```java
package model;

import java.io.File;

public class Engine
{
    static final double HST_RATE = 0.13;
    static final double MIN_SHIPPING_WAIVER = 70;
    static final double SHIPPING_COST = 5;
    static Engine engine = null;

    private ItemDAO itemDAO;
    private String xmlPOFolderPath;
    private String xmlPOProcessedFolderPath;
    private int count = 0;

    private Engine() throws Exception
    {
        itemDAO = ItemDAO.getInstance();
        initializeFilePath();
        initializeCount();
    }
    public synchronized static Engine getInstance() throws Exception
    {
        try
        {
            if (engine == null)
                engine = new Engine();
            return engine;
        }
        catch (Exception e)
        {
            throw new Exception ("Issue at instatiating engine", e);
        }
    }
    public List<ItemBean> getItems(String catalog) throws Exception
    {
        try
        {
            return itemDAO.getItems(catalog, null);
        }
        catch (Exception e)
        {
            throw new Exception ("Fail to get items", e);
        }
    }
    public List<ItemBean> getItems(String catalog, String orderBy) throws
  Exception
```

```
57      {
58          try
59          {
60              return itemDAO.getItems(catalog, orderBy);
61          }
62          catch (Exception e)
63          {
64              throw new Exception ("Fail to get items", e);
65          }
66      }
67
68      public List<ItemBean> searchItem(String number) throws Exception
69      {
70          List<ItemBean> result = new ArrayList<ItemBean>();
71          result.add(getItem(number));
72          //TODO
73          return result;
74      }
75      public ItemBean getItem(String number) throws Exception
76      {
77          try
78          {
79              return itemDAO.getItem(number);
80          }
81          catch (Exception e)
82          {
83              throw new Exception("Fail to find item of number " + number);
84          }
85      }
86
87      public List<CatalogBean> getCatalogs() throws Exception
88      {
89          try
90          {
91              return itemDAO.getCatalogs();
92          }
93          catch (Exception e)
94          {
95              throw new Exception ("Fail to get catalogs",e);
96          }
97      }
98
99      //return the 'existing item' in the stact if found, else return null;
100     private ItemBean isExistingItem(List<ItemBean> items, String number)
101     {
102         ItemBean item = null;
103         for(int i = 0; i < items.size() && item == null; i++)
104         {
```

```
105             if (items.get(i).getNumber().equals(number))
106                 item = items.get(i);
107         }
108         return item;
109     }
110 
111     private void updateOrderTotalPrice(OrderBean order)
112     {
113         List<ItemBean> items = order.getItems();
114         double total = 0;
115         for(ItemBean item: items)
116         {
117             total = total + (item.getPrice() * item.getQuantity());
118         }
119         order.setTotal(total);
120         order.setHST(total*HST_RATE);
121 
122         //set shipping cost
123         if (total == 0 || total > MIN_SHIPPING_WAIVER )
124             order.setShipping(0);
125         else
126             order.setShipping(SHIPPING_COST);
127 
128         order.setGrandTotal(order.getTotal() + order.getHST() +
    order.getShipping());
129     }
130 
131     public OrderBean orderAddItem(OrderBean order, String number, String
    qty) throws Exception
132     {
133         int quantity;
134         try
135         {
136             quantity = Integer.parseInt(qty);
137         }
138         catch(Exception e)
139         {
140             throw new Exception("Invalid quantity entries " + qty);
141         }
142 
143         try
144         {
145             ItemBean item = isExistingItem(order.getItems(), number);
146             if(item != null)
147             {
148                 item.setQuantity(item.getQuantity() + quantity);
149             }
150             else
```

```
151             {
152                 item = this.getItem(number);
153                 item.setQuantity(quantity);
154                 order.getItems().add(item);
155             }
156             updateOrderTotalPrice(order);
157             return order;
158 
159         }
160         catch (Exception e)
161         {
162             throw new Exception("Fail to add "+ qty +" of item number " +
    number + " into order");
163         }
164     }
165 
166     private void orderDelItems(OrderBean order, List<String> toDelete)
167     {
168         List<ItemBean> items = order.getItems();
169         for (String e: toDelete)
170         {
171             ItemBean item = isExistingItem(items, e);
172             if(item !=null)
173                 items.remove(isExistingItem(order.getItems(), e));
174         }
175     }
176     private void orderUpdateQty (OrderBean order, Map<String, String>
    itemsQty) throws Exception
177     {
178         List<ItemBean> items = order.getItems();
179         Set<String> toUpdate = itemsQty.keySet();
180         for(String itemNo: toUpdate)
181         {
182             ItemBean item = isExistingItem(items, itemNo);
183             if (item!=null)
184             {
185                 int quantity;
186                 try
187                 {
188                     quantity = Integer.parseInt(itemsQty.get(itemNo));
189                     if (quantity < 0 )
190                         throw new Exception ();
191                 }
192                 catch (Exception e)
193                 {
194                     throw new Exception ("Invalid quantity entries!");
195                 }
196                 if (quantity == 0)
```

```
197                     items.remove(item);
198                 item.setQuantity(quantity);
199             }
200         }
201     }
202 
203     public OrderBean updateOrder(OrderBean order, Map<String, String>
    itemsQty, List<String> toDelete) throws Exception
204     {
205         orderDelItems(order, toDelete);
206         orderUpdateQty(order, itemsQty);
207         updateOrderTotalPrice(order);
208         return order;
209     }
210 
211      public  void jaxbObjectToXML(OrderBean order, String id, String path)
    throws Exception
212      {
213                 //Create JAXB Context
214                 JAXBContext jaxbContext = JAXBContext.newInstance
    (OrderBean.class);
215 
216                 //Create Marshaller
217                 Marshaller jaxbMarshaller = jaxbContext.createMarshaller();
218 
219                 //Required formatting??
