

# Eclipse and Java: Introducing Persistence

## Companion Tutorial Document

Version 1.2 (October 26, 2007)

By Mark Dexter

### Table of Contents

Introduction & Setup.....	2
Tutorial Target Audience.....	2
Tutorial Objectives.....	2
Tutorial Approach.....	3
Getting The Most From This Tutorial.....	3
Downloading and Installing Eclipse.....	3
Lesson Outlines.....	4
Lesson 1 .....	4
Lesson 2 .....	4
Lesson 3.....	5
Lesson 4.....	5
Lesson 5.....	5
Lesson 6.....	5
Lesson 7.....	5
Lesson 8.....	6
Lesson 9.....	6
Lesson 10.....	6
Lesson 11.....	6
Lesson 12.....	6
Additional Resources.....	7
Lesson-Specific Resources.....	7
General Resources.....	7
Eclipse Websites.....	7
Java Websites.....	7
Java Books.....	8
Alphabetical Index by Lesson.....	8
Code Snapshots.....	11
Lesson 1 – Scrapbook.....	11
Lesson 2 – AllTests.....	11
Lesson 2 – MyUtilitiesTest (in process).....	11
Lesson 3 – MyUtilitiesTest (test saveStringToFile).....	12
Lesson 3 – MyUtilities (Auto-generated method stubs from Quick Fix).....	12
Lesson 4 – MyUtilities (saveStringToFile complete).....	13
Lesson 4 – Optional Scrapbook Examples.....	13
Lesson 5 – MyUtilities (saveStringToFile() and getStringFromFile() complete).....	14
Lesson 6 – MyUtilitiesTest (with createMyLibrary() method).....	15
Lesson 7 – MyUtilitiesTest (with convertToXML() method).....	16
Lesson 7 – MyUtilities (with completed convertToXML() and convertFromXML() methods).....	18

# Eclipse and Java: Introducing Persistence Tutorial Companion Document

Lesson 8 – MyUtilities (with completed saveMyLibraryToXMLFile() and getMyLibraryFromXMLFile() methods).....	19
Lesson 8 – MyUtilitiesTest (with completed saveToXMLFile() method).....	20
Lesson 9 – MyUtilities (convertToXML() and convertFromXML() methods modified).....	22
Lesson 10 – manifest.txt file.....	24
Lesson 10 – MyLibrary (with updated main() method).....	24
Lesson 11 – Final MyUtilitiesTest (with saveToSerialFile() method).....	27
Lesson 11 – MyUtilities (with saveMyLibraryToSerialFile() method).....	29
Lesson 12 – Final Book Class.....	31
Lesson 12 – Final MyLibrary Class.....	32
Lesson 12 – Final MyUtilities Class.....	35
Lesson 12 – Final Person Class.....	37
Lesson 12 – Final AllTests Class.....	38

## Introduction & Setup

This document is designed to accompany the “Eclipse And Java: Introducing Persistence” video tutorial, which is available at <http://eclipsetutorial.sourceforge.net/>. Before starting this tutorial, please download the following files from the website:

- Persistence-Tutorial-Companion-Document.PDF – this document.
- persistencetutorial.zip – zip file for importing the Eclipse PersistenceTutorial project. This is done as part of Lesson 1.
- persistence-lesson01.zip through persistence-lesson12.zip – zip files for the 12 lessons in the tutorial

To play the lessons:

1. Unzip each lesson's zip file into a directory on your system.
2. Find the file called “lessonxx.html”, where xx is the lesson number 01-12.
3. Open this file with your browser (e.g., Internet Explorer or Firefox). The lesson should play inside the browser. Note that you need to have the Adobe Flash player installed on your system. This can be downloaded from Adobe at <http://www.adobe.com/products/flashplayer/>.

## Tutorial Target Audience

This tutorial is targeted for people who have complete the tutorial “Eclipse and Java for Total Beginners” or who have some familiarity with Eclipse and Java.

## Tutorial Objectives

The objectives of this tutorial are to:

- Demonstrate how to save Java objects to disk files, using either XML or object serialization
- Demonstrate how to use the JUnit 4.0 unit testing framework in Eclipse and how to convert from JUnit 3.8 to 4.0 in Eclipse

- Demonstrate time-saving Eclipse features related to Java development
- Demonstrate using Eclipse to explore the Java class libraries

## Tutorial Approach

The tutorial is organized around the following activities. We will start with the MyLibrary project which was created in the “Total Beginners” tutorial.

- Import the MyLibrary project
- Write methods to save and retrieve text files, to convert objects to XML strings, and to save and retrieve serialized objects
- Use the “test-first” approach to develop all methods.
- Write a “test drive” program and create an executable JAR file, and run the JAR file from the system console.

Concepts are introduced as needed during the development of the sample application.

## Getting The Most From This Tutorial

This tutorial can be used as an in-depth demonstration of persistence in Java. However, if you want to learn how to program in Java, the following approach is recommended:

- Have Eclipse installed and ready to go.
- Work side-by-side with the lessons, pausing and rewinding as needed.
- Use this guide as needed.
- Consult other resources as needed to understand the topics covered in greater depth.
- Keep a positive attitude!

## Downloading and Installing Eclipse

Before Installing Eclipse, you need to have the Java JDK (Java development kit) installed on your computer. This are available at <http://java.sun.com/javase/downloads/index.jsp>. IMPORTANT NOTE: You need to install Java version 1.5 or later to run the tutorial. Earlier versions do not support generics (used in the MyLibrary project) or the JUnit 4.0 unit test framework. Also, although you can run Eclipse with the JRE, we need the JDK for this tutorial, since we'll be using the attached Java source code.

Installing the JDK is reasonably simple. Detailed, step-by-step instructions, if needed, are available in the PDF Eclipse Tutorial at the <https://www.arctechsoftware.com/tutorial/welcomePage.do>. (Follow the link to “Beginning Eclipse”.)

This tutorial is based on Eclipse 3.3. Here are the steps to install Eclipse 3.3 from [www.eclipse.org](http://www.eclipse.org):

- Navigate to [www.eclipse.org/downloads](http://www.eclipse.org/downloads)
- Select “Eclipse IDE for Java Developers”. If your platform is Linux or MacOSX, be sure to

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

select the link to the right. Note that you can use “Eclipse IDE for Java EE Developers”, “Eclipse for RCP/Plug-in Developers”, or “Eclipse Classic” as well. All of these include the Java development portions of Eclipse used in this tutorial.

- On the [www.eclipse.org/downloads](http://www.eclipse.org/downloads) page, follow the link “Find out more”. Scroll your browser to display the far right-hand side of the screen to the column “Tutorials and Help”. The first tutorial is a Screencam tutorial that steps you through downloading and installing Eclipse on Windows.

The Eclipse installation is very straightforward. There is no installation program. Instead, you just create the top-level folder and then unzip the file inside this folder. In Windows XP, for example, just copy the zip file to your root directory (e.g., “C:\”) and then unzip the downloaded zip file. Note: You need to use an unzip program. Do not use the Windows File Explorer unzip. This will create a folder called “C:\eclipse”. The Eclipse programs will be created in several subfolders (configuration, features, plugins, readme). The procedure for Linux is similar, except you unzip the .tar.gz file.

