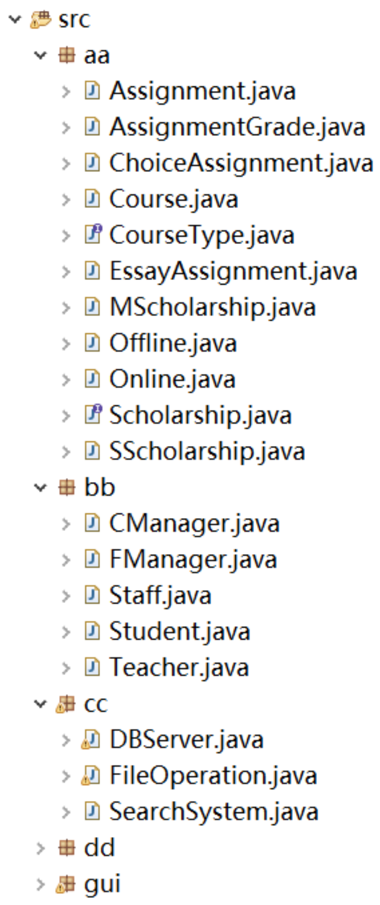
document

Abstract

In this document, firstly I would introduce the basic programming of my project and then the answers for the given requirements are demonstrated.

Programming

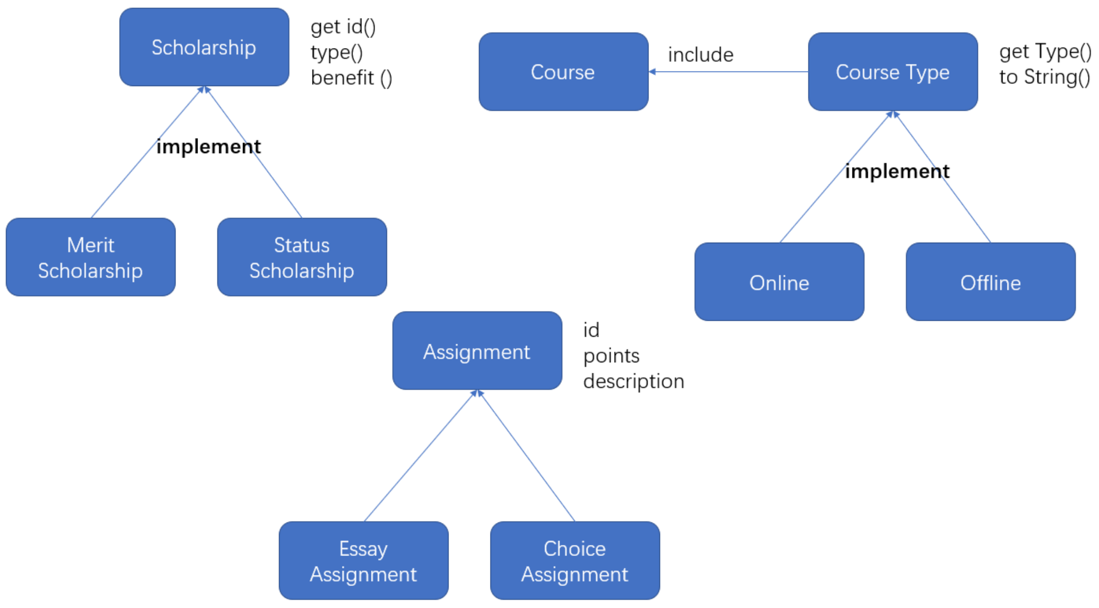


```
src
  aa
    Assignment.java
    AssignmentGrade.java
    ChoiceAssignment.java
    Course.java
    CourseType.java
    EssayAssignment.java
    MScholarship.java
    Offline.java
    Online.java
    Scholarship.java
    SScholarship.java
  bb
    CManager.java
    FManager.java
    Staff.java
    Student.java
    Teacher.java
  cc
    DBServer.java
    FileOperation.java
    SearchSystem.java
  dd
  gui
```

aa package contains School entities, bb package contains Human entities, and cc is for data storage and dd package is for testing and debugging, while gui package is used as interaction.

