

In this assignment, students were asked to program a grade system.

There are five courses in the system including Database, Data Structure, Operating System, Mathematics, and System Design.

Several functions should be included in this system:

1. Add

add a student name and the grade of a course. It allows users to continue to add if they choose not to input enter.

1. Delete

Delete grade of one course or delete all the grades of a student.

1. Search

Search a student’s all grades or a selected course. If there is no input name of the student, it will prompt and exit to last level.

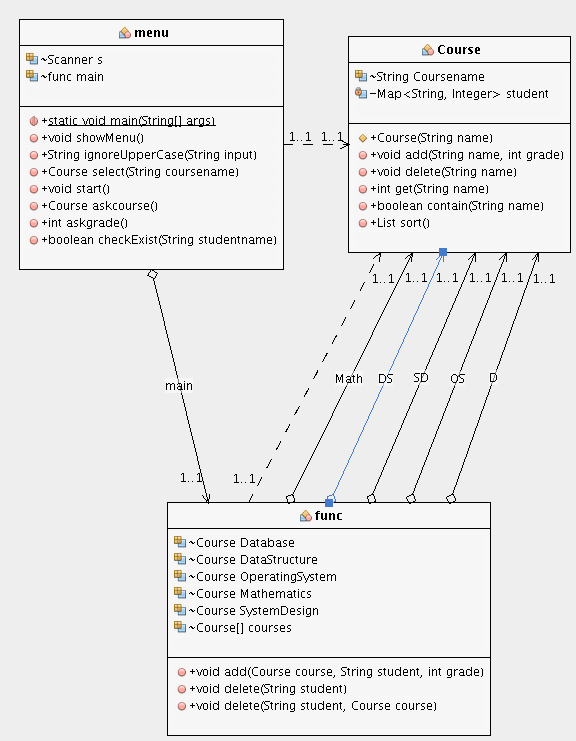
1. Update

Update an existed grade of a student. If there is no input name of the student, it will prompt and exit to last level.

1. List

By providing a course name, list all the grades and students in descending orders of the grades. If students with same grades, list in ascending alphabetical order of names. And display the average of the grades.

As for the structure of the system, here is the UML diagram.



**Diagram 1:** UML structure of the system

In this diagram, the arrows from Func to Course are shorthand. Math, DS, SD, OS, D represent Mathematics, Data Structure, System Design, Operating System, Database.

There are three classes including Course, Func, and Menu. The Course class is the basic class in this system, which stores the information of student’s name and grade in each course. The func class is focusing on storing different courses, and add provides encapsulation add method.

As for Menu class, it provides a main method to run the program. Meanwhile, menu structure is in this class including case statements related to various opeartions. In addition, some validation methods are included in this class.

A Map structure is used and named as student, whose key represents the name of student in String data type and value represents the grades of student in Int data type limited from 0 to 100. The Map structure is supposed to be suitable for storing data in this system, since It stores data in the form of key and value pairs where every key is unique, which helps to find the grade of a student.

For example, the data is stored in this way:

**Key        :       Value**

John      :          81

Peter     :          75

In each course, there is a Map to store the data, which is convenient for searching information by given key.

In Course class, the list function depends on the sort () method, which is the core algorithm in list courses. The method override the comparator to define the rules of sorting.

The pseudo code:

compare

//input Map.Entry<String, Integer> o1, Map.Entry<String, Integer> o2

If (o1.value == o2.value) then

return o1.getKey().compareTo(o2.getKey())

else

return -(o1.getValue().compareTo(o2.getValue()))

End

In this code, o1.getKey().compareTo(o2.getKey()) means that it uses the compareTo method to compare the key of o1 and o2 (name of students), which is in ascending order. -(o1.getValue().compareTo(o2.getValue()))

Means that it uses compareTo method to compare the value of o1 and o2 (grade of students), which is in descending order.

The time complexity of this method is **O(1)**.

However, when Collections.sort() is applied in map related to this method, it actually makes use of merge sort. The time complexity of merge sort is O(n logn), which can be calculated by iterative mehod.

Therefore, the time complexity of this sort is O(1)\* O(n logn) = **O(n logn)**.

As for add function, the time complexity is **O(1)**, without any iteration.

The update function, it focuses on a selected course, but it requires to search the key of the map, Time complexity: **O(n)**.

