Persistence

Saving and retrieving objects to/from XML files

Produced by:

Dr. Siobhán Drohan
Mr. Colm Dunphy
Mr. Diarmuid O'Connor
Dr. Frank Walsh

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/

Persistence – lack of (volatility - volatile)

Our **Shop** App

Shop V4.0

implemented the **CRUD** process

⚠ Problem: All entered **data is lost** if we **close our application** (or lose power)

Shop V5.0

use XML to make our **data persistent** beyond the life of our app

Solution: Store our objects from memory to XML **files**.

Shop **V5.0** (using XML)

- For our XML persistence, we will use a component called **Xstream**.

- **XStream**
  - is a simple **library**
    to serialize objects to XML and back again.
  - is called a **component**
    and we must download the **jar** file it is stored in,
    and incorporate it into our project.

http://x-stream.github.io/index.html

New Version of Xstream - **xstream-1.4.11.1**

- **Images in the slides may refer to a different version of Xstream (e.g. 1.4.10.0).**

  The current version is xstream-1.4.11.1

  http://repo1.maven.org/maven2/com/thoughtworks/xstream/xstream/1.4.11.1/xstream-1.4.11.1.jar

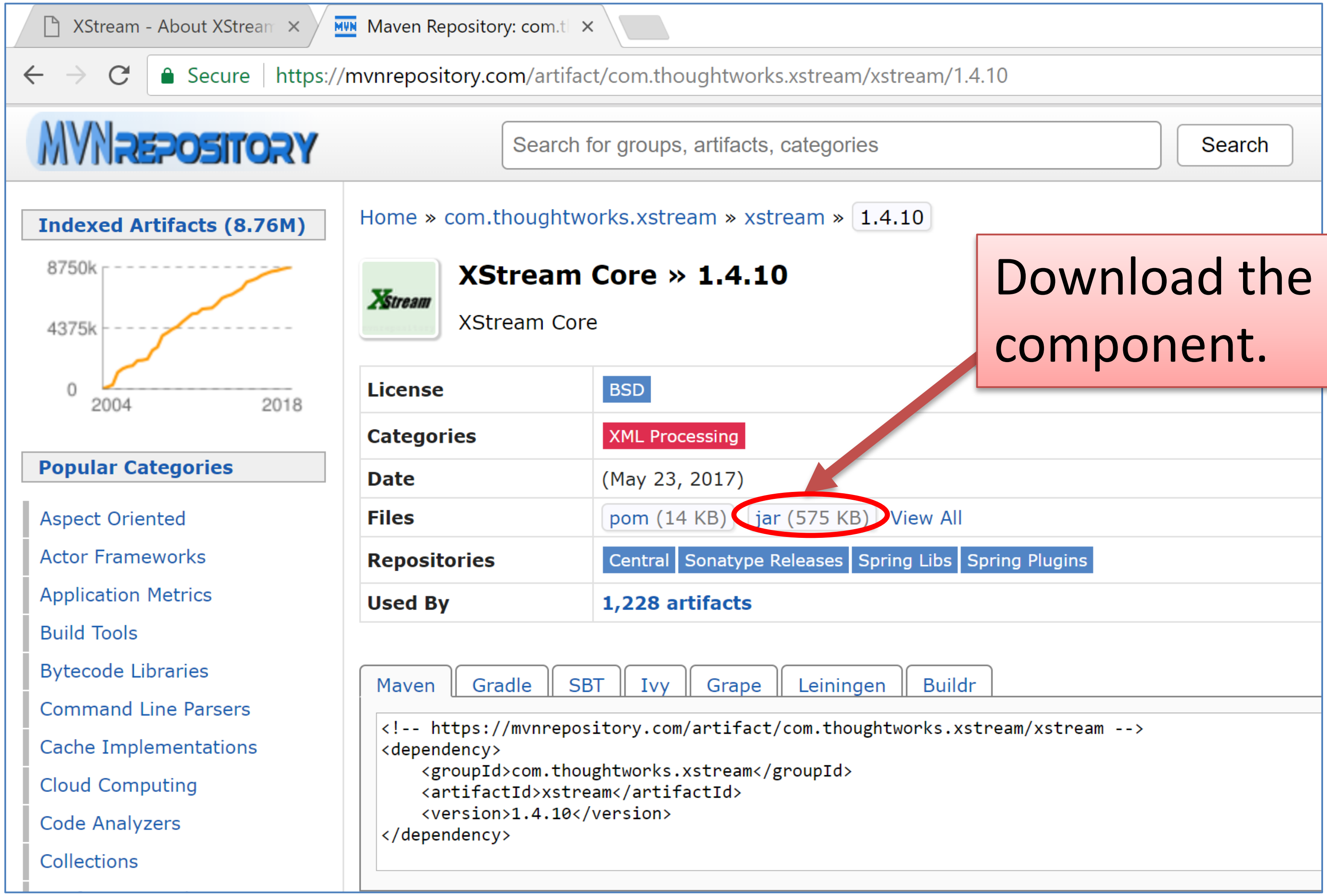
Shop V5.0 (using XML) - **STEPS**

1. Download the **xstream.jar** component
   - 1.1 Add it to your Shop project.

2. **Store** Class
   - Write the **save()**, and **load()** methods.