Another option is to install the EasyEclipse distribution of Eclipse (<http://www.easyeclipse.org>). This is a free third-party distribution that includes a Windows installation program that installs Java and Eclipse all in one step. You should install version 1.3 (based on Eclipse 3.3) or later of the Expert Java or Desktop Java distributions.

## Lesson Outlines

### Lesson 1

- Create Java project in Eclipse using Import from archive file
- Learn how to export a project
- Review of MyLibrary classes
- Introduce try / catch blocks
- Create a Scrapbook page
- Run try / catch blocks inside Scrapbook
- Check Javadoc location and Source attachment

### Lesson 2

- Convert from JUnit 3 to JUnit 4
- Introduce Java Annotations
- Plan methods for converting objects to XML text files
- Discuss Static Methods
- Start test method for saveStringToFile() method

### Lesson 3

- Complete test method for saveStringToFile() method
- Discuss unit tests as class documentation in agile software development
- Start actual saveStringToFile() method

### Lesson 4

- Finish on saveStringToFile() method
- Discuss BufferedWriter and FileWriter classes
- Discuss exceptions and throws keyword

### Lesson 5

- Finish getStringFromFile() method
- Look at TODO and Tasks View
- Run MyUtilitiesTest JUnit test
- Read a Stack Trace

### Lesson 6

- Download and add XStream XML program library to our project
- Attach source code to Java language classes to our project
- Learn how to browse the Java source code
- Learn how to use the Eclipse Hierarchy view
- Discuss the benefits of XML format
- Write createMyLibrary() method in MyUtilitiesTest class

### Lesson 7

- Write test method for convertToXML() and convertFromXML() methods
- Create convertToXML() and convertFromXML() methods
- Look at XStream web tutorial
- Test conversion methods

## Lesson 8

- Write test method to test conversion to and from XML disk file
- Create `saveMyLibraryToXMLFile()` and `getMyLibraryFromXMLFile()` methods
- Test our methods

## Lesson 9

- Use Eclipse XML editor to examine XML file
- Explore XML object reference options in XStream
- Use the Eclipse Compare With Local History feature to compare versions of XML files

## Lesson 10

- Use History View and Compare With Local for Java Source
- Modify `MyLibrary` main method and run as Java application in Eclipse
- Learn about Java class and JAR files
- Create JAR manifest file with Class-Path
- Set Java compiler compliance level
- Create JAR file in Eclipse
- Execute JAR file in Windows and Linux

## Lesson 11

- Overview of Java Object Serialization
- Discuss the pros and cons of object serialization and XML
- Create test method for `saveMyLibraryToSerialFile()` and `getMyLibraryFromSerialFile()` methods
- Write `saveMyLibraryToSerialFile()` method

## Lesson 12

- Write `getMyLibraryToSerialFile()` method
- Add `Serializable` interface and `serialVersionUID` to classes
- Modify and run `AllTests` to test the entire application
- Look at Java Compiler Errors/Warning options

## Additional Resources

### Lesson-Specific Resources

This section lists resources that relate specifically to topics covered in different lessons.

#### Lesson 2

- <http://today.java.net/pub/a/today/2006/12/07/junit-reloaded.html>. JUnit Reloaded article that compares JUnit 4 to JUnit 3 with simple examples.
- Annotations are discussed in Sun Java Tutorials at <http://java.sun.com/docs/books/tutorial/java/javaOO/annotations.html>.
- For a more in-depth discussion of Java annotations, see “Thinking in Java”, by Bruce Eckel.

#### Lesson 6

- <http://xstream.codehaus.org/>. XStream XML library web site.

#### Lessons 7 - 9

- <http://xstream.codehaus.org/tutorial.html>. XStream XML library web site “2-Minute Tutorial”.

### General Resources

There are many general resources available for learning more about Eclipse and Java. These are just a few that I’ve found helpful.

### Eclipse Websites

- [www.eclipse.org/resources](http://www.eclipse.org/resources). This lists a number of articles, books, presentations, demonstrations and other resources on a variety of topics related to Eclipse.
- eclipse.newcomer newsgroup. This is a friendly, active newsgroup where newcomers to Eclipse can ask questions. The search feature of this and other newsgroups can be especially valuable, since there is a good chance that your question has already been asked and answered.
- Beginning Eclipse Tutorial on ArcTech Software LLC website. Written tutorial to help you install Eclipse and Java. Login required to download. It has a very good section on downloading and installing the Java JDK. Link to tutorial is <https://www.arctechsoftware.com/tutorial/tutorial.do?subcatId=1>. Link to home page is <https://www.arctechsoftware.com/tutorial/welcomePage.do>.

### Java Websites

- The Java Tutorials from Sun (<http://java.sun.com/docs/books/tutorial/java/index.html>). The gold standard for learning Java, and it’s free.

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

- JavaRanch Big Moose Saloon web site (<http://saloon.javaranch.com/cgi-bin/ubb/ultimatebb.cgi?category=1>). This has a variety of forums, including Java in General (beginner), Java in General (intermediate), and many other Java topics. Very active and friendly, with knowledgeable moderators.
- The Java Developers Almanac 1.4 (<http://www.exampledepot.com/>). Contains Java code samples for many topics.

### Java Books

- Head First Java, by Kathy Sierra & Bert Bates. Excellent, fun, creative book for Java and OOP beginners.
- Thinking In Java, by Bruce Eckel. Excellent, thorough reference for Java. For all levels of programmer.
- Effective Java, by Joshua Bloch. Concise book documents specific recommendations for Java best practices. For intermediate to advanced programmers.

### Alphabetical Index by Lesson

Topic	Lesson
agile software development.....	3
AllTests class.....	1, 2
Alt-Left Arrow.....	6
Alt-Right Arrow.....	6
annotations.....	2
assertFalse.....	3
assertTrue.....	3
Back arrow.....	6, 9
BufferedInputStream.....	12
BufferedOutputStream class.....	11
BufferedReader.....	5
BufferedWriter.....	4
Build Path.....	6
casting operator.....	7
Class-Path in JAR file.....	10
Compare current with local.....	10
Compare With Local History.....	9, 10
Compiler compliance level.....	10
compiler errors – identifying in Eclipse.....	7
compiler preferences for errors / warnings.....	12
compiler warnings.....	12
convertFromXML() method.....	7
convertToXML() method.....	7

