

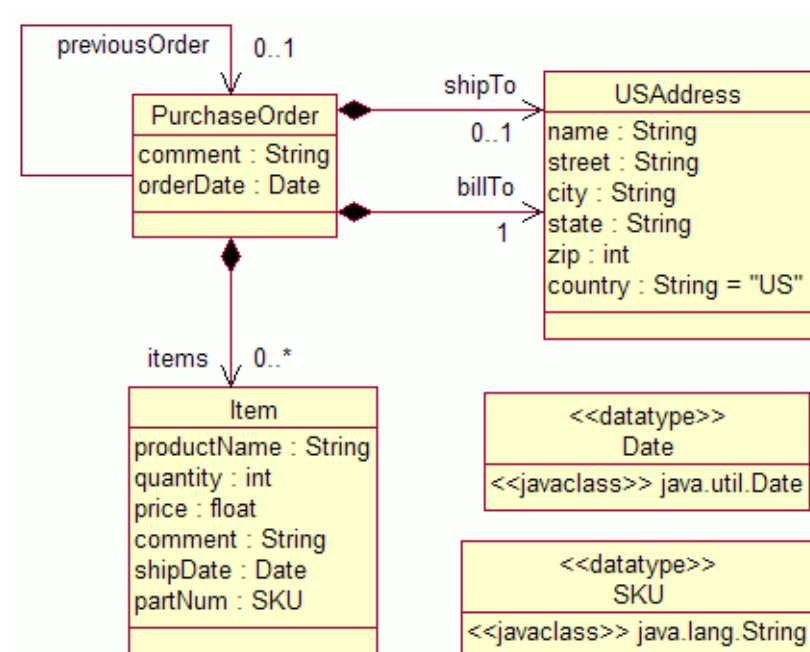
## Exercise 2: The Static API: Creating, Loading, Saving

### What This Exercise Is About

Using the Purchase Order model you generated in [Exercise 1](#), you will now load a model instance and make changes to it programmatically.

In [Exercise 1](#), we generated an editor that integrates into the Eclipse IDE. EMF can also be used in standalone applications, headless Eclipse applications, and RCP (Rich Client Platform) applications. In all cases, you can use the same APIs to load, save, and modify your data. In this exercise, you will work on a simple, standalone application, with no dependency on the Eclipse platform.

For your reference, here again is the UML class diagram describing the purchase order model:



### What You Should Be Able To Do

At the end of the lab, you should be able to:

- Create and configure a Java project for standalone development with EMF
- Programmatically create an instance of a generated model
- Modify items in a model instance
- Load and save model instance data

### Required Materials

- Eclipse 3.2
- Eclipse Modeling Framework (EMF) 2.2

### Exercise Instructions

This exercise is carried out entirely using the Eclipse Software Development Kit (SDK) version 3.2 with the Eclipse Modeling Framework (EMF) 2.2 installed into it. The exercise instructions refer to this product as either Eclipse or as "the workbench."

In your workspace, there should be a [EMF\\_Workshop/Exercise2\\_Create\\_Load\\_Save](#) folder that contains these instructions, as well as the following three files:

- [CreatePOInstance.java](#) shows how to create a model instance programmatically. Review it to get a feel for what's required to perform this step. Note that the code does not show how to **save** the generated instance, which we will be doing in this exercise.
- [po.xml](#) contains the instance data that would be generated from [CreatePOInstance.java](#), if that data was then serialized to a file. This will be your starting point for this exercise. We will first load this instance, then change it, and save the modified data to a second file.
- [Exercise2.java](#) is the class you will modify to perform the work noted above.

The solutions to this exercise can be installed using the cheat sheet.

---

## Directions

### Step A: Create & Configure a Java Project

1. Switch to the [Java Development](#) perspective, if not already there.
  - a. Select [Window -> Open Perspective -> Java](#).

**Cheat** Step 2 can be skipped using the Cheatsheet steps [Import required EMF plugins](#) and [Import required XSD plugin](#).