220                 jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
    Boolean.TRUE);
221 
222                //Store XML to File
223                 File filePath = new File(path + String.format("%s.xml", id));
224                 if (!filePath.exists()) {
225                         filePath.createNewFile();
226                 }
227 
228                 //Writes XML file to file-system
229                 jaxbMarshaller.marshal(order, filePath);
230       }
231 
232 
233     public OrderBean convertFromXMLFileToObject(File file,String user) {
234         JAXBContext jaxbContext;
235         try
236         {
237             jaxbContext = JAXBContext.newInstance(OrderBean.class);
238             Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();
239             OrderBean order = (OrderBean) jaxbUnmarshaller.unmarshal(file);
240 
```

```
241             if (order.getAccount() != null && order.getAccount().equals(user))
242                 return order;
243         }
244         catch (JAXBException e)
245         {
246             e.printStackTrace();
247         }
248 
249         return null;
250 
251     }
252     public ArrayList<String> getXMLLinks(String user, String path) {
253         ArrayList<String> names = new ArrayList<>();
254         File[] files = new File(path).listFiles();
255 
256         for (File file : files) {
257             if (!file.isDirectory() && file.getName().contains(".xml")) {
258 
259                   OrderBean order = convertFromXMLFileToObject(file, user);
260                   if (order != null)
261                   {
262                       String[] f = file.getName().split(".xml");
263                       names.add(f[0]);
264                   }
265             }
266         }
267         return names;
268     }
269 
270 
271     public String getXmlPOProcessedFolderPath() {
272         return xmlPOProcessedFolderPath;
273     }
274 
275     public String getXmlFolderPath() {
276         return xmlPOFolderPath;
277     }
278 
279     public int increment() {
280         count++;
281         return count;
282     }
283 
284 
285 
286     private void initializeFilePath() {
287 
288         if (xmlPOFolderPath == null) {
```

```
289             xmlPOFolderPath = System.getProperty("user.dir") + "/appData/
    PO/";
290             File filePath = new File(xmlPOFolderPath);
291             //Create folder if they don't exist
292             if (!filePath.exists()) {
293                 try{
294                     if (filePath.getParentFile().exists())
295                         filePath.getParentFile().mkdirs();
296 
297                     filePath.mkdirs();
298                 }
299 
300                 catch(SecurityException se){
301                     //handle it
302                 }
303             }
304         }
305 
306         if (xmlPOProcessedFolderPath == null) {
307             xmlPOProcessedFolderPath = System.getProperty("user.dir") + "/
    appData/PO_processed/";
308             File filePath = new File(xmlPOProcessedFolderPath);
309             if (!filePath.exists())
310                     filePath.mkdirs();
311 
312         }
313     }
314 
315      private void initializeCount() {
316 
317         File[] files = new File(xmlPOFolderPath).listFiles();
318 
319         for (File file : files) {
320             if (!file.isDirectory() && file.getName().contains(".xml")) {
321                     String[] f = file.getName().split(".xml");
322                     int num = Integer.parseInt(f[0].split("_")[1]);
323                     if (num > count)
324                         count = num;
325             }
326         }
327 
328         files = new File(xmlPOProcessedFolderPath).listFiles();
329         for (File file : files) {
330             if (!file.isDirectory() && file.getName().contains(".xml")) {
331                     String[] f = file.getName().split(".xml");
332                     int num = Integer.parseInt(f[0].split("_")[1]);
333                     if (num > count)
334                         count = num;
```

```
            }
        }
    }

}
```