Persistence

Saving and retrieving objects to/from XML files

Produced by: Dr. Siobhán Drohan
Mr. Colm Dunphy

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 **library 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.
    1. Lib folder
    2. Configure the library in IntelliJ

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

Current Version of Xstream - **xstream-1.4.16**

- **Any images in the slides that refer to a different version of Xstream should be replaced with the current version.**

  The current version is xstream-1.4.16

  https://x-stream.github.io/download.html

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

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

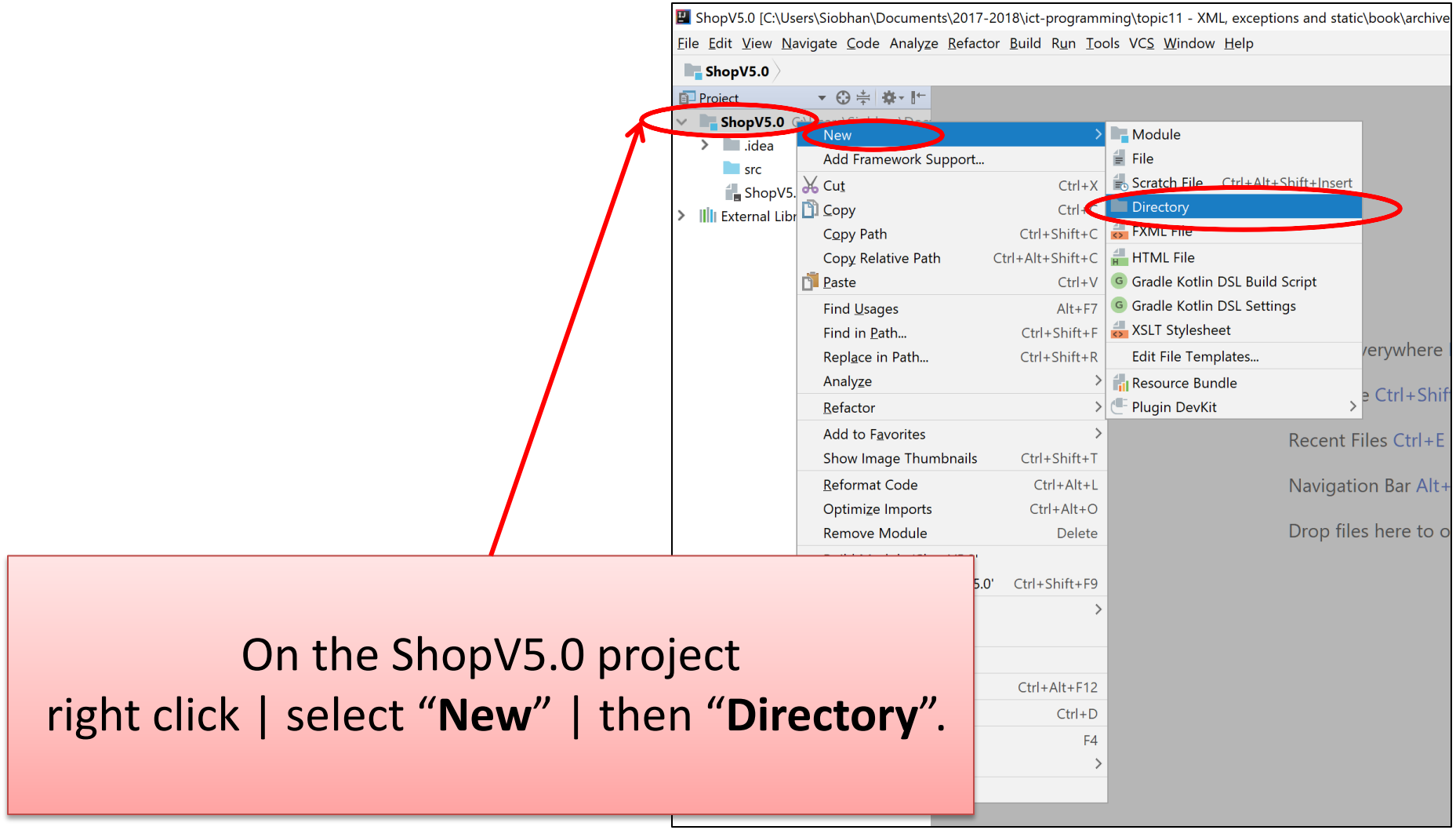
2. **Store** Class
   - Write the **save()**, and **load()** methods.

