```
// RentRep.java - GENERATE THE RENTAL PROFILE REPORT
//
// MODULE INDEX
// NAME                 CONTENTS
// Event                Event structure
// Rental               Rental structure
// Stats                Statistics structure
// Stats.Stats          Statistics structure constructor
// Stats.add            Add statistics structures
// Stats.format         Format a statistics structure
// Outlet               Outlet data structure
// readString           Read a string from the log file
// readEvent            Read an event record from the log file
// main                 Main line
//
// MAINTENANCE HISTORY
// DATE         PROGRAMMER AND DETAILS
// 17-09-13     JS      Original
//
//-----------------------------------------------------------------------------

// IMPORTS

import java.util.*;
import java.text.*;
import java.io.*;

//-----------------------------------------------------------------------------

public class RentRep {

    //-------------------------------------------------------------------------

    // DEFINITIONS

    static final long MILLIS_PER_DAY = 24L*60L*60L*1000L;

    //-------------------------------------------------------------------------

    // EVENT STRUCTURE

    public static class Event {
        public long     eventDate;      // Event date in days since epoch
        public String   outlet;         // Outlet name
        public String   operation;      // Operation (OUT or IN)
        public String   vehicleTag;     // Vehicle registration tag
        public String   vehicleType;    // Vehicle type
        public int      odometer;       // Odometer reading
        public String   licenceNo;      // Customer's driver's licence number
        public String   custName;       // Customer's name
        public int      amountPaid;     // Amount paid by the customer
    }

    //-------------------------------------------------------------------------

    // RENTAL STRUCTURE

    public static class Rental {
        long            dateOut;        // Date the vehicle was rented out
        int             odometerOut;    // Odometer reading when rented out
```

```
    int             amountPaid;     // Amount paid by the customer
}

//---------------------------------------------------------------------

// STATISTICS STRUCTURE

public static class Stats {
    TreeSet<String> custSet;        // Customer set
    long            daysRented;     // Vehicle-days rented
    long            distanceDriven; // Distance driven
    long            rentalReceived; // Rental received

    // Constructor

    Stats ()
    {
        custSet = new TreeSet<String> ();
        daysRented = 0;
        distanceDriven = 0;
        rentalReceived = 0;
    }

    // Add One Statistics Structure to Another

    void
    add (
        Stats       s)              // Value to add
    {
        custSet.addAll (s.custSet);
        daysRented += s.daysRented;
        distanceDriven += s.distanceDriven;
        rentalReceived += s.rentalReceived;
    }

    // Format a Statistics Structure

    String
    format ()
    {
        return String.format ("%9d%8d%9d%9d",
            custSet.size(), daysRented, distanceDriven, rentalReceived);
    }
}

//---------------------------------------------------------------------

// OUTLET DATA STRUCTURE

public static class Outlet {
    TreeMap<String,Stats>   vehicleMap; // Vehicle statistics map
}

//---------------------------------------------------------------------

// GLOBAL VARIABLES

static FileReader   inpReader;      // Input stream reader
static int          inpChar;        // Input character

//---------------------------------------------------------------------
```

```
// READ A STRING FROM THE LOG FILE

static String
ReadString (
    char            termChar)       // Terminating character
throws IOException
{
    String          st;             // String value

    st = new String ();
    while (inpChar != '\t' && inpChar != '\n' && inpChar != -1) {
        st += (char)inpChar;
        inpChar = inpReader.read();
    }
    if (inpChar != termChar)
        throw new IOException("Missing tab");
    inpChar = inpReader.read();
    return st;
}

//---------------------------------------------------------------------------

// READ AN EVENT RECORD FROM THE LOG FILE

static Event
ReadEvent ()
{
    int             yr, mo, dy;     // Date components
    boolean         negative;       // Negative flag
    Event           event;          // Event structure
    String          field;          // Field string
    SimpleDateFormat dateFormat;    // Date formatter

    event = new Event ();
    try {
        // Check for end-of-file

        if (inpChar == -1) return null;

        // Read the date

        field = ReadString ('\t');
        dateFormat = new SimpleDateFormat ("yyyy-MM-dd");
        dateFormat.setTimeZone (new SimpleTimeZone (0, "UTC"));
        event.eventDate = dateFormat.parse(field).getTime()
            / MILLIS_PER_DAY;

        // Skip the time

        ReadString ('\t');

        // Read the other fields

        event.outlet = ReadString ('\t');
        event.operation = ReadString ('\t');
        event.vehicleTag = ReadString ('\t');
        event.vehicleType = ReadString ('\t');
        event.odometer = Integer.parseInt (ReadString('\t'));
        event.licenceNo = ReadString ('\t');
        event.custName = ReadString ('\t');
```

```
        event.amountPaid = Integer.parseInt (ReadString('\n'));
    }
    catch (Exception e) {
        System.err.println ("Exception: " + e.toString());
        System.exit (1);
    }
    return event;
}

//---------------------------------------------------------------------------

// MAIN LINE

public static void
main (
    String[]        argv)           // Argument values
{
    SimpleDateFormat dateFormat;    // Date formatter
    long            fromDate;       // From date in days since epoch
    long            toDate;         // To date in days since epoch
    Event           event;          // Event structure
    Rental          rental;         // Rental structure
    TreeMap<String,Rental> rentalMap; // Rental map
    long            totalDays;      // Total rental days
    long            interDays;      // Intersecting days
    long            interFromDate;  // Intersection from-date
