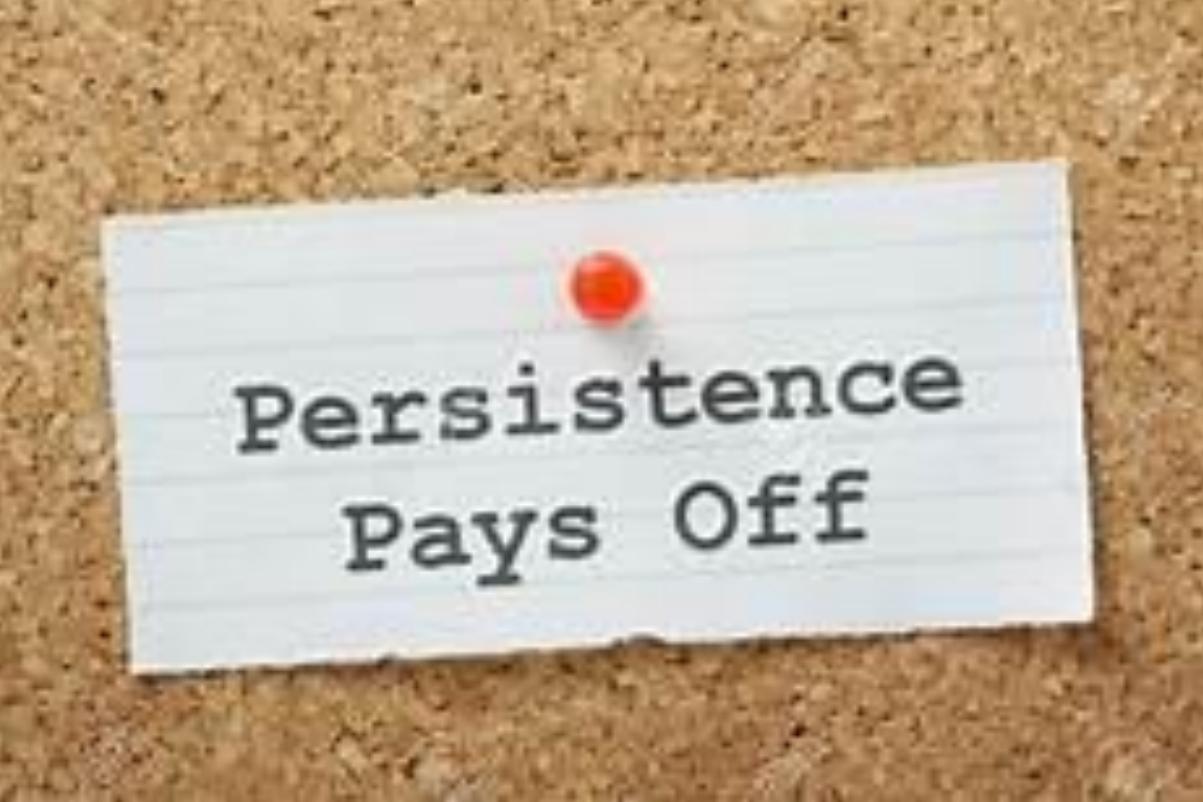


Persistence

Saving and retrieving objects to/from XML files

Produced Ms. Mairéad Meagher
by: Dr. Siobhán Drohan
 Ms Siobhan Roche



Persistence
Pays Off

Our Shop App



Shop V4.0 - implemented the **CRUD** process



Problem: All entered **data is lost** if we **close our application**

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.





Shop V5.0 (using XML) - STEPS

1. Download the **xstream-1.4.20.jar** component
 - Add it to your Shop project.

2. **Store** Class

- Write the load(), save() methods.

3. **Driver** Class

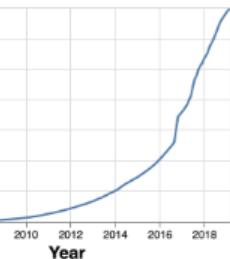
- include extra load and save functionality to the menu.

Download the component

https://mvnrepository.com/artifact/com.thoughtworks.xstream/xstream/1.4.20

EPOSITORY Search for groups, artifacts, categories

Artifacts (37.7M)



Home » com.thoughtworks.xstream » xstream » 1.4.20

XStream Core » 1.4.20

XStream is a simple and fast library to serialize objects with minimal footprint.