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

1. Download the **component**

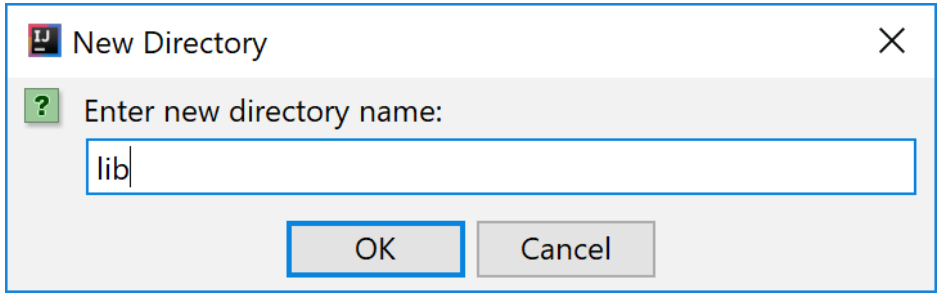
https://x-stream.github.io/download.html

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

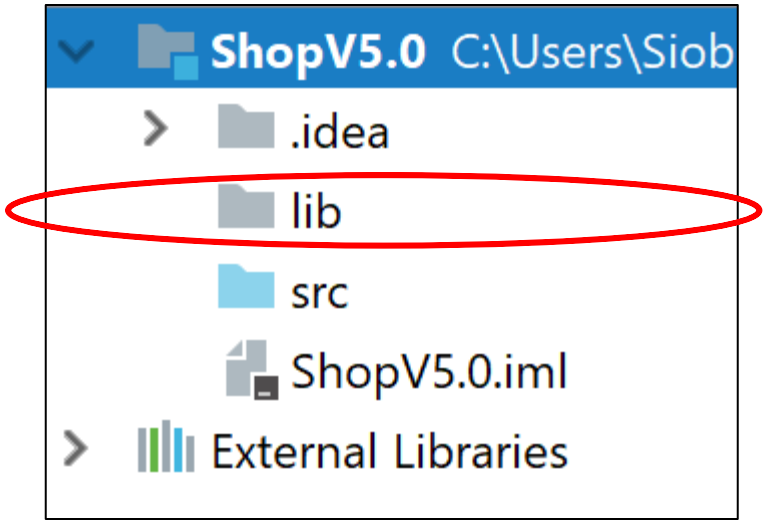


On the ShopV5.0 project
right click | select “**New**” | then “**Directory**”.