2. Import the required EMF dependencies to the workspace as linked binary projects.
  - a. Right-click in the Package Explorer and select [Import...](#) (or select [File -> Import/Export...](#)).
  - b. Select [Plug-in Development -> Plug-ins and Fragments](#). Hit [Next](#).
  - c. Select [The target platform \(as specified in Preferences\)](#).
  - d. Select [Select from all plug-ins and fragments found at the specified location](#).
  - e. Select [Binary projects with linked content](#). Hit [Next](#).
  - f. Using the text field or by scrolling in the left-hand list, search for:
    - [org.eclipse.emf.common](#)
    - [org.eclipse.emf.ecore](#)
    - [org.eclipse.emf.ecore.change](#)
    - [org.eclipse.emf.ecore.xml](#)
    - [org.eclipse.xsd](#)
  - g. These plugins contain the runtime EMF libraries we'll need in this and following exercises. Select them, then hit [Add -->](#) to add them to the right-hand list. Hit [Finish](#).

**Cheat** Steps 3 to 8 can be skipped using the Cheatsheet step [Create Exercises project](#).

3. Create a new Java Project, called [Exercises](#), with a separate source and output folders called [src](#) and [bin](#). Add the EMF dependencies and the existing [com.example.po](#) project, which contains the generated model, to the new project's build path.
  - a. Right-click in the Package Explorer, and select [New -> Project...](#), or just hit [CTRL-N](#).
  - b. Select [Java Project](#) and Hit [Next](#).
  - c. Name the project [Exercises](#).
  - d. Select the radio button for [Create separate source and output folders](#).
  - e. Click [Configure default...](#)

- f. Under **Source and output folder**, select **Folders**. **Source folder name** and **Output folder name** should be set to **src** and **bin**, respectively. Click **OK**. Hit **Next**.
  - g. Switch to the **Projects** tab, and click **Add...**
  - h. Select the following projects:
    - **com.example.po**
    - **org.eclipse.emf.common**
    - **org.eclipse.emf.ecore**
    - **org.eclipse.emf.ecore.change**
    - **org.eclipse.emf.ecore.xml**
    - **org.eclipse.xsd**
  - i. Click **OK**, and then **Finish** to close the wizard.
4. Review the classpath file for the Exercises project.
    - a. Eclipse stores a project's build path information in a **.classpath** file. To see it, switch to the Navigator view (**Window -> Show View -> Navigator**), then open the **Exercises** project folder.
    - b. Open the **.classpath** file to see what Eclipse has generated for you. Eclipse uses this information to ensure that the correct **.class** and **.jar** files are on the classpath when compiling the project.
    - c. Switch back to the Package Explorer view.
  5. Create a **data** folder in the root of the Exercises project.
    - a. Select the root of the Exercises project in the Package Explorer.
    - b. Right-click and select **New -> Folder**, or just hit **CTRL-N** and browse for **General -> Folder**.
    - c. Set the name to be **data**. Hit **Finish**.
  6. Copy **po.xml** from the Exercise 2 source folder to the **Exercises/data** folder.
    - a. Select **po.xml**, then hit **CTRL-C** to copy (or **Edit -> Copy**).
    - b. Select the new data folder created in the last step, **Exercises/data**, and hit **CTRL-V** to paste (or **Edit -> Paste**).
  7. Create a package under the src folder in the Exercises project, called **exercises**.
    - a. Select the src folder under Exercises in the Package Explorer.
    - b. Right-click and select **New -> Package**, or just hit **CTRL-N** and browse for **Java -> Package**.
    - c. Set the name to be **exercises**. Hit **Finish**.
  8. Copy java code from the Exercise 2 source folder into **Exercises/src/exercises**.
    - a. Select **CreatePOInstance.java** and **Exercise2.java**, then hit **CTRL-C** to copy (or **Edit -> Copy**).
    - b. Select the new package created in the last step, **Exercises/src/exercises**, and hit **CTRL-V** to paste (or **Edit -> Paste**).
- 

## Step B: Load the Model Instance

1. Switch to the **Java Development** perspective, if not already there.
  - a. Select **Window -> Open Perspective -> Java**.
2.  Open the file **Exercise2.java**.
3. Work your way through the loading portion of the exercise by filling in the missing code for each of the first six TODO comments.
  - a. The last TODO in this step is to print out the shipTo address data.
  - b. You can activate Eclipse's code completion by pressing **CTRL-SPACE**. Selecting a class or member name and then pressing **F2** will display the Javadoc; **F3** will jump to the declaration. These shortcuts provide a good way to explore the APIs for EMF and your model.
  - c. If you run into trouble or are pressed for time, you can copy code from the **JPages** folder under the Exercise 2 source folder.