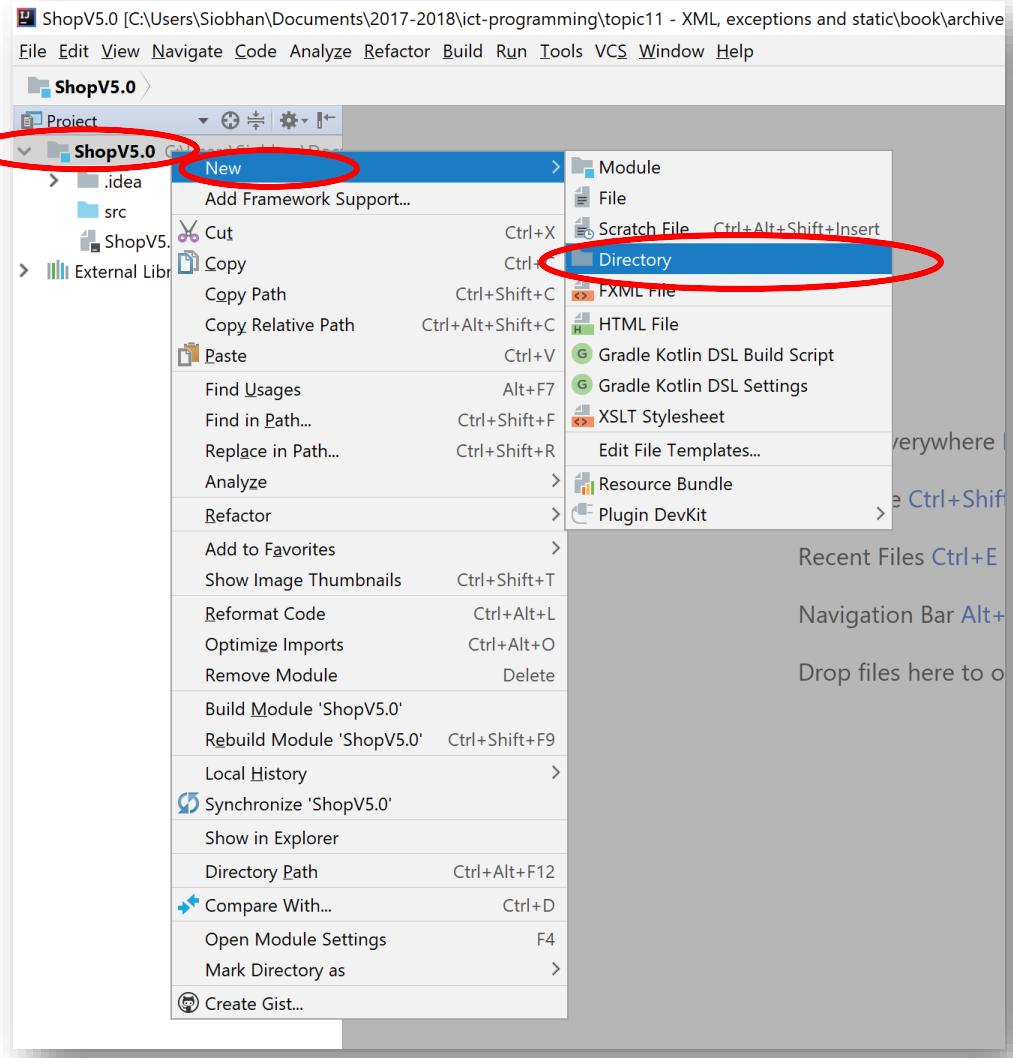
License	BSD 3-clause
Categories	XML Processing
Tags	xml processing
Date	Dec 24, 2022
Files	pom (24 KB) jar (629 KB) View All
Repositories	Central
Ranking	#199 in MvnRepository (See Top Artifacts) #6 in XML Processing

Download the **xstream-1.4.20.jar** component.

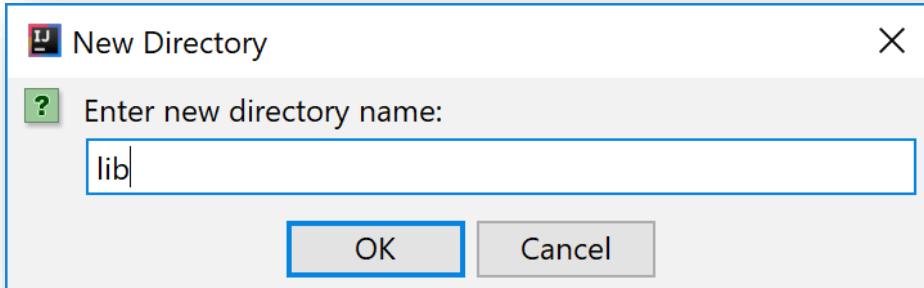
https://mvnrepository.com/artifact/com.thoughtworks.xstream/xstream/1.4.20

Adding a component to the lib folder -1

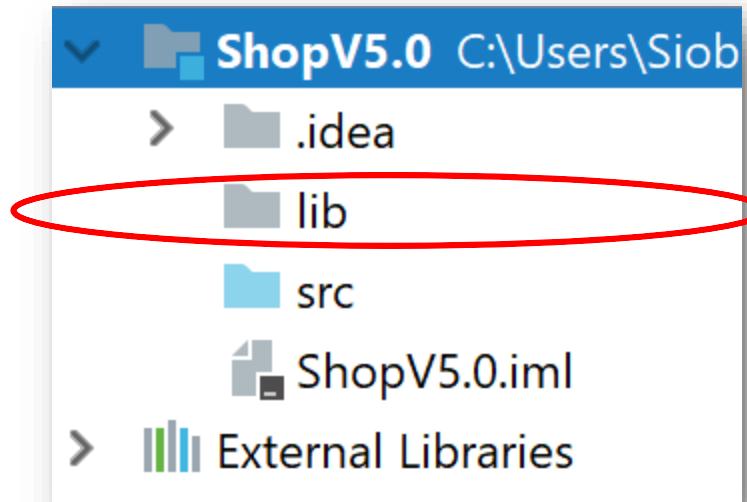
On the ShopV5.0 project...
right click
select “New”
then “Directory”.