Eclipse and Java: Introducing Persistence  
Tutorial Companion Document

createMyLibrary() method.....	7
CTRL-/ (toggle in-line comment).....	4
CTRL-Shift-P – find matching bracket.....	5
CTRL-Z (undo).....	4
Declaration.....	6
Exception class.....	11
Executable JAR file.....	10
extreme programming.....	3
F3 Type Hierarchy.....	6
F4 Open Declaration.....	6
File class.....	3
FileInputStream.....	12
FileReader.....	5
FileWriter.....	4
finally block.....	4
Forward arrow.....	6
getMyLibraryFromSerialFile() method.....	12
getMyLibraryFromXMLFile() method.....	8
getStringFromFile().....	5
Hierarchy view.....	6
History View.....	10
import static.....	1, 2
Inline refactor command.....	11
instanceof operator.....	7
IOException.....	4
JAR file manifest.....	10
JAR file structure.....	10
java -jar command.....	10
Java annotations.....	1
Java Build Path.....	6
Java source.....	6
java.io package.....	4
JDK.....	6
JRE version.....	10
JUnit 3.....	1, 2
JUnit 4.....	1, 2
Local History Preferences.....	9
nested constructors.....	4
nested try / catch blocks.....	4
newline character.....	3
NotSerializableException.....	12
object references in XML.....	9
object serialization.....	11
ObjectInputStream.....	11, 12
ObjectInputStream class.....	11
ObjectOutputStream class.....	11

Eclipse and Java: Introducing Persistence  
Tutorial Companion Document

Open Run Dialog.....	1, 2
printStackTrace.....	5
Quick Fix.....	1, 2
readObject() method.....	11, 12
Refactor / Inline.....	11
Run as Java application.....	10
run configuration.....	2
run or launch configuration.....	1
run single JUnit test method.....	12
saveMyLibraryToSerialFile() method.....	11
saveMyLibraryToXMLFile() method.....	8
saveStringToFile().....	4
Serializable interface.....	12
serialization, object: pros and cons .....	11
serialVersionUID field.....	12
src.zip.....	6
Stack Trace output.....	5
static methods.....	1, 2
StringBuilder.....	5
System.out.println code assist.....	10
Tasks view.....	5
test-first development.....	1, 2
throws keyword.....	4
TODO comment.....	5
toggle line comments.....	4
Type Hierarchy.....	6
undo.....	4
while loop.....	5
writeObject() method.....	11
XML editor – in Eclipse.....	9
XML: pros and cons.....	11
XStream Java library.....	6
XStream program library.....	7
XStream program options.....	9
XStream tutorial.....	7
xstream.fromXML() method.....	7
xstream.toXML() method.....	7
@Test.....	1, 2

## Code Snapshots

The following pages contain code snapshots as of the end of each lesson. Use these to check your code or to help you fix any problems you might have. If needed, you can copy and past this code into your Eclipse Java source files. Also, if you want to start the tutorial in the middle, these can help you catch up to the correct point.

### Lesson 1 – Scrapbook

```
long x = 1;
try {
    x = 1 / 0;
}
catch (ArithmeticException e) {
    x = 0;
    e.printStackTrace();
}
finally {
    System.out.println("This will print. x=" + x);
}
System.out.println("Life goes on...");
```

### Lesson 2 – AllTests

```
package org.persistence.tutorial;

import org.junit.runner.*;
import org.junit.runners.*;

@RunWith(Suite.class)
@Suite.SuiteClasses(value={
    PersonTest.class,
    BookTest.class,
    MyLibraryTest.class
})

public class AllTests {
}
```

### Lesson 2 – MyUtilitiesTest (in process)

```
package org.persistence.tutorial;
import org.junit.*;
import static org.junit.Assert.*;

public class MyUtilitiesTest {

    @Test
    public void saveStringToFile() {

    }

}
```

```
}
```

## Lesson 3 – MyUtilitiesTest (test saveStringToFile)

```
package org.persistence.tutorial;

import java.io.File;

import org.junit.*;
import static org.junit.Assert.*;

public class MyUtilitiesTest {

    @Test
    public void saveStringToFile() {
        String saveString = "this is test line one\n" +
            "this is test line two\n";

        File testFile = new File("testsavetostring.txt");
        testFile.delete();
        assertFalse("File should not exist",
            testFile.exists());

        assertTrue("File should have been saved",
            MyUtilities.saveStringToFile("testsavestring.txt",
                saveString));

        String newString = MyUtilities.getStringFromFile(
            "testsavestring.txt");
        assertTrue("Save and get strings should be equal",
            saveString.equals(newString));

        assertFalse("File should not be saved",
            MyUtilities.saveStringToFile(
                "non-existent directory/thisshouldfail.txt",
                saveString));

        String emptyString = MyUtilities.getStringFromFile(
            "badfilename.txt");
        assertTrue("String should be empty",
            emptyString.length() == 0);
    }
}
```

## Lesson 3 – MyUtilities (Auto-generated method stubs from Quick Fix)

```
package org.persistence.tutorial;

public class MyUtilities {

    public static boolean saveStringToFile(String string, String saveString) {
        // TODO Auto-generated method stub
        return false;
    }

    public static String getStringFromFile(String string) {
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
        // TODO Auto-generated method stub
        return null;
    }
}
```

### Lesson 4 – MyUtilities (saveStringToFile complete)

```
package org.persistence.tutorial;

import java.io.*;

public class MyUtilities {

    public static boolean saveStringToFile(String fileName,
        String saveString) {
        boolean saved = false;
        BufferedWriter bw = null;

        try {
            bw = new BufferedWriter(
                new FileWriter(fileName));

            try {
                bw.write(saveString);
                saved = true;
            }
            finally {
                bw.close();
            }
        }
        catch (IOException ex) {
            ex.printStackTrace();
        }
        return saved;
    }

    public static String getStringFromFile(String string) {
        // TODO Auto-generated method stub
        return null;
    }
}
```

### Lesson 4 – Optional Scrapbook Examples

Below are some examples of code to try in the scrapbook to demonstrate how nested try / catch blocks work. In the example below, there is an error in the outer try block. Nothing in the inner try / finally block will execute. Only the catch block will execute.

```
// example 1: error in outer try block
long x = 0;
try {
    x = 1 / 0;
    try {
        x = 1 / 1;
    }
}
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
    }  
    finally {  
        x = 1 / 1;  
        System.out.println("inner finally executes");  
    }  
}  
catch (Exception e) {  
    System.out.println("exception caught by outer block");  
}  
System.out.println("life goes on...");
```

Change the example as follows to put an error in the inner try block. Now, the inner finally block will execute, as well as the catch block.

```
/ example 2: error in inner try block  
long x = 0;  
try {  
    x = 1 / 1;  
    try {  
        x = 1 / 0;  
    }  
    finally {  
        x = 1 / 1;  
        System.out.println("inner finally executes");  
    }  
}  
catch (Exception e) {  
    System.out.println("exception caught by outer block");  
}  
System.out.println("life goes on...");
```

On your own, try putting the error in the finally block to see what happens. Also, try the example with no errors to prove that the finally block executes.

