```
/*
 * Course class
 *
 * This model represents a course in an event
 *
 */
package uk.ac.aber.awf1;

import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.LinkedList;
import java.util.Scanner;

/**
 *
 * @author fewstera
 */
class Course {

    private char id;
    private LinkedList<Node> nodes = new LinkedList<Node>();
    private UserMenu userMenu;

    Course(UserMenu userMenu, char id){
        this.userMenu = userMenu;
        this.id = id;
    }

    public char getId(){
        return this.id;
    }

    public LinkedList<Node> getNodes() {
        return nodes;
    }

    public void listOptions() {
        System.out.print("\n\n\n\n\nModify Course " + getId() + "\n"
                + "============================================\n"
                + "What would you like to do?\n"
                + "1. Add a new node\n"
                + "2. Remove a node\n"
                + "3. Return to modify event menu\n\n"
                + "Enter your selection: ");
        int selection;

        try { //Attempt to receive keyboard input from user
            Scanner in = new Scanner(System.in);
            selection = in.nextInt();
        } catch (InputMismatchException IO) {
            selection = 0; //Invalid user input so set selection to 0
        }

        switch (selection) {
            case 1:
                addNode();
                listOptions();
                break;
            case 2:
                removeNode();
                listOptions();
                break;
            case 3:
                break;
            default:
                System.out.println("\n\n\nERROR: Unexpected input, please enter"
                        + " only the number of your selection. "
                        + "Please try again");
                listOptions(); //Unexpected output, try again
```

```
            break;

    }
}

private void addNode() {
    if(nodes.size()==0){
        System.out.println("\n\nSelect Starting Node\n"
            + "=========================================\n"
            + "Select the start of the course\n");
        int count = 0;
        for (Node node : userMenu.getNodes()) {
            count++;
            System.out.println(count + ". " + node.getId());
        }
        System.out.println(++count + ". Cancel");
        System.out.print("\nEnter the value of your choice: ");
        int selection;
        try { //Attempt to receive keyboard input from user
            Scanner in = new Scanner(System.in);
            selection = in.nextInt();
        } catch (InputMismatchException IO) {
            System.out.print("\n\nInvalid input. Returning to modify "
                    + "course page");
            selection = count; //Invalid user input so set selection to count
        }
        if((selection>0)&&(selection<count)){
            nodes.add(userMenu.getNodes().get(selection-1));
            System.out.print("\n\nAdded new node!");
        }
    }else{
        System.out.println("\n\nAdd an extra node\n"
            + "=========================================\n"
            + "Below is a list of nodes linked to the "
                + "last node (" + nodes.getLast().getId() + "). \n"
                + "Select which node you would like to add to the list.\n");

        int count = 0;
        ArrayList<Node> availableNodes = availableNodes();
        for (Node node : availableNodes) {
            count++;
            System.out.println(count + ". " + node.getId());
        }
        System.out.println(++count + ". Cancel");
        System.out.print("\nEnter the value of your choice: ");
        int selection;
        try { //Attempt to receive keyboard input from user
            Scanner in = new Scanner(System.in);
            selection = in.nextInt();
        } catch (InputMismatchException IO) {
            System.out.print("\n\nInvalid input. Returning to modify "
                    + "course page");
            selection = count; //Invalid user input so set selection to count
        }
        if((selection>0)&&(selection<count)){
            nodes.add(availableNodes.get(selection-1));
            System.out.print("\n\nAdded new node!");
        }
    }
}

private void removeNode() {
    if(nodes.size()==0){
        System.out.print("\n\nThere isn't any nodes to remove");
    }else{
        System.out.print("\n\nRemove a node\n"
            + "=========================================\n"
            + "Please select from the following options\n"
            + "1. Remove first node\n"
```

```
                + "2. Remove last node\n"
                + "3. Cancel");
            int selection;
            try { //Attempt to receive keyboard input from user
                Scanner in = new Scanner(System.in);
                selection = in.nextInt();
            } catch (InputMismatchException IO) {
                System.out.print("\n\nInvalid input. Returning to modify "
                        + "course page");
                selection = 3; //Invalid user input so set selection to count
            }
            if(selection==1){