As for search function, when search with no course, it will search for all the course with the student name, time complexity: O(5\*n) = **O(n)**. When search a selected course, the time complexity is **O(n).**

As for the delete function, it is similar to search function, which is considered to have two situations. The complexity is **O(n)**.

Appendix:

**Menu.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package cse104ass2;

import java.util.List;

import java.util.Map;

import java.util.Scanner;

/\*\*

\*

\* @author Hongyi.Wu

\*/

public class Menu {

public static void main(String[] args){

// TODO code application logic here

Menu test = new Menu();

//start the program

while (true)

test.start();

}

Scanner s = new Scanner(System.in);

Func main = new Func();

public void showMenu(){

System.out.println();

System.out.println(

"Welcome to the students’ grades system, functions provided include the following:");

System.out.println(

"o Add – to add a new student name together with course/grade info into the system");

System.out.println(

"o Update – to update the grade of an existing course of a student");

System.out.println(

"o Search – to enquire about the grades of a specific student in the system");

System.out.println("o Delete – to deleteastudent’s info");

System.out.println(

"o List – to list all grades of a specific course in descending order");

System.out.println(

"o Quit – to exit from the current level of interactions");

System.out.println();

}

public String ignoreUpperCase(String input){

return input.toLowerCase();

}//ignore each input about the capitalization

public Course select(String coursename){

if (coursename.equals("data structure"))

return main.DataStructure;

if (coursename.equals("database"))

return main.Database;

if (coursename.equals("mathematics"))

return main.Mathematics;

if (coursename.equals("operating system"))

return main.OperatingSystem;

if (coursename.equals("system design"))

return main.SystemDesign;

else

return null;

}//match input with courses

public void start(){

showMenu();

//firstly show the menu

String in = ignoreUpperCase(s.nextLine());

//input ignores capitalization

switch (in) {

case "add":

System.out.print("Enter student’s name: ");

String addname = ignoreUpperCase(s.nextLine());

//judge if it is in the system

for (Course course : main.courses) {

if (course.contain(addname)) {

System.out.println(addname + " is already in the system.");

break;

}

}

while (true) {

Course addcourse = askcourse();

if (addcourse == null) {

System.out.println("stop adding!");

break;

}

if (!addcourse.contain(addname)) {

int addgrade = askgrade();

addcourse.add(addname, addgrade);

} else {

System.out.println(

"there is already the grade in this course");

}

}

System.out.print("new entered: " + addname + " ");

for (Course course : main.courses) {

if (course.contain(addname))

System.out.print(course.Coursename + "/" + course.get(

addname) + " ");

}

break;

case "update":

System.out.print("Enter student’s name: ");

String updatename = ignoreUpperCase(s.nextLine());

//input ignores capitalization

if (!checkExist(updatename))

break;

//if no this name in system, stop it

Course updatecourse = askcourse();

while (updatecourse == null)

updatecourse = askcourse();

//if input a blank, continues to ask

if (updatecourse.contain(updatename)) {

int updaterecord = updatecourse.get(updatename);

int updategrade = askgrade();

//then ask the grade

updatecourse.add(updatename, updategrade);

System.out.print("student" + updatename + ", " +

updatecourse.Coursename + " grade updated from " +

updaterecord + " to " + updategrade);

} else {

System.out.println("There is no grade to update");

}

break;

case "search":

System.out.print("Enter student’s name: ");

String searchname = ignoreUpperCase(s.nextLine());

Course searchcourse = askcourse();

//if blank, search all the courses

if (searchcourse == null) {

int sum = 0;

int counter = 0;

System.out.print("Student: " + searchname + " ");

for (Course course : main.courses) {

if (course.contain(searchname)){

System.out.print(course.Coursename + "/" + course.get(

searchname) + "; ");

sum += course.get(searchname);

counter++;

}

}

System.out.print("Average/"+(sum/counter)+'.');

} else {

System.out.print("Student: " + searchname + " ");

if (searchcourse.contain(searchname)){

System.out.print(searchcourse.Coursename + "/" +

searchcourse.get(searchname) + "; ");

}

}

//else, search the focus course

break;

case "list":

Course listcourse = askcourse();

while (listcourse == null)

listcourse = askcourse();

//if input a blank, continues to ask

List sorted = listcourse.sort();

if (sorted == null) {

System.out.println("No grades in this course.");

break;

}//special situations

System.out.print(listcourse.Coursename + ": ");

int sum = 0;

for (Object s : sorted) {

Map.Entry temp = (Map.Entry)s;

sum += (int)temp.getValue();

