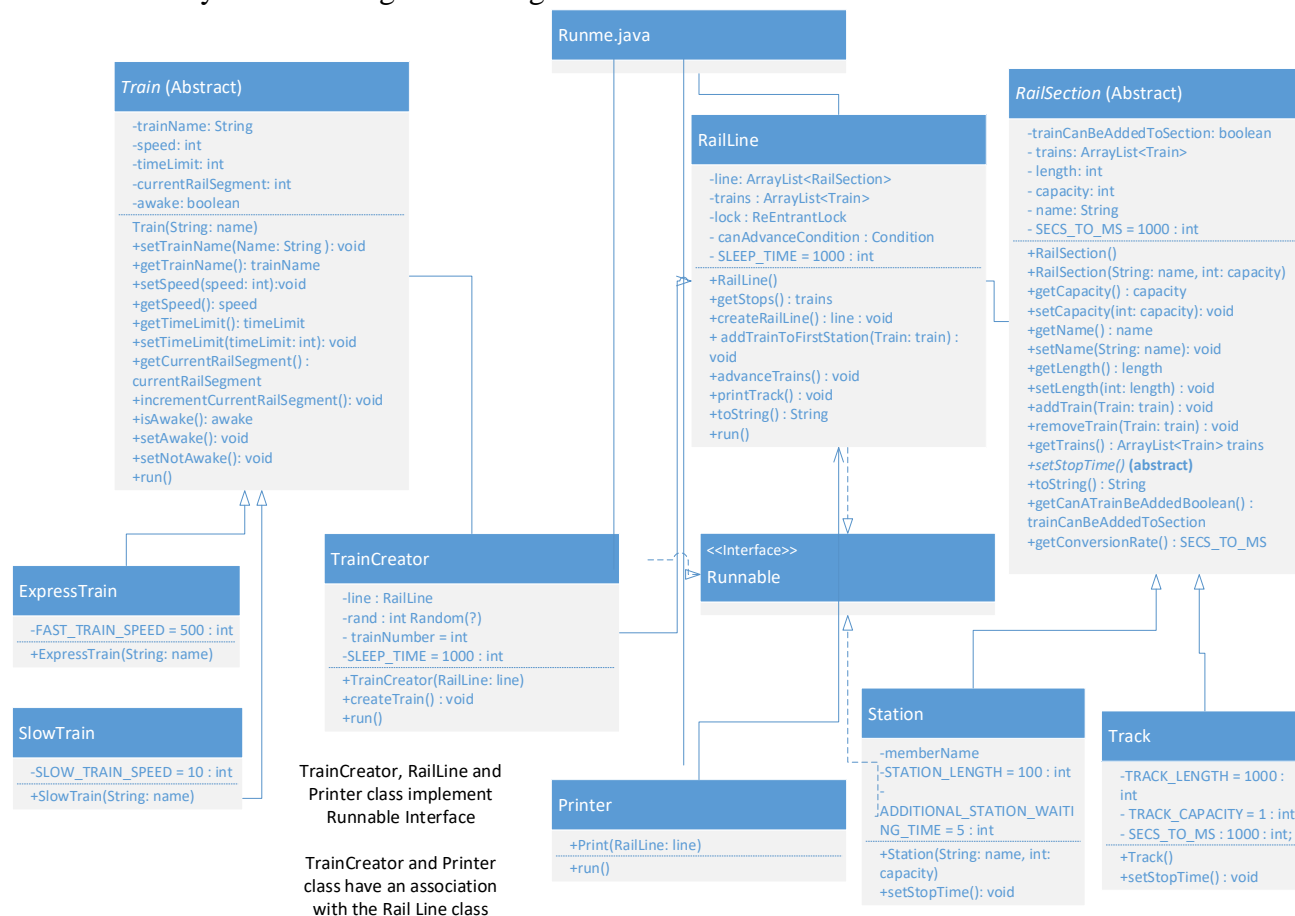My program consists of nine classes plus an additional "Runme" class that starts the program. It can be illustrated by the following UML Diagram.