    long            interToDate;    // Intersection to-date
    Stats           stats;          // Statistics structure
    Outlet          outlet;         // Outlet structure
    TreeMap<String,Outlet> outletMap; // Outlet statistics map
    Stats           outletTotal;    // Outlet total
    Stats           grandTotal;     // Grand total

    // Decode the parameters

    if (argv.length != 2) {
        System.out.println ("Usage: java RentRep from-date to-date");
        System.exit (1);
    }
    dateFormat = new SimpleDateFormat ("dd-MM-yy");
    dateFormat.setTimeZone (new SimpleTimeZone (0, "UTC"));
    try {
        fromDate = dateFormat.parse(argv[0]).getTime()
            / MILLIS_PER_DAY;
        toDate = dateFormat.parse(argv[1]).getTime()
            / MILLIS_PER_DAY;
    }
    catch (ParseException e) {
        System.out.println ("Exception: " + e.toString());
        System.exit (1);
        fromDate = toDate = 0;
    }

    // Validate the date range

    if (toDate < fromDate) {
        System.out.println ("Error: toDate < fromDate");
        System.exit (1);
    }

    // Open the input stream
```

```
try {
    inpReader = new FileReader ("RENTLOG");
    inpChar = inpReader.read();
}
catch (Exception e) {
    System.err.println ("Exception: " + e.toString());
    System.exit (1);
}

// Create the rental map

rentalMap = new TreeMap<String,Rental> ();

// Create the outlet statistics map

outletMap = new TreeMap<String,Outlet> ();

// Process each record in the log file

while ((event = ReadEvent()) != null) {

    // If the operation is a vehicle being rented out,
    // create a new entry for the rental in the rental map

    if (event.operation.equals ("OUT")) {

        // Load a new rental structure

        rental = new Rental ();
        rental.dateOut = event.eventDate;
        rental.odometerOut = event.odometer;
        rental.amountPaid = event.amountPaid;

        // Store the rental structure in the rental map

        rentalMap.put (event.vehicleTag, rental);
    }

    // If the operation is a vehicle being returned,
    // post the rental statistics to the report tables

    else if (event.operation.equals ("IN")) {

        // Look up the rental data.  If there is no rental data
        // assume this is a mismatched return and quietly disregard
        // the entry.

        rental = rentalMap.get (event.vehicleTag);
        if (rental != null) {

            // Only process rentals that are not empty and intersect
            // the reporting period

            if (
                rental.dateOut <= event.eventDate &&
                rental.dateOut <= toDate &&
                event.eventDate >= fromDate
            ) {
                // Calculate the total rental days
```

```
                totalDays = event.eventDate - rental.dateOut + 1;

                // Calculate the intersecting rental days

                if (rental.dateOut > fromDate)
                        interFromDate = rental.dateOut;
                else
                        interFromDate = fromDate;
                if (toDate < event.eventDate)
                        interToDate = toDate;
                else
                        interToDate = event.eventDate;
                interDays = interToDate - interFromDate + 1;

                // Look up or initialise the statistics row for
                // the vehicle/outlet combination

                outlet = outletMap.get (event.outlet);
                if (outlet == null) {
                    outlet = new Outlet ();
                    outlet.vehicleMap = new TreeMap<String,Stats> ();
                    outletMap.put (event.outlet, outlet);
                }
                stats = outlet.vehicleMap.get (event.vehicleType);
                if (stats == null) {
                    stats = new Stats ();
                    outlet.vehicleMap.put (event.vehicleType, stats);
                }

                // Add the rental to the statistics

                stats.custSet.add (event.licenceNo);
                stats.daysRented += interDays;
                stats.distanceDriven +=
                    (event.odometer - rental.odometerOut) * interDays
                        / totalDays;
                stats.rentalReceived +=
                    (event.amountPaid + rental.amountPaid) * interDays
                        / totalDays;
            }
        }

        // Remove the opening rental entry

        rentalMap.remove (event.vehicleTag);
    }
}

// Display the report

System.out.println ("RENTAL PROFILE REPORT");
System.out.print ("For ");
System.out.print (dateFormat.format(new Date(fromDate*MILLIS_PER_DAY)));
System.out.print (" to ");
System.out.print (dateFormat.format(new Date(toDate*MILLIS_PER_DAY)));
System.out.println ();
System.out.println ();

System.out.print ("                                ");
System.out.println ("No of    Days Distance   Rental");
System.out.print ("Outlet     Vehicle        ");
```

```
        System.out.println ("Customers  Rented   Driven Received");

        grandTotal = new Stats ();
        for (Map.Entry<String,Outlet> outletEntry : outletMap.entrySet()) {
            System.out.println ();
            System.out.printf ("%-11s", outletEntry.getKey());
            outlet = outletEntry.getValue();
            outletTotal = new Stats ();
            for (Map.Entry<String,Stats> vehicleEntry :
              outlet.vehicleMap.entrySet()) {
                System.out.printf ("%-15s", vehicleEntry.getKey());
                stats = vehicleEntry.getValue ();
                System.out.println (stats.format());
                System.out.printf ("%-11s", "");
                outletTotal.add (stats);
            }
            System.out.printf ("%-15s     ----   -----  ------- --------\n","");
            System.out.printf ("%-11s%-15s", "", "");
            System.out.println (outletTotal.format());
            grandTotal.add (outletTotal);
        }
        System.out.printf ("%-11s", "");
        System.out.printf ("%-15s     ----   -----  ------- --------\n", "");
        System.out.printf ("%-11s%-15s", "Grand total", "");
        System.out.println (grandTotal.format());
        System.out.printf ("%-26s     ====   =====  ======= ========\n", "");
    }
}
```