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

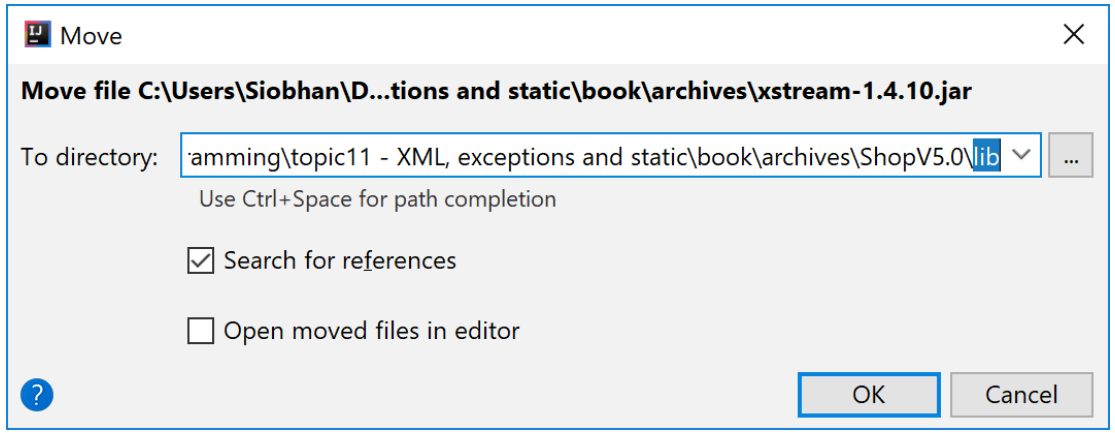


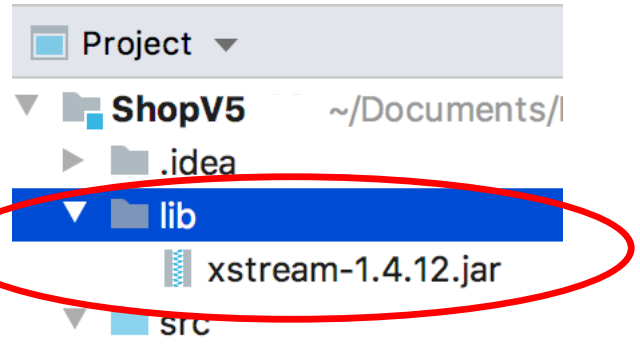
Call the new directory "**lib**".



*you can also create this in Explorer/finder/terminal

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

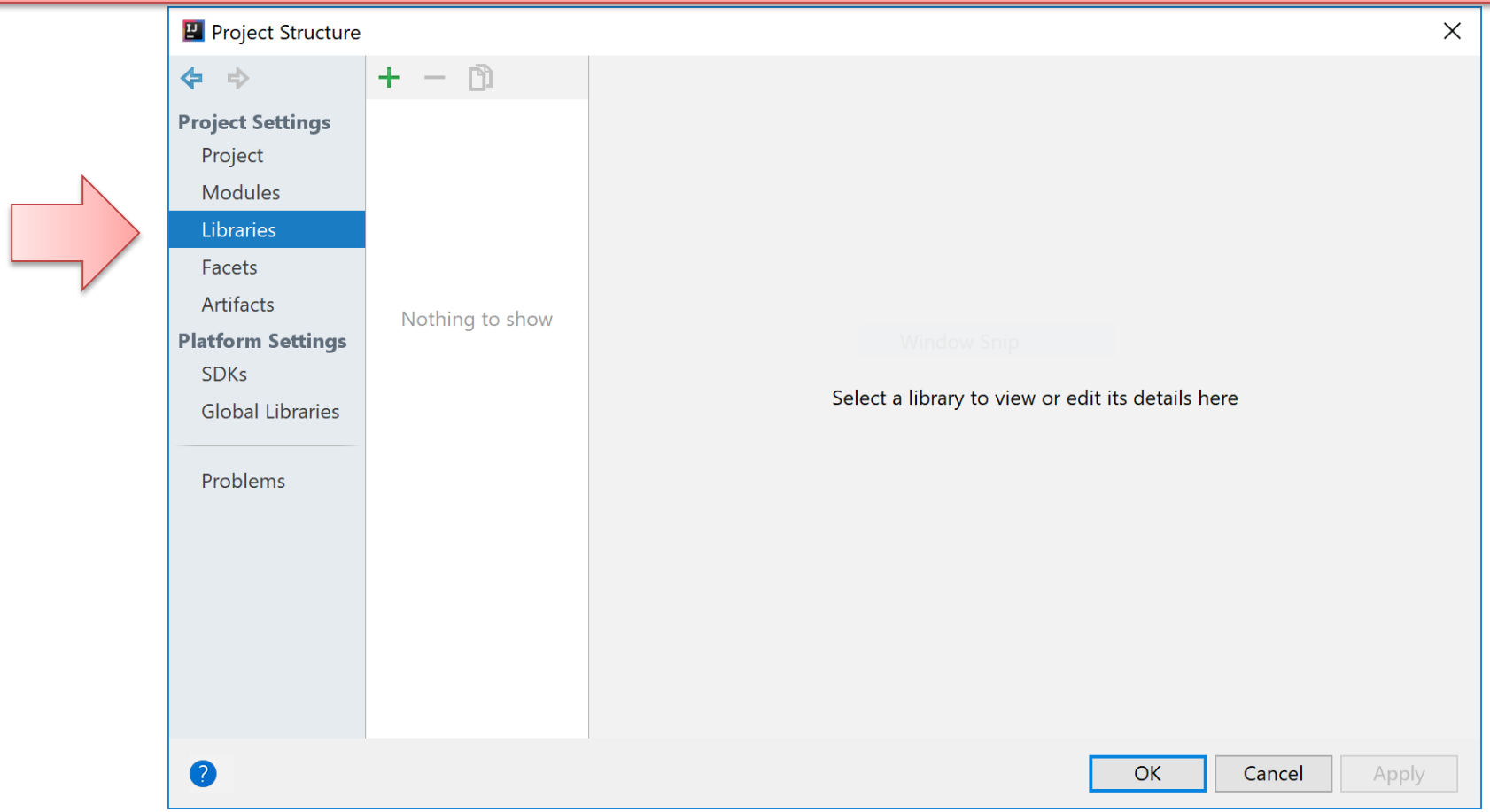




Drag the **xstream.jar** file into the **lib** folder.

