HOMEWORK 2 - SPRING 2025

HOMEWORK 2 - due Tuesday, February 25th no later than 7:00PM

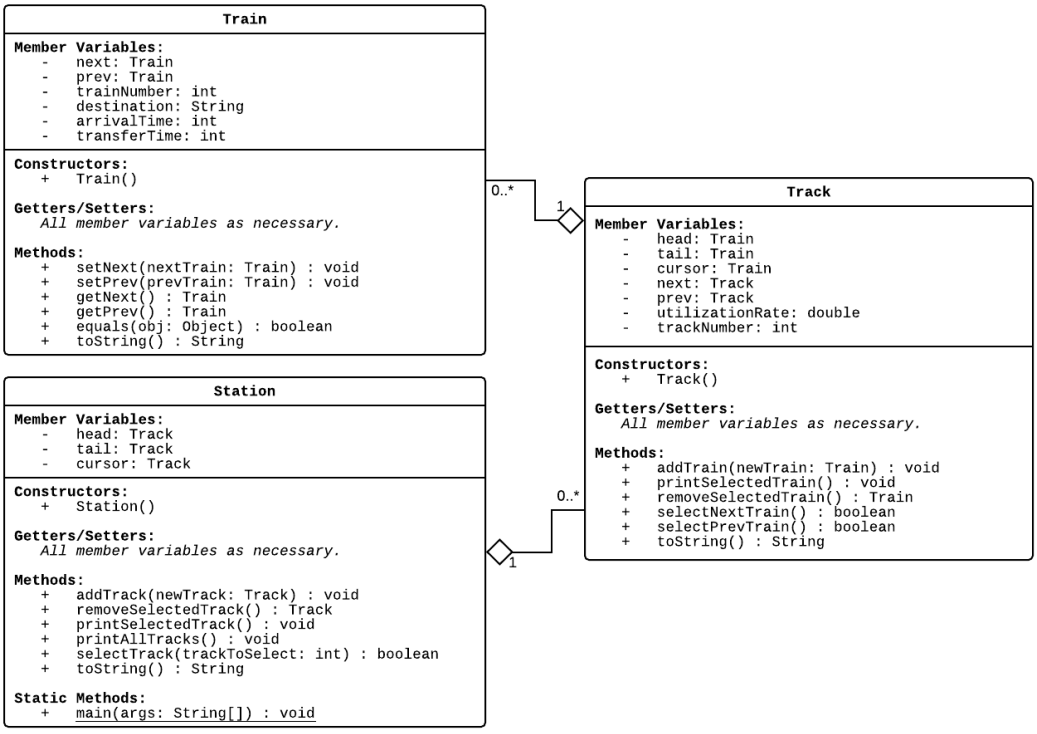
REMINDERS:

- **Submit your files only via CodeGrade on the Content page in Brightspace. Access CodeGrade by clicking "HW2 - CODEGRADE SUBMISSION LINK" for your submission.**
- **Use of a package is optional. If you wish to use it, make sure to name it "hw2" (all in lowercase). Otherwise, you will lose points.**
- **Be sure your code follows the coding style for CSE214.**
- **Make sure you read the warnings about academic dishonesty. *Remember, all work you submit for homework or exams MUST be your own work.***
- **You are not allowed to use ArrayList, LinkedList, Vector or any other Java API Data Structure classes to implement this assignment.**
- **You may use Scanner, InputStreamReader, or any other class that you wish for keyboard input.**

Scheduling trains that use the same railway is a tough, yet important, problem to solve when it comes to transportation. One train being even several minutes off can lead to a butterfly effect of congestion, delays, and many frustrated passengers. It can also be hazardous if two trains are mistakenly scheduled to arrive on the same track at the same time. Assuming that there are no delays and all trains arrive and depart on time, your job is to implement an interface to interact with all trains in the same station in the form of a linked list. A Station will contain a list of Tracks, and each Track will contain a list of Trains. Linked lists are efficient in these types of situations because the list can be easily modified without changing the whole structure of the existing list.