### Lesson 5 – MyUtilities (saveStringToFile() and getStringFromFile() complete)

```
package org.persistence.tutorial;  
  
import java.io.*;  
  
public class MyUtilities {  
  
    public static boolean saveStringToFile(String fileName,  
        String saveString) {  
        boolean saved = false;  
        BufferedWriter bw = null;  
  
        try {  
            bw = new BufferedWriter(  
                new FileWriter(fileName));  
  
            try {  
                bw.write(saveString);  
                saved = true;  
            }  
            finally {
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
        bw.close();
    }
    }
    catch (IOException ex) {
        ex.printStackTrace();
    }
    return saved;
}

public static String getStringFromFile(String fileName) {
    BufferedReader br = null;
    StringBuilder sb = new StringBuilder();

    try {
        br = new BufferedReader(
            new FileReader(fileName));
        try {
            String s;
            while ((s = br.readLine()) != null) {
                // add linefeed back since stripped by readLine()
                sb.append(s);
                sb.append("\n");
            }
        }
        finally {
            br.close();
        }
    }
    catch (IOException ex) {
        ex.printStackTrace();
    }
    return sb.toString();
}
}
```

### Lesson 6 – MyUtilitiesTest (with createMyLibrary() method)

```
package org.persistence.tutorial;

import java.io.File;

import org.junit.*;
import static org.junit.Assert.*;

public class MyUtilitiesTest {

    @Test
    public void saveStringToFile() {
        String saveString = "this is test line one\n" +
            "this is test line two\n";

        File testFile = new File("testsavetostring.txt");
        testFile.delete();
        assertFalse("File should not exist",
            testFile.exists());

        assertTrue("File should have been saved",
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
        MyUtilities.saveStringToFile("testsavestring.txt",
                                    saveString));

String newString = MyUtilities.getStringFromFile(
    "testsavestring.txt");
assertTrue("Save and get strings should be equal",
    saveString.equals(newString));

assertFalse("File should not be saved",
    MyUtilities.saveStringToFile(
        "non-existent directory/thisshouldfail.txt",
        saveString));

String emptyString = MyUtilities.getStringFromFile(
    "badfilename.txt");
assertTrue("String should be empty",
    emptyString.length() == 0);
}

public MyLibrary createMyLibrary() {
    Book b1;
    Book b2;
    Person p1;
    Person p2;
    MyLibrary ml;

    b1 = new Book("Book1");
    b2 = new Book("Book2");
    p1 = new Person();
    p1.setName("Fred");
    p2 = new Person();
    p2.setName("Sue");
    ml = new MyLibrary("Test");

    ml.addBook(b1);
    ml.addBook(b2);
    ml.addPerson(p1);
    ml.addPerson(p2);
    ml.checkOut(b1, p1);
    return ml;
}
}
```

### Lesson 7 – MyUtilitiesTest (with convertToXML() method)

```
package org.persistence.tutorial;

import java.io.File;

import org.junit.*;
import static org.junit.Assert.*;

public class MyUtilitiesTest {

    @Test
    public void saveStringToFile() {
        String saveString = "this is test line one\n" +
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
        "this is test line two\n";

File testFile = new File("testsavetostring.txt");
testFile.delete();
assertFalse("File should not exist",
           testFile.exists());

assertTrue("File should have been saved",
           MyUtilities.saveStringToFile("testsavestring.txt",
                                       saveString));

String newString = MyUtilities.getStringFromFile(
    "testsavestring.txt");
assertTrue("Save and get strings should be equal",
           saveString.equals(newString));

assertFalse("File should not be saved",
            MyUtilities.saveStringToFile(
                "non-existent directory/thisshouldfail.txt",
                saveString));

String emptyString = MyUtilities.getStringFromFile(
    "badfilename.txt");
assertTrue("String should be empty",
           emptyString.length() == 0);
}

public MyLibrary createMyLibrary() {
    Book b1;
    Book b2;
    Person p1;
    Person p2;
    MyLibrary ml;

    b1 = new Book("Book1");
    b2 = new Book("Book2");
    p1 = new Person();
    p1.setName("Fred");
    p2 = new Person();
    p2.setName("Sue");
    ml = new MyLibrary("Test");

    ml.addBook(b1);
    ml.addBook(b2);
    ml.addPerson(p1);
    ml.addPerson(p2);
    ml.checkOut(b1, p1);
    return ml;
}

@Test public void convertToXML() {
    MyLibrary startMyLibrary = createMyLibrary();
    String testXMLOut = MyUtilities.convertToXML(startMyLibrary);
    MyLibrary endMyLibrary =
        MyUtilities.convertFromXML(testXMLOut);
    assertEquals("Test", endMyLibrary.getName());
    assertEquals(2, endMyLibrary.getBooks().size());
    assertEquals(2, endMyLibrary.getPeople().size());
    assertEquals("Fred", endMyLibrary.getBooks().
        get(0).getPerson().getName());
}
}
```

```
}
```

## Lesson 7 – MyUtilities (with completed convertToXML() and convertFromXML() methods)

```
package org.persistence.tutorial;

import java.io.*;

import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.io.xml.DomDriver;

public class MyUtilities {

    public static boolean saveStringToFile(String fileName,
        String saveString) {
        boolean saved = false;
        BufferedWriter bw = null;

        try {
            bw = new BufferedWriter(
                new FileWriter(fileName));

            try {
                bw.write(saveString);
                saved = true;
            }
            finally {
                bw.close();
            }
        }
        catch (IOException ex) {
            ex.printStackTrace();
        }
        return saved;
    }

    public static String getStringFromFile(String fileName) {
        BufferedReader br = null;
        StringBuilder sb = new StringBuilder();

        try {
            br = new BufferedReader(
                new FileReader(fileName));

            try {
                String s;
                while ((s = br.readLine()) != null) {
                    // add linefeed back since stripped by readLine()
                    sb.append(s);
                    sb.append("\n");
                }
            }
            finally {
                br.close();
            }
        }
        catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
        return sb.toString();
    }

    public static String convertToXML(MyLibrary ml) {
        XStream xstream = new XStream(new DomDriver());
        return xstream.toXML(ml);
    }

    public static MyLibrary convertFromXML(String XMLString) {
        MyLibrary ml = null;
        XStream xstream = new XStream(new DomDriver());
        Object obj = xstream.fromXML(XMLString);
        if (obj instanceof MyLibrary) {
            ml = (MyLibrary) obj;
        }
        return ml;
    }
}
```

