```java
package control;

import adt.CircularDoublyLinkedList;
import adt.ListInterface;
import adt.exampleAdt.*;
import boundary.TutorManagementUI;
import dao.*;
import entity.Tutor;
import java.time.*;
import java.util.Comparator;
import java.util.Iterator;
import utility.*;

/**
 *
 * @author Yip Zi Yan
 */
public class TutorManagement {

    private final String fileName = "tutor.dat";
    DAO tutorDAO = new DAO(fileName);
    TutorManagementUI tutorUI = new TutorManagementUI();
    ListInterface<Tutor> tutorList = new CircularDoublyLinkedList<>();

//    seeder(generate fake data)
//    TutorSeeder seeder = new TutorSeeder();
    public TutorManagement() {
//        tutorDAO.saveToFile(seeder.getTutorList());
        tutorList = tutorDAO.retrieveFromFile();
        Tutor.setTotalTutor(tutorList.getNumberOfEntries());
    }

    public void startUI() {

        int choice;
        do {
            choice = tutorUI.getMenuChoice();
            switch (choice) {
                case 1:
                    displayTutorList();
                    break;
                case 2:
                    addNewTutor();
                    break;
                case 3:
                    searchTutorUI();
                    break;
                case 4:
                    updateTutorUI();
                    break;
                case 5:
                    filterTutorUI();
                    break;
                case 6:
                    generateReportUI();
                    break;
                case 7:
                    return;
                case 0:
                    MessageUI.displayExitMessage();
                    System.exit(0);
            }
        } while (choice != 0);

    }

    public static void main(String[] args) {
        TutorManagement teachingControl = new TutorManagement();
        teachingControl.startUI();
    }

    private void addNewTutor() {
        if (!tutorUI.addTutorMenu()) {
            return;
        }
        int currentId;

        try {
            currentId = Integer.parseInt(tutorList.getLastEntry()
                    .getTutorId().substring(1)) + 1;
        } catch (Exception e) {
            System.out.println("Something went wrong.");
            GeneralUtil.systemPause();
            return;
        }

        do {
            Tutor newTutor = tutorUI.addTutor(currentId);

            if (newTutor != null) {
                tutorList.add(newTutor);
                tutorDAO.saveToFile(tutorList);
            }

        } while (tutorUI.contAction("Anymore To Add? [Y|N] > "));

    }

    private void displayTutorList() {
        tutorUI.displayAllTutor("Tutor List", getAllTutor(),
                true, 78);
    }

    private String getAllTutor() {
        String outputStr = "";
        int number = 0;

        Iterator<Tutor> it = tutorList.getIterator();
        while (it.hasNext()) {
            outputStr += String.format("%2d.  ", ++number)
                    + it.next() + "\n";

        }
        return outputStr;
    }

    private void searchTutorUI() {
        int choice;
        do {
            choice = tutorUI.findTutorMenu();
            switch (choice) {
                case 1:
                    searchTutorBy("name",
```

```java
                            "Search Tutor Name > ");
                    break;
                case 2:
                    searchTutorBy("email",
                            "Search Tutor Email > ");
                    break;
                case 3:
                    searchTutorBy("id",
                            "Search Tutor ID > ");
                    break;
            }
        } while (choice != 0);
    }

    private void searchTutorBy(String attribute, String queryQuestion) {
        String query = tutorUI.getTutorQuery(queryQuestion, "find")
                .toLowerCase();
        String outputStr = "";
        Iterator<Tutor> it = tutorList.getIterator();
        int number = 0;

        while (it.hasNext()) {
            Tutor matchTutor = it.next();
            switch (attribute) {
                case "name":
                    if (matchTutor.getTutorName().toLowerCase().startsWith(query)) {
                        outputStr += String.format("%2d.  ", ++number)
                                + matchTutor + "\n";
                    }
                    break;
                case "email":
                    if (matchTutor.getEmail().toLowerCase().contains(query)) {
                        outputStr += String.format("%2d.  ", ++number)
                                + matchTutor + "\n";
                    }
                    break;
                case "id":
                    if (matchTutor.getTutorId().toLowerCase().equals(query)) {
                        outputStr += String.format("%2d.  ", ++number)
                                + matchTutor + "\n";
                    }
                    break;
            }

        }

        tutorUI.displayFindResult(outputStr);

    }

    private void updateTutorUI() {
        int choice;
        do {
            tutorUI.displayAllTutor("Tutor List", getAllTutor(),
                    false, 78);
            choice = tutorUI.updateTutorMenu();
            switch (choice) {
                case 1:
                    updateTutor("num", "Enter No. > ");
                    break;
                case 2:
                    updateTutor("id", "Enter Tutor ID > ");
                    break;

            }
        } while (choice != 0);
    }

    private void updateTutor(String searchBy, String queryQuestion) {
        String query = tutorUI.getTutorQuery(queryQuestion, "update")
                .toLowerCase();
        Tutor updateTutor = null;
        Iterator<Tutor> it = tutorList.getIterator();

        switch (searchBy) {
            case "id":