UML

The UML Diagram for all the classes specified below is as follows:



SPECIFICATIONS

1. Train

Write a fully-documented class named Train that contains basic information for a train approaching a station. Required information includes the train number, the train's destination, the arrival time of the train to its track (in 24-hour format between 0000 - 2359), and the transfer time for how long the train waits at the station (in minutes). You may add additional fields as needed. The following is a partial specification and it is up to you to fill in the details:

- `public Train()`
  - **Constructor for Train (you may include an additional constructor with parameters)**
- `public void setNext(Train nextTrain)`
  - **Sets the next Train in the list.**
  - ***NOTE: The next Train should point to the train scheduled to arrive after this one. There can be a gap in time between the departure of this train and the arrival of the next one, but there shouldn't be another train in between this train and its pointer to the next train.***
- `public void setPrev(Train prevTrain)`
  - **Sets the previous Train in the list.**
  - ***NOTE: Similar to the last note, the previous train should be the train arriving before this train. There shouldn't be a train between this train and its pointer to the previous train.***
- `public Train getNext()`
  - **Returns the next Train in the list. If there is no next Train, returns `null`.**
- `public Train getPrev()`
  - **Returns the previous Train in the list. If there is no previous Train, returns `null`.**
- `public boolean equals(Object o)`
  - **Check if this Train object is equal to the supplied parameter.**
  - ***NOTE: Two Train objects are considered equal if and only if their Train numbers are equal.***
- `public String toString()`
  Returns a textual representation of all the information for this Train object. See sample I/O.
- **Getters/Setters for all fields as necessary.**

2. Track

Write a fully-documented class named Track that contains references to the head and tail of a list of Train nodes, as well as a cursor representing the selected Train node. You also want to keep in mind the Track utilization rate for what percentage of the day the track is being used (total time of trains waiting at the track / total minutes in a day). Each track also contains a track number, which must be unique among all Track objects (there can only be one 'Track 1'). The `Train` objects stored in your Track must be sorted according to the arrival times of the trains. Your class will follow this specification, but you have to fill in the details:

- `public Track()`
  - **Constructor for Track (you may include an additional constructor with parameters).**
- `public void addTrain(Train newTrain)`
  - **Adds a new Train to the specified Track list. After insertion, the currently selected train for this Track object should be updated to be the newly inserted Train.**
  - ***NOTE: Since we are using a 24-hour time format, the last possible time a train can depart is at 2359 hours and the earliest a train can arrive is at 0000 hours. The Track list is ordered by time, where the head node should be the train to arrive at the station first, and the tail node should be the train to depart from the station last. The user inputs the arrival time in proper 24-hour time format (e.g. do not accept 0070, but do accept 0110) and transfer time in minutes (which is used to calculate the departure time which is also in valid 24-hour time format).***
  - ***NOTE: You do NOT need to handle the case where a Train arrives one day and leaves the next. e.g., a Train can not arrive at 2345 and depart at 0015.***
  - ***NOTE: Make sure that two trains can't be scheduled for the same track at the same time. If the user tries to input an arrival and departure time that conflicts with an already scheduled train, alert the user that the train could not be scheduled.***
  - ***NOTE: Make sure that a Train can not be scheduled if it has the same train number as an already scheduled Train in the same track.***
  - ***HINT: There are 1440 minutes in a day.***
- `public void printSelectedTrain()`
  - **Prints the data of the selected Train.**
- `public Train removeSelectedTrain()`
  - **Removes the selected Train from the track and returns a reference to it. If there is no Train selected, return `null`.**
  - **The selected Train should now be the Train node after the one which was just removed. If there is no Train after the one which was just removed, the selected Train should now be the node before the one which was just removed. If the Train removed was the only one in the Track list, then the selected Train should now be `null`.**
- `public boolean selectNextTrain()`
  - **Moves the reference of the selected Train node forwards to the next Train if it exists and returns `true`. If there is no next Train, then the selected Train should remain the same and return `false`. If there is no selected Train, you should throw an appropriate exception.**
- `public boolean selectPrevTrain()`
  - **Moves the reference of the selected Train node backwards to the previous Train if it exists and returns `true`.**

If there is no previous Train, then the selected Train should remain the same and return `false`. If there is no selected Train, you should throw an appropriate exception.