3. **Driver** Class
   - include extra load and save functionality to the menu.

1. Download the **component**



XStream - About XStrean
Maven Repository: com.t
Secure
https://mvnrepository.com/artifact/com.thoughtworks.xstream/xstream/1.4.10
MVNREPOSITORY
Search for groups, artifacts, categories
Search
Indexed Artifacts (8.76M)
8750k
4375k
0
2004
2018
Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections

Home » com.thoughtworks.xstream » xstream » 1.4.10

XStream Core » 1.4.10

XStream Core

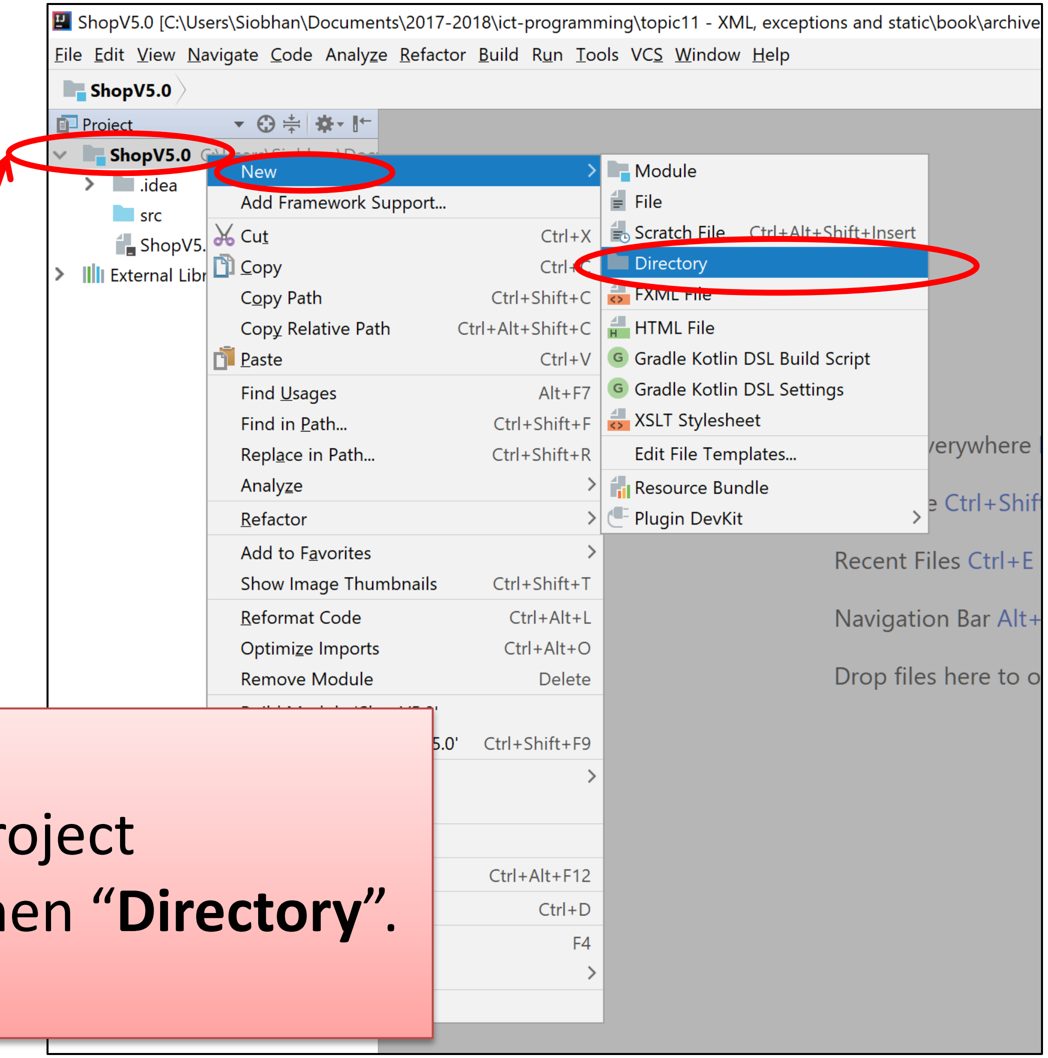
License	BSD
Categories	XML Processing
Date	(May 23, 2017)
Files	pom (14 KB) jar (575 KB) View All
Repositories	Central Sonatype Releases Spring Libs Spring Plugins
Used By	**1,228 artifacts**

```
<!-- https://mvnrepository.com/artifact/com.thoughtworks.xstream/xstream -->
<dependency>
    <groupId>com.thoughtworks.xstream</groupId>
    <artifactId>xstream</artifactId>
    <version>1.4.10</version>
</dependency>
```

Download the **xstream.jar** component.