### Lesson 8 – MyUtilities (with completed saveMyLibraryToXMLFile() and getMyLibraryFromXMLFile() methods)

```
package org.persistence.tutorial;

import java.io.*;

import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.io.xml.DomDriver;

public class MyUtilities {

    public static boolean saveStringToFile(String fileName,
        String saveString) {
        boolean saved = false;
        BufferedWriter bw = null;

        try {
            bw = new BufferedWriter(
                new FileWriter(fileName));

            try {
                bw.write(saveString);
                saved = true;
            }
            finally {
                bw.close();
            }
        }
        catch (IOException ex) {
            ex.printStackTrace();
        }
        return saved;
    }

    public static String getStringFromFile(String fileName) {
        BufferedReader br = null;
        StringBuilder sb = new StringBuilder();
    }
}
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
        try {
            br = new BufferedReader(
                new FileReader(fileName));
            try {
                String s;
                while ((s = br.readLine()) != null) {
                    // add linefeed back since stripped by readLine()
                    sb.append(s);
                    sb.append("\n");
                }
            }
            finally {
                br.close();
            }
        }
        catch (IOException ex) {
            ex.printStackTrace();
        }
        return sb.toString();
    }

    public static String convertToXML(MyLibrary ml) {
        XStream xstream = new XStream(new DomDriver());
        return xstream.toXML(ml);
    }

    public static MyLibrary convertFromXML(String XMLString) {
        MyLibrary ml = null;
        XStream xstream = new XStream(new DomDriver());
        Object obj = xstream.fromXML(XMLString);
        if (obj instanceof MyLibrary) {
            ml = (MyLibrary) obj;
        }
        return ml;
    }

    public static boolean saveMyLibraryToXMLFile(String fileName,
        MyLibrary ml) {
        return saveStringToFile(fileName, convertToXML(ml));
    }

    public static MyLibrary getMyLibraryFromXMLFile(String fileName) {
        return convertFromXML(getStringFromFile(fileName));
    }
}
```

### Lesson 8 – MyUtilitiesTest (with completed saveToXMLFile() method)

```
package org.persistence.tutorial;

import java.io.File;
import org.junit.*;
import static org.junit.Assert.*;

public class MyUtilitiesTest {

    @Test
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
public void saveStringToFile() {
    String saveString = "this is test line one\n" +
        "this is test line two\n";

    File testFile = new File("testsavetostring.txt");
    testFile.delete();
    assertFalse("File should not exist",
        testFile.exists());

    assertTrue("File should have been saved",
        MyUtilities.saveStringToFile("testsavestring.txt",
            saveString));

    String newString = MyUtilities.getStringFromFile(
        "testsavestring.txt");
    assertTrue("Save and get strings should be equal",
        saveString.equals(newString));

    assertFalse("File should not be saved",
        MyUtilities.saveStringToFile(
            "non-existent directory/thisshouldfail.txt",
            saveString));

    String emptyString = MyUtilities.getStringFromFile(
        "badfilename.txt");
    assertTrue("String should be empty",
        emptyString.length() == 0);
}

public MyLibrary createMyLibrary() {
    Book b1;
    Book b2;
    Person p1;
    Person p2;
    MyLibrary ml;

    b1 = new Book("Book1");
    b2 = new Book("Book2");
    p1 = new Person();
    p1.setName("Fred");
    p2 = new Person();
    p2.setName("Sue");
    ml = new MyLibrary("Test");

    ml.addBook(b1);
    ml.addBook(b2);
    ml.addPerson(p1);
    ml.addPerson(p2);
    ml.checkOut(b1, p1);
    return ml;
}

@Test public void convertToXML() {
    MyLibrary startMyLibrary = createMyLibrary();
    String testXMLOut = MyUtilities.convertToXML(startMyLibrary);
    MyLibrary endMyLibrary =
        MyUtilities.convertFromXML(testXMLOut);
    assertEquals("Test", endMyLibrary.getName());
    assertEquals(2, endMyLibrary.getBooks().size());
    assertEquals(2, endMyLibrary.getPeople().size());
    assertEquals("Fred", endMyLibrary.getBooks().
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
        get(0).getPerson().getName());
    }

    @Test public void saveToXMLFile() {
        MyLibrary startMyLibrary = createMyLibrary();
        String fileName = "testmylibrary.xml";
        File testFile = new File(fileName);
        testFile.delete();
        assertFalse("File should not exist",
            testFile.exists());
        assertTrue("File should have been saved",
            MyUtilities.saveMyLibraryToXMLFile(
                fileName, startMyLibrary));

        MyLibrary endMyLibrary =
            MyUtilities.getMyLibraryFromXMLFile(fileName);
        assertEquals("Test", endMyLibrary.getName());
        assertEquals(2, endMyLibrary.getBooks().size());
        assertEquals(2, endMyLibrary.getPeople().size());
        assertEquals("Fred", endMyLibrary.getBooks().
            get(0).getPerson().getName());
    }
}
```

### Lesson 9 – MyUtilities (convertToXML() and convertFromXML() methods modified)

```
package org.persistence.tutorial;

import java.io.File;
import org.junit.*;
import static org.junit.Assert.*;

public class MyUtilitiesTest {

    @Test
    public void saveStringToFile() {
        String saveString = "this is test line one\n" +
            "this is test line two\n";

        File testFile = new File("testsavetostring.txt");
        testFile.delete();
        assertFalse("File should not exist",
            testFile.exists());

        assertTrue("File should have been saved",
            MyUtilities.saveStringToFile("testsavestring.txt",
                saveString));

        String newString = MyUtilities.getStringFromFile(
            "testsavestring.txt");
        assertTrue("Save and get strings should be equal",
            saveString.equals(newString));

        assertFalse("File should not be saved",
            MyUtilities.saveStringToFile(
                "non-existent directory/thisshouldfail.txt",
                saveString));
    }
}
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
String emptyString = MyUtilities.getStringFromFile(
    "badfilename.txt");
assertTrue("String should be empty",
    emptyString.length() == 0);
}

public MyLibrary createMyLibrary() {
    Book b1;
    Book b2;
    Person p1;
    Person p2;
    MyLibrary ml;

    b1 = new Book("Book1");
    b2 = new Book("Book2");
    p1 = new Person();
    p1.setName("Fred");
    p2 = new Person();
    p2.setName("Sue");
    ml = new MyLibrary("Test");

    ml.addBook(b1);
    ml.addBook(b2);
    ml.addPerson(p1);
    ml.addPerson(p2);
    ml.checkOut(b1, p1);
    return ml;
}

@Test public void convertToXML() {
    MyLibrary startMyLibrary = createMyLibrary();
    String testXMLOut = MyUtilities.convertToXML(startMyLibrary);
    MyLibrary endMyLibrary =
        MyUtilities.convertFromXML(testXMLOut);
    assertEquals("Test", endMyLibrary.getName());
    assertEquals(2, endMyLibrary.getBooks().size());
    assertEquals(2, endMyLibrary.getPeople().size());
    assertEquals("Fred", endMyLibrary.getBooks().
        get(0).getPerson().getName());
}

@Test public void saveToXMLFile() {
    MyLibrary startMyLibrary = createMyLibrary();
    String fileName = "testmylibrary.xml";
    File testFile = new File(fileName);
    testFile.delete();
    assertFalse("File should not exist",
        testFile.exists());
    assertTrue("File should have been saved",
        MyUtilities.saveMyLibraryToXMLFile(
            fileName, startMyLibrary));
    MyLibrary endMyLibrary =
        MyUtilities.getMyLibraryFromXMLFile(fileName);
    assertEquals("Test", endMyLibrary.getName());
    assertEquals(2, endMyLibrary.getBooks().size());
    assertEquals(2, endMyLibrary.getPeople().size());
    assertEquals("Fred", endMyLibrary.getBooks().
        get(0).getPerson().getName());
}
}
```