- `public String toString()`
  - Returns a neatly formatted list of all the trains scheduled on this Track. See sample I/O.
- Getters/Setters for all fields as necessary.

3. Station

Write a fully-documented class named Station. This class will contain a list of `Track` objects (which are themselves lists of `Train` objects) that the user can switch between in the menu. It also contains the main method that allows the user to interact with the `Trains` in the `Tracks` using the commands specified below:

- `public Station()`
  - Constructor for Station (you may include an additional constructor with parameters).
- `public void addTrack(Track newTrack)`
  - Adds newTrack to this Station. The new Track object must be inserted into the list such that the list is sorted by track numbers. If there is already a Track object in this Station with the same track number as `newTrack`, do not insert the new Track and throw an appropriate exception. Note that after inserting a Track, the currently selected Track is set to the newly inserted Track.
- `public Track removeSelectedTrack()`
  - Removes the selected `Track` object from this Station and returns it. If there is no Track selected, returns `null`.
  - The selected Track should now be the Track node directly after the one which was removed. If there is no Track node after the one which was removed, then selected Track should now be the node directly before the one which was removed. If the Track which was removed was the only one in Station list, then the selected Track should now be `null`.
- `public void printSelectedTrack()`
  - Prints the selected Track list. See Sample I/O.
- `public void printAllTracks()`
  - Prints all Track lists in this Station.
- `public boolean selectTrack(int trackToSelect)`
  - Moves the reference of the selected Track node to the node which has the same `trackNumber` as the given `trackToSelect` parameter if it exists and returns `true`. If there is no Track with a trackNumber which matches `trackToSelect`, then the selected Track object should remain the same and return `false`.
- `public String toString()`
  - Returns a neatly formatted representation of this Station object. See Sample I/O.
- Getters/setters for all fields as necessary.
- `public static void main(String[] args)`
  - The main method that allows the user to interact with Trains and Tracks at the Station.
  - Menu:
    - `A` - Add new Train <Train Number> <Destination> <Arrival Time> <Transfer Time>
    - `N` - Select next Train
    - `V` - Select previous Train
    - `R` - Remove selected Train
    - `P` - Print selected Train
    - `TPS` - Print selected Track
    - `TPA` - Print all Tracks
    - `TS` - Switch Track <Track Number>
    - `TA` - Add Track <Track Number>
    - `TR` - Remove selected Track
    - `SI` - Print Station information (NOTE: The time complexity of this selection must be proportional to the number of Tracks in the station, and be irrelevant to the number of Trains amongst the Tracks.)
    - `Q` - Quit

4. You will need classes to handle the exceptions thrown (see class specifications above for exception classes you need).

NOTE: You may include additional methods, variables, or classes as necessary or as you find convenient.

INPUT FORMAT

- Each menu operation is entered on its own line and should be case insensitive (i.e. 'q' and 'Q' are the same).
- Check to make sure that the position, if required, is valid. If not, print an error message and return to the main menu.
- For the Add commands, if the input information is valid, construct the object accordingly. Otherwise, print an error message and return to the main menu.
- You may assume Strings are at most 25 characters long.

OUTPUT FORMAT

- Each command should output the result (as shown in the sample IO below) after each operation is performed.