                nodes.removeFirst();
            }else if(selection==2){
                nodes.removeLast();
            }
        }
    }

    private ArrayList<Node> availableNodes(){
        ArrayList<Node> returnList = new ArrayList<Node>();
        Node lastNode = nodes.getLast();
        for(Track track : userMenu.getTracks()){
            if(track.getFrom().getId()==lastNode.getId()){
                returnList.add(track.getTo());
            }else if(track.getTo().getId()==lastNode.getId()){
                returnList.add(track.getFrom());
            }
        }
        return returnList;
    }

    public String toFormatedString(){
        String returnString = getId() + " " + nodes.size();
        for(Node node: nodes){
            returnString =  returnString + " " + node.getId();
        }
        return returnString;
    }
}
```

```
/*
 * Entrant class
 *
 * This model represents an entrant on the course
 *
 */
package uk.ac.aber.awf1;

/**
 *
 * @author fewstera
 */
class Entrant {
   private Course course;
   private String name;

   Entrant(Course course, String name){
       this.course = course;
       this.name = name;
   }

    public Course getCourse() {
        return course;
    }

    public String getName() {
        return name;
    }

    public String toFormatedString(){
        return getCourse().getId() + " " + getName();
    }
}
```

```
/*
 * Event class
 *
 * Responsible for handling all actions on an event
 *
 */
package uk.ac.aber.awf1;

import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.LinkedList;
import java.util.Scanner;

/**
 *
 * @author fewstera
 */
class Event {

    private String name, date, timeString;
    private LinkedList<Course> courses = new LinkedList<Course>();
    private LinkedList<Entrant> entrants = new LinkedList<Entrant>();
    private UserMenu userMenu;

    Event(UserMenu userMenu, String eventName,
            String eventDate, String eventTimeString){
        this.userMenu = userMenu;
        this.name = eventName;
        this.date = eventDate;
        this.timeString = eventTimeString;
    }

    public String getName() {
        return name;
    }

    public String getDate() {
        return date;
    }

    public String getTimeString() {
        return timeString;
    }

    public void listOptions() {
        System.out.print("\n\n\n\n\nModify " + getName() + "\n"
                + "==========================================\n"
                + "What would you like to do?\n"
                + "1. Add a new course\n"
                + "2. Add or remove nodes to or from a course\n"
                + "3. Remove a course\n"
                + "4. Add an entrant\n"
                + "5. Remove an entrant\n"
                + "6. Return to main menu\n\n"
                + "Enter your selection: ");
        int selection;

        try { //Attempt to receive keyboard input from user
            Scanner in = new Scanner(System.in);
            selection = in.nextInt();
        } catch (InputMismatchException IO) {
            selection = 0; //Invalid user input so set selection to 0
        }

        //Switch case to determine the users
        //selection, and call the appropiate method.
        switch (selection) {
            case 1:
                addNewCourse();
```

```
            listOptions();
            break;
        case 2:
            modifyCourse();
            listOptions();
            break;
        case 3:
            removeCourse();
            listOptions();
            break;
        case 4:
            addEntrant();
            listOptions();
            break;
        case 5:
            removeEntrant();
            listOptions();
            break;
        case 6:
            break;
        default:
            System.out.println("\n\n\nERROR: Unexpected input, please enter"
                    + " only the number of your selection. Please try again");
            listOptions(); //Unexpected output, try again
            break;

    }
}

private void addNewCourse() {
    System.out.print("\n\nAdd new course\n"
            + "==========================================\n"
            + "Note: Prefixs must be a single character\n\n"
            + "Please enter a prefix for your course: ");
    Scanner in = new Scanner(System.in);
    char letter = in.next().toUpperCase().charAt(0);
    if(Character.isLetter(letter)){
        if(getCourse(letter)==null){
            courses.add(new Course(userMenu, letter));
        }else{
            System.out.print("\n\nA course already exists with "
                    + ""
                    + "this prefix. try again");
            addNewCourse();
        }
    }else{
         System.out.print("\n\nInvalid prefix, try again.");
         addNewCourse();
    }

}

private Course getCourse(char letter) {
    for(Course course : courses){
        if(course.getId()==letter){
            return course;
        }
    }
    return null;
}

private void removeCourse() {
    System.out.print("\n\nRemove a course\n"
            + "==========================================\n"
            + "Which course would you like to remove?\n");
