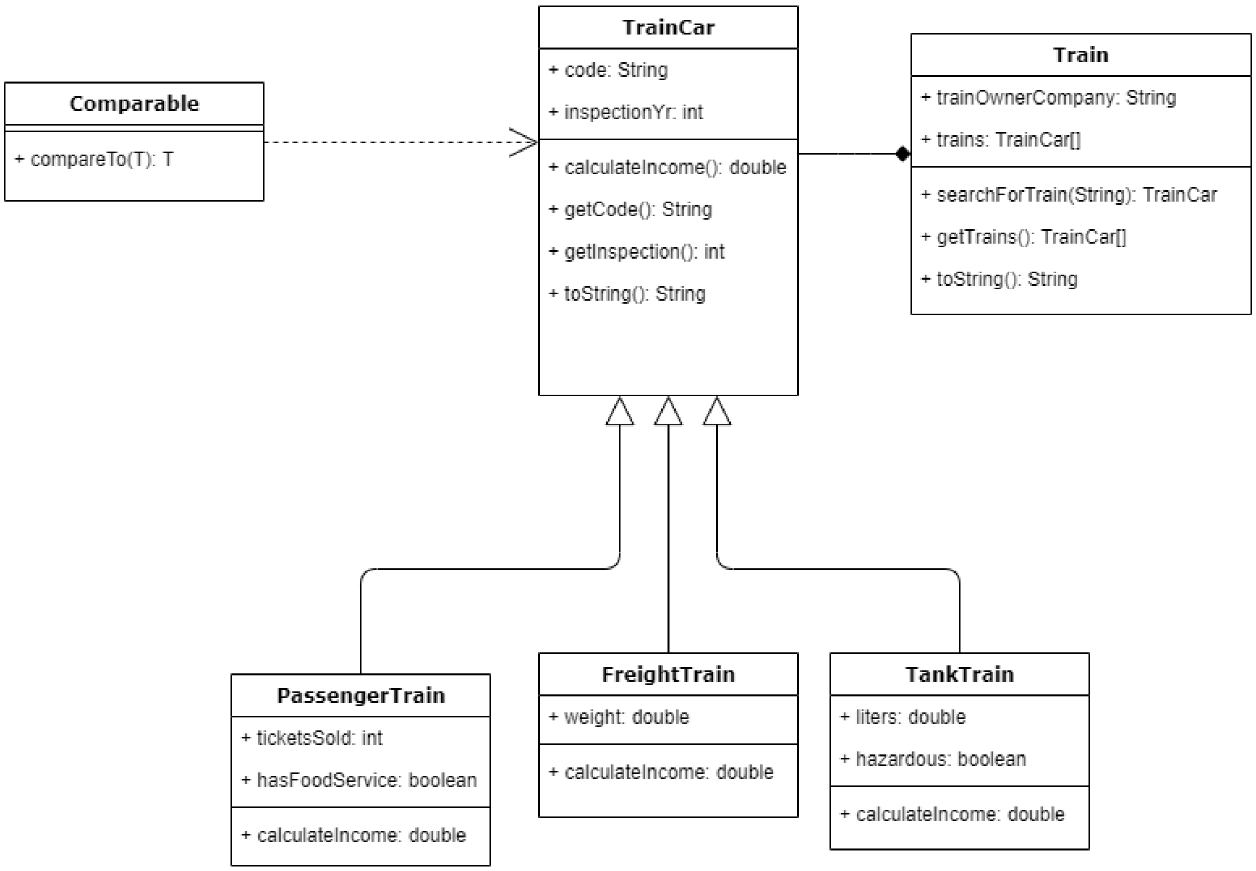
Isaac Ray Shoebottom
CS 1083
Assignment 4
3429069

Section A :



Section B

Source Code for Train:

```
/**
 * Train class
 * @author Isaac Shoebottom (3429069)
 */

public class Train {
    String trainOwnerCompany;
    TrainCar[] trains;

    /**
     * Train constructor
     * @param trainOwnerCompany The owner of the train
     * @param trains The train cars the train has
     */
    Train(String trainOwnerCompany, TrainCar[] trains) {
        this.trainOwnerCompany = trainOwnerCompany;
        this.trains = trains;
    }

    /**
     * Searches for the the train based on the code
     * @param code The code that is being searched for
     * @return The train object that is returned if it is found
     */
    public TrainCar searchForTrain(String code) {
        for(TrainCar train: trains) {
            if(train.getCode().compareToIgnoreCase(code) == 0) {
                return train;
            }
        }
        return null;
    }

    /**
     * Get a copy of the train car array
     * @return A copy of the train car array
     */
    public TrainCar[] getTrains() {
        return trains.clone();
    }
```

```
    /**
     * Textual representation of the the array
     * @return A string containing a textual representation of the array
     */
    public String toString() {
        String str = trainOwnerCompany + "\n";
        for(TrainCar train : trains) {
            str += train.toString();
        }
        return str;
    }
}
```

