```java
package model;

import java.io.FileWriter;

public class Model {
    private static Model instance = null;
    private static ItemDAO itemDao;
    private static CategoryDAO catDao;

    private Model() {
        try {
            itemDao = new ItemDAO();
            catDao = new CategoryDAO();
        } catch(Exception e) {
            System.out.println("DB ERROR: " + e.getMessage());
            e.printStackTrace();
        }
    }

    public static Model getInstance() {
        if (instance == null) {
            instance = new Model();
        }
        return instance;
    }

    public List<Item> getFoods() throws Exception {
        return getFoods("0");
    }

    public List<Item> getFoods(String limit) throws Exception {
        int l = Integer.parseInt(limit);
        return itemDao.getAll(l);
    }

    public Item getFood(String val) throws Exception {
        return this.getFoodBy("id", val);
    }

    public Item getFoodBy(String by, String val) throws Exception
{
        return this.getFoodBy(by, val, false);
    }

```

```
60      public Item getFoodBy(String by, String val,
61              Boolean like) throws Exception {
62          return itemDao.findOneBy(by, val, like);
63      }
64
65      public List<Item> getFoodsByMultiple(String val) throws
    Exception {
66          return itemDao.getAllByMultiple(val, true, null);
67      }
68
69      public List<Item> getFoodsBy(String by,
70              String val) throws Exception {
71          return itemDao.getAllBy(by, val, true);
72      }
73
74      public List<Item> getFoodsBy(String by,
75              String val, Boolean like) throws Exception {
76          return getFoodsBy(by, val, like, "0");
77      }
78
79      public List<Item> getFoodsBy(String by, String val,
80              Boolean like, String limit) throws Exception {
81          int l = Integer.parseInt(limit);
82          return itemDao.getAllBy(by, val, like, l);
83      }
84
85      public List<Category> getCategories() throws Exception {
86          return catDao.getAll();
87      }
88
89      public Category getCategory(String cat) throws Exception {
90          return catDao.findOne(cat);
91      }
92
93      public Map<String, List<Item>> getCatNameWithFoods() throws
    Exception {
94          Map<String, List<Item>> result = new HashMap<String,
    List<Item>>();
95
96          for (Category cat: this.getCategories()) {
97              List<Item> items = new ArrayList<Item>();
98              for (Item item: this.getFoods()) {
99                  if (cat.getId() == item.getCatID()) {
100                     items.add(item);
```

```
101                     result.put(cat.getName(), items);
102                 }
103             }
104         }
105 
106         return result;
107     }
108 
109     public Map<Category, List<Item>> getCatsWithFoods() throws
    Exception {
110         Map<Category, List<Item>> result = new HashMap<Category,
    List<Item>>();
111 
112         for (Category cat: this.getCategories()) {
113             List<Item> items = new ArrayList<Item>();
114             for (Item item: this.getFoods()) {
115                 if (cat.getId() == item.getCatID()) {
116                     items.add(item);
117                     result.put(cat, items);
118                 }
119             }
120         }
121 
122         return result;
123     }
124 
125     public synchronized StringWriter createPO(String filePath,
126             int orderNum, String xslFilename,
127             Order order, Account user) throws Exception
128     {
129         OrderDAO orderDao = new OrderDAO(filePath);
130         String fileName =
    orderDao.getOrderFileName(user.getUsername(), orderNum);
131 
132         JAXBContext jaxbContext =
    JAXBContext.newInstance(order.getClass());
133         Marshaller marshaller = jaxbContext.createMarshaller();
134         marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
    Boolean.TRUE);
135         marshaller.setProperty(Marshaller.JAXB_FRAGMENT,
    Boolean.TRUE);
136 
137         StringWriter sw = new StringWriter();
138         sw.write("<?xml version=\"1.0\" encoding=\"UTF-8\"
```

```
    standalone=\"yes\"?>");
139         sw.write("<?xml-stylesheet type=\"text/xsl\" href=\"" +
    xslFilename + "\"?>\n");
140 
141         marshaller.marshal(order, new StreamResult(sw));
142         FileWriter orderFileWrite =
    orderDao.getFileWriter(fileName);
143         orderFileWrite.write(sw.toString());
144         orderFileWrite.close();
145 
146         return sw;
147     }
148 
149 
150     public static void main(String[] args) {
151         Model model = Model.getInstance();
152         try {
153             System.out.println(model.getCategories().toString());
154             System.out.println(model.getFoods().toString());
155         } catch(Exception e) {
156             System.out.println("ERROR: " + e.getMessage());
157         }
158     }
159 }
160 
```