```java
/*
 * 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 boundary;

import adt.ArrList;
import adt.DoublyLinkedList;
import adt.LinkedListInterface;
import adt.ListInterface;
import adt.OrderClause;
import adt.StackInterface;
import entity.Course;
import entity.Course.Sem;
import entity.CourseCodeComparator;
import entity.CourseProgram;
import entity.CreditHoursComparator;
import entity.SemesterComparator;
import entity.TitleComparator;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Iterator;
import utility.Command;
import utility.ConsoleColor;
import utility.InputValue;
import utility.MessageUI;

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

    private LinkedListInterface<CourseProgram> cp = new DoublyLinkedList<>();
    private ListInterface<Course> courses = new ArrList<>();
    private final InputValue iv = new InputValue();

    /**
     * Constructs a CourseGenerateReportMaintenanceUI with the given course
     * programs and courses.
     *
     * @param cp The linked list of course programs.
     * @param courses The list of courses.
     */
    public CourseGenerateReportMaintenanceUI(LinkedListInterface<CourseProgram> cp, ListInterface<Course> courses) {
        this.cp = cp;
        this.courses = courses;
    }

    private final LocalDateTime myDateObj = LocalDateTime.now();
    DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("dd-MM-yyyy");
    DateTimeFormatter myFormatObj2 = DateTimeFormatter.ofPattern("h:mm a");
    String formattedDate = myDateObj.format(myFormatObj);
    String formattedTime = myDateObj.format(myFormatObj2);
    private final CourseCodeComparator cCodeC = new CourseCodeComparator();
    private final CreditHoursComparator cHoursC = new CreditHoursComparator();
    private final TitleComparator titleC = new TitleComparator();
    private final SemesterComparator semC = new SemesterComparator();

    /**
     * Displays the header for the course report menu.
     */
    public void displayHeader() {
        Command.cls();
        System.out.println("\t\t\t\t==============================================");
        System.out.println("\t\t\t\t||            Course Report Menu            ||");
        System.out.println("\t\t\t\t==============================================");
    }

    /**
     * Displays the options available in the course report menu.
     */
    public void displayReportMenu() {
        System.out.println("\t\t\t\t1. Generate Course and Program Report");
        System.out.println("\t\t\t\t2. Generate Course Report");
        System.out.println("\t\t\t\t0. Exit");
    }

    /**
     * Gets the user's choice from the course report menu.
     *
     * @return The user's choice.
     */
    public int getChoices() {
        int choice;
        do {
            System.out.print("\t\t\t\tEnter choice: ");
            choice = iv.readInteger();
            if (choice > 2 || choice < 0) {
                MessageUI.displayInvalidChoiceMessage();
            }
        } while (choice > 2 || choice < 0);
        return choice;
    }

    /**
     * Displays the header for the course program report.
     */
    public void displayCoursePReportHeader() {
        Command.cls();
        System.out.println("\t\t\t\t==============================================");
        System.out.println("\t\t\t\t||           Course Program Report           ||");
        System.out.println("\t\t\t\t==============================================");
    }

    /**
     * Displays the header for the course report.
     */
    public void displayCourseReportHeader() {
        Command.cls();
        System.out.println("\t\t\t\t==============================================");
        System.out.println("\t\t\t\t||              Course Report                ||");
        System.out.println("\t\t\t\t==============================================");
    }

    /**
     * Displays a progress animation, indicating the generation of a report.
     */
    public void progress() {
        var anim = "|/-\\";
        for (int i = 0; i <= 100; i = i + 50) {
            Command.progressPercentage(i, 100);
            try {
                Thread.sleep(500);
            } catch (Exception e) {
            }
        }
        MessageUI.displaySuccessConfirmationMessage("Generating Report");
        Command.pressEnterToContinue();
    }

    /**
     * Displays the course program report, showing course information grouped by
     * program.