Source code for TrainCar:

```
/**
 * TrainCar class
 * @author Isaac Shoebottom (3429069)
 */

import java.text.NumberFormat;

public abstract class TrainCar implements Comparable<TrainCar> {
    private final String code;
    private final int inspectionYr;

    /**
     * Calculates income
     * @return The income of that car
     */
    abstract double calculateIncome();


    /**
     * TrainCar constructor
     * @param code The unique identifier for the train car
     * @param inspectionYr The year the train was last inspected
     */
    TrainCar(String code, int inspectionYr) {
        this.code = code;
        this.inspectionYr = inspectionYr;
    }


    /**
     * Method to compare to train cars to determine the natural order
     * @param other The train car to be compared against
     * @return -1, 0, 1 based on if the object is greater than the object
being compared against
     */
    public int compareTo(TrainCar other) {
        String obj1 = String.valueOf(code.charAt(0)).toUpperCase();
        String obj2 =
String.valueOf(other.getCode().charAt(0)).toUpperCase();
        if (obj1.compareTo(obj2) > 0) {
            return 1;
        }
        else if (obj1.compareTo(obj2) < 0) {
            return -1;
        }
        else {
            int obj1Int = Integer.parseInt(String.valueOf(code.charAt(1) +
code.charAt(2) + code.charAt(3)));
```

```
            int obj2Int =
Integer.parseInt(String.valueOf(other.getCode().charAt(1) +
other.getCode().charAt(2) + other.getCode().charAt(3)));
            if (obj1Int < obj2Int) {
                return 1;
            }
            else if (obj1Int > obj2Int) {
                return -1;
            }
            return 0;
        }
    }

    /**
     * Gets the train cars unique code
     * @return The unique code
     */
    public String getCode() {
        return code;
    }

    /**
     * Gets the inspection year
     * @return The inspection year
     */
    public int getInspectionYr() {
        return inspectionYr;
    }

    /**
```

```
     * String representing the train car
     * @return A string representing the attributes of a train car
     */
    public String toString(){
        NumberFormat cf = NumberFormat.getCurrencyInstance();
        String income = cf.format(calculateIncome());
        return "[Code: " + code + "\tInspection Year: "+ inspectionYr +
"\tIncome: "+ income + "]\n";
    }

}
```

Source code for PassengerTrain:

```
/**
 * PassengerTrain class
 * @author Isaac Shoebottom (3429069)
 */

public class PassengerTrain extends TrainCar {
    int ticketsSold;
    boolean hasFoodService;

    /**
     * Constructor for a passenger train
     * @param code The trains unique code
     * @param inspectionYr The year the train was last inspected
     * @param ticketsSold The number of tickets the train has sold
     * @param hasFoodService If the train car has food service
     */
    PassengerTrain(String code, int inspectionYr, int ticketsSold, boolean
hasFoodService) {
        super(code, inspectionYr);
        this.ticketsSold = ticketsSold;
        this.hasFoodService = hasFoodService;
    }

    /**
     * The income calculator specific to passenger train
     * @return The income of that train
     */
    @Override
    double calculateIncome() {
        return (hasFoodService) ? ticketsSold*150 : ticketsSold*120;
    }
}
```

Source code for FreightTrain:

```
/**
 * FreightTrain class
 * @author Isaac Shoebottom (3429069)
 */

public class FreightTrain extends TrainCar {
    double weight;

    /**
     * The constructor for a freight train
     * @param code The unique code of this train
     * @param inspectionYr The year this train was last inspected
     * @param weight The weight of the mass the train is carrying
     */
    FreightTrain(String code, int inspectionYr, double weight) {
        super(code, inspectionYr);
        this.weight = weight;
    }

    /**
     * The income calculator specific to freight train
     * @return The income of that train
     */
    @Override
    double calculateIncome() {
        if (weight < 100000) {
            return 11500;
        }
        else {
            return 25000;
        }
    }
}
```

Source code for TankTrain:

```
/**
 * TankTrain class
 * @author Isaac Shoebottom (3429069)
 */

public class TankTrain extends TrainCar {
    double litres;

    boolean hazardous;


    /**

     * The constructor for the tank train

     * @param code The unique code for the tank train

     * @param inspectionYr The year the train was last inspected

     * @param litres The litres the train is carrying

     * @param hazardous If the train is carrying hazardous materials

     */

    TankTrain(String code, int inspectionYr, double litres, boolean
hazardous) {

        super(code, inspectionYr);

        this.litres = litres;

        this.hazardous = hazardous;

    }


    /**

     * The income calculator specific to tank train

     * @return The income of that train

     */

    @Override

    double calculateIncome() {

        return (hazardous) ? litres*17 : litres*9.5;

    }

}
```