4. Run your code. There are several ways to do this.
  - o Right-click in the body of the Java file, and select **Run As -> Java Application**.
  - o Select from the **Run** menu **Run As -> Java Application**.
  - o Click the Run button in the toolbar (a green circle with a white triangle in it, or a "play" button) and select **Run As -> Java Application**.
  - o Hit **SHIFT-ALT-X** (eXecute). Wait one second for the context menu to open, then then hit **J** (Java Application).

You should see the following output:

```
name: Scarlet O'Hara
street: 321 Backwoods Lane
city: Louisville
state: AL
zip: 67655
```

---

### Step C: Modify the Data

1. The next two TODOs describe a change that you should make to the data. Fill in the missing code. To remove an item according to its part number, you'll need to iterate through all of the items in the purchase order, checking that attribute.
  2. You can try running the application again, though no additional output will be produced.
- 

### Step E: Save the Modified Instance

1. Fill in the code for the final TODO, to put the updated purchase order in a new resource and serialize it to the console and a new file.
2. Run again from the Run menu, the Run button, a right-click, or hit **CTRL-F11**. You should see the following output:

```
name: Scarlet O'Hara
street: 321 Backwoods Lane
city: Louisville
state: AL
zip: 67655
<?xml version="1.0" encoding="UTF-8"?>
<po:order xmlns:po="http://www.example.com/po" orderDate="2006-03-20">
  <shipTo>
    <name>Scarlet O'Hara</name>
    <street>321 Backwoods Lane</street>
    <city>Louisville</city>
    <state>AL</state>
    <zip>67655</zip>
  </shipTo>
  <billTo>
    <name>Rhett Butler</name>
    <street>123 Iditarod Lane</street>
    <city>Nome</city>
    <state>AK</state>
    <zip>34582</zip>
  </billTo>
  <po:comment>Plasma Television (STV999876) is discontinued and
  has been removed from the order.</po:comment>
  <items partNum="SWH123456">
    <productName>Wireless Headphones</productName>
    <quantity>2</quantity>
    <price>75.0</price>
```

```
<po:comment>Backordered</po:comment>
<shipDate>2006-03-23</shipDate>
</items>
</po:order>
```

3. Compare the contents of the new `po-updated.xml` file to the above console output.
    - a. Select the `data` folder in the Package Explorer. Right click and select **Refresh** from the pop-up menu, or just hit **F5**. This refreshes the workspace view of that folder based on the state of the actual file system.
    - b. Open the new `po-updated.xml` file in the text editor.
- 


### Step D: [Optional] Run Code On The Commandline (No Eclipse)

1. If you need to know how to run an Eclipse launch configuration without Eclipse, follow these steps:
    - a. Switch to the Debug perspective (**Window -> Open Perspective -> Debug**).
    - b. Right-click the last entry in the Debug view, and select Properties.
    - c. If you execute the given command line from the Exercises folder (or save to a `.bat` or `.sh` file in the root of the Exercises project folder), you will produce the same `data/po-updated.xml` file as in the preceding step.
    - d. Under Windows, you may not get console output unless you change `javaw` to `java`.
    - e. Under Linux, you may have to set execute permission on the script file before you can execute it.
- 

### Step E: [Optional] Open the Updated Model Instance in the PurchaseOrder Model Editor

1. If you want to open the `po-updated.xml` file in your PurchaseOrder model editor, launch your second workbench, then copy the file into its workspace.
  2. Rename it to have a `.po` extension so that when double-clicked, Eclipse will automatically open it in your model editor.
- 

### Step F: [Optional] Update CreatePOInstance.java to Serialize the In-Memory Instance

1.  Open the file `CreatePOInstance.java`.
  2. `CreatePOInstance.java` shows how to create an instance of a purchase order in memory, but does not show how to serialize that data to a file. If you would like to do so, use what you've learned above to implement a `savePOData()` method that saves the purchase order to `po.xml`.
  3. The solution is available in [EMF\\_Workshop/Solution2\\_Optional/Exercises/src/exercises/CreatePOInstance.java](#).
- 

## Summary

You have learned how to load, change, and save an instance of an EMF model programmatically. You created a standalone EMF application, with no Eclipse dependencies. Although you used Eclipse to manage your classpath, you then ran the resulting code without Eclipse, directly in the Java virtual machine. Finally, you opened the modified instance in your PurchaseOrder model editor to graphically review the changes that your application made.