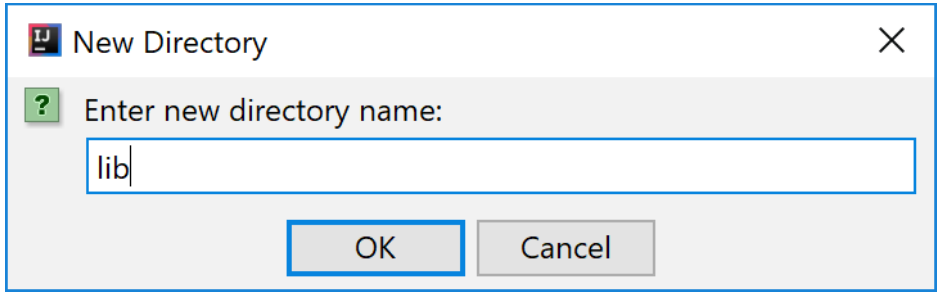
https://mvnrepository.com/artifact/com.thoughtworks.xstream/xstream/1.4.10

1.1 Adding a component to the **lib folder** -1



On the ShopV5.0 project
right click | select “**New**” | then “**Directory**”.

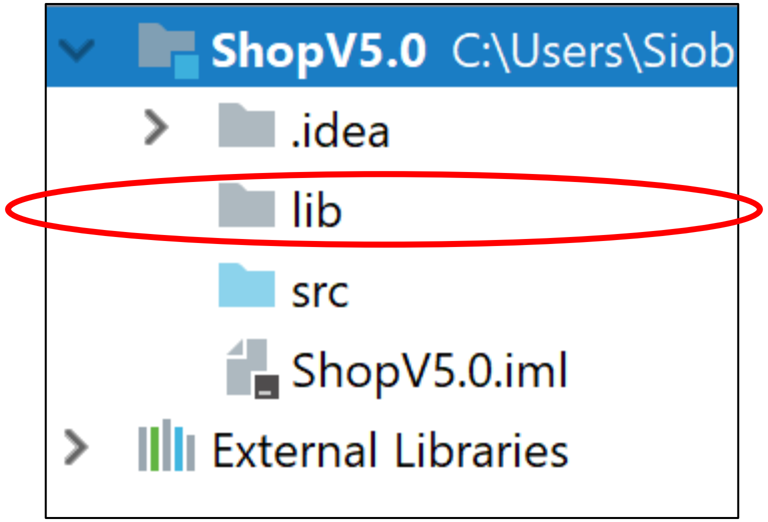
1.1 Adding a component to the **lib folder** - 2


New Directory

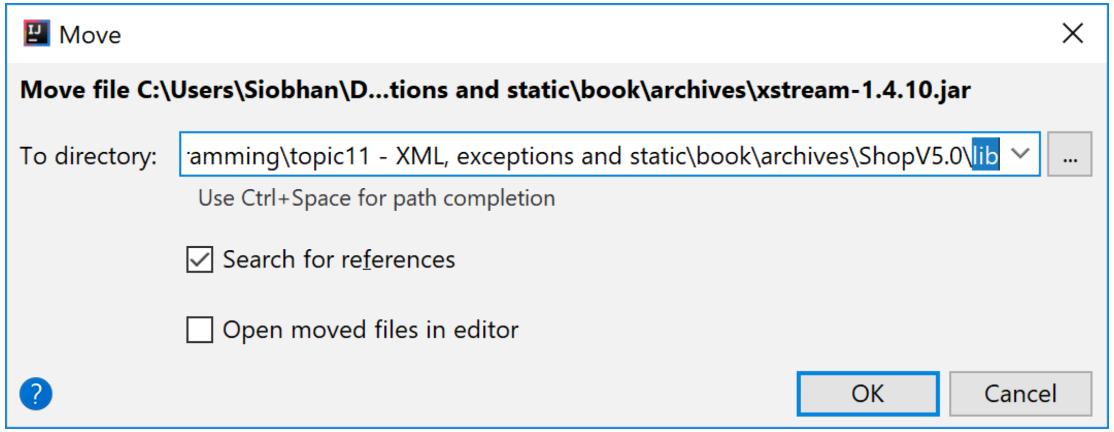
Enter new directory name:

lib

Call the new directory “lib”.



1.1 Adding a component to the **lib folder** - 3



Move

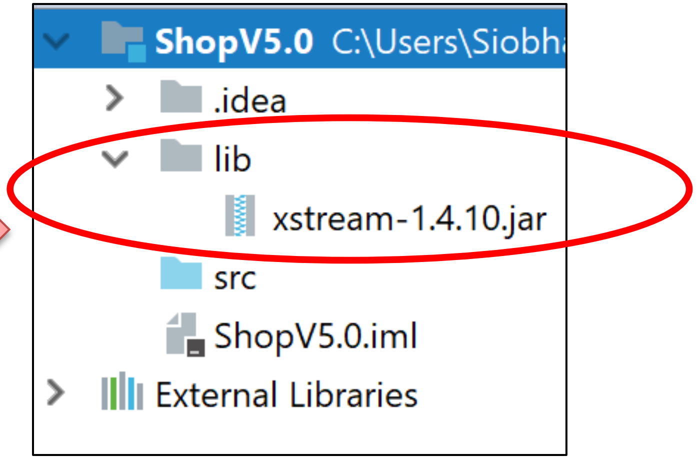
Move file C:\Users\Siobhan\D...tions and static\book\archives\xstream-1.4.10.jar