Runme.java

Train (Abstract)
- -trainName: String
- -speed: int
- -timeLimit: int
- -currentRailSegment: int
- -awake: boolean
- Train(String: name)
- +setTrainName(Name: String): void
- +getTrainName(): trainName
- +setSpeed(speed: int):void
- +getSpeed(): speed
- +getTimeLimit(): timeLimit
- +setTimeLimit(timeLimit: int): void
- +getCurrentRailSegment() : currentRailSegment
- +incrementCurrentRailSegment(): void
- +isAwake(): awake
- +setAwake(): void
- +setNotAwake(): void
- +run()

RailLine
- -line: ArrayList<RailSection>
- -trains : ArrayList<Train>
- -lock : ReEntrantLock
- - canAdvanceCondition : Condition
- - SLEEP_TIME = 1000 : int
- +RailLine()
- +getStops() : trains
- +createRailLine() : line : void
- + addTrainToFirstStation(Train: train) : void
- +advanceTrains() : void
- +printTrack() : void
- +toString() : String
- +run()

RailSection (Abstract)
- -trainCanBeAddedToSection: boolean
- - trains: ArrayList<Train>
- - length: int
- - capacity: int
- - name: String
- - SECS_TO_MS = 1000 : int
- +RailSection()
- +RailSection(String: name, int: capacity)
- +getCapacity() : capacity
- +setCapacity(int: capacity): void
- +getName() : name
- +setName(String: name): void
- +getLength() : length
- +setLength(int: length) : void
- +addTrain(Train: train) : void
- +removeTrain(Train: train) : void
- +getTrains() : ArrayList<Train> trains
- +setStopTime() (abstract)
- +toString() : String
- +getCanATrainBeAddedBoolean() : trainCanBeAddedToSection
- +getConversionRate() : SECS_TO_MS

ExpressTrain
- -FAST_TRAIN_SPEED = 500 : int
- +ExpressTrain(String: name)

SlowTrain
- -SLOW_TRAIN_SPEED = 10 : int
- +SlowTrain(String: name)

TrainCreator
- -line : RailLine
- -rand : int Random(?)
- - trainNumber = int
- -SLEEP_TIME = 1000 : int
- +TrainCreator(RailLine: line)
- +createTrain() : void
- +run()

TrainCreator, RailLine and Printer class implement Runnable Interface

TrainCreator and Printer class have an association with the Rail Line class

<<Interface>> Runnable

Station
- -memberName
- -STATION_LENGTH = 100 : int
- -ADDITIONAL_STATION_WAITING_TIME = 5 : int
- +Station(String: name, int: capacity)
- +setStopTime(): void

Track
- -TRACK_LENGTH = 1000 : int
- - TRACK_CAPACITY = 1 : int;
- - SECS_TO_MS : 1000 : int;
- +Track()
- +setStopTime() : void

Printer
- +Print(RailLine: line)
- +run()

The "ExpressTrain" and "SlowTrain" classes will inherit from the abstract "Train" class. Express and slow trains differ in speed; therefore, each class has a constant to reflect this.

The train class implements the Runnable interface. Although an instance of this class cannot be instantiated as the class itself is abstract, the subclasses will inherit the run method. The aim of this method is to force a train thread to sleep for an allotted time on a rail section that was calculated using the setStopTime() method in the track/station subclasses.

The "Station" and "Track" classes will inherit from the abstract "RailSection" class. The different lengths of each rail section are reflected by the constants in either subclass. For a track segment, the capacity is fixed as a constant. Meanwhile, a station's capacity is determined by the value that is passed to the constructor. The Rail Section class has a constant, SECS_TO_MS, that both subclasses invoke to convert stop times into milliseconds and get the correct time that a train occupies a rail section.

The "TrainCreator" class is responsible for creating trains indefinitely, the "Printer" class is responsible for printing the status of the rail line indefinitely and the "RailLine" class is responsible for advancing trains indefinitely. These three classes implement Runnable and a thread object is created per class. The thread object created from the "TrainCreator" instance will place a train on the first station. If there is not enough room, the thread will go into the waiting state. This allows the thread created from the "RailLine" class to call the advanceTrains method, which advances trains

along the railway line and updates the shared RailLine object. When successful, the thread will call the signalAll method to enable the thread from the "TrainCreator" class to awake from the waiting state, reclaim the lock and finish adding a new train to the first station.