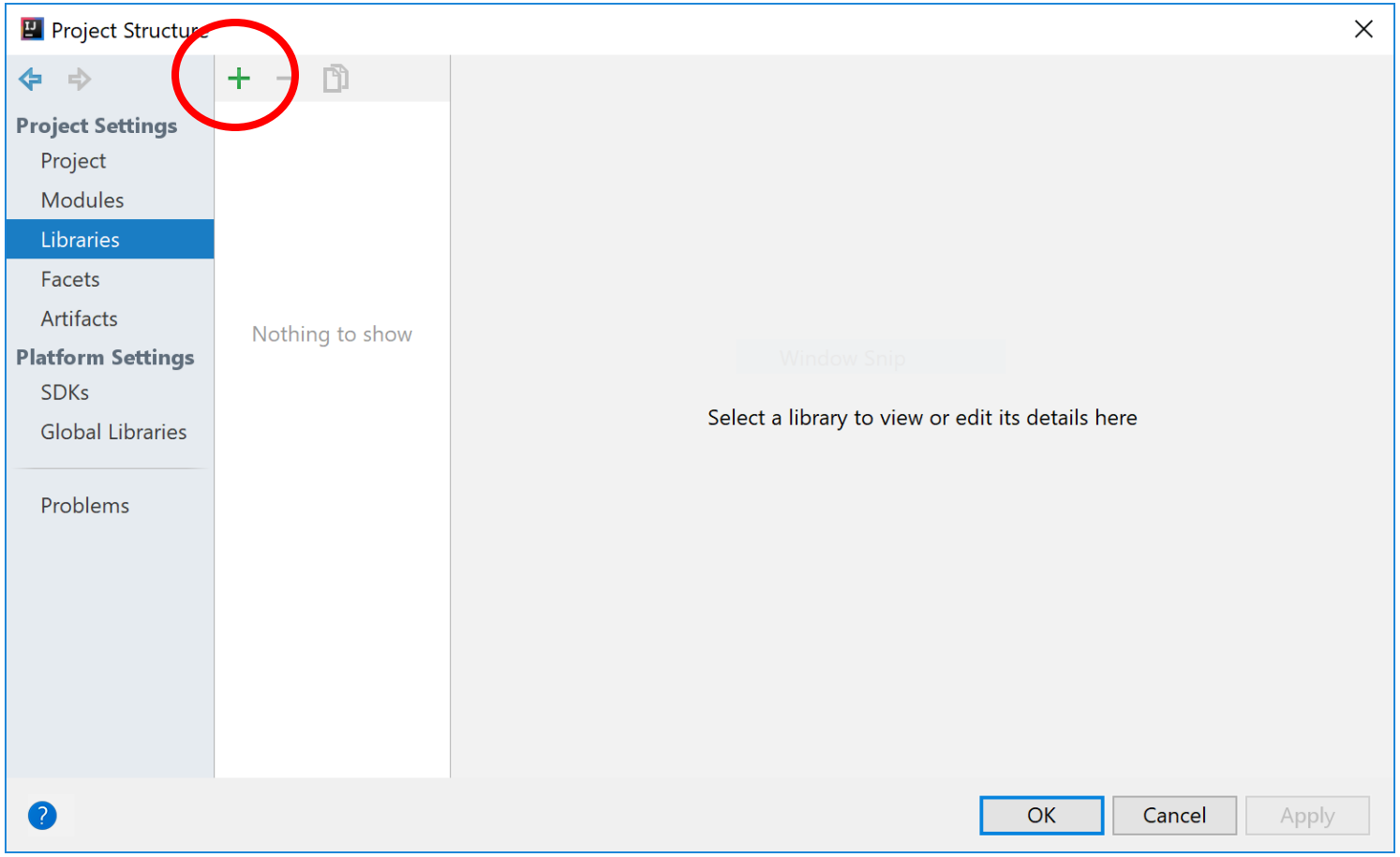
1.1 Adding the component to your **build path** - 1