To directory: 'amming\topic11 - XML, exceptions and static\book\archives\ShopV5.0\lib

Use Ctrl+Space for path completion

☑ Search for references

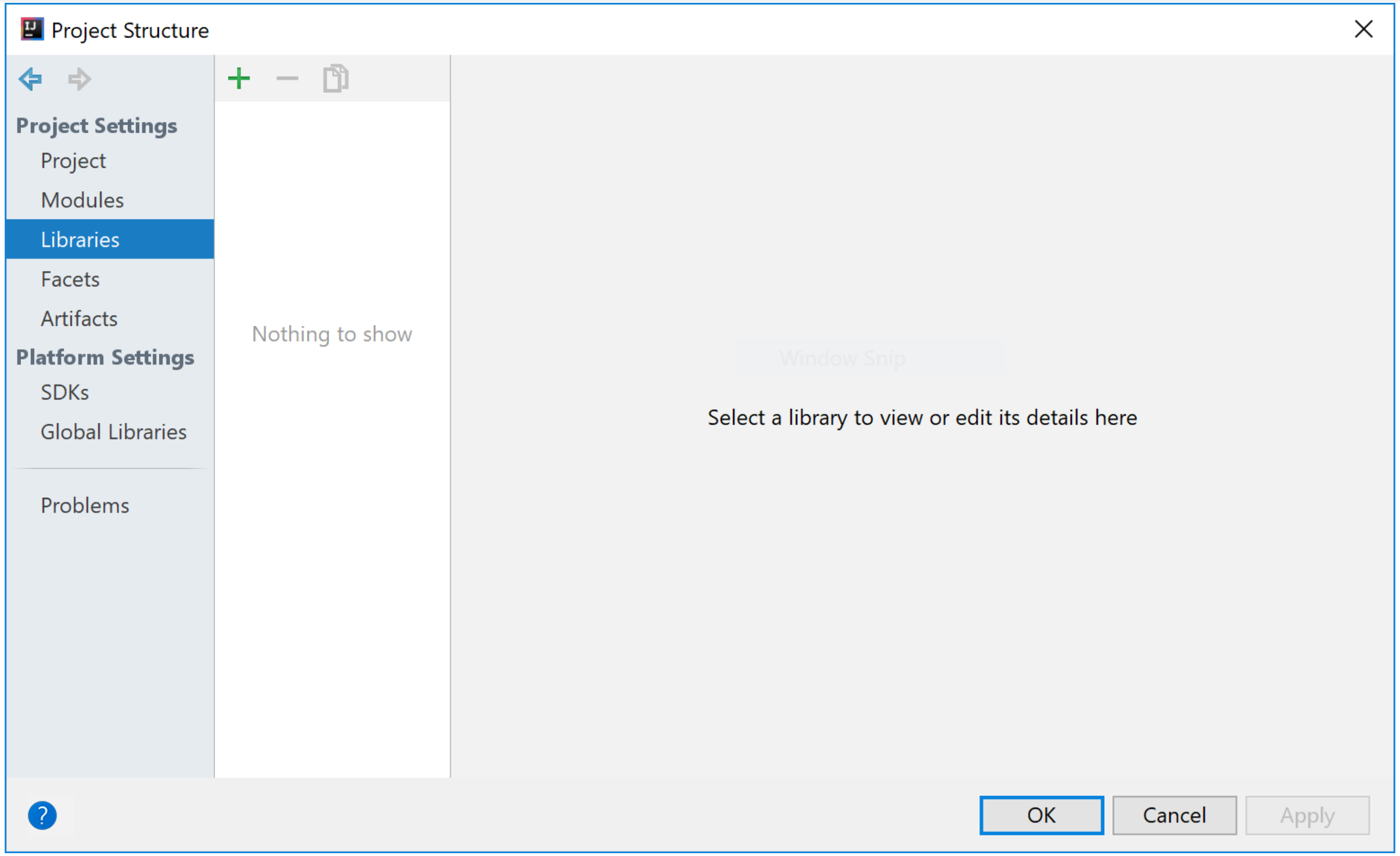
□ Open moved files in editor



Drag the **xstream.jar** file into the lib folder.

