```java
package model;
import java.io.BufferedReader;

public class Engine {
    private static Engine instance = null;

    private static String posDir = System.getProperty("user.home")
+ "/ws_4413/.metadata/.plugins/org.eclipse.wst.server.core/tmp0/
wtpwebapps/eFoods/POs/";
    private String processedOrdersFileName;
    private String reportsDir;
    private String outTo;
    private List<Product> products;
    private List<String> processed;
    private Report report;


    // ------------------------------------------------------
    private Engine() throws Exception {
        this(posDir);
    }

    private Engine(String POsDir) throws Exception {
        this(POsDir, null);
    };

    private Engine(String POsDir, String processedOrdersFileName)
throws Exception {
        products = new ArrayList<Product>();
        processed = new ArrayList<String>();
        report = new Report();

        this.setPOsDir(POsDir);
        if (this.checkPOsDir(POsDir)) {
            this.processedOrdersFileName =
processedOrdersFileName;
            this.getProcessedFromFile(processedOrdersFileName);

            processPOs(posDir);
            report.setProducts(products);
            this.saveProcessedToFile();
        }
    }

```

```
66      private Engine(String POsDir, String processedOrdersFileName,
    String reportsDir, String outTo) throws Exception {
67          products = new ArrayList<Product>();
68          processed = new ArrayList<String>();
69          report = new Report();
70
71          this.setPOsDir(POsDir);
72          this.reportsDir = reportsDir;
73          this.outTo = outTo;
74
75          if (this.checkPOsDir(POsDir)) {
76              this.processedOrdersFileName =
    processedOrdersFileName;
77              this.getProcessedFromFile(processedOrdersFileName);
78
79              processPOs(posDir);
80              report.setProducts(products);
81              this.saveProcessedToFile();
82              this.saveReportToFile(outTo);
83          }
84      }
85
86
87      // ------------------------------------------------------
88      public static Engine getInstance(String POsDir, String
    processedOrdersFileName, String reportsDir, String outTo) throws
    Exception {
89          if (instance == null) {
90              instance = new Engine(POsDir, processedOrdersFileName,
    reportsDir, outTo);
91          }
92          return instance;
93      }
94
95      public static Engine getInstance(String POsDir, String
    processedOrdersFileName) throws Exception {
96          if (instance == null) {
97              instance = new Engine(POsDir,
    processedOrdersFileName);
98          }
99          return instance;
100     }
101
102     public static Engine getInstance(String POsDir) throws
```

```
    Exception {
103         return getInstance(POsDir, null);
104     }
105 
106     public static Engine getInstance() throws Exception {
107         return getInstance(posDir);
108     }
109 
110 
111     // ------------------------------------------------
112     public List<String> getProcessed() {
113         return processed;
114     }
115 
116     public List<Product> getProducts() {
117         return products;
118     }
119 
120     public void setPOsDir(String POsDir) {
121         if (POsDir.equals("null") || POsDir.equals("none")) {
122             posDir = posDir;
123         } else if (checkPOsDir(POsDir) ) {
124             posDir = POsDir;
125         }
126     }
127 
128     public boolean checkPOsDir(String POsDir) {
129         if (POsDir != null && !POsDir.isEmpty()) {
130             return true;
131         } else if (POsDir.equals("null") || POsDir.equals("none"))
    {
132             return true;
133         }
134         return false;
135     }
136 
137 
138     // ------------------------------------------------
139     public synchronized void processPOs(File[] files) throws
    Exception {
140         for (File f: files) {
141             processPO(f);
142         }
143     }
```

```
144
145     public synchronized void processPOs(String dirName) throws
    Exception {
146         processPOs(new File(dirName));
147     }
148
149     public synchronized void processPOs(File dir) throws Exception
    {
150         processPOs(dir.listFiles());
151     }
152
153
154     // ------------------------------------------------
155     public synchronized void processPO(File f, List<Product>
    products, List<String> processed) throws Exception {
156         if (getFileExtension(f).equals(".xml")) {
157             if (!processed.contains(f.getName())) {
158
159                 DocumentBuilderFactory dbFactory =
    DocumentBuilderFactory.newInstance();
160                 DocumentBuilder dBuilder =
    dbFactory.newDocumentBuilder();
161                 Document doc = dBuilder.parse(f);
162                 doc.getDocumentElement().normalize();
163
164                 NodeList items = doc.getElementsByTagName("item");
165                 for (int i=0; i<items.getLength(); i++) {
166                     Node item = items.item(i);
167                     if (item.getNodeType() == Node.ELEMENT_NODE) {
168
169                         Element e = (Element) item;
170
171                         String itemNum = e.getAttribute("number");
172                         String itemName =
    e.getElementsByTagName("name").item(0).getTextContent();
173                         String itemQtyVal =
    e.getElementsByTagName("quantity").item(0).getTextContent();
174
175                         if (itemNum != null && itemNum != "") {
176                             int itemQty =