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



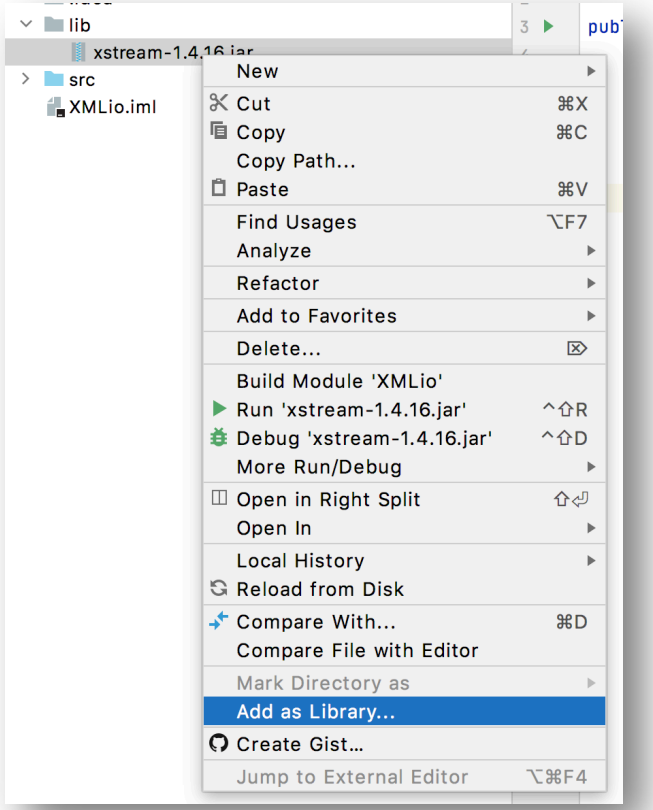
1.1 Adding the component to your **build path** - 2