1.1 Adding the component to your **build path** - 1

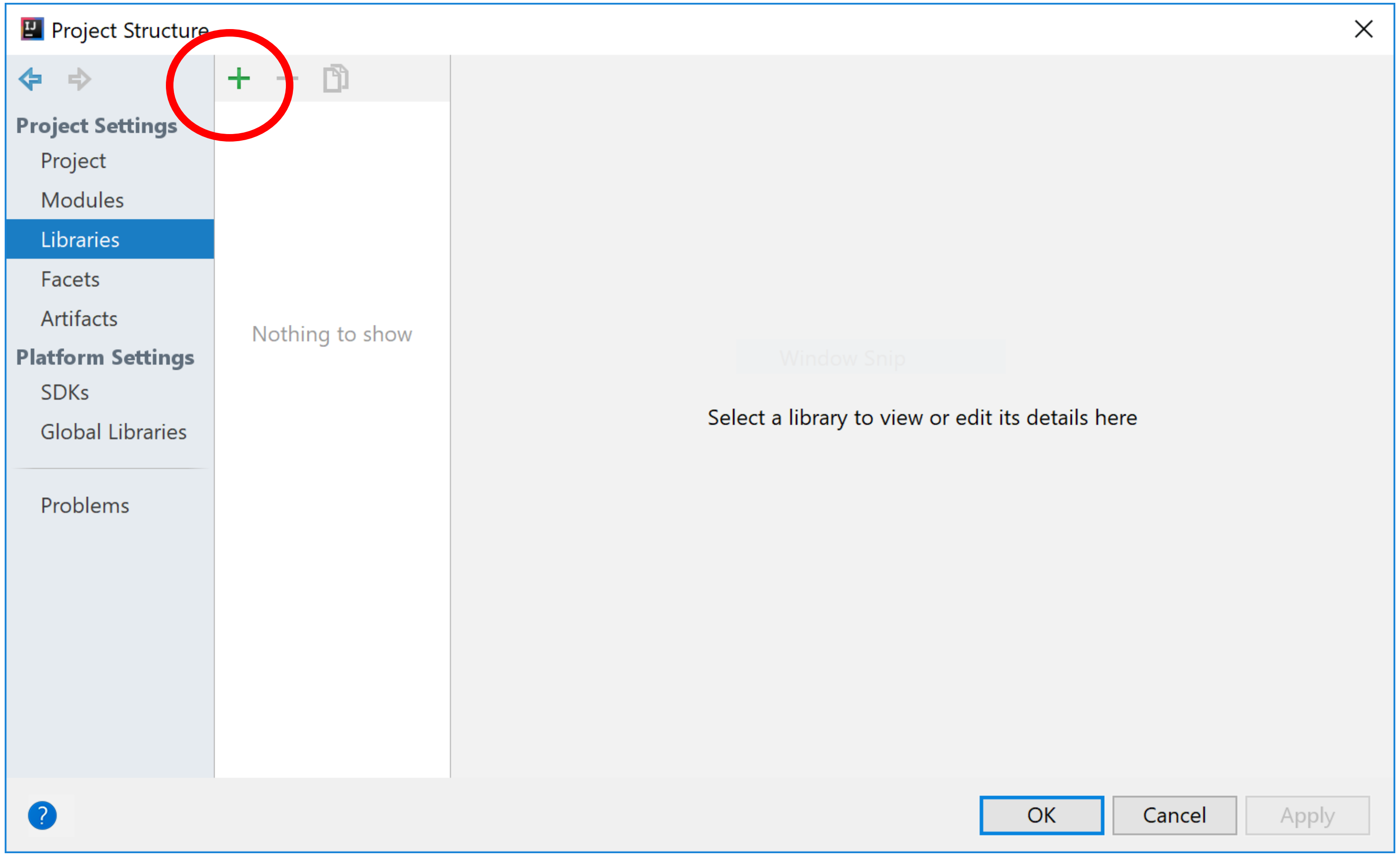
From **File** menu, select **Project Structure.** Click on **Libraries.**