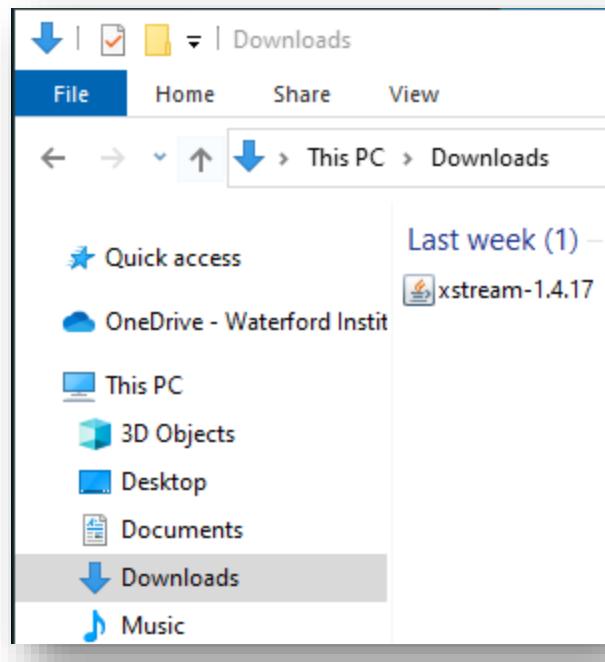
Adding a component to the lib folder - 2



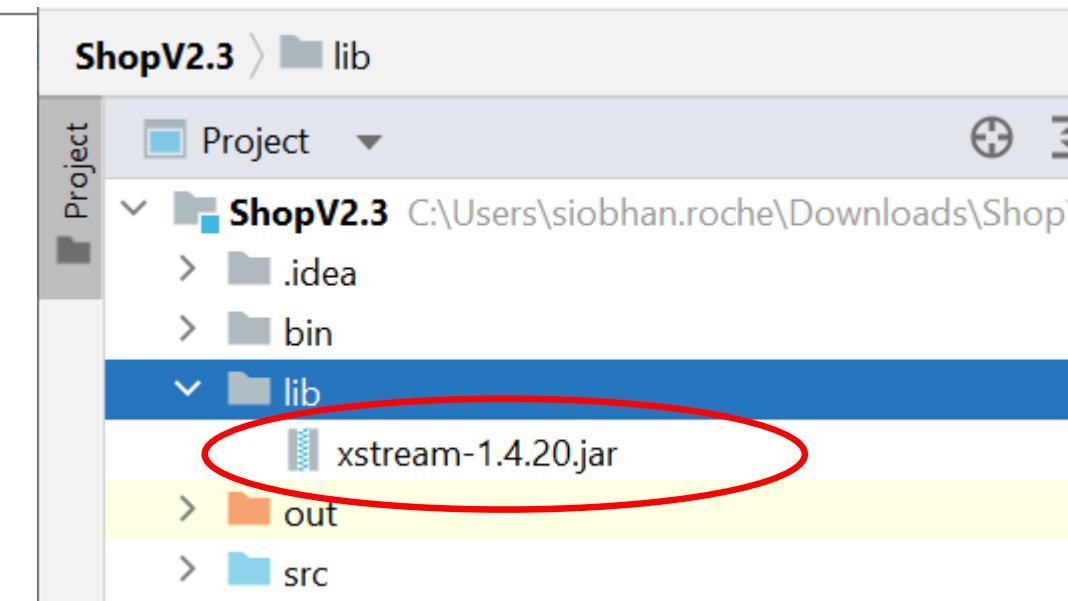
Call the new
directory “lib”.



