```
// Workshop.java - WORKSHOP CLASS
//
// MAINTENANCE HISTORY
// DATE             PROGRAMMER AND DETAILS
// 23-09-14  MPF    Original
//
//-----------------------------------------------------------------------------

import java.io.*;
import java.util.*;
import java.lang.Thread;

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

// CLASS DECLARATION

public class Workshop {

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

    // PERFORM CUSTOMISATION

    public static Data.CarMap
    readCarMap (
       String       file,        // Customisation file
       OutputStreamWriter errorOut,     // Error output stream
       Machines     machines)    // Machines API
    throws Exception
    {
       Data.CarMap  carMap;             // Car map
       BufferedReader     reader;             // Reader
       String       line;        // Single-line data
       String[]     values;             // Values
       String       carNo;       // Car serial number
       Data.AccessoryMap accessoryMap;  // Accessory map

       carMap = new Data.CarMap();
       reader = new BufferedReader(new InputStreamReader(
           new FileInputStream(file)));

       while ((line = reader.readLine()) != null) {
           values = line.split (",", -1);
           if (values.length != 5) {
              System.out.println ("Bad line " + line);
           } else {
              carNo = values[0];
              accessoryMap = new Data.AccessoryMap();

              readSku (carNo, values[1], AccessoryType.RIM, accessoryMap,
                 errorOut, machines);
              readSku (carNo, values[2], AccessoryType.TYRE, accessoryMap,
                 errorOut, machines);
              readSku (carNo, values[3], AccessoryType.AUDIO, accessoryMap,
                 errorOut, machines);
              readSku (carNo, values[4], AccessoryType.GPS, accessoryMap,
```

```
            errorOut, machines);

          if (carMap.containsKey (carNo))
              System.out.println ("Car duplicate " + carNo);
          else
              carMap.put (carNo, accessoryMap);
        }
    }

    return carMap;
}


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

// READ SKU

public static void
readSku (
    String       carNo,       // Car serial number
    String       skuId,       // SKU identifier
    AccessoryTypeaccType,     // Accessory type
    Data.AccessoryMap accessoryMap,  // Accessory map
    OutputStreamWriter errorOut,     // Error output stream
    Machines     machines)    // Machines API
throws IOException
{
    if (skuId.length() > 0) {
        if (machines.isValidAccessory (accType, skuId))
            accessoryMap.put (accType, skuId);
        else {
            errorOut.append (carNo);
            errorOut.append (": Invalid SKU ");
            errorOut.append (skuId);
            errorOut.append ("\n");
        }
    }
}

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

// PERFORM CUSTOMISATION

public static void
performCustomisation (
    String       file,        // Customisation file
    String       errorFile,   // Error file
    Machines     machines)    // Machines API
throws Exception
{
    Data.CarMap  carMap;             // Car map
    List<Station>stationList; // Station list
    Station             newStation;  // New station
    boolean             allCompleted;// All stations have completed
    OutputStreamWriter errorOut;     // Error output stream
```

```
        errorOut = new OutputStreamWriter(new FileOutputStream(errorFile,true));
        carMap = readCarMap(file, errorOut, machines);

        // Initialise stations

        stationList = new ArrayList<Station> ();
        for (int stationId : machines.getServiceableStations())
            stationList.add (new Station (stationId, carMap, machines));

        // Process until all stations have no more cars

        do {
            allCompleted = true;
            for (Station station : stationList) {
               if (! station.process ())
                   allCompleted = false;
            }
            // Do not sleep during test
            //Thread.sleep (1000);

        } while (!allCompleted);

        // Log error for all invalid cars

        for (String carNo : carMap.keySet()) {
            errorOut.append ("Invalid car ");
            errorOut.append (carNo);
            errorOut.append ("\n");
        }
        errorOut.close();
    }
}
```