    int count = 0;
    for (Course course : courses) {
        count++;
        System.out.println(count + ". " + course.getId());
```

```
        }
        System.out.println(++count + ". Cancel");
        System.out.print("\nEnter the value of your choice: ");
        int selection;
        try { //Attempt to receive keyboard input from user
            Scanner in = new Scanner(System.in);
            selection = in.nextInt();
        } catch (InputMismatchException IO) {
            System.out.print("\n\nInvalid input. Returning to modify page.");
            selection = count; //Invalid user input so set selection to count
        }
        if((selection>0)&&(selection<count)){
            courses.remove(selection-1);
            System.out.print("\n\nCourse removed.");
        }
    }

    private void addEntrant() {
        if(courses.size()==0){
            System.out.print("\n\nYou must have atleast one course "
                    + "before you can add entrants");
        }else{
            String entrantName;
            Course entrantCourse = null;
            System.out.print("\n\nAdd a new entrant to " + this.getName() + "\n"
                    + "==========================================\n"
                    + "Enter the entrants name: ");

            Scanner in = new Scanner(System.in);
            entrantName = in.nextLine();

            boolean courseSelected = false;
            while(!courseSelected){
                System.out.println("Which course is " + entrantName + " on?");
                int count = 0;
                for (Course course : courses) {
                    count++;
                    System.out.println(count + ". " + course.getId());
                }
                System.out.print("\nEnter the value of your choice: ");
                int selection = 0;
                try { //Attempt to receive keyboard input from user
                    selection = in.nextInt();
                    if((selection>0)&&(selection<=count)){
                        entrantCourse = courses.get(selection-1);
                        courseSelected = true;
                    }

                } catch (InputMismatchException IO) {
                    System.out.print("\n\nInvalid input. Try again");
                }
            }
            entrants.add(new Entrant(entrantCourse, entrantName));
            System.out.print("\n\nAdded " + entrantName + "to " + this.getName());
        }
    }

    private void removeEntrant() {
        System.out.print("\n\nRemove an entrant\n"
                + "==========================================\n"
                + "Which entrant would you like to remove?\n");
        int count = 0;
        for (Entrant entrant : entrants) {
            count++;
            System.out.println(count + ". " + entrant.getName());
        }
        System.out.println(++count + ". Cancel");
        System.out.print("\nEnter the value of your choice: ");
        int selection;
```

```
        try { //Attempt to receive keyboard input from user
            Scanner in = new Scanner(System.in);
            selection = in.nextInt();
        } catch (InputMismatchException IO) {
            System.out.print("\n\nInvalid input. Returning to modify event page.");
            selection = count; //Invalid user input so set selection to count
        }
        if((selection>0)&&(selection<count)){
            entrants.remove(selection-1);
            System.out.print("\n\nEntrant removed.");
        }
    }

    private void modifyCourse() {
        System.out.println("\n\nModify a course\n"
                + "==========================================\n"
                + "Which course would you like to modify?");
        int count = 0;
        for (Course course : courses) {
            count++;
            System.out.println(count + ". " + course.getId());
        }
        System.out.println(++count + ". Cancel");
        System.out.print("\nEnter the value of your choice: ");
        int selection;
        try { //Attempt to receive keyboard input from user
            Scanner in = new Scanner(System.in);
            selection = in.nextInt();
        } catch (InputMismatchException IO) {
            System.out.print("\n\nInvalid input. Returning to modify event page");
            selection = count; //Invalid user input so set selection to count
        }
        if((selection>0)&&(selection<count)){
            courses.get(selection-1).listOptions();
        }
    }

    //Gnerate the text for an event file
    public String generateEventFile(){
        return getName() + "\n" + getDate() + "\n" + getTimeString();
    }

    //Generates the text for a course file
    public String generateCoursesFile(){
        String returnString = "";
        for(Course course: courses){
            returnString = returnString + course.toFormatedString() + "\n";
        }
        return returnString;
    }

    //Generates the text for an entrants file
    public String generateEntrantsFile(){
        String returnString = "";
        int count = 1;