## Lesson 10 – manifest.txt file

```
Manifest-Version: 1.0
Main-Class: org.persistence.tutorial.MyLibrary
Class-Path: xstream-1.2.2.jar
```

## Lesson 10 – MyLibrary (with updated main() method)

```
package org.persistence.tutorial;

import java.util.ArrayList;

public class MyLibrary {

    String name;
    ArrayList<Book> books;
    ArrayList<Person> people;

    public MyLibrary(String name) {
        this.name = name;
        books = new ArrayList<Book>();
        people = new ArrayList<Person>();
    }

    public String getName() {
        return name;
    }

    public ArrayList<Book> getBooks() {
        return books;
    }

    public ArrayList<Person> getPeople() {
        return people;
    }

    public void addBook(Book b1) {
        this.books.add(b1);
    }

    public void removeBook(Book b1) {
        this.books.remove(b1);
    }

    public void addPerson(Person p1) {
        this.people.add(p1);
    }

    public void removePerson(Person p1) {
        this.people.remove(p1);
    }
}
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
public boolean checkOut(Book b1, Person p1) {
    int booksOut = this.getBooksForPerson(p1).size();
    if ((b1.getPerson() == null) &&
        (booksOut < p1.getMaximumBooks()))
    {
        b1.setPerson(p1);
        return true;
    }
    else {
        return false;
    }
}

public boolean checkIn(Book b1) {
    if (b1.getPerson() != null) {
        b1.setPerson(null);
        return true;
    }
    else {
        return false;
    }
}

public ArrayList<Book> getBooksForPerson(Person p1) {
    ArrayList<Book> result = new ArrayList<Book>();
    for (Book aBook : this.getBooks()) {
        if ((aBook.getPerson() != null) &&
            (aBook.getPerson().getName() == p1.getName()))
        {
            result.add(aBook);
        }
        // if (aBook.getPerson().getName() ==
        //     p1.getName()) {
        //     result.add(aBook);
        // }
    }
    return result;
}

public ArrayList<Book> getAvailableBooks() {
    ArrayList<Book> result = new ArrayList<Book>();
    for (Book book : this.getBooks()) {
        if (book.getPerson() == null) {
            result.add(book);
        }
    }
    return result;
}

public ArrayList<Book> getUnavailableBooks() {
    ArrayList<Book> result = new ArrayList<Book>();
    for (Book book : this.getBooks()) {
        if (book.getPerson() != null) {
            result.add(book);
        }
    }
    return result;
}
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
public static void main(String[] args) {
    // create a new mylibrary
    MyLibrary testLibrary = new MyLibrary("Test Drive Library");
    Book b1 = new Book("War and Peace");
    b1.setAuthor("Tolstoy");
    Book b2 = new Book("Great Expectations");
    b2.setAuthor("Charles Dickens");

    Person jim = new Person();
    jim.setName("Jim");
    Person sue = new Person();
    sue.setName("Sue");

    testLibrary.addBook(b1);
    testLibrary.addBook(b2);
    testLibrary.addPerson(jim);
    testLibrary.addPerson(sue);

    System.out.println("Just Created New Library");
    testLibrary.printStatus();

    testLibrary.checkOut(b1, sue);
    System.out.println("Checked out War and Peace to Sue");
    testLibrary.printStatus();

    testLibrary.checkIn(b1);
    testLibrary.checkOut(b2, jim);

    testLibrary.printStatus();

    MyUtilities.
        saveMyLibraryToXMLFile("testmain.xml", testLibrary);
    MyLibrary newMyLibrary = MyUtilities.
        getMyLibraryFromXMLFile("testmain.xml");
    System.out.println("Printing information from saved xml file");
    newMyLibrary.printStatus();
}

private void printStatus() {
    System.out.println("Status Report of MyLibrary: \n" +
        this.toString());

    for (Book thisBook : this.getBooks()) {
        System.out.println(thisBook);
    }

    for (Person thisPerson : this.getPeople()) {
        int count = this.getBooksForPerson(thisPerson).size();
        System.out.println(thisPerson + " (has " + count +
            " of my books)");
    }

    System.out.println("Books Available: "
        + this.getAvailableBooks().size());
    System.out.println("--- End of Status Report---\n");
}

@Override
public String toString() {
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
        return this.getName() +  
        ": " + this.getBooks().size() + " books; "  
        + this.getPeople().size() + " people.";  
    }  
}
```

### Lesson 11 – Final MyUtilitiesTest (with saveToFile() method)

```
package org.persistence.tutorial;  
  
import java.io.File;  
import org.junit.*;  
import static org.junit.Assert.*;  
  
public class MyUtilitiesTest {  
  
    @Test  
    public void saveStringToFile() {  
        String saveString = "this is test line one\n" +  
            "this is test line two\n";  
  
        File testFile = new File("testsavetostream.txt");  
        testFile.delete();  
        assertFalse("File should not exist",  
            testFile.exists());  
  
        assertTrue("File should have been saved",  
            MyUtilities.saveStringToFile("testsavestring.txt",  
                saveString));  
  
        String newString = MyUtilities.getStringFromFile(  
            "testsavestring.txt");  
        assertTrue("Save and get strings should be equal",  
            saveString.equals(newString));  
  
        assertFalse("File should not be saved",  
            MyUtilities.saveStringToFile(  
                "non-existent directory/thisshouldfail.txt",  
                saveString));  
  
        String emptyString = MyUtilities.getStringFromFile(  
            "badfilename.txt");  
        assertTrue("String should be empty",  
            emptyString.length() == 0);  
    }  
  
    public MyLibrary createMyLibrary() {  
        Book b1;  
        Book b2;  
        Person p1;  
        Person p2;  
        MyLibrary ml;  
  
        b1 = new Book("Book1");  
        b2 = new Book("Book2");  
        p1 = new Person();  
        p1.setName("Fred");  
        p2 = new Person();  
    }  
}
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
p2.setName("Sue");
ml = new MyLibrary("Test");

ml.addBook(b1);
ml.addBook(b2);
ml.addPerson(p1);
ml.addPerson(p2);
ml.checkOut(b1, p1);
return ml;
}

@Test public void convertToXML() {
    MyLibrary startMyLibrary = createMyLibrary();
    String testXMLOut = MyUtilities.convertToXML(startMyLibrary);
    MyLibrary endMyLibrary =
        MyUtilities.convertFromXML(testXMLOut);
    assertEquals("Test", endMyLibrary.getName());
    assertEquals(2, endMyLibrary.getBooks().size());
    assertEquals(2, endMyLibrary.getPeople().size());
    assertEquals("Fred", endMyLibrary.getBooks().
        get(0).getPerson().getName());
}

@Test public void saveToXMLFile() {
    MyLibrary startMyLibrary = createMyLibrary();
    String fileName = "testmylibrary.xml";
    File testFile = new File(fileName);
    testFile.delete();
    assertFalse("File should not exist",
        testFile.exists());
    assertTrue("File should have been saved",
        MyUtilities.saveMyLibraryToXMLFile(
            fileName, startMyLibrary));
    MyLibrary endMyLibrary =
        MyUtilities.getMyLibraryFromXMLFile(fileName);
    assertEquals("Test", endMyLibrary.getName());
    assertEquals(2, endMyLibrary.getBooks().size());
    assertEquals(2, endMyLibrary.getPeople().size());
    assertEquals("Fred", endMyLibrary.getBooks().
        get(0).getPerson().getName());
}

@Test public void saveToSerialFile() {
    MyLibrary startMyLibrary = createMyLibrary();
    String fileName = "testmylibrary.ser";
    File testFile = new File(fileName);
    testFile.delete();
    assertFalse("File should not exist",
        testFile.exists());
    assertTrue("File should have been saved",
        MyUtilities.saveMyLibraryToSerialFile(
            fileName, startMyLibrary));
    MyLibrary endMyLibrary =
        MyUtilities.getMyLibraryFromSerialFile(fileName);
    assertEquals("Test", endMyLibrary.getName());
    assertEquals(2, endMyLibrary.getBooks().size());
    assertEquals(2, endMyLibrary.getPeople().size());
    assertEquals("Fred", endMyLibrary.getBooks().
        get(0).getPerson().getName());
}
```

```
}
```