- **All menu operations must be accompanied by a message indicating what operation was performed and whether or not it was successful.**

SAMPLE INPUT/OUTPUT

```
// Comments in green, input in red, output in black.

|-------------------------------------------------------------------------|
| Train Options                       | Track Options                     |
|    A. Add new Train                 |    TA. Add Track                  |
|    N. Select next Train             |    TR. Remove selected Track      |
|    V. Select previous Train         |    TS. Switch Track               |
|    R. Remove selected Train         |   TPS. Print selected Track       |
|    P. Print selected Train          |   TPA. Print all Tracks           |
|-------------------------------------------------------------------------|
| Station Options                                                         |
|   SI. Print Station Information                                         |
|    Q. Quit                                                              |
|-------------------------------------------------------------------------|

Choose an operation: A

Enter train number: 12340
Enter train destination: Port Jeff
Enter train arrival time: 0000
Enter train transfer time: 15

Train not added: There is no Track to add the Train to!

// Menu not shown in sample i/o

Choose an operation: TA

Enter track number: 1

Track 1 added to the Station.

// Menu not shown in sample i/o

Choose an operation: TA

Enter track number: 1

Track not added: Track 1 already exists.

// Menu not shown in sample i/o

Choose an operation: A

Enter train number: 12340
Enter train destination: Port Jeff
Enter train arrival time: 0000
Enter train transfer time: 15

Train No. 12340 to Port Jeff added to Track 1.

// Menu not shown in sample i/o

Choose an operation: A

Enter train number: 233
Enter train destination: Penn Station
Enter train arrival time: 0070
Enter train transfer time: 10

Train not added: Invalid arrival time.

// Menu not shown in sample i/o

Choose an operation: A

Enter train number: 233
Enter train destination: Penn Station
Enter train arrival time: 0020
Enter train transfer time: 10

Train No. 233 to Penn Station added to Track 1.

// Menu not shown in sample i/o

Choose an operation: TPS
```

```
Track 1 (1.74% Utilization Rate):
Selected  Train Number     Train Destination         Arrival Time     Departure Time
--------------------------------------------------------------------------------
             12340          Port Jeff                   0000             0015
    *          233          Penn Station                0020             0030
// The selected train in the Track list is represented by the asterisk (*)

// Menu not shown in sample i/o

Choose an operation: V

Cursor has been moved to previous train.

// Menu not shown in sample i/o

Choose an operation: P

Selected Train:
    Train Number: 12340
    Train Destination: Port Jeff
    Arrival Time: 0000
    Departure Time: 0015

// Menu not shown in sample i/o

Choose an operation: A

Enter train number: 454532
Enter train destination: Huntington
Enter train arrival time: 0025
Enter train transfer time: 10

Train not added: There is a Train already scheduled on Track 1 at that time!

// Menu not shown in sample i/o

Choose an operation: TPS

Track 1 (1.74% Utilization Rate):
Selected  Train Number     Train Destination         Arrival Time     Departure Time
--------------------------------------------------------------------------------
    *        12340          Port Jeff                   0000             0015
               233          Penn Station                0020             0030

// Menu not shown in sample i/o

Choose an operation: N

Cursor has been moved to next train.

// Menu not shown in sample i/o

Choose an operation: N

Selected train not updated: Already at end of Track list.

// Menu not shown in sample i/o

Choose an operation: A

Enter train number: 1231
Enter train destination: Ronkonkoma
Enter train arrival time: 1435
Enter train transfer time: 20

Train No. 1231 to Ronkonkoma added to Track 1.

// Menu not shown in sample i/o

Choose an operation: TPS

Track 1 (3.13% Utilization Rate):
Selected  Train Number     Train Destination         Arrival Time     Departure Time
--------------------------------------------------------------------------------
             12340          Port Jeff                   0000             0015
               233          Penn Station                0020             0030