Adding a component to the lib folder - 3

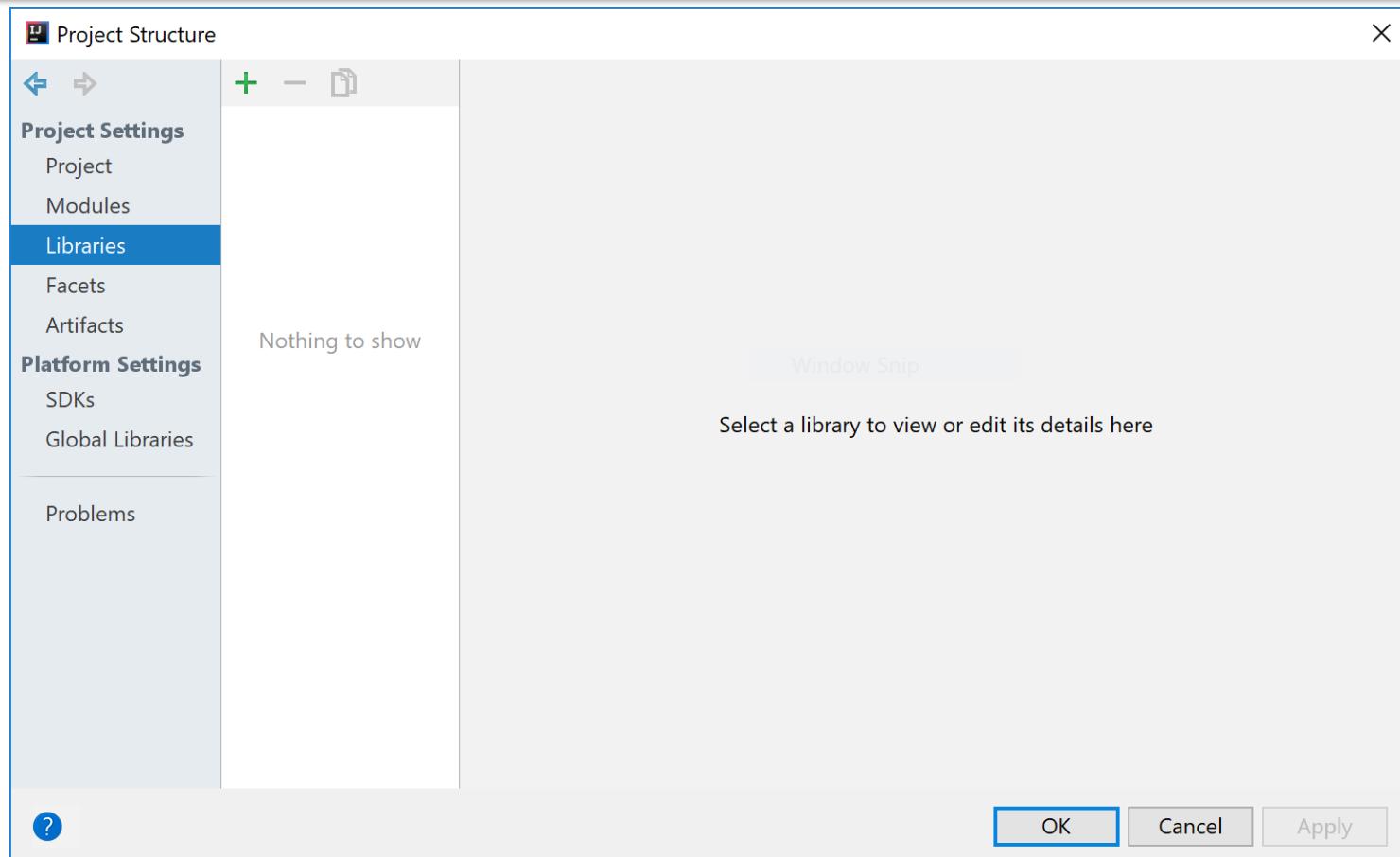


Drag the
xstream-1.4.20.jar
file into the lib
folder.



Adding the component to your build path - 1

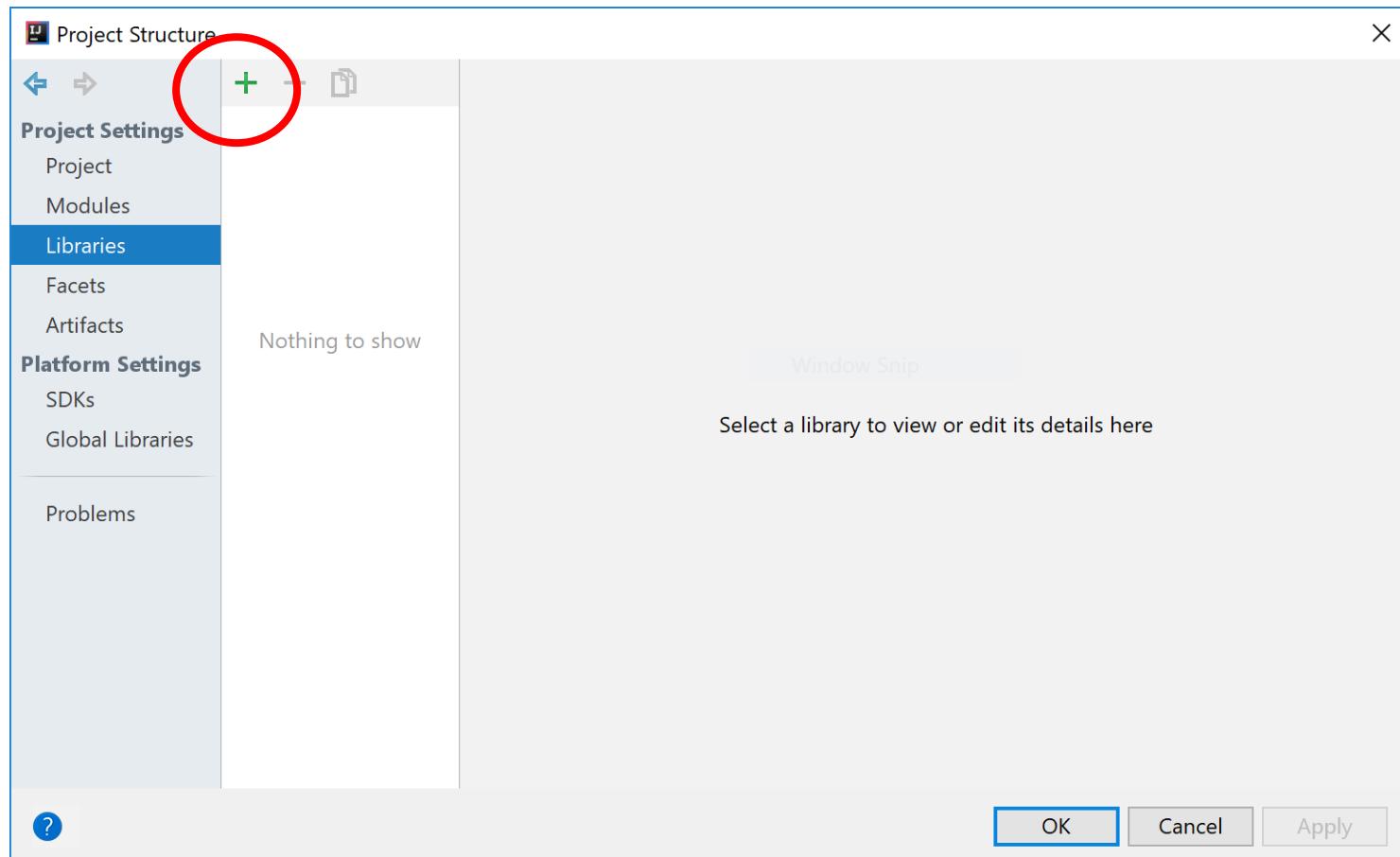
From **File** menu, select **Project Structure**. Click on **Libraries**.