Select a library to view or edit its details here

1.1 Adding the component to your **build path** - 2

To add a library to your build path, click on the green +

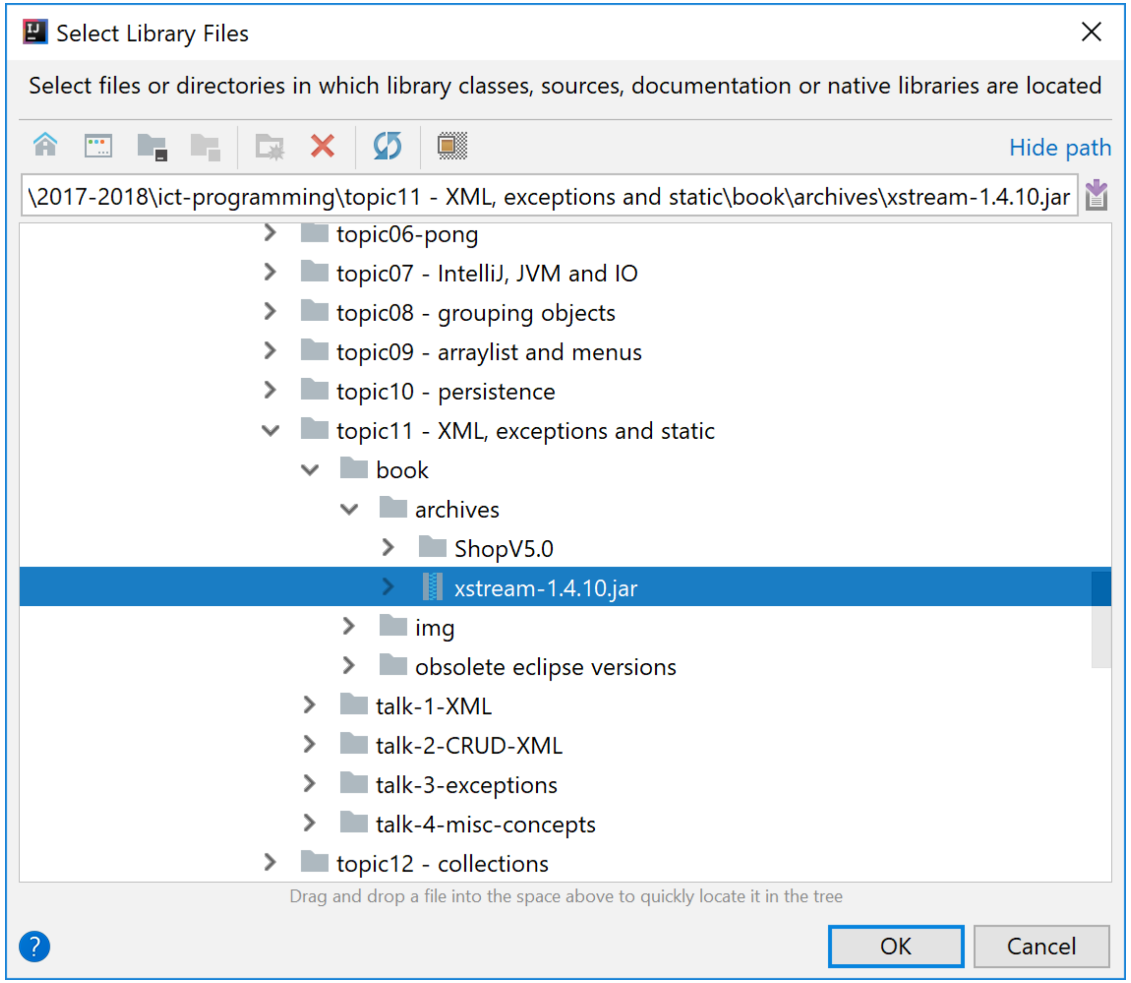


Nothing to show

Select a library to view or edit its details here

1.1 Adding the component to your **build path** - 3

Select **Java** and locate your library…click **OK** (a few times!)



Select Library Files

Select files or directories in which library classes, sources, documentation or native libraries are located

\2017-2018\ict-programming\topic11 - XML, exceptions and static\book\archives\xstream-1.4.10.jar

- topic06-pong
- topic07 - IntelliJ, JVM and IO
- topic08 - grouping objects
- topic09 - arraylist and menus
- topic10 - persistence
- topic11 - XML, exceptions and static
  - book
    - archives
      - ShopV5.0
      - xstream-1.4.10.jar
    - img
    - obsolete eclipse versions
  - talk-1-XML
  - talk-2-CRUD-XML
  - talk-3-exceptions
  - talk-4-misc-concepts
- topic12 - collections

Drag and drop a file into the space above to quickly locate it in the tree

Shop V5.0 (using XML) - **STEPS**

1. Download the **xstream.jar** component
   - Add it to your Shop project.

2. **Store** Class
   - Write the **save()**, and **load()** methods.

3. **Driver** Class
   - include extra load and save functionality to the menu.

Store
.java

save()

```
@SuppressWarnings("unchecked")

   public void save() throws Exception
   {
       XStream xstream = new XStream(new DomDriver());      // 1.

       ObjectOutputStream out = xstream.createObjectOutputStream
                                    (new FileWriter("products.xml"));     // 2.


       out.writeObject(products);   // 3.

       out.close();             // 4.
   }
```

// 1. Initialize an xstream object variable
// 2. Use it to initialize an ObjectOutputStream to a specific file
// 3. Write out the objects you want saved e.g. products
// 4. Close the stream / file

Store

.java

load()

To prevent warning messages

```
    @SuppressWarnings("unchecked")
    public void load() throws Exception
    {
        XStream xstream = new XStream(new DomDriver());  // 1.

        ObjectInputStream is = xstream.createObjectInputStream
                                    (new FileReader("products.xml"));      // 2.


        products = (ArrayList<Product>) is.readObject(); // 3.


        is.close();   // 4.
    }
```

// 1. Initialize an xstream object variable

// 2. Use it to initialize an Object**Input**Stream from a specific file

// 3. Call the **is.readObject()** method to assign values to the object e.g. products

// 4. Close the stream / file

*Updated to remove JAVA security warning

Store

.java

load()

```
public void load() throws Exception
{
    XStream xstream = new XStream(new DomDriver());

     // ------------------ PREVENT SECURITY WARNINGS-----------------------------
     // The Product class is what we are reading in.
     // Modify to include others if needed

     Class<?>[] classes = new Class[] { Product.class };
     XStream.setupDefaultSecurity(xstream);
     xstream.allowTypes(classes);
    // --------------------------------------------------------------------------

    ObjectInputStream is = xstream.createObjectInputStream
                                (new FileReader("products.xml"));
    products = (ArrayList<Product>) is.readObject();
    is.close();
}
```

Store

.java

Reusing save() and load() code

To use the **load()** & **save()** code in another project, change:

1. The **type** of object stored in the ArrayList.
2. The **name** of the xml file
3. The **name** of the ArrayList object.

