```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package control;

import adt.*;
import boundary.CourseMaintenanceUI;
import boundary.CourseProgramMaintenanceUI;
import dao.DAO;
import dao.Initializer;
import entity.Course;
import entity.Course.Sem;
import entity.CourseProgram;
import entity.Program;
import java.io.IOException;
import utility.*;

/**
 *
 * @author Chew Lip Sin
 */
public class CourseMaintenance {

//    private ListInterface<Course> courseList = new ArrList<>();
    private final CourseMaintenanceUI courseUI = new CourseMaintenanceUI();
    private final Sort s = new Sort();
    private static final DAO<Course> cDAO = new DAO<>();
    private static final DAO<Program> pDAO = new DAO<>();
    private final DAO<CourseProgram> cpDAO = new DAO<>();
    private static final CourseProgramMaintenance cpm = new CourseProgramMaintenance();
    private final Initializer in = new Initializer();

    public CourseMaintenance() {
    }

    public void runCourseMaintenance() throws IOException, InterruptedException {
        ListInterface<Course> courseList = cDAO.retrieveFromFile("course.dat");
//        ListInterface<Course> courseList = in.initializeCourse();
//        cDAO.saveToFile(courseList, "course.dat");
        ListInterface<Program> programList = pDAO.retrieveFromFile("program.dat");
//        pDAO.saveToFile(programList, "program.dat");
//        ListInterface<Program> programList = in.ProgramInitializer();
//        pDAO.saveToFile(programList, "program.dat");
```

```java
        int choice;
        do {
            Command.cls();
            choice = courseUI.getMenuChoices();
            switch (choice) {
                case 0:
                    break;
                case 1:
                    addNewCourse(courseList);
                    break;
                case 2:
                    removeCourse(courseList);
                    break;
                case 3:
                    searchCourse(courseList, "search");
                    break;
                case 4:
                    searchCourse(courseList, "ammend");
                    break;
                case 5:
                    listedCourse(courseList);
                    break;
                case 6:
                    CourseProgramMaintenance coursePM = new CourseProgramMaintenance();
                    coursePM.runCourseProgramMaintenance(courseList, programList);
                    break;
                case 7:
                    CourseGenerateReportMaintenance courseGRM = new
CourseGenerateReportMaintenance();
                    courseGRM.runCourseGenerateReportMaintenance();
                    break;
                default:
                    MessageUI.displayInvalidChoiceMessage();

            }
        } while (choice != 0);

    }

    private void addNewCourse(ListInterface<Course> courseList) {
        boolean loop = false;
        do {
            Command.cls();
            boolean newCourseFull, confirm = true;
            Course newCourse = courseUI.inputCourseDetails(courseList);
            newCourseFull = (!newCourse.equals(new Course(null, null, 0, null)));
            if (newCourseFull == true) {
                confirm = askConfirmation(newCourse, "New", "add");
                if (confirm == true) {
                    courseList.add(newCourse);
                    cDAO.saveToFile(courseList, "course.dat");
                    MessageUI.displaySuccessConfirmationMessage("Added");
                    loop = courseUI.getConfirmationChoice("add", 0);
                } else {
                    loop = false;
                }
            } else {
                loop = false;
            }
            newCourse = null;
            newCourseFull = false;
        } while (loop);
    }

    public boolean askConfirmation(Course newCourse, String val, String val2) {
        courseUI.displayCourse(newCourse, val);
        return courseUI.getConfirmationChoice(val2, 1);
    }

    private void removeCourse(ListInterface<Course> courseList) {
        boolean loop = false;
        do {
            Command.cls();
            boolean deleteCourse, confirm;
            int courseSelected = courseUI.inputRemoveCode(courseList);
            deleteCourse = (courseSelected > -1);
            if (deleteCourse == true) {
                confirm = askConfirmation(courseList.getEntry(courseSelected), "Remove", "remove");
                if (confirm == true) {
                    LinkedListInterface<CourseProgram> courseProgramList = cpDAO.dLLRetrieveFromFile("courseProgram.dat");
                    cpm.deleteCourse(courseList.getEntry(courseSelected), courseProgramList);
                    courseList.remove(courseSelected);
                    cDAO.saveToFile(courseList, "course.dat");
                    MessageUI.displaySuccessConfirmationMessage("Removed");
                    Command.pressEnterToContinue();
                    loop = courseUI.getConfirmationChoice("remove", 0);
                } else {
                    loop = false;
                }
            } else {
                loop = false;
            }
        } while (loop);
    }

    private void searchCourse(ListInterface<Course> courseList, String val) {
        boolean loop = true;
        do {
            Command.cls();
            int choice = courseUI.getSearchMenuChoices();
            switch (choice) {
                case 0:
                    loop = false;
                    break;
                case 1:
                    searchCourseCode(courseList, val);
                    break;
                case 2:
                    searchCourseTitle(courseList, val);
                    break;
                default:
                    MessageUI.displayInvalidChoiceMessage();

            }
        } while (loop);
    }

    public void searchCourseCode(ListInterface<Course> courseList, String val) {
        boolean loop;
        do {
            int found = courseUI.getSearchCourseCode(courseList);
            if (found == -1) {
                MessageUI.displayNotFoundMessage();
            } else {
                courseUI.displayCourseDetailsFounded(courseList, found);
                if (val == "ammend") {
                    ammendCourse(courseList, found);
                }
            }
            loop = courseUI.getConfirmationChoice(val, 0);
        } while (loop == true);

    }

    private void searchCourseTitle(ListInterface<Course> courseList, String val) {
        boolean loop = true;
        ListInterface<Course> courseList2 = new ArrList();
        do {
            int found = courseUI.getSearchCourseTitle(courseList, courseList2);
            if (found != 0) {