Adding the component to your build path - 2

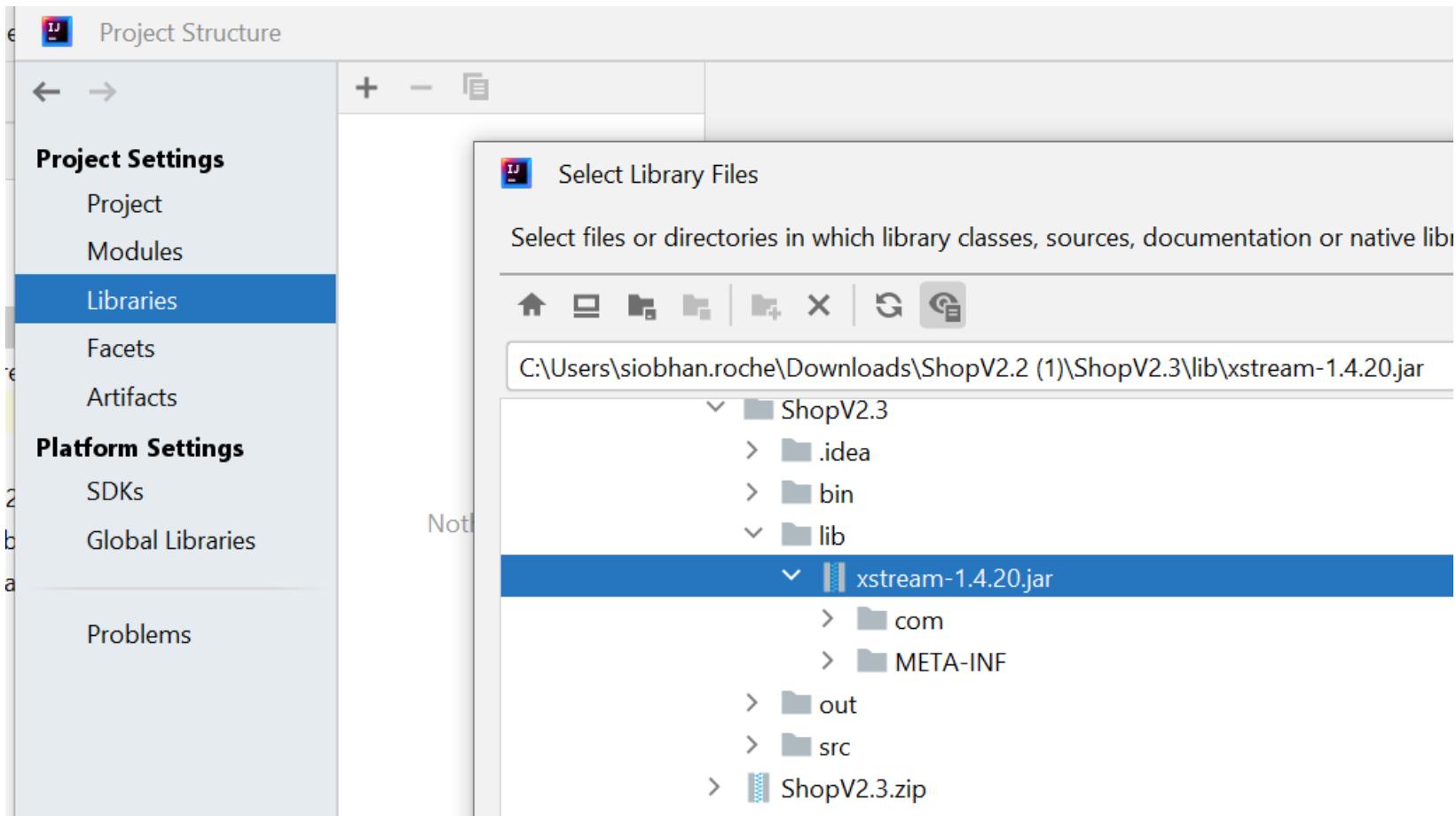
To add a library to
your build path:

click on the +



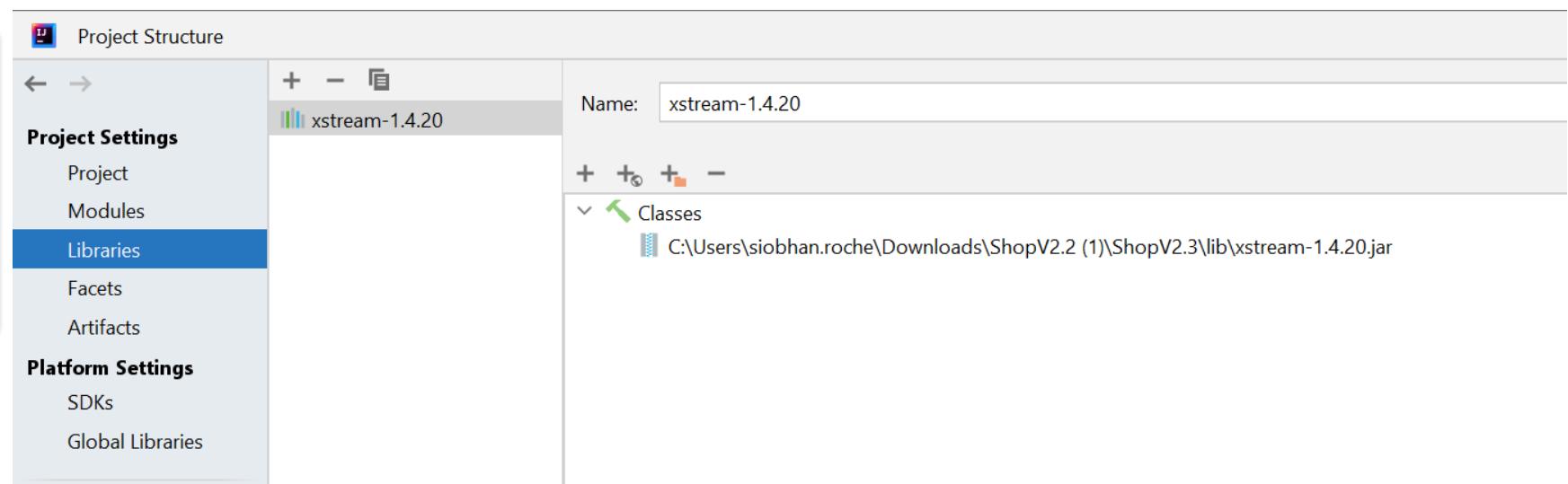
Adding the component to your build path - 3

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



Adding the component to your build path - 4

XStream is now
added as a library –
this means the
compiler can find it!





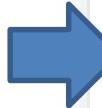
Shop V5.0 (using XML) - STEPS

1. Download the **xstream-1.4.17.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.



```
C  Store
m  Store()
m  getProducts(): ArrayList<Product>
m  numberOfRowsInSection(): int
m  isValidIndex(int): boolean
m  add(Product): boolean
m  findProduct(int): Product
m  listProducts(): String
m  deleteProduct(int): Product
m  updateProduct(int, String, int, double, boolean): boolean
m  cheapestProduct(): Product
m  listCurrentProducts(): String
m  averageProductPrice(): double
m  listProductsAboveAPrice(double): String
m  load(): void
m  save(): void
f  products: ArrayList<Product>
```

// 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

```
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
}
```

```
C Store
m Store()
m getProducts(): ArrayList<Product>
m numberOfProducts(): int
m isValidIndex(int): boolean
m add(Product): boolean
m findProduct(int): Product
m listProducts(): String
m deleteProduct(int): Product
m updateProduct(int, String, int, double, boolean): boolean
m cheapestProduct(): Product
m listCurrentProducts(): String
m averageProductPrice(): double
m listProductsAboveAPrice(double): String
m load(): void
m save(): void
f products: ArrayList<Product>
```

// 1. Provide a list of classes that you want to serialise
// 2. Initialise an XStream object and set up security rules
// 3. Use XStream object to initialize an ObjectInputStream from a specific file
// 4. Call the is.readObject() method to assign values to the object e.g. products
// 5. Close the stream / file

```
@SuppressWarnings("unchecked")
public void load() throws Exception {
    //list of classes that you wish to include in the serialisation, separated by a comma
    Class<?>[] classes = new Class[] { Product.class };    //1

    //setting up the xstream object with default security and the above classes
    XStream xstream = new XStream(new DomDriver());
    XStream.setupDefaultSecurity(xstream);
    xstream.allowTypes(classes);                                //2

    //doing the actual serialisation to an XML file
    ObjectInputStream is = xstream.createObjectInputStream(new
    FileReader("products.xml"));    //3
    products = (ArrayList<Product>) is.readObject();    //4
    is.close();    //5
}
```

```
C  Store
m  Store()
m  getProducts(): ArrayList<Product>
m  int numberOfProducts()
m  boolean isValidIndex(int)
m  boolean add(Product)
m  Product findProduct(int)
m  String listProducts()
m  Product deleteProduct(int)
m  boolean updateProduct(int, String, int, double, boolean)
m  Product cheapestProduct()
m  String listCurrentProducts()
m  double averageProductPrice()
m  String listProductsAboveAPrice(double)
m  void load()
m  void save()
f  ArrayList<Product> products
```

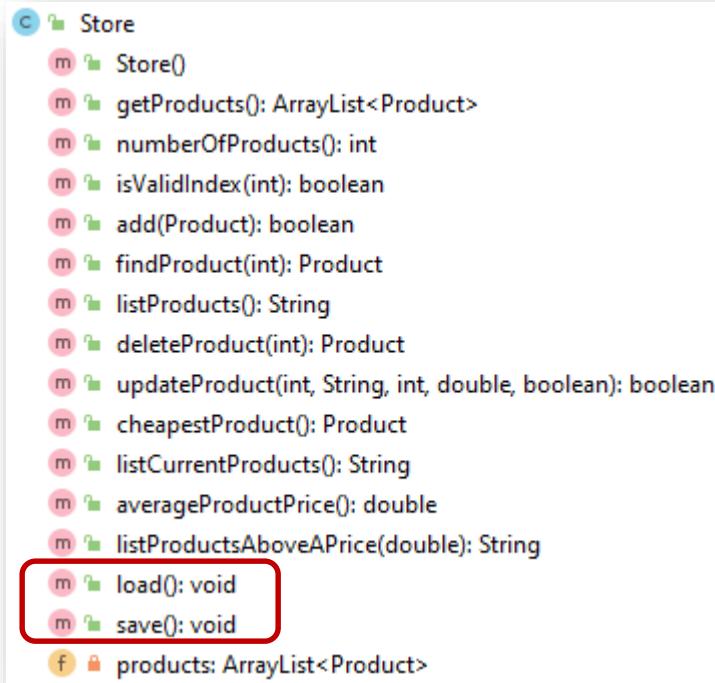
To use the **load()** & **save()** code in another project, change:

load()

1. the class **type** in the list of classes to serialise.
2. the **name** of the xml file.
3. the **type** of object stored in the ArrayList.
4. the **name** of the ArrayList object.

save()

1. the **name** of the xml file.
2. the **name** of the ArrayList object.