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



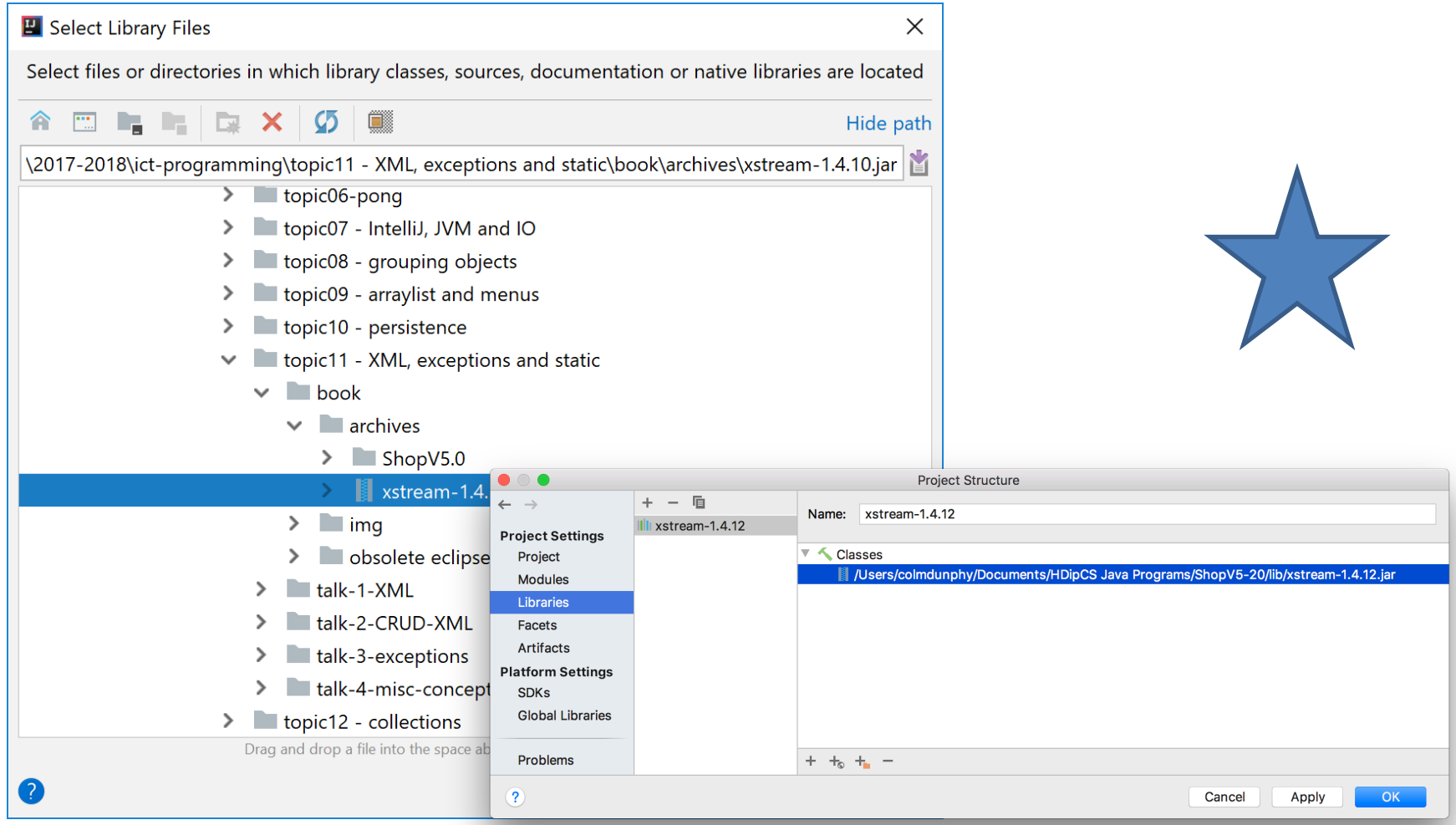
OR

Right click the .jar file and select “Add as Library”



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

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



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.

- Now that you have the library component (.jar) file installed in IntelliJ you can use it.

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

Security Updates

Store
.java

save()

```
@SuppressWarnings("unchecked")

    public void save() 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 by modifying the next line,
        // add additional classes inside the braces, comma separated

        Class<?>[] classes = new Class[] { Product.class };

        XStream.setupDefaultSecurity(xstream);
        xstream.allowTypes(classes);
        // ---------------------------------------------------------------------------
        ObjectOutputStream out = xstream.createObjectOutputStream
                                        (new FileWriter("products.xml"));
        out.writeObject(products);
        out.close();
    }
```

*Updated to remove JAVA security warning

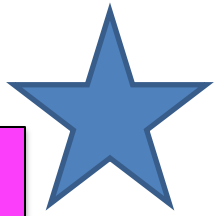
Store
.java

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 by modifying the next line,
    // add additional classes inside the braces, comma separated

    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

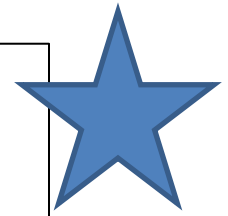


To use the **load()** & **save()** code in another project, change:

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**

Just add them at the top of the file

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


```
<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>
```

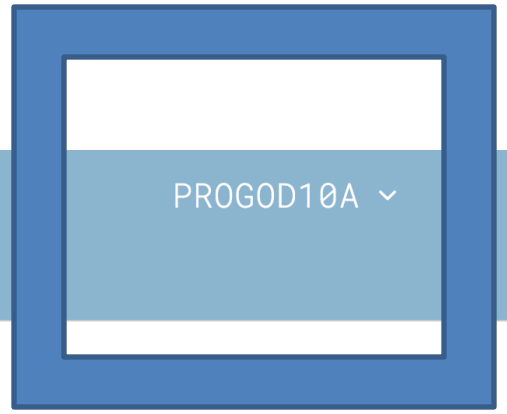
When the **save** option is selected from the menu, this **XML file** is created

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

Reading/Writing Other **Formats** Using Other **Libraries**

- Challenge, try using a library for **JSON** or **YAML** to read/write out your data to these formats
- Remember:
  - Download the library as a jar file
  - Add it to the **lib folder**
    - create it if it doesn't exist
    - and add it to the **build path**

For YAML try this: https://www.baeldung.com/java-snake-yaml



PROGOD10A

Colm

0/63

LAUNCH QUIZZES ROOMS REPORTS **RESULTS**

Prog Fund Week 10 unit 1

Finish

Show Names Show Responses Show Results

NAME	SCORE %	1	2	3	4	5	6	7	8	9	10	11	12
– •••••••	0%												
– •••••••	0%												
– •••••••	0%												

Any Questions?