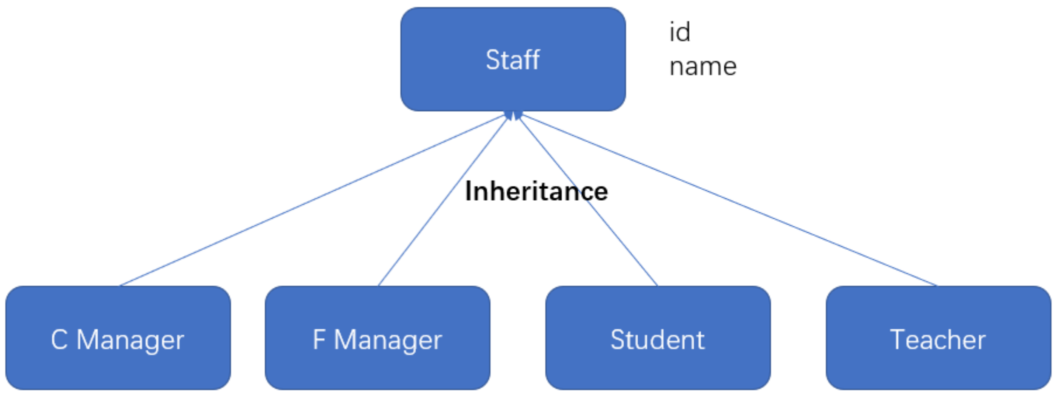
For aa package

Those classes could be grouped into as below,



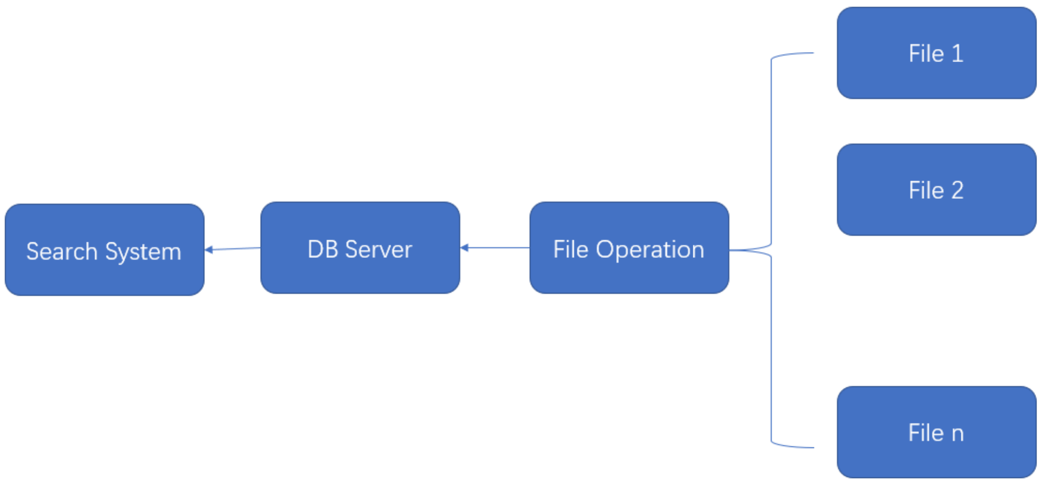
For bb package

Those classes could be grouped into as below,



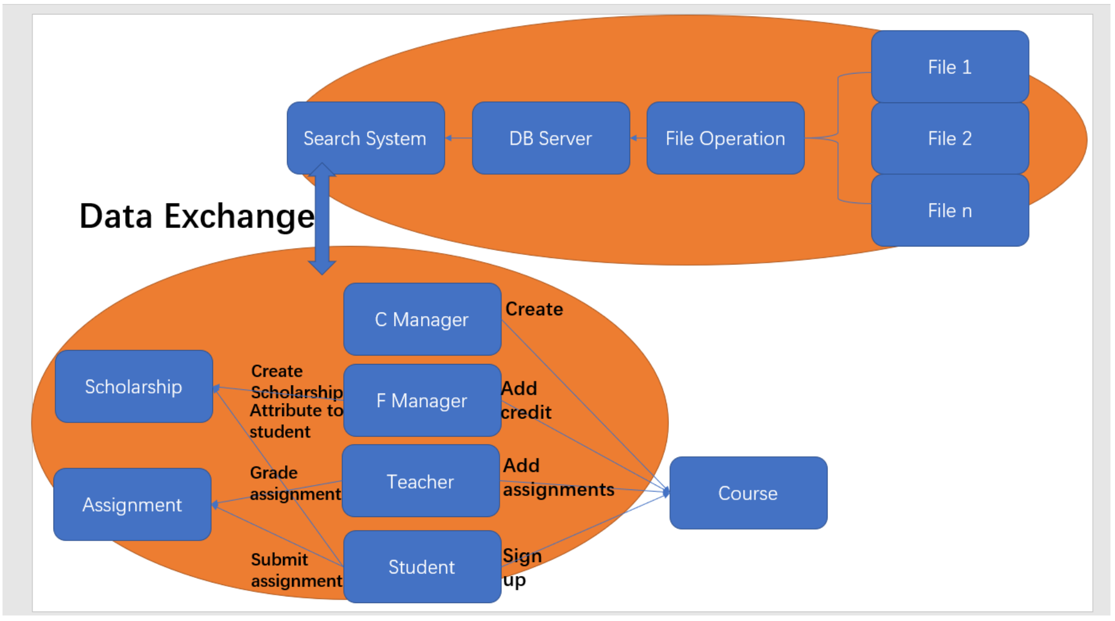
For cc package

Those classes could be illustrated as below,



1, describe model

While for the connections of all the entities is:



(1) There are types of entities, human entity and school entity,
   School objects, Course, Assignment, Scholarship
   Human staff, Course Manager, Finance Manager, Teacher, Student

(2) And for the relationships of different entities,
   Course, Assignment (1 to N), (One course includes many assignments)

   Teacher, Course (1 to N), (One teacher teaches many courses)

   Student, Course (1 to N), (One student signs up many courses)

Course Manager, Finance Manager, Student and Teacher could share some common features, so do assignments (inheritance)

Course Type and Scholarship could use interface.

Course Manager create Course with certain attributes such as id, name, capacity and type and so on, Finance Manager add credit for each course, and Teacher create assignments for the taught course, and then Student could sign up course or not.

Finance Manager create scholarship and could attribute scholarships to students.

For every course, if student sign up it, student needs to submit answer for every assignment of this course and teacher could grade it.

(3) Files are used to store data, because human entities could be grouped into staff List, and course could be stored in a List, and the same to assignment List,
File Operation is used to read and write data from and to files,
DB Server is used to store the whole data as lists,
And Search System is used to search certain entities by id, name or some other features, such as assignment, course, student.
This is the reason why almost every entity needs to be given an id.

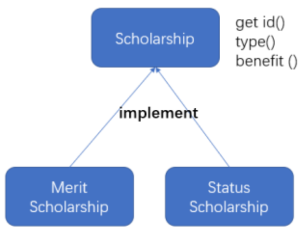
2, where to use Collection

(1) **List**, there are lots of List uses in Courses, Student and Teacher classes,
Because each Couse includes several assignments, Student sign up some courses, and Teacher teaches a few courses, List here is used which is simple and efficient, when store and retrieve data, and every time only one instance is stored or retrieved.

(2) **Map**, several classes also use Map to store data, for example, the assignments grades of a student for a certain course, every time there are two types of data are processed, assignment id and grade, therefore key and value of Map is very efficent