Required Packages

```
import com.thoughtworks.xstream.XStream;  
import com.thoughtworks.xstream.io.xml.DomDriver;  
  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.ObjectInputStream;  
import java.io.ObjectOutputStream;
```

Note: you need to import these additional packages.



Shop V5.0 (using XML) - STEPS

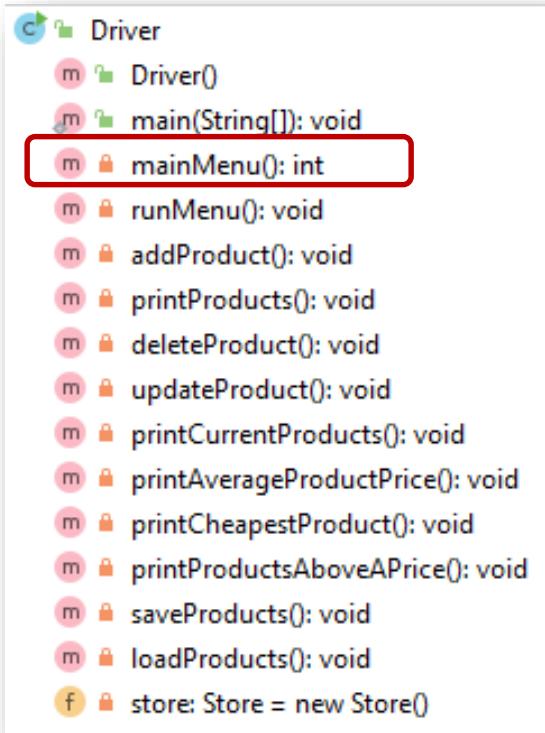
1. Download the **xstream-1.4.17.jar** component
 - Add it to your Shop project.

2. **Store** Class

- Write the load(), save() methods.

3. **Driver** Class

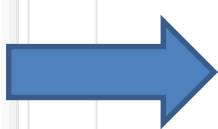
- include extra load and save functionality to the menu.

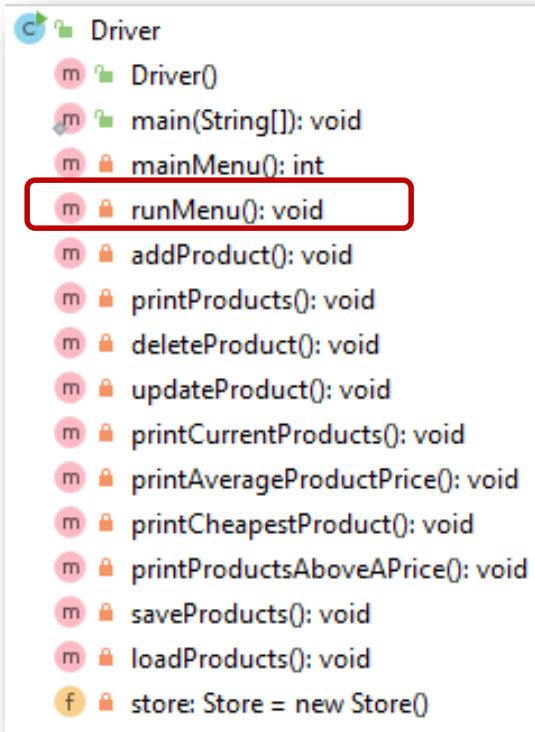


```
Driver
  m Driver()
  m main(String[]): void
  m mainMenu(): int
  m runMenu(): void
  m addProduct(): void
  m printProducts(): void
  m deleteProduct(): void
  m updateProduct(): void
  m printCurrentProducts(): void
  m printAverageProductPrice(): void
  m printCheapestProduct(): void
  m printProductsAboveAPrice(): void
  m saveProducts(): void
  m loadProducts(): void
  f store: Store = new Store()
```

Adding Save and Load functionality to the menu.

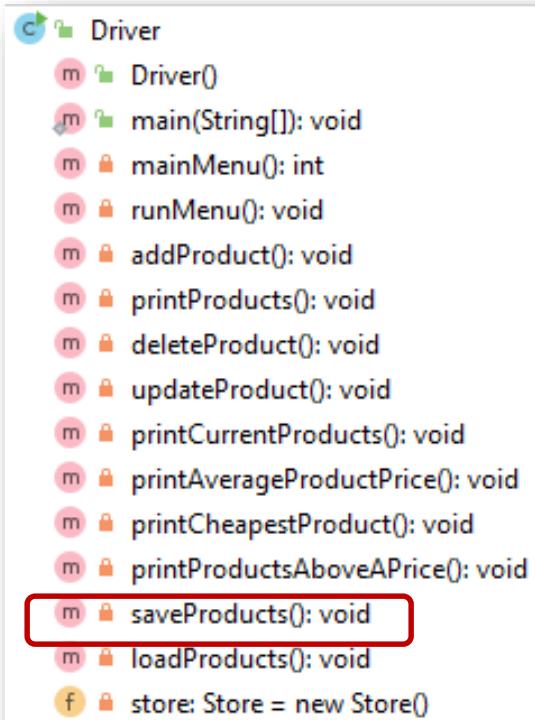
```
private int mainMenu() {
    return ScannerInput.readNextInt( prompt: """
    |                               Shop Menu
    | -----
    |   1) Add a product
    |   2) List the Products
    |   3) Update a product
    |   4) Delete a product
    | -----
    |   5) List the current products
    |   6) Display average product unit cost
    |   7) Display cheapest product
    |   8) List products that are more expensive than a given price
    | -----
    |   9) Save products to products.xml
    |  10) Load products from products.xml
    |  0) Exit
    | -----
    ==>> """ );
}
```