Store

.java

Required Packages

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

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

Note: you need to import these additional **packages.**

Shop V5.0 (using XML) - **STEPS**

1. Download the **xstream.jar** component
   - Add it to your Shop project.

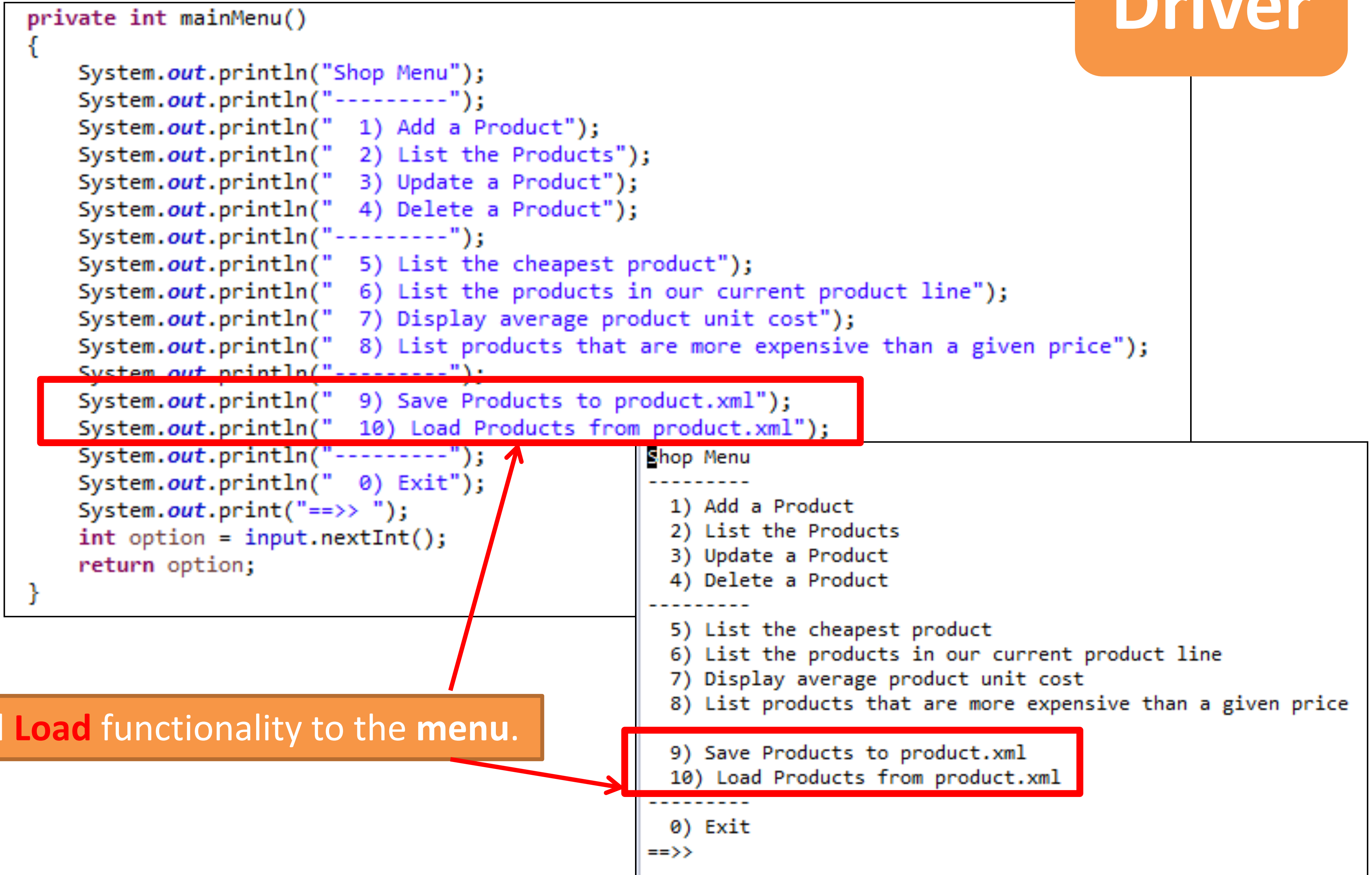
2. **Store** Class
   - Write the **load()**, **save()** methods.

3. **Driver** Class
   - include extra **load** and **save** functionality to the menu.