3, where to use Interface

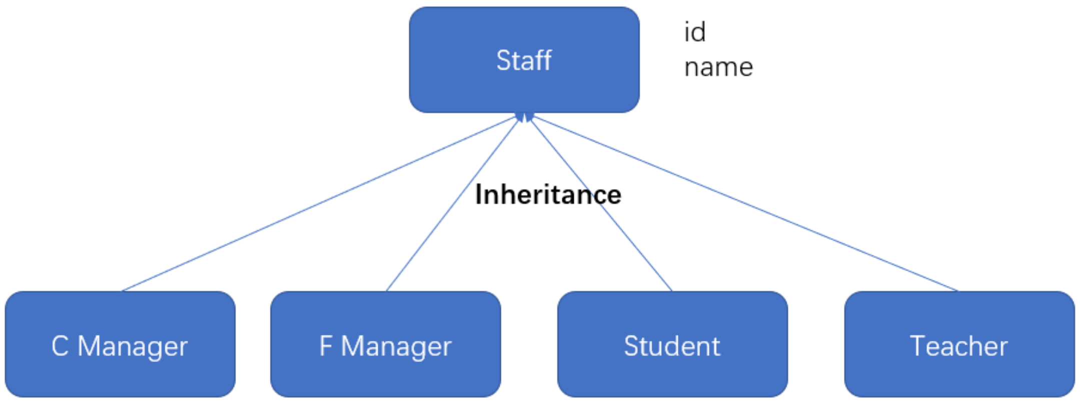
(1) Scholarship, and Course Type,



There are two kinds of scholarships, based on merit and status, which could be implemented by using interface, more efficient and better compatible, simultaneously flexible.

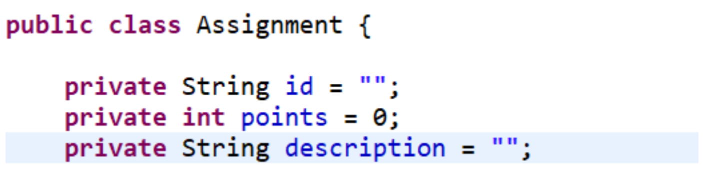
4, where to use Inheritance

(1) Staff, which is used to group Course Manager, Finance Manager, Teacher and Student,



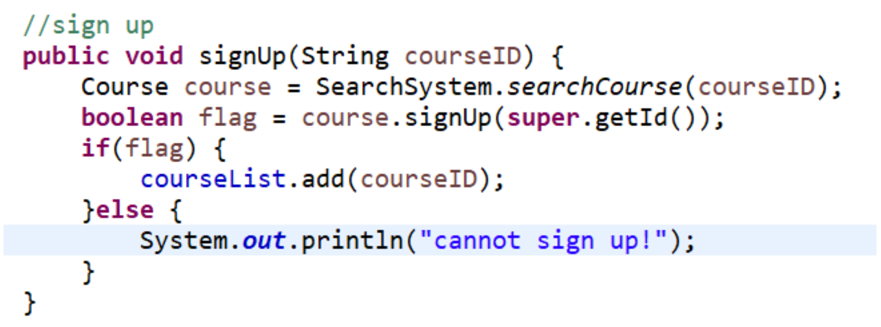
Because those human entities share some the same features, grouping them could be easier to manage the data of them.

5, example of variable declaration



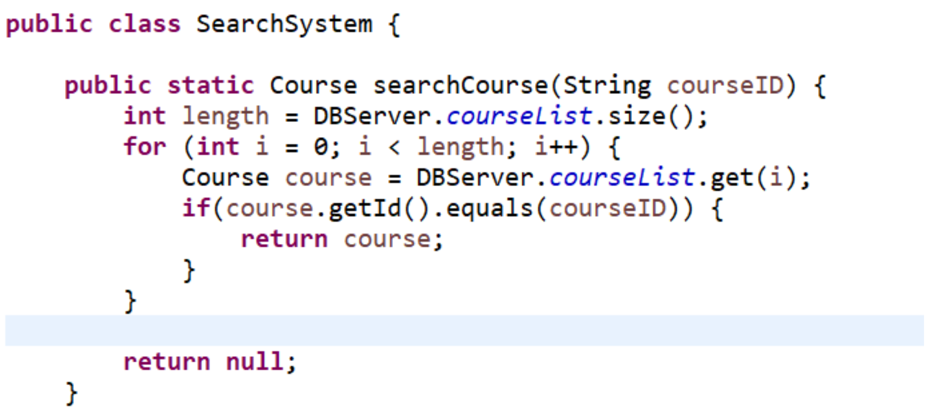
```
public class Assignment {

    private String id = "";
    private int points = 0;
    private String description = "";
```