## Lesson 11 – MyUtilities (with saveMyLibraryToSerialFile() method)

```
package org.persistence.tutorial;

import java.io.*;

import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.io.xml.DomDriver;

public class MyUtilities {

    public static boolean saveStringToFile(String fileName,
        String saveString) {
        boolean saved = false;
        BufferedWriter bw = null;

        try {
            bw = new BufferedWriter(
                new FileWriter(fileName));

            try {
                bw.write(saveString);
                saved = true;
            }
            finally {
                bw.close();
            }
        }
        catch (IOException ex) {
            ex.printStackTrace();
        }
        return saved;
    }

    public static String getStringFromFile(String fileName) {
        BufferedReader br = null;
        StringBuilder sb = new StringBuilder();

        try {
            br = new BufferedReader(
                new FileReader(fileName));

            try {
                String s;
                while ((s = br.readLine()) != null) {
                    // add linefeed back since stripped by readLine()
                    sb.append(s);
                    sb.append("\n");
                }
            }
            finally {
                br.close();
            }
        }
        catch (IOException ex) {
            ex.printStackTrace();
        }
        return sb.toString();
    }
}
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
public static String convertToXML(MyLibrary ml) {
    XStream xstream = new XStream(new DomDriver());
    xstream.setMode(XStream.ID_REFERENCES);
    xstream.alias("person", Person.class);
    xstream.alias("book", Book.class);
    xstream.alias("mylibrary", MyLibrary.class);
    return xstream.toXML(ml);
}

public static MyLibrary convertFromXML(String XMLString) {
    MyLibrary ml = null;
    XStream xstream = new XStream(new DomDriver());
    xstream.setMode(XStream.ID_REFERENCES);
    xstream.alias("person", Person.class);
    xstream.alias("book", Book.class);
    xstream.alias("mylibrary", MyLibrary.class);
    Object obj = xstream.fromXML(XMLString);
    if (obj instanceof MyLibrary) {
        ml = (MyLibrary) obj;
    }
    return ml;
}

public static boolean saveMyLibraryToXMLFile(String fileName,
    MyLibrary ml) {
    return saveStringToFile(fileName, convertToXML(ml));
}

public static MyLibrary getMyLibraryFromXMLFile(String fileName) {
    return convertFromXML(getStringFromFile(fileName));
}

public static boolean saveMyLibraryToSerialFile(String fileName,
    MyLibrary ml) {
    boolean saved = false;
    try {
        ObjectOutputStream oos =
            new ObjectOutputStream(
                new BufferedOutputStream(
                    new FileOutputStream(fileName)));

        // inner try block
        try {
            oos.writeObject(ml);
            saved = true;
        }
        finally {
            oos.close();
        }
    }
    catch (Exception ex) {
        ex.printStackTrace();
    }
    return saved;
}

public static MyLibrary getMyLibraryFromSerialFile(String fileName) {
    // TODO Auto-generated method stub
    return null;
}
}
```

## Lesson 12 – Final Book Class

```
package org.persistence.tutorial;

import java.io.Serializable;

public class Book implements Serializable{

    /**
     *
     */
    private static final long serialVersionUID = -4946021242876591956L;
    public String title;
    public String author;
    private Person person;

    public Book(String string) {
        this.title = string;
        this.author = "unknown author";
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getTitle() {
        return title;
    }

    public void setPerson(Person p2) {
        this.person = p2;
    }

    public Person getPerson() {
        return this.person;
    }

    @Override
    public String toString() {
        String available;
        if (this.getPerson() == null) {
            available = "Available";
        }
        else {
            available = "Checked out to " +
                this.getPerson().getName();
        }
        return this.getTitle() +
            " by " + this.getAuthor() +
            "; " + available;
    }
}
```