     */
    public void displayCourseProgramReport() {
        String line = "";
        sortByProgramID();
        sortById();
        String oldCC = "";
        System.out.println("");
        System.out.println(String.format("\t\t\t%-20s|  %-20s|%-20s", "Course Code", "Program Code", "Main/Elective"));
        for (int i = 0; i < 65; i++) {
            line += "-";
        }
        Iterator<CourseProgram> it = cp.getIterator();
        while (it.hasNext()) {
            CourseProgram cp2 = it.next();
            if (oldCC.equals(cp2.getCourseCode())) {
                System.out.print(String.format("\t\t\t%-20s|", ""));
                String elective = strElective(cp2.isIsElective());
                System.out.println(String.format("  %-20s|  %-20s", cp2.getProgramCode(), elective));

            } else {
                System.out.println("\t\t\t" + line);
                System.out.print(String.format("\t\t\t%-20s|", cp2.getCourseCode()));
                String elective = strElective(cp2.isIsElective());
                System.out.println(String.format("  %-20s|  %-20s", cp2.getProgramCode(), elective));

            }
            oldCC = cp2.getCourseCode();

        }

        System.out.println("\t\t\t" + line);
        displayReportFooter();
        System.out.println("");
        System.out.print("\t\t\t");
        Command.pressEnterToContinue();
    }

    /**
     * Sorts the list of course programs by their course codes.
     */
    public void sortById() {
        cp.orderBy((c1, c2)
                -> c1.getCourseCode().compareTo(c2.getCourseCode()) < 0
                ? OrderClause.MOVE_FORWARD : OrderClause.MOVE_BACKWARD);
    }

    /**
     * Sorts the list of course programs by their program codes.
     */
    public void sortByProgramID() {
        cp.orderBy((c1, c2)
                -> c1.getProgramCode().compareTo(c2.getProgramCode()) < 0
                ? OrderClause.MOVE_FORWARD : OrderClause.MOVE_BACKWARD);
    }

    /**
     * Converts a Boolean value indicating whether a course is elective or not
     * into a string representation.
     *
     * @param isElective True if the course is elective, false otherwise.
     * @return The string representation "Elective" if true, or "Main" if false.
     */
    private String strElective(boolean isElective) {
        if (isElective == true) {
            return "Elective";
        } else {
            return "Main";
        }

    }

    /**
     * Displays the footer information for the generated report.
     */
    public void displayReportFooter() {
        System.out.println("\n\t\t\tReport Generated by: University Education System");
        System.out.println("\t\t\tReport Generated at: " + formattedDate + " " + formattedTime);
    }

    /**
     * Gets user choices for generating course reports and maintains a stack of
     * chosen reports.
     *
     * @param choice A stack containing the user's report choices.
     * @return The updated stack of report choices.
     */
    public StackInterface<String> getCourseReportMenu(StackInterface<String> choice) {
        ListInterface<String> typeReport = new ArrList<>();
        boolean checkCode = false;
        boolean checkCredit = false;
        boolean checkSem = false;
        typeReport.add("Course Code Report");
        typeReport.add("Credit Hour Report");
        typeReport.add("Semester Report");

        int choice2;
        do {
            Command.cls();
            System.out.println("\t\t\t\t==============================================");
            System.out.println("\t\t\t\t||              Course Report Menu          ||");
            System.out.println("\t\t\t\t==============================================");
            System.out.println("\t\t\t\t1. Course Code Report");
            System.out.println("\t\t\t\t2. Credit Hour Report");
            System.out.println("\t\t\t\t3. Semester Report");
            System.out.println("\t\t\t\t4. Undo");
            System.out.println("\t\t\t\t0. Exit/Continue");
            System.out.print("\t\t\t\tEnter your choice(One report only choose one time): ");
            choice2 = iv.readInteger();
            if (choice2 < 0 || choice2 > 4) {
                MessageUI.displayInvalidChoiceMessage();
            } else if (choice2 == 1 && !checkCode) {
                choice.push(typeReport.getEntry(1));
                checkCode = true;
            } else if (choice2 == 2 && !checkCredit) {
                choice.push(typeReport.getEntry(2));
                checkCredit = true;
            } else if (choice2 == 3 && !checkSem) {
                choice.push(typeReport.getEntry(3));
                checkSem = true;
            } else if (choice2 == 4) {
                if (!choice.isEmpty()) {
                    String popOut = choice.pop();
                    if (popOut.equals(typeReport.getEntry(1))) {
                        checkCode = false;
                    } else if (popOut.equals(typeReport.getEntry(2))) {
                        checkCredit = false;
                    } else if (popOut.equals(typeReport.getEntry(3))) {
                        checkSem = false;
                    }
                } else {
                    MessageUI.printFormattedText("\t\t\t\tNothing can Undo\n", ConsoleColor.YELLOW);
                }
            } else if (choice2 == 0) {
```

```java
                System.out.println("");
            } else {
                MessageUI.printFormattedText("\t\t\t\tYou have been assigned value into the stack!\n", ConsoleColor.YELLOW);
            }
            System.out.print("\t\t\t\t");
            Command.pressEnterToContinue();
        } while (choice2 != 0);
        return choice;
    }

    /**
     * Displays a report of courses grouped by their course codes.
