Packages in Java

Using a derivation of the Model View Controller

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

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

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

Using Packages

Evolving Shop to use a Package Structure