//display the result of the program

System.out.print(temp.getKey() + "/" + temp.getValue() + "; ");

}

int avg = sum / (sorted.size());

//caculate the average score

System.out.print("Average/" + avg + ".");

break;

case "delete":

System.out.print("Enter student’s name: ");

String deletename = ignoreUpperCase(s.nextLine());

Course deletecourse = askcourse();

if (!checkExist(deletename)) {

break;

//no this name then stop

} else if (deletecourse == null) {

System.out.print("Course deleted: " + deletename + "; ");

main.delete(deletename);

//delete all the course of the student

} else {

if (deletecourse.contain(deletename)) {

System.out.print("Course deleted: " + deletename + "; " +

deletecourse.Coursename + "/" + deletecourse.get(

deletename) + "; ");

deletecourse.delete(deletename);

} else {

System.out.print("No " + deletename + " found.");

}

}

//find the course of the student and delete it

break;

case "quit":

System.out.println("Bye-Bye");

System.exit(0);

break;

default:

System.out.println("Please enter the above commands.");

}

s.nextLine();

}

public Course askcourse(){

//ask the course including in demand, if it is not, ask again

boolean flag = false;

String course;

do {

flag = false;

System.out.print("Enter course name(can be blank): ");

course = ignoreUpperCase(s.nextLine());

if (course.equals(""))

return null;

if (select(course) == null) {

System.out.println("no this course, please input again");

flag = true;

}

} while (flag);

return select(course);

}

public int askgrade(){

//ask to input a valid grade

boolean con = false;

int number = 0;

do {

System.out.print("NewGrade: ");

String in = s.nextLine();

try {

number = Integer.parseInt(in);

con = false;

} catch (NumberFormatException e) {

con = true;

}

if (!((number >= 0) && (number <= 100)))

// judge if it is from 0 to 100

con = true;

} while (con);

// if not, input again

return number;

}

public boolean checkExist(String studentname){

//check if the student exists in system

boolean flag = false;

for (Course course : main.courses) {

if (course.contain(studentname))

flag = true;

}

if (flag == false) {

System.out.println("No " + studentname + " found");

return false;

}

return true;

}

}

**Func.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package cse104ass2;

/\*\*

\*

\* @author Hongyi.Wu

\*/

public class Func {

Course Database = new Course("Database");

Course DataStructure = new Course("Data Structure");

Course OperatingSystem = new Course("Operating System");

Course Mathematics = new Course("Mathematics");

Course SystemDesign = new Course("System Design");

Course[] courses = { Database, DataStructure, OperatingSystem, Mathematics, SystemDesign };

//create five courses and put in an array

public void add(Course course, String student, int grade){

course.add(student, grade);

}//store the student and the grade in a coruse

public void delete(String student){

for (Course course : courses) {

if (course.contain(student)) {

System.out.print(course.Coursename + "/" + course.get(student) + "; ");

course.delete(student);

}

}

}////delete the student and the grade in a coruse

public void delete(String student, Course course){

if (course.contain(student))

course.delete(student);

//delte a student in one course

}

}

**Course.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package cse104ass2;

import java.util.Collections;

import java.util.Comparator;

import java.util.HashMap;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;

/\*\*

\*

\* @author Hongyi.Wu

\*/

public class Course {

String Coursename;

private Map<String, Integer> student = new HashMap<String, Integer>();

// add the map to store students' name and grades

public Course(String name){

Coursename = name;

}

public void add(String name, int grade){

student.put(name, grade);

} // encapsulation

public void delete(String name){

student.remove(name);

} // encapsulation

public int get(String name){

return student.get(name);

} // encapsulation

public boolean contain(String name){

return student.containsKey(name);

} // encapsulation

public List sort(){ // the method to list grades in a course

if (student.isEmpty()) {

return null;

} // if there is no students, return null

List list = new LinkedList(student.entrySet());

Collections.sort(list, new Comparator<Map.Entry<String, Integer> >(){

@Override

public int compare(Map.Entry<String, Integer> o1, Map.Entry<String, Integer> o2){

if (o1.getValue().compareTo(o2.getValue()) == 0) {

return o1.getKey().compareTo(o2.getKey());

//if students have the same grades, sort them in alphabetical order

} else {

return -(o1.getValue().compareTo(o2.getValue()));

//else sort them in grade descending order

}

}

});

return list;

}

}