     */
    public void displayCourseCodeReport() {
        ArrList.insertionSort(courses, cCodeC, "asc");
        Iterator<Course> itA = courses.getIterator();
        Iterator<Course> itB = courses.getIterator();
        Iterator<Course> itF = courses.getIterator();
        Iterator<Course> itM = courses.getIterator();

        String line = "\t\t\t";
        for (int i = 0; i < 65; i++) {
            line += "-";
        }
        int countA = 1, countB = 1, countF = 1, countM = 1;
        System.out.println(line);
        System.out.println("\t\t\tCourse Code Start with 'A'");
        while (itA.hasNext()) {
            Course course = itA.next();
            if (course.getCourseCode().charAt(0) == 'A') {
                System.out.println(String.format("\t\t\t%2d. %8s %s", countA, course.getCourseCode(), course.getTitle()));
                countA++;
            }
        }

        System.out.println(String.format("\t\t\tTotal = %d", countA - 1));
        System.out.println(line);
        System.out.println("\t\t\tCourse Code Start with 'B'");
        while (itB.hasNext()) {
            Course course = itB.next();
            if (course.getCourseCode().charAt(0) == 'B') {
                System.out.println(String.format("\t\t\t%2d. %8s %s", countB, course.getCourseCode(), course.getTitle()));
                countB++;

            }
        }
        System.out.println(String.format("\t\t\tTotal = %d", countB - 1));
        System.out.println(line);
        System.out.println("\t\t\tCourse Code Start with 'F'");
        while (itF.hasNext()) {
            Course course = itF.next();
            if (course.getCourseCode().charAt(0) == 'F') {
                System.out.println(String.format("\t\t\t%2d. %8s %s", countF, course.getCourseCode(), course.getTitle()));
                countF++;
            }
        }
        System.out.println(String.format("\t\t\tTotal = %d", countF - 1));
        System.out.println(line);
        System.out.println("\t\t\tCourse Code Start with 'M'");
        while (itM.hasNext()) {
            Course course = itM.next();
            if (course.getCourseCode().charAt(0) == 'M') {
                System.out.println(String.format("\t\t\t%2d. %8s %s", countM, course.getCourseCode(), course.getTitle()));
                countM++;
            }
        }
        System.out.println(String.format("\t\t\tTotal = %d", countM - 1));

    }

    /**
     * Displays a report of courses grouped by their credit hours.
     */
    public void displayCreditHoursReport() {
        ArrList.insertionSort(courses, cHoursC, "asc");
        String line = "";
        int count1 = 1, count2 = 1, count3 = 1, count4 = 1, count5 = 1;
        for (int i = 0; i < 65; i++) {
            line += "-";
        }
        System.out.println("\t\t\t" + line);
        System.out.println("\t\t\tCourse with 1 credit hour:");
        for (int i = 1; i <= courses.size(); i++) {
            if (courses.getEntry(i).getCreditHours() == 1) {
                System.out.println(String.format("\t\t\t%2d. %s %s", count1, courses.getEntry(i).getCourseCode(), courses.getEntry(i).getTitle()));
                count1++;
            }
        }
        System.out.println(String.format("\t\t\tTotal = %d", count1 - 1));
        System.out.println("\t\t\t" + line);
        System.out.println("\t\t\tCourse with 2 credit hours:");
        for (int i = 1; i <= courses.size(); i++) {
            if (courses.getEntry(i).getCreditHours() == 2) {
                System.out.println(String.format("\t\t\t%2d. %s %s", count2,
courses.getEntry(i).getCourseCode(), courses.getEntry(i).getTitle()));
                count2++;
            }
        }
        System.out.println(String.format("\t\t\tTotal = %d", count2 - 1));
        System.out.println("\t\t\t" + line);
        System.out.println("\t\t\tCourse with 3 credit hours:");

        for (int i = 1; i <= courses.size(); i++) {
            if (courses.getEntry(i).getCreditHours() == 3) {
                System.out.println(String.format("\t\t\t%2d. %s %s", count3,
courses.getEntry(i).getCourseCode(), courses.getEntry(i).getTitle()));
                count3++;
            }
        }
        System.out.println(String.format("\t\t\tTotal = %d", count3 - 1));
        System.out.println("\t\t\t" + line);
        System.out.println("\t\t\tCourse with 4 credit hours:");

        for (int i = 1; i <= courses.size(); i++) {
            if (courses.getEntry(i).getCreditHours() == 4) {
                System.out.println(String.format("\t\t\t%2d. %s %s", count4,
courses.getEntry(i).getCourseCode(), courses.getEntry(i).getTitle()));
                count4++;
            }
        }
        System.out.println(String.format("\t\t\tTotal = %d", count4 - 1));
        System.out.println("\t\t\t" + line);
        System.out.println("\t\t\tCourse with 5 credit hours:");

        for (int i = 1; i <= courses.size(); i++) {
            if (courses.getEntry(i).getCreditHours() == 5) {
                System.out.println(String.format("\t\t\t%2d. %s %s", count5,
courses.getEntry(i).getCourseCode(), courses.getEntry(i).getTitle()));
                count5++;
            }
        }
        System.out.println(String.format("\t\t\tTotal = %d", count5 - 1));
```

```
    }

    /**
     * Displays a report of courses grouped by their intake semester (January or
     * July).
     */
    public void displayCourseSemesterReport() {
        Sem janu = Sem.JAN;
        Sem july = Sem.JUL;

        Sem alll = Sem.ALL;

        ArrList.insertionSort(courses, titleC, "asc");
        ArrList.insertionSort(courses, semC, "asc");
        int countJan = 0, countJuly = 0;
        Iterator<Course> itJan = courses.getIterator();
        Iterator<Course> itJuly = courses.getIterator();
        String line = "";
        for (int i = 0; i < 65; i++) {
            line += "-";
        }
        System.out.println("\t\t\t" + line);
        System.out.println("\t\t\tCourse intake on January:");
        while (itJan.hasNext()) {
            Course course = itJan.next();
            int all = course.compareSem(alll);
            int jan = course.compareSem(janu);
            if (all == 0 || jan == 0) {
                countJan++;
                System.out.println(String.format("\t\t\t%2d. %8s %s", countJan, course.getCourseCode(), course.getTitle()));
            }
        }
        System.out.println("\t\t\tTotal = " + countJan);
        System.out.println("\t\t\t" + line);
        System.out.println("\t\t\tCourse intake on July:");
        while (itJuly.hasNext()) {
            Course course = itJuly.next();
            int all = course.compareSem(alll);
            int jul = course.compareSem(july);
            if (all == 0 || jul == 0) {
                countJuly++;
                System.out.println(String.format("\t\t\t%2d. %8s %s", countJuly, course.getCourseCode(), course.getTitle()));
            }
        }
```

```java
        System.out.println("\t\t\tTotal = " + countJuly);
    }
}
```