    Integer.parseInt(itemQtyVal);
177
178                             Product p = new Product(itemNum,
    itemName, itemQty);
```

```
179                             if (!products.contains(p)) {
180                                 products.add(p);
181                             } else {
182                                 p =
    products.get(products.indexOf(p));
183                                 p.increaseQty(itemQty);
184                                 products.set(products.indexOf(p),
    p);
185                             }
186                         }
187                     }
188                 }
189
190                 processed.add(f.getName());
191             }
192         }
193     }
194
195     public synchronized void processPO(File f) throws Exception {
196         processPO(f, this.products, this.processed);
197     }
198
199
200     // -----------------------------------------------------
201     public synchronized JsonObject createJsonReport(List<Product>
    products) throws Exception {
202         JsonObject json = new JsonObject();
203         JsonElement jsonElement = new Gson().toJsonTree(report);
204         json.add("data", jsonElement);
205         return json;
206     }
207
208     public synchronized JsonObject createJsonReport() throws
    Exception {
209         return createJsonReport(this.products);
210     }
211
212
213     // -----------------------------------------------------
214     public synchronized String createXmlReport(List<Product>
    products) throws Exception {
215         JAXBContext jc = JAXBContext.newInstance(Report.class);
216         Marshaller m = jc.createMarshaller();
217         StringWriter sw = new StringWriter();
```

```
218         m.marshal(new Report(), sw);
219         return sw.toString();
220     }
221 
222     public synchronized String createXmlReport() throws Exception
    {
223         return createXmlReport(this.products);
224     }
225 
226     public synchronized void saveReportToFile(String outTo) throws
    Exception {
227         this.checkDir(this.reportsDir);
228         String reportFile =
    (this.reportsDir.charAt(this.reportsDir.length()-1) == '/') ?
    this.reportsDir : this.reportsDir + "/";
229         reportFile = reportFile + report.getGeneratedOn();
230 
231         String reportCreated = "";
232 
233         if (outTo.toLowerCase().equals("xml")) {
234             reportFile = reportFile + ".xml";
235             reportCreated = this.createXmlReport();
236         } else if (outTo.toLowerCase().equals("json")) {
237             reportFile = reportFile + ".json";
238             reportCreated = this.createJsonReport().toString();
239         }
240         FileWriter fw = new FileWriter(reportFile);
241         fw.write(reportCreated);
242         fw.close();
243     }
244 
245 
246     // ------------------------------------------------------
247     public synchronized List<String> getProcessedFromFile(String
    filename) throws Exception {
248         File f = this.checkProcessedOrdersFile(filename);
249         BufferedReader br = new BufferedReader(new FileReader(f));
250         String line;
251         while ((line = br.readLine()) != null) {
252             if (!processed.contains(line)) {
253                 processed.add(line);
254             }
255         }
256         br.close();
```

```
257         return processed;
258     }
259 
260     public synchronized List<String> getProcessedFromFile() throws
    Exception {
261         return
    this.getProcessedFromFile(this.processedOrdersFileName);
262     }
263 
264 
265     // ------------------------------------------------
266     public synchronized void saveProcessedToFile(String filename)
    throws Exception {
267         File f = this.checkProcessedOrdersFile(filename);
268         FileWriter fw = new FileWriter(f);
269         for (String str: processed) {
270             if (!str.isEmpty() && str != null) {
271                 fw.write(str + "\n");
272             }
273         }
274         fw.close();
275     }
276 
277     public synchronized void saveProcessedToFile() throws
    Exception {
278         saveProcessedToFile(this.processedOrdersFileName);
279     }
280 
281 
282     // ------------------------------------------------
283     public synchronized File checkProcessedOrdersFile(String
    filename) throws Exception {
284         File f = new File(filename);
285         if (!f.exists()) {
286             f.createNewFile();
287         }
288         return f;
289     }
290 
291     public synchronized File checkDir(String dirName) throws
    Exception {
292         File f = new File(dirName);
293         if (!f.exists()) {
294             f.mkdir();
```

```
295         }
296         return f;
297     }
298 
299     public synchronized File checkProcessedOrdersFile() throws
    Exception {
300         return
    checkProcessedOrdersFile(this.processedOrdersFileName);
301     }
302 
303 
304     // ------------------------------------------------------
305     private synchronized static String getFileExtension(File file)
    {
306         String name = file.getName();
307         int lastIndexOf = name.lastIndexOf(".");
308         if (lastIndexOf == -1) {
309             return ""; // empty extension
310         }
311         return name.substring(lastIndexOf);
312     }
313 
314     public synchronized void clearProducts() {
315         products.clear();
316     }
317 
318     public synchronized void clearProcessed() {
319         processed.clear();
320     }
321 
322 
323 }
324 
```