6, example of conditional control flow



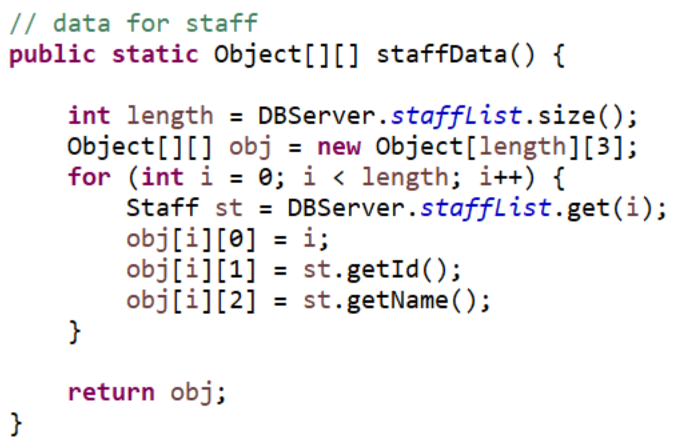
```
//sign up
public void signUp(String courseID) {
    Course course = SearchSystem.searchCourse(courseID);
    boolean flag = course.signUp(super.getId());
    if(flag) {
        courseList.add(courseID);
    }else {
        System.out.println("cannot sign up!");
    }
}
```

7, example of loop



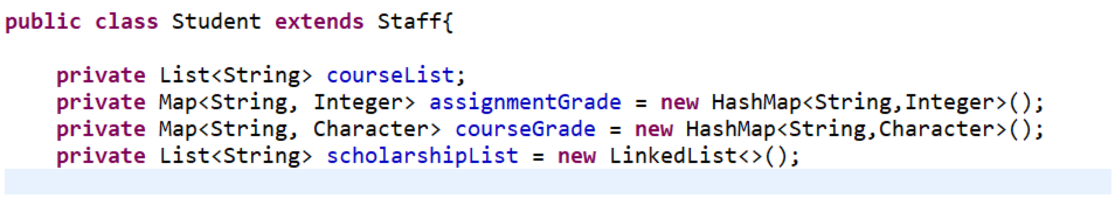
```
public class SearchSystem {

    public static Course searchCourse(String courseID) {
        int length = DBServer.courseList.size();
        for (int i = 0; i < length; i++) {
            Course course = DBServer.courseList.get(i);
            if(course.getId().equals(courseID)) {
                return course;
            }
        }

        return null;
    }
```

8, example of Primitive type variable and Object type variable



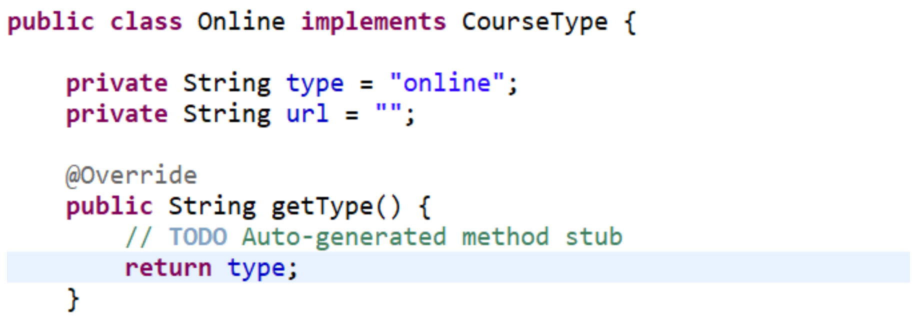
```
// data for staff
public static Object[][] staffData() {

    int length = DBServer.staffList.size();
    Object[][] obj = new Object[length][3];
    for (int i = 0; i < length; i++) {
        Staff st = DBServer.staffList.get(i);
        obj[i][0] = i;
        obj[i][1] = st.getId();
        obj[i][2] = st.getName();
    }

    return obj;
}
```

9, example of Inheritance



```
public class Student extends Staff{

    private List<String> courseList;
    private Map<String, Integer> assignmentGrade = new HashMap<String,Integer>();
    private Map<String, Character> courseGrade = new HashMap<String,Character>();
    private List<String> scholarshipList = new LinkedList<>();
```

10, example of function overriding



```java
public class Online implements CourseType {

    private String type = "online";
    private String url = "";

    @Override
    public String getType() {
        // TODO Auto-generated method stub
        return type;
    }
```