        for(Entrant entrant: entrants){
            returnString = returnString + count + " "
                    + "" + entrant.toFormatedString() + "\n";
            count++;
        }
        return returnString;
    }

}
```

```
/*
 * Main class
 */
package uk.ac.aber.awf1;

/**
 *
 * @author fewstera
 */
public class EventCreator {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        UserMenu userMenu = new UserMenu();
    }
}
```

```
/*
 * Node Class
 *
 * This class represnts a node on the course
 *
 */
package uk.ac.aber.awf1;

/**
 *
 * @author fewstera
 */
class Node {

    private int id;
    private String type;

    public Node(int id, String type){
        this.id = id;
        this.type = type;
    }

    public int getId() {
        return id;
    }

    public String getType() {
        return type;
    }

}
```

```
/*
 * Track class
 *
 * This model represents a Track on a course
 *
 */
package uk.ac.aber.awf1;

/**
 *
 * @author fewstera
 */
class Track {
    private Node to, from;

    Track(Node from, Node to){
        this.from = from;
        this.to = to;
    }

    public Node getTo() {
        return to;
    }

    public Node getFrom() {
        return from;
    }


}
```

```
/*
 * User Menu Class
 *
 * This class is responsible for handling the users navigation through the system
 *
 */
package uk.ac.aber.awf1;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.LinkedList;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 *
 * @author fewstera
 */
class UserMenu {
    private ArrayList<Track> tracks = new ArrayList<Track>();
    private ArrayList<Node> nodes = new ArrayList<Node>();
    private LinkedList<Event> events = new LinkedList<Event>();


    public UserMenu(){
        System.out.println("Welcome to the event creation system.");
        loadNodesFile();
        loadTracksFile();
        topLevelMenu();
    }

    private void loadNodesFile(){
        System.out.print("\n\nPlease enter the location of the nodes file: ");
        Scanner in = new Scanner(System.in);
        String nodesFile = in.nextLine();
        try{
            BufferedReader br = new BufferedReader(new FileReader(nodesFile));
            String line;
            while((line = br.readLine()) != null) {
                String[] stringParts = line.split(" ");
                nodes.add(new
                        Node(Integer.parseInt(stringParts[0]), stringParts[1]));
            }
            System.out.println("\n\nLoaded Successfully!");
        } catch (FileNotFoundException ex) {
            System.out.print("\n\nThe file you entered was not found "
                    + "please try again.");
            loadNodesFile();
        } catch (IOException ex) {
            System.out.println("\n\nError reading file exiting.");
            System.exit(0);
        }

    }

    private void loadTracksFile(){
        System.out.print("\n\nPlease enter the location of the tracks file: ");
        Scanner in = new Scanner(System.in);
        String tracksFile = in.nextLine();
        try{
            BufferedReader br = new BufferedReader(new FileReader(tracksFile));
```

```
            String line;
            while((line = br.readLine()) != null) {
                String[] stringParts = line.split(" ");
                Node to, from;
                to = getNode(Integer.parseInt(stringParts[1]));
                from = getNode(Integer.parseInt(stringParts[2]));
                if((to==null)||(from==null)){
                    System.out.println("\n\nError parsing tracks file, exiting.");
                    System.exit(0);
                }
                tracks.add(new Track(to, from));
            }
            System.out.println("\n\nLoaded Successfully!");
        } catch (FileNotFoundException ex) {
            System.out.print("\n\nThe file you entered was not found "
                    + "please try again.");
            loadTracksFile();
        } catch (IOException ex) {
            System.out.println("\n\nError reading file exiting.");
            System.exit(0);
        }

    }

    private Node getNode(int id) {
        for(Node node : nodes){
            if(node.getId()==id){
                return node;
            }
        }
        return null;
    }

    private void topLevelMenu() {
        System.out.println("\n\nMain Menu\n"
                + "============================================\n"
                + "Please choose from the following options\n"
                + "1. Create a new event\n"
                + "2. Add entrants and courses to an event\n"
                + "3. Remove an event\n"
                + "4. Generate files for events\n"
                + "5. Exit"
                + "");

        int selection;

        try { //Attempt to receive keyboard input from user
            Scanner in = new Scanner(System.in);
            selection = in.nextInt();
        } catch (InputMismatchException IO) {
            selection = 0; //Invalid user input so set selection to 0
        }

        //Switch case to determine the users selection, and call the appropiate
        //method.