Source code for Driver

```
/**

 * Driver
 * @author Isaac Shoebottom (3429069)
 */

import java.util.Scanner;

public class Driver {
    /**
     * Main method
     * @param args program arguments
     */
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String companyName = scan.nextLine();
        int trainsLength = Integer.parseInt(scan.nextLine());
        TrainCar[] tempArr = new TrainCar[trainsLength];
        Sorter<TrainCar> sorter = new Sorter<>();
        for(int i = 0; i < trainsLength; i++) {
            String code = scan.next();
            int inspectionYear = Integer.parseInt(scan.next());
            if (code.charAt(0)  == 'F') {
                double weight = Double.parseDouble(scan.next());
                tempArr[i] = new FreightTrain(code, inspectionYear, weight);
            }
            if (code.charAt(0)  == 'P') {
                int ticketsSold = Integer.parseInt(scan.next());
                boolean isServiceTrain = Boolean.parseBoolean(scan.next());
                tempArr[i] = new PassengerTrain(code, inspectionYear,
ticketsSold, isServiceTrain);
            }
            if (code.charAt(0)  == 'T') {
                double litres = Double.parseDouble(scan.next());

                boolean isHazardous = Boolean.parseBoolean(scan.next());

                tempArr[i] = new TankTrain(code, inspectionYear, litres,
isHazardous);

            }

        }

        Train train = new Train(companyName, tempArr);


        System.out.print(train.toString());

        TrainCar[] tempArrSorted = train.getTrains();

        sorter.selectionSort(tempArrSorted);

        System.out.print("Sorted Data:\n");


        for (TrainCar trainCar: tempArrSorted) {

            System.out.println(trainCar.getCode() + "\t$" +
trainCar.calculateIncome());

        }


        System.out.print("Search Results:\n");


        String code;

        while (scan.hasNext()) {

            code = scan.next();

            if (train.searchForTrain(code) != null) {

                System.out.print("Train Car " + code + " found\n");

            }

            else if (train.searchForTrain(code) == null){

                System.out.print("Train Car " + code + " NOT found\n");

            }
        }

    }
}
```

Section C

Input 1:

```
CN Railway

5

P714 2012 85 true

F019 2018 75000

F221 2016 105000

T102 2017 35000 true

P904 2018 78 false

T102

F220

P714
```

Output 1:

```
CN Railway

[Code: P714      Inspection Year: 2012     Income: $12,750.00]

[Code: F019      Inspection Year: 2018     Income: $11,500.00]

[Code: F221      Inspection Year: 2016     Income: $25,000.00]

[Code: T102      Inspection Year: 2017     Income: $595,000.00]

[Code: P904      Inspection Year: 2018     Income: $9,360.00]

Sorted Data:

F019     $11500.0

F221     $25000.0

P904     $9360.0

P714     $12750.0
```

```
T102     $595000.0

Search Results:

Train Car T102 found

Train Car F220 NOT found

Train Car P714 found


Process finished with exit code 0
```

Input 2

```
CN Railway

7

P714 2012 85 true

F019 2018 75000

F221 2016 105000

F222 2016 1050000

T102 2017 35000 true

P904 2018 78 false

P905 2018 75 false

T102

F220

P714

P905
```

Output 2:

```
CN Railway

[Code: P714      Inspection Year: 2012    Income: $12,750.00]

[Code: F019      Inspection Year: 2018    Income: $11,500.00]

[Code: F221      Inspection Year: 2016    Income: $25,000.00]

[Code: F222      Inspection Year: 2016    Income: $25,000.00]

[Code: T102      Inspection Year: 2017    Income: $595,000.00]
```

```
[Code: P904     Inspection Year: 2018    Income: $9,360.00]

[Code: P905     Inspection Year: 2018    Income: $9,000.00]

Sorted Data:

F019    $11500.0

F222    $25000.0

F221    $25000.0

P905    $9000.0

P904    $9360.0

P714    $12750.0

T102    $595000.0

Search Results:

Train Car T102 found

Train Car F220 NOT found

Train Car P714 found

Train Car P905 found


Process finished with exit code 0
```

Input 3:

```
CN Railway

3

P714 2012 85 true

T102 2017 35000 true

P904 2018 78 false

T102

F220

P714

P906
```

Output 3:

```
CN Railway

[Code: P714      Inspection Year: 2012    Income: $12,750.00]

[Code: T102      Inspection Year: 2017    Income: $595,000.00]

[Code: P904      Inspection Year: 2018    Income: $9,360.00]

Sorted Data:

P904     $9360.0

P714     $12750.0

T102     $595000.0

Search Results:

Train Car T102 found

Train Car F220 NOT found

Train Car P714 found

Train Car P906 NOT found


Process finished with exit code 0
```