20Composition%20Aggregation%20Inheritence.pptx?d=w2b9a78bad219473cad80e2990c8ddaa2&csf=1&web=1&e=64Bw3K

# 7. ХАВСРАЛТ

Person.java:

public class Person {  
 protected String name;  
 protected String SSNum;  
 protected int age;  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setSSNum(String SSNum) {  
 this.SSNum = SSNum;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getSSNum() {  
 return SSNum;  
 }  
  
 public int getAge() {  
 return age;  
 }  
 public void showPerson(){  
 System.*out*.println("name: "+getName());  
 System.*out*.println("SSNum: "+getSSNum());  
 System.*out*.println("age: "+getAge());  
 }  
}

spouse.java:

import java.util.Date;  
  
public class Spouse extends Person{  
 protected Date AnniversaryDate;  
  
 public Date getAnniversaryDate() {  
 return AnniversaryDate;  
 }  
  
 public void setAnniversaryDate(Date anniversaryDate) {  
 AnniversaryDate = anniversaryDate;  
 }  
  
 public Spouse(String name,int age,Date anniversaryDate) {  
 setName(name);  
 setAge(age);  
 AnniversaryDate = anniversaryDate;  
 }  
}

employee.java:

import java.util.Date;  
import java.util.Vector;  
  
public class Employee extends Person{  
 protected String CompanyId;  
 protected String Title;  
 protected Date StartDate;  
 protected Division division;  
 protected Spouse spouse;  
 protected Vector<Child> child;  
 protected Vector<JobDescription> jobDescription;  
  
  
 public void setCompanyId(String companyId) {  
 CompanyId = companyId;  
 }  
  
 public void setTitle(String title) {  
 Title = title;  
 }  
  
 public void setStartDate(Date startDate) {  
 StartDate = startDate;  
 }  
  
 public String getCompanyId() {  
 return CompanyId;  
 }  
  
 public String getTitle() {  
 return Title;  
 }  
  
 public Date getStartDate() {  
 return StartDate;  
 }  
  
 public void setDivision(Division division){this.division=division;}  
  
 public void setJobDescription(JobDescription jobDescription) {  
 this.jobDescription.addElement(jobDescription);  
 }  
 public void setSpouse(Spouse spouse) {  
 this.spouse = spouse;  
 }  
 public void addJobDescription(JobDescription jobDescription) {  
 this.jobDescription.addElement(jobDescription);  
 }  
 public void addChild(Child child) {  
 this.child.addElement(child);  
 }  
 public Employee(String name1, String SSnum, int age1, String companyId, String title, Date startDate,Division division,JobDescription jobDescription) {  
 setName(name1);  
 setSSNum(SSnum);  
 setAge(age1);  
 CompanyId = companyId;  
 Title = title;  
 StartDate = startDate;  
 this.child = new Vector<Child>();  
 this.jobDescription= new Vector<JobDescription>();  
 setDivision(division);  
 setJobDescription(jobDescription);  
 }  
 public void showData(){  
 this.showPerson();  
 System.*out*.println("CompanyId: "+getCompanyId());  
 System.*out*.println("Title: "+getTitle());  
 System.*out*.println("Start date: "+getStartDate());  
 }  
}

child.java:

public class Child extends Person{  
 protected String FavoriteToy;  
  
 public void setFavoriteToy(String favoriteToy) {  
 FavoriteToy = favoriteToy;  
 }  
  
 public String getFavoriteToy() {  
 return FavoriteToy;  
 }  
  
 public Child(String name,String favoriteToy) {  
 setName(name);  
 FavoriteToy = favoriteToy;  
 }  
}

division.java:

public class Division{  
 private String DivisionName;  
  
 public void setDivisionName(String divisionName) {  
 DivisionName = divisionName;  
 }  
  
 public String getDivisionName() {  
 return DivisionName;  
 }  
  
 public Division(String divisionName) {  
 DivisionName = divisionName;  
 }  
  
}

jobdescription.java:

public class JobDescription {  
 private String Description;  
  
 public void setDescription(String description) {  
 Description = description;  
 }  
  
 public String getDescription() {  
 return Description;  
 }  
  
 public JobDescription(String description) {  
 Description = description;  
 }  
}

main.java:

import java.util.Date;  
public class Main{  
 public static void main(String[] args) {  
 Division IS = new Division("Information system");  
 Division CS = new Division("Computer science");  
  
 JobDescription AI = new JobDescription("AI engineer");  
 JobDescription SA = new JobDescription("Systems architect");  
  
 Spouse Sender = new Spouse("Sender", 20, new Date(2022, 05, 12));  
 Spouse Soylnym = new Spouse("Soylnym", 20, new Date(2012, 12, 31));  
  
 Child Binderya = new Child("Binderya", "mobile legends");  
 Child Duulim = new Child("Duulim", "genshin");  
  
 Employee Bilguun = new Employee("Bilguun", "TSE02", 20, "30v0719", "microsoft", new Date(2020, 12, 30), CS, AI);  
 Bilguun.setDivision(CS);  
 Bilguun.addJobDescription(SA);  
 Bilguun.setSpouse(Sender);  
 Bilguun.addChild(Binderya);  
 Employee Enkhmunkh = new Employee("Enkhmunkhh", "HS15", 21, "31g3679", "google", new Date(2022, 3, 12), IS, SA);  
 Enkhmunkh.setDivision(IS);  
 Enkhmunkh.setSpouse(Soylnym);  
 Enkhmunkh.setJobDescription(AI);  
 Enkhmunkh.addChild(Duulim);  
 Bilguun.showData();  
 Enkhmunkh.showData();  
 }  
}