                while (it.hasNext()) {
                    Tutor matchTutor = it.next();

                    if 
(matchTutor.getTutorId().toLowerCase().equals(query)) {
                        updateTutor = matchTutor;
                    }

                }
                break;
            case "num":
                try {
                updateTutor = tutorList
                        .getEntry(Integer.parseInt(query) - 1);
            } catch (Exception e) {
            }
            break;
        }

        updateSelectedField(updateTutor);

    }

    private void updateSelectedField(Tutor updateTutor) {
        int choice;
        boolean isRemove = false;
        do {
            tutorUI.displaySelectedTutor(updateTutor);
            if (updateTutor == null) {
                return;
            }

            choice = tutorUI.selectAttributeToUpdate(updateTutor);
            switch (choice) {
                case 1:
                    updateTutorName(updateTutor);
                    break;
                case 2:
                    updateTutorGender(updateTutor);
                    break;
                case 3:
                    updateTutorStatus(updateTutor);
                    break;
                case 4:
```

```java
                        updateSalary(updateTutor);
                        break;
                    case 5:
                        isRemove = removeTutor(updateTutor);
                        break;
                }
                if (isRemove) {
                    tutorUI.removedClosure();
                }
            } while (choice != 0 && !isRemove);
        }

        private void updateTutorName(Tutor updateTutor) {
            String newName = tutorUI.updateTutorName();
            if (newName == null) {
                return;
            }

            updateTutor.setTutorName(newName);
            tutorDAO.saveToFile(tutorList);
        }

        private void updateTutorGender(Tutor updateTutor) {
            boolean confirmToChange = tutorUI
                    .updateTutorGender(updateTutor.isFemale());
            if (!confirmToChange) {
                return;
            }

            if (updateTutor.isFemale()) {
                updateTutor.setGender('M');
            } else {
                updateTutor.setGender('F');
            }
            tutorDAO.saveToFile(tutorList);

        }

        private void updateSalary(Tutor updateTutor) {
            double newSalary = tutorUI.updateTutorSalary();
            if (newSalary == -1) {
```

```
            return;
        }

        updateTutor.setSalary(newSalary);
        tutorDAO.saveToFile(tutorList);

    }

    private void updateTutorStatus(Tutor updateTutor) {
        String newStatus = tutorUI
                .updateTutorStatus(updateTutor.getStatus());
        if (newStatus == null) {
            return;
        }

        updateTutor.setStatus(newStatus);
        tutorDAO.saveToFile(tutorList);

    }

    private boolean removeTutor(Tutor updateTutor) {
        boolean isDeleted = false;

        if (tutorUI.removeTutor()) {
            isDeleted = tutorList.remove(updateTutor);
            tutorDAO.saveToFile(tutorList);
        }
        return isDeleted;
    }

    private void filterTutorUI() {
        int choice;
        StackInterface<String> filter = new ArrayStack<>();

        do {

            choice = tutorUI.filterTutorMenu(filter.toString());
            switch (choice) {
                case 1:
                    filter.push("FT");
                    break;
```

```java
                case 2:
                    filter.push("PT");
                    break;
                case 3:
                    filter.push("RS");
                    break;
                case 4:
                    filter.push("RT");
                    break;
                case 5:
                    filter.pop();
                    break;
                case 6:
                    filterTutor(filter);
                    break;
            }

        } while (choice != 0);
    }

    private void filterTutor(StackInterface<String> selectedFilter) {
        adt.exampleAdt.ListInterface<String> filter = new ArrayList<>();
        int number = 0;
        String output = "";

        while (!selectedFilter.isEmpty()) {
            filter.add(selectedFilter.pop());
        }

        for (int i = filter.getNumberOfEntries(); i > 0; i--) {
            Iterator<Tutor> it = tutorList.getIterator();
            while (it.hasNext()) {
                Tutor matchTutor = it.next();
                if (matchTutor.getStatus().equals(filter.getEntry(i))) {
                    output += String.format("%2d.  ", ++number)
                            + matchTutor + "\n";
                }
            }
        }

        tutorUI.displayFilteredTutor(output);

    }

    private void generateReportUI() {
        int choice;
        int year, month;
        do {
            choice = tutorUI.generateTutorReportMenu();

            switch (choice) {
                case 1:
                    generateSalaryReport();
                    break;
                case 2:
                    generateRecruitmentReport();
                    break;
            }
        } while (choice != 0);
    }

    private void generateSalaryReport() {
        int year = tutorUI.getReportYear();
        int month = tutorUI.getReportMonth();
        String reportHeader = String.format("%s %d/%02d",
                "Monthly Tutor Salary Report", year, month);
        double totalSalary = 0.0;

        Iterator<Tutor> it = tutorList.getIterator();
        YearMonth yearMonth = YearMonth.of(year, month);
        LocalDateTime selectedDate = yearMonth
                .atEndOfMonth().atTime(23, 59, 59);

        ListInterface<Tutor> validTutor = new