    *         1231          Ronkonkoma                  1435             1455

// Menu not shown in sample i/o

Choose an operation: R

Train No. 1231 to Ronkonkoma has been removed from Track 1.

// Menu not shown in sample i/o
```

```
Choose an operation: TPS

Track 1 (1.74% Utilization Rate):
Selected  Train Number     Train Destination         Arrival Time     Departure Time
--------------------------------------------------------------------------------
            12340          Port Jeff                    0000             0015
    *         233          Penn Station                 0020             0030

// Menu not shown in sample i/o

Choose an operation: SI

Station (1 track):
    Track 1: 2 trains arriving (1.74% Utilization Rate)

// Menu not shown in sample i/o

Choose an operation: TA

Enter track number: 2

Track 2 added to the Station.

// Menu not shown in sample i/o

Choose an operation: TS

Enter the Track number: 1

Switched to Track 1.

// Menu not shown in sample i/o

Choose an operation: A

Enter train number: 12340
Enter train destination: Huntington
Enter train arrival time: 1405
Enter train transfer time: 10

Train not added: There is already a Train with that number!

// Menu not shown in sample i/o

Choose an operation: TS

Enter the Track number: 9

Could not switch to Track 9: Track 9 does not exist.

// Menu not shown in sample i/o

Choose an operation: TS

Enter the Track number: 2

Switched to Track 2.

// Menu not shown in sample i/o

Choose an operation: SI

Station (2 tracks):
    Track 1: 2 trains arriving (1.74% Utilization Rate)
    Track 2: 0 trains arriving (0.00% Utilization Rate)

// Menu not shown in sample i/o

Choose an operation: TPS

Track 2 (0.00% Utilization Rate):
Selected  Train Number     Train Destination         Arrival Time     Departure Time
--------------------------------------------------------------------------------

// Menu not shown in sample i/o

Choose an operation: A

Enter train number: 9394
Enter train destination: Huntington
Enter train arrival time: 1405
Enter train transfer time: 10

Train No. 9394 to Huntington added to Track 2.
```

```
// Menu not shown in sample i/o

Choose an operation: TPS

Track 2 (0.69% Utilization Rate):
Selected  Train Number     Train Destination         Arrival Time     Departure Time
--------------------------------------------------------------------------------
    *         9394          Huntington                   1405             1415

// Menu not shown in sample i/o

Choose an operation: TPA

// The selected train the cursor is pointing to in each Track list is represented by the asterisk (*)
Track 1 (1.74% Utilization Rate):
Selected  Train Number     Train Destination         Arrival Time     Departure Time
--------------------------------------------------------------------------------
            12340          Port Jeff                    0000             0015
    *         233          Penn Station                 0020             0030

Track 2* (0.69% Utilization Rate): // asterisk (*) denotes selected track
Selected  Train Number     Train Destination         Arrival Time     Departure Time
--------------------------------------------------------------------------------
    *         9394          Huntington                   1405             1415

// Menu not shown in sample i/o

Choose an operation: TS

Enter the Track number: 4

Could not switch tracks: Track 4 does not exist.

// Menu not shown in sample i/o

Choose an operation: TR

Closed Track 2.

// Menu not shown in sample i/o

Choose an operation: TPA

Track 1* (1.74% Utilization Rate): // asterisk (*) denotes selected track
Selected  Train Number     Train Destination         Arrival Time     Departure Time
--------------------------------------------------------------------------------
            12340          Port Jeff                    0000             0015
    *         233          Penn Station                 0020             0030

// Menu not shown in sample i/o

Choose an operation: TA

Enter track number: 9

Track 9 added to the Station.

// Menu not shown in sample i/o

Choose an operation: TA

Enter track number: 5

Track 5 added to the Station.

// Menu not shown in sample i/o

Choose an operation: SI

Station (3 tracks):
    Track 1: 2 trains arriving (1.74% Utilization Rate)
    Track 5: 0 trains arriving (0.00% Utilization Rate)
    Track 9: 0 trains arriving (0.00% Utilization Rate)

// Menu not shown in sample i/o

Choose an operation: Q

Program terminating normally...
```