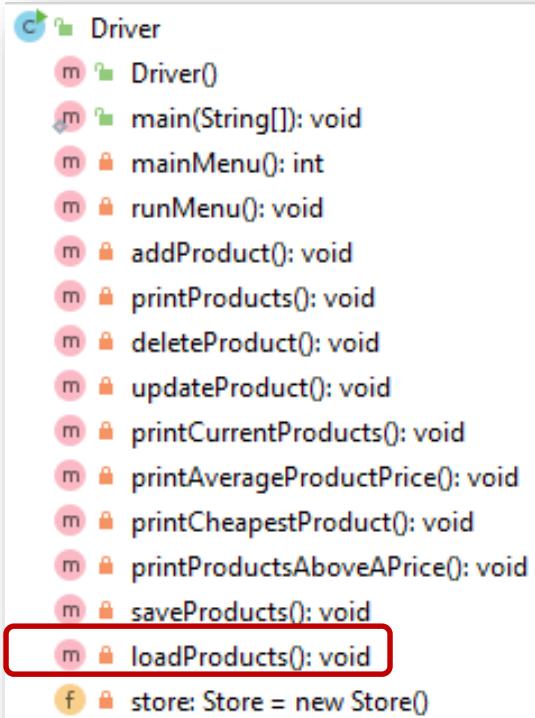
Adding **Save** and **Load** functionality to the menu.

```
switch (option) {  
    case 1 -> addProduct();  
    case 2 -> printProducts();  
    case 3 -> updateProduct();  
    case 4 -> deleteProduct();  
    case 5 -> printCurrentProducts();  
    case 6 -> printAverageProductPrice();  
    case 7 -> printCheapestProduct();  
    case 8 -> printProductsAboveAPrice();  
    case 9 -> saveProducts();  
    case 10 -> loadProducts();  
    default -> System.out.println("Invalid option entered: " + option);  
}
```



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

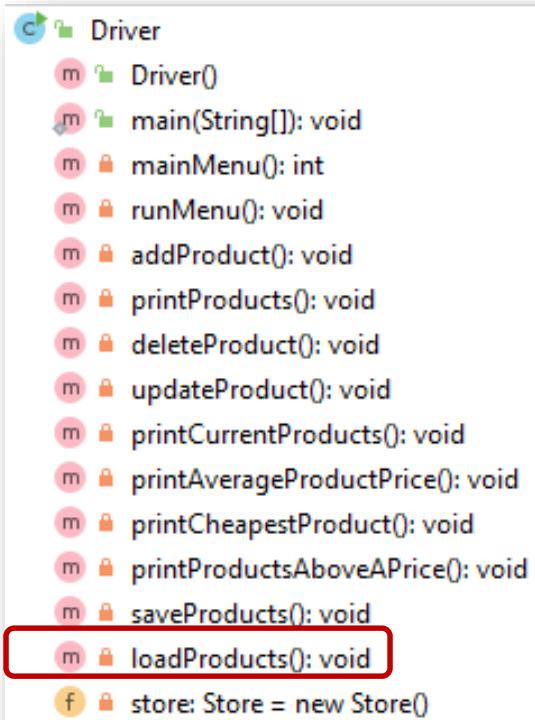
```
//save all the products in the store to a file on the hard disk
private void saveProducts() {
    try {
        store.save();
    } catch (Exception e) {
        System.err.println("Error writing to file: " + e);
    }
}
```



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

```
//save all the products in the store to a file on the hard disk
private void saveProducts() {
    try {
        store.save();
    } catch (Exception e) {
        System.err.println("Error writing to file: " + e);
    }
}
```

```
//load all the products into the store from a file on the hard disk
private void loadProducts() {
    try {
        store.load();
    } catch (Exception e) {
        System.err.println("Error reading from file: " + e);
    }
}
```



try/catch

The `save()` and `load()` methods in the `Store` class can throw an Exception. So, if we want to use, say, the `load()`, method, we need to write code to “handle” an Exception being thrown. This is where `try/catch` comes in and we will look at this construct in the next slide deck.

```
//load all the products into the store from a file on the hard disk
private void loadProducts() {
    try {
        store.load();
    } catch (Exception e) {
        System.err.println("Error reading from file: " + e);
    }
}
```

A screenshot of an XML editor window titled "products.xml". The code is displayed in a syntax-highlighted format. The XML structure includes an "object-stream" root element, which contains a "list" element. Inside the "list" element, there are two "Product" elements, each with attributes for productName, productCode, unitCost, and inCurrentProductLine. The code is numbered from 1 to 16.

```
<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>23321</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**.

Self Study Questions

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

**Any
Questions?**