## Lesson 12 – Final MyLibrary Class

```
package org.persistence.tutorial;

import java.io.Serializable;
import java.util.ArrayList;

public class MyLibrary implements Serializable {

    /**
     *
     */
    private static final long serialVersionUID = 169520141385923186L;
    String name;
    ArrayList<Book> books;
    ArrayList<Person> people;

    public MyLibrary(String name) {
        this.name = name;
        books = new ArrayList<Book>();
        people = new ArrayList<Person>();
    }

    public String getName() {
        return name;
    }

    public ArrayList<Book> getBooks() {
        return books;
    }

    public ArrayList<Person> getPeople() {
        return people;
    }

    public void addBook(Book b1) {
        this.books.add(b1);
    }

    public void removeBook(Book b1) {
        this.books.remove(b1);
    }

    public void addPerson(Person p1) {
        this.people.add(p1);
    }

    public void removePerson(Person p1) {
        this.people.remove(p1);
    }

    public boolean checkOut(Book b1, Person p1) {
        int booksOut = this.getBooksForPerson(p1).size();
        if ((b1.getPerson() == null) &&
            (booksOut < p1.getMaximumBooks()))
        {
            b1.setPerson(p1);
        }
    }
}
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
        return true;
    }
    else {
        return false;
    }
}

public boolean checkIn(Book b1) {
    if (b1.getPerson() != null) {
        b1.setPerson(null);
        return true;
    }
    else {
        return false;
    }
}

public ArrayList<Book> getBooksForPerson(Person p1) {
    ArrayList<Book> result = new ArrayList<Book>();
    for (Book aBook : this.getBooks()) {
        if ((aBook.getPerson() != null) &&
            (aBook.getPerson().getName() == p1.getName()))
        {
            result.add(aBook);
        }
        // if (aBook.getPerson().getName() ==
        //     p1.getName()) {
        //     result.add(aBook);
        // }
    }
    return result;
}

public ArrayList<Book> getAvailableBooks() {
    ArrayList<Book> result = new ArrayList<Book>();
    for (Book book : this.getBooks()) {
        if (book.getPerson() == null) {
            result.add(book);
        }
    }
    return result;
}

public ArrayList<Book> getUnavailableBooks() {
    ArrayList<Book> result = new ArrayList<Book>();
    for (Book book : this.getBooks()) {
        if (book.getPerson() != null) {
            result.add(book);
        }
    }
    return result;
}

public static void main(String[] args) {
    // create a new mylibrary
    MyLibrary testLibrary = new MyLibrary("Test Drive Library");
    Book b1 = new Book("War and Peace");
    b1.setAuthor("Tolstoy");
    Book b2 = new Book("Great Expectations");
    b2.setAuthor("Charles Dickens");
}
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
    Person jim = new Person();
    jim.setName("Jim");
    Person sue = new Person();
    sue.setName("Sue");

    testLibrary.addBook(b1);
    testLibrary.addBook(b2);
    testLibrary.addPerson(jim);
    testLibrary.addPerson(sue);

    System.out.println("Just Created New Library");
    testLibrary.printStatus();

    testLibrary.checkOut(b1, sue);
    System.out.println("Checked out War and Peace to Sue");
    testLibrary.printStatus();

    testLibrary.checkIn(b1);
    testLibrary.checkOut(b2, jim);

    testLibrary.printStatus();

    MyUtilities.
        saveMyLibraryToXMLFile("testmain.xml", testLibrary);
    MyLibrary newMyLibrary = MyUtilities.
        getMyLibraryFromXMLFile("testmain.xml");
    System.out.println("Printing information from saved xml file");
    newMyLibrary.printStatus();
}

private void printStatus() {
    System.out.println("Status Report of MyLibrary: \n" +
        this.toString());

    for (Book thisBook : this.getBooks()) {
        System.out.println(thisBook);
    }

    for (Person thisPerson : this.getPeople()) {
        int count = this.getBooksForPerson(thisPerson).size();
        System.out.println(thisPerson + " (has " + count +
            " of my books)");
    }

    System.out.println("Books Available: "
        + this.getAvailableBooks().size());
    System.out.println("--- End of Status Report---\n");
}

@Override
public String toString() {
    return this.getName() +
        ": " + this.getBooks().size() + " books; "
        + this.getPeople().size() + " people.";
}
}
```

## Lesson 12 – Final MyUtilities Class

```
package org.persistence.tutorial;

import java.io.*;

import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.io.xml.DomDriver;

public class MyUtilities {

    public static boolean saveStringToFile(String fileName,
        String saveString) {
        boolean saved = false;
        BufferedWriter bw = null;

        try {
            bw = new BufferedWriter(
                new FileWriter(fileName));

            try {
                bw.write(saveString);
                saved = true;
            }
            finally {
                bw.close();
            }
        }
        catch (IOException ex) {
            ex.printStackTrace();
        }
        return saved;
    }

    public static String getStringFromFile(String fileName) {
        BufferedReader br = null;
        StringBuilder sb = new StringBuilder();

        try {
            br = new BufferedReader(
                new FileReader(fileName));

            try {
                String s;
                while ((s = br.readLine()) != null) {
                    // add linefeed back since stripped by readLine()
                    sb.append(s);
                    sb.append("\n");
                }
            }
            finally {
                br.close();
            }
        }
        catch (IOException ex) {
            ex.printStackTrace();
        }
        return sb.toString();
    }

    public static String convertToXML(MyLibrary ml) {
        XStream xstream = new XStream(new DomDriver());
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
xstream.setMode(XStream.ID_REFERENCES);
xstream.alias("person", Person.class);
xstream.alias("book", Book.class);
xstream.alias("mylibrary", MyLibrary.class);
return xstream.toXML(ml);
}

public static MyLibrary convertFromXML(String XMLString) {
    MyLibrary ml = null;
    XStream xstream = new XStream(new DomDriver());
    xstream.setMode(XStream.ID_REFERENCES);
    xstream.alias("person", Person.class);
    xstream.alias("book", Book.class);
    xstream.alias("mylibrary", MyLibrary.class);
    Object obj = xstream.fromXML(XMLString);
    if (obj instanceof MyLibrary) {
        ml = (MyLibrary) obj;
    }
    return ml;
}

public static boolean saveMyLibraryToXMLFile(String fileName,
    MyLibrary ml) {
    return saveStringToFile(fileName, convertToXML(ml));
}

public static MyLibrary getMyLibraryFromXMLFile(String fileName) {
    return convertFromXML(getStringFromFile(fileName));
}

public static boolean saveMyLibraryToSerialFile(String fileName,
    MyLibrary ml) {
    boolean saved = false;
    try {
        ObjectOutputStream oos =
            new ObjectOutputStream(
                new BufferedOutputStream(
                    new FileOutputStream(fileName)));

        // inner try block
        try {
            oos.writeObject(ml);
            saved = true;
        }
        finally {
            oos.close();
        }
    }
    catch (Exception ex) {
        ex.printStackTrace();
    }
    return saved;
}

public static MyLibrary getMyLibraryFromSerialFile(String fileName) {
    MyLibrary ml = null;
    try {
        ObjectInputStream ois =
            new ObjectInputStream(
                new BufferedInputStream(
                    new FileInputStream(fileName)));

        try {
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
        Object obj = ois.readObject();
        if (obj instanceof MyLibrary) {
            ml = (MyLibrary) obj;
        }
    }
    finally {
        ois.close();
    }
}
catch (Exception ex) {
    ex.printStackTrace();
}
return ml;
}
}
```

### Lesson 12 – Final Person Class

```
package org.persistence.tutorial;

import java.io.Serializable;

/**
 * @author Mark Dexter
 * @version 1.0
 * The Person class represents people who
 * check out books from a MyLibrary.
 */
public class Person implements Serializable{
    /**
     *
     */
    private static final long serialVersionUID = -549832220463532971L;
    // fields
    private String name; // person's name
    private int maximumBooks; // # books can check out

    // constructors
    public Person() {
        name = "unknown name";
        maximumBooks = 3;
    }

    public Person(String name, int maximumBooks) {
        super();
        this.name = name;
        this.maximumBooks = maximumBooks;
    }

    public int getMaximumBooks() {
        return maximumBooks;
    }

    // methods
    public String getName() {
        return name;
    }
}
```

## Eclipse and Java: Introducing Persistence Tutorial Companion Document

```
public void setMaximumBooks(int maximumBooks) {
    this.maximumBooks = maximumBooks;
}

public void setName(String newName) {
    this.name = newName;
}

public String toString() {
    return this.getName() + " (" + this.getMaximumBooks()
        + " books)";
}
}
```

### Lesson 12 – Final AllTests Class

```
package org.persistence.tutorial;

import org.junit.runner.*;
import org.junit.runners.*;

@RunWith(Suite.class)
@Suite.SuiteClasses(value={
    PersonTest.class,
    BookTest.class,
    MyLibraryTest.class,
    MyUtilitiesTest.class
})

public class AllTests {
}
```