        switch (selection) {
            case 1:
                createNewEvent();
                break;
            case 2:
                selectEventToModify();
                topLevelMenu();
                break;
            case 3:
                removeEvent();
                topLevelMenu();
                break;
            case 4:
                generateFiles();
```

```
            topLevelMenu();
            break;
        case 5:
            break;
        default:
            System.out.println("\n\nERROR: Unexpected input, please enter "
                    + "only the number of your selection. Please try again");
            topLevelMenu(); //Unexpected output, try again
            break;

    }

}

private void createNewEvent() {
    String eventName, eventDate, timeString = "";

    System.out.print("\n\nCreate new event\n"
            + "==========================================\n"
            + "Enter the event name: ");

    Scanner in = new Scanner(System.in);
    eventName = in.nextLine();

    System.out.print("Enter the event date:  ");
    eventDate = in.nextLine();

    boolean correctTime = false;
    while(!correctTime){
        System.out.print("Enter the time of the event (HH:MM):  ");
        timeString = in.next();
        Pattern pattern = Pattern.compile("([01][0-9]|2[0-3]):[0-5][0-9]");
        Matcher matcher = pattern.matcher(timeString);
        if(matcher.matches()){
            correctTime = true;
        }else{
            System.out.print("\n\nError incorrect time, try again\n\n");
        }
    }
    events.add(new Event(this, eventName, eventDate, timeString));
    System.out.print("\n\nEvent created successfully!");
    topLevelMenu();

}

private void removeEvent() {
    System.out.println("\n\nRemove an event\n"
            + "==========================================\n"
            + "Which event would you like to delete?");
    int count = 0;
    for (Event event : events) {
        count++;
        System.out.println(count + ". " + event.getName());
    }
    System.out.println(++count + ". Cancel");
    System.out.print("\nEnter the value of your choice: ");
    int selection;
    try { //Attempt to receive keyboard input from user
        Scanner in = new Scanner(System.in);
        selection = in.nextInt();
    } catch (InputMismatchException IO) {
        System.out.print("\n\nInvalid input. Returning to main menu");
        selection = count; //Invalid user input so set selection to count
    }
    if((selection>0)&&(selection<count)){
        events.remove(selection-1);
        System.out.print("\n\nEvent removed.");
    }
}
```

```
private void selectEventToModify() {
    System.out.println("\n\nSelect an event\n"
            + "=========================================\n"
            + "Which event would you like to modify?");
    int count = 0;
    for (Event event : events) {
        count++;
        System.out.println(count + ". " + event.getName());
    }
    System.out.println(++count + ". Cancel");
    System.out.print("\nEnter the value of your choice: ");
    int selection;
    try { //Attempt to receive keyboard input from user
        Scanner in = new Scanner(System.in);
        selection = in.nextInt();
    } catch (InputMismatchException IO) {
        System.out.print("\n\nInvalid input. Returning to main menu");
        selection = count; //Invalid user input so set selection to count
    }
    if((selection>0)&&(selection<count)){
        events.get(selection-1).listOptions();
    }
}

public ArrayList<Track> getTracks() {
    return tracks;
}

public ArrayList<Node> getNodes() {
    return nodes;
}

private void generateFiles() {
    System.out.println("\n\nGenerate Event Files\n"
            + "=========================================\n"
            + "Directory to save files to: ");
    Scanner in = new Scanner(System.in);

    String directoryPath = in.nextLine();
    File file = new File(directoryPath);
    //Check the entered text is a directory
    if (file.isDirectory()) {
        for(Event event:events){
            String eventFolderPath =
                    directoryPath + File.separator + event.getName();
            (new File(eventFolderPath)).mkdirs();
            //Make new DIRs for each event

            File eventFile = new
                    File(eventFolderPath + File.separator + "event.txt");
            File coursesFile = new
                    File(eventFolderPath + File.separator + "courses.txt");
            File entrantsFile = new
                    File(eventFolderPath + File.separator + "entrants.txt");

            try {
                //Write to all the files
                DataOutputStream eventFileOuts =
                        new DataOutputStream(new
                        FileOutputStream(eventFile, false));
                DataOutputStream coursesFileOuts =
                        new DataOutputStream(new
                        FileOutputStream(coursesFile, false));
                DataOutputStream entrantsFileOuts =
                        new DataOutputStream(new
                        FileOutputStream(entrantsFile, false));
                eventFileOuts.write(event.generateEventFile().getBytes());
                coursesFileOuts.write(event.generateCoursesFile().getBytes());
```

```
                    entrantsFileOuts.write(event.generateEntrantsFile().getBytes());

                } catch (IOException ex) {
                    System.out.println("Unable to create event files. Quiting");
                    System.exit(0);
                }

            }
            System.out.println("\n\nSaved files successfully!");
        } else {
            System.out.println("\n\nThis is not a directory, please try again");
            generateFiles();
        }
    }


}
```