                if (found == -1) {
                    MessageUI.displayNotFoundMessage();
                } else if (found == 1) {
                    if (val == "ammend") {

                        int index = courseUI.getCourseAmmend(courseList, courseList2);
                        if (index != -1) {
                            ammendCourse(courseList, index);
                        }else{
                            loop = false;
                        }
                    } else {
                        courseUI.displayCourseFounded(courseList2);
                    }
                }
                courseList2.clear();
                loop = courseUI.getConfirmationChoice(val, 0);
            } else if (found == 0) {
                loop = false;
            }
        } while (loop);
    }

    private void ammendCourse(ListInterface<Course> courseList, int found) {
        int choice = 0;
        Course tempCourse = courseList.getEntry(found + 1);
        Course tempCourse2 = tempCourse;
        String newCourseCode = tempCourse.getCourseCode();
        String newTitle = tempCourse.getTitle();
        Sem newSem = tempCourse.getSemester();
        int newCreditHours = tempCourse.getCreditHours();
//         do {
        do {
            Command.cls();
            choice = courseUI.getAmmendMenuChoices();

            switch (choice) {
                case 0:
                    break;
                case 1:
                    newCourseCode = courseUI.inputCourseCode(courseList);
                    if (newCourseCode.equals("0")) {
                        newCourseCode = tempCourse.getCourseCode();
                    }
                    break;
                case 2:
                    newTitle = courseUI.inputTitle();
                    if (newTitle.equals("0")) {
                        newTitle = tempCourse.getTitle();
                    }
                    break;
                case 3:
                    newCreditHours = courseUI.inputCreditHour();
                    if (newCreditHours == 0) {
                        newCreditHours = tempCourse.getCreditHours();
                    }
                    break;
                case 4:
                    newSem = courseUI.inputSemester();
                    if (newSem == null) {
                        newSem = tempCourse.getSemester();
                    }
                    break;
                default:
                    MessageUI.displayInvalidChoiceMessage();
            }
        } while (choice != 0);
        tempCourse = new Course(newCourseCode, newTitle, newCreditHours, newSem);
        tempCourse.setCreatedAt(tempCourse2.getCreatedAt());
        boolean match = courseList.contains(tempCourse);
        int match2 = tempCourse2.compareSem(tempCourse.getSemester());
        if (!match || match2 != 0) {
            boolean confirm = askConfirmation(tempCourse, "Ammend", "ammend");
            if (confirm == true) {
                LinkedListInterface<CourseProgram> cp = new DoublyLinkedList<>();
                cp = cpDAO.dLLRetrieveFromFile("courseProgram.dat");

                cpm.modifyCourse(tempCourse, tempCourse2, cp);
                tempCourse.update();
                courseList.replace(found + 1, tempCourse);
                cDAO.saveToFile(courseList, "course.dat");
                MessageUI.displaySuccessConfirmationMessage("Ammend");
                Command.pressEnterToContinue();
//                      loop = courseUI.getConfirmationChoice("ammend", 0);
            }
        } else {
            MessageUI.printFormattedText("No changing\n", ConsoleColor.YELLOW);
        }
//        } while (loop);

    }

    public void listedCourse(ListInterface<Course> courseList) {
        int choice;
        s.insertionSort(courseList, "courseCode");
        courseUI.listAllCourses(courseList);
        do {
            choice = courseUI.getSortMenu(courseList);
            Command.cls();
            switch (choice) {
                case 0:
                    break;
                case 1:
                    ascListedCourse(courseList);
                    break;
                case 2:
                    desListedCourse(courseList);
                    break;
                default:
                    MessageUI.displayInvalidChoiceMessage();
            }
        } while (choice != 0);
    }

    private void ascListedCourse(ListInterface<Course> courseList) {
        int choice;
        do {
            choice = courseUI.getSortMenuChoice(courseList, "Ascending Order");
            switch (choice) {
                case 0:
                    break;
                case 1:
                    s.insertionSort(courseList, "courseCode");
                    break;
                case 2:
                    s.insertionSort(courseList, "title");
                    break;
                case 3:
                    s.insertionSort(courseList, "creditHours");
                    break;
                case 4:
                    s.insertionSort(courseList, "semester");
                    break;
```

```java
                case 5:
                    s.insertionSort(courseList, "createdAt");
                    break;
                case 6:
                    s.insertionSort(courseList, "updatedAt");
                    break;
                default:
                    MessageUI.displayInvalidChoiceMessage();
            }
            courseUI.listAllCourses(courseList);
        } while (choice != 0);
    }

    private void desListedCourse(ListInterface<Course> courseList) {
        int choice;
        do {
            choice = courseUI.getSortMenuChoice(courseList, "Descending Order");
            switch (choice) {
                case 0:
                    break;
                case 1:
                    s.insertionSortDes(courseList, "courseCode");
                    break;
                case 2:
                    s.insertionSortDes(courseList, "title");
                    break;
                case 3:
                    s.insertionSortDes(courseList, "creditHours");
                    break;
                case 4:
                    s.insertionSortDes(courseList, "semester");
                    break;
                case 5:
                    s.insertionSortDes(courseList, "createdAt");
                    break;
                case 6:
                    s.insertionSortDes(courseList, "updatedAt");
                    break;
                default:
                    MessageUI.displayInvalidChoiceMessage();
            }
            courseUI.listAllCourses(courseList);
        } while (choice != 0);

    }

```

```java
}
```