3. Driver Class

menu

Driver



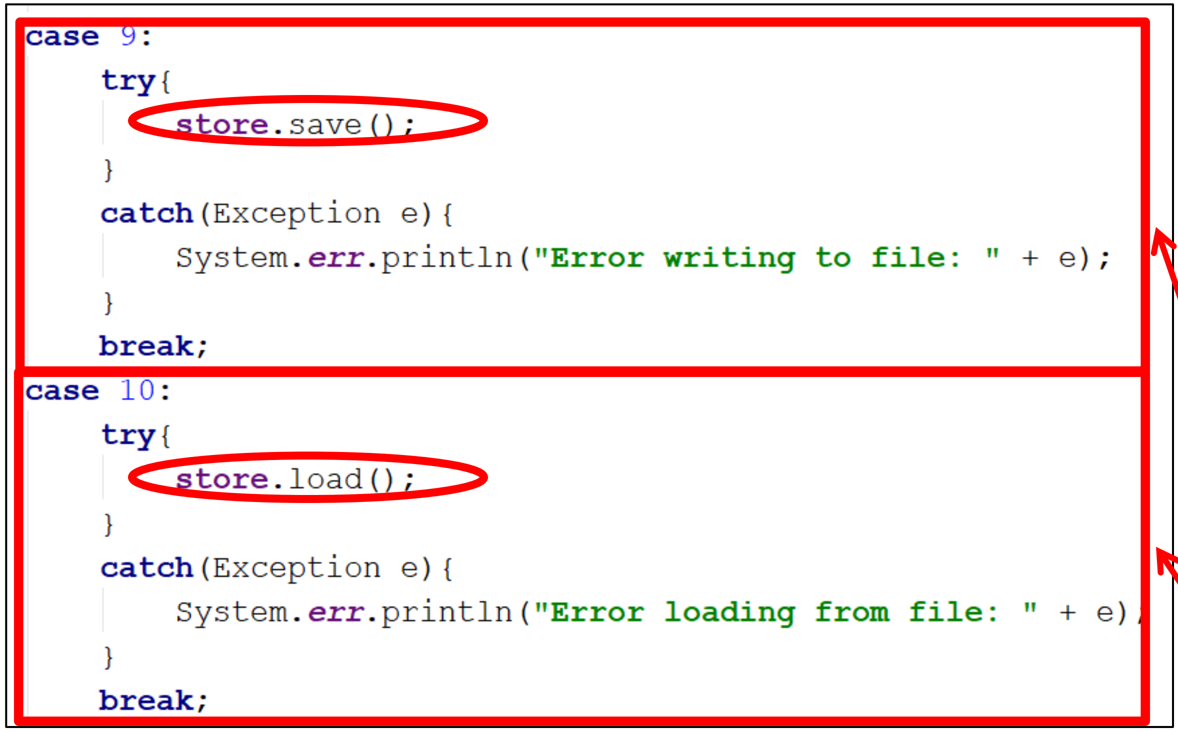
```
private int mainMenu()
{
    System.out.println("Shop Menu");
    System.out.println("---------");
    System.out.println("  1) Add a Product");
    System.out.println("  2) List the Products");
    System.out.println("  3) Update a Product");
    System.out.println("  4) Delete a Product");
    System.out.println("---------");
    System.out.println("  5) List the cheapest product");
    System.out.println("  6) List the products in our current product line");
    System.out.println("  7) Display average product unit cost");
    System.out.println("  8) List products that are more expensive than a given price");
    System.out.println("---------");
    System.out.println("  9) Save Products to product.xml");
    System.out.println("  10) Load Products from product.xml");
    System.out.println("---------");
    System.out.println("  0) Exit");
    System.out.print("==>> ");
    int option = input.nextInt();
    return option;
}
```

```
Shop Menu
---------
  1) Add a Product
  2) List the Products
  3) Update a Product
  4) Delete a Product
---------
  5) List the cheapest product
  6) List the products in our current product line
  7) Display average product unit cost
  8) List products that are more expensive than a given price
---------
  9) Save Products to product.xml
  10) Load Products from product.xml
---------
  0) Exit
==>>
```

Add **Save** and **Load** functionality to the **menu**.

try/catch

Driver



```
case 9:
    try{
        store.save();
    }
    catch(Exception e){
        System.err.println("Error writing to file: " + e);
    }
    break;
case 10:
    try{
        store.load();
    }
    catch(Exception e){
        System.err.println("Error loading from file: " + e);
    }
    break;
```

Inside a **try/catch block**,

- Call the **save** method for **option 9**.
- Call the **load** method for **option 10**.

Add **Save** and **Load** functionality to the **menu**.

XML file

```xml
<object-stream>
  <list>
    <Product>
      <productName>24 inch monitor</productName>
      <productCode>3423</productCode>
      <unitCost>129.99</unitCost>
      <inCurrentProductLine>true</inCurrentProductLine>
    </Product>
    <Product>
      <productName>14 inch monitor</productName>
      <productCode>2322</productCode>
      <unitCost>109.99</unitCost>
      <inCurrentProductLine>true</inCurrentProductLine>
    </Product>
  </list>
</object-stream>
```

When the **save** option is selected from the menu, this **XML file** is created

The XML file is located in your **root project directory**.

Questions

1. What file type do we store Java components in?
2. Which Java component did we use for serializing objects?
3. What 2 methods do we have to write to use this component?

Any Questions?