CircularDoublyLinkedList<>();

        //get tutor in date range and calculate salary
        while (it.hasNext()) {
            Tutor next = it.next();
            if (isValidTutor(next, selectedDate)) {
                validTutor.add(next);
                totalSalary += next.getSalary();
```

```java
            }
        }

        if (isEmptyTutorReport(validTutor)) {
            return;
        }
        sortTutorList(validTutor, reportHeader, totalSalary);

    }

    private boolean isEmptyTutorReport(ListInterface<Tutor> validTutor) {
        if (validTutor.isEmpty()) {
            System.err.println("No Tutor In Selected Date.");
            GeneralUtil.systemPause();
            return true;
        }
        return false;
    }

    private void generateRecruitmentReport() {
        int year = tutorUI.getReportYear();
        int month = tutorUI.getReportMonth();
        String reportHeader = String.format("%s %d/%02d",
                "Monthly Tutor Recruitment Report", year, month);

        Iterator<Tutor> it = tutorList.getIterator();

        ListInterface<Tutor> validTutor = new 
CircularDoublyLinkedList<>();

        //get tutor in selected date
        while (it.hasNext()) {
            Tutor next = it.next();
            if (isValidTutor(next, year, month)) {
                validTutor.add(next);
            }
        }

        if (isEmptyTutorReport(validTutor)) {
            return;
        }

        sortTutorList(validTutor, reportHeader, -1);
    }

    private void sortTutorList(ListInterface<Tutor> validTutor, String reportHeader, double totalSalary) {
        int pageSize;
        int choice;
        do {
            choice = tutorUI.sortSelection();
            switch (choice) {
                case 1:
                    validTutor.sortBy(Comparator.comparing(Tutor::getTutorName),
                            true);
                    break;
                case 2:
                    validTutor.sortBy(Comparator.comparing(Tutor::getTutorName),
                            false);
                    break;
                case 3:
                    validTutor.sortBy(Comparator.comparing(Tutor::getTutorId),
                            true);
                    break;
                case 4:
                    validTutor.sortBy(Comparator.comparing(Tutor::getSalary),
                            true);
                    break;
                case 5:
                    validTutor.sortBy(Comparator.comparing(Tutor::getSalary),
                            false);
                    break;
            }

            if (choice != 0) {
                pageSize = tutorUI
                        .getPageSize(validTutor.getNumberOfEntries());
                reportPreview(validTutor, reportHeader, pageSize, totalSalary);
            }

        } while (choice != 0);
    }

    private boolean isValidTutor(Tutor tutor, LocalDateTime selectedDate) {
        return !tutor.getCreated_at().isAfter(selectedDate) && (tutor.isWorking());
    }

    private boolean isValidTutor(Tutor tutor, int year, int month) {
        return year == tutor.getCreated_at().getYear()
                && month == tutor.getCreated_at().getMonthValue()
                && (tutor.isWorking());
    }

    private void reportPreview(ListInterface<Tutor> report, String reportHeader,
            int pageSize, double totalSalary) {
        String choice;
        Paginator page = new Paginator(report, pageSize);
        String currentPage = getPageContent(page.jumpTo(0), report);
        do {

            if (currentPage == "") {
                currentPage = getPageContent(page.jumpTo(page.currentPage),
                        report);
            }

            tutorUI.displayAllTutor(reportHeader,
                    currentPage, false, 90);

            System.out.printf("Page No: %-130d < 1 .. %d >\n",
                    page.currentPage + 1, page.pageNumber);
            if (totalSalary > 0) {
                System.out.printf("Total Salary To Pay: RM %,.2f\n\n", totalSalary);
            }

            if (page.isEndOfPage()) {
                MessageUI.displayInfoMessage(String.format("%88s", "END OF PAGES"));
            }

            choice = tutorUI.pageController().toLowerCase();
            switch (choice) {
                case ">":
                    currentPage = getPageContent(page.nextPage(), report);
                    break;
                case ">|":
                    currentPage = getPageContent(page.toEnd(), report);
                    break;
                case "<":
                    currentPage = getPageContent(page.prevPage(), report);
                    break;
                case "|<":
                    currentPage = getPageContent(page.toStart(), report);
                    break;
                default:
                    if (choice.matches("[0-9]+")) { // is integer
                        currentPage = getPageContent(page.jumpTo(Integer.parseInt(choice) - 1), report);
                    } else if (!choice.equals("exit")) {
                        System.err.println("Invalid command.");
                        GeneralUtil.systemPause();
                    }
                    break;
            }

        } while (!choice.equals("exit"));
    }

    private String getPageContent(ListInterface<Tutor> list, ListInterface<Tutor> original) {
        String outputStr = "";
        try {
            int number = original.indexOf(list.getFirstEntry());
            Iterator<Tutor> it = list.getIterator();
            while (it.hasNext()) {
                outputStr += String.format("%2d.  ", ++number)
                        + it.next() + "\n";

            }
        } catch (Exception e) {
            GeneralUtil.systemPause();
        }

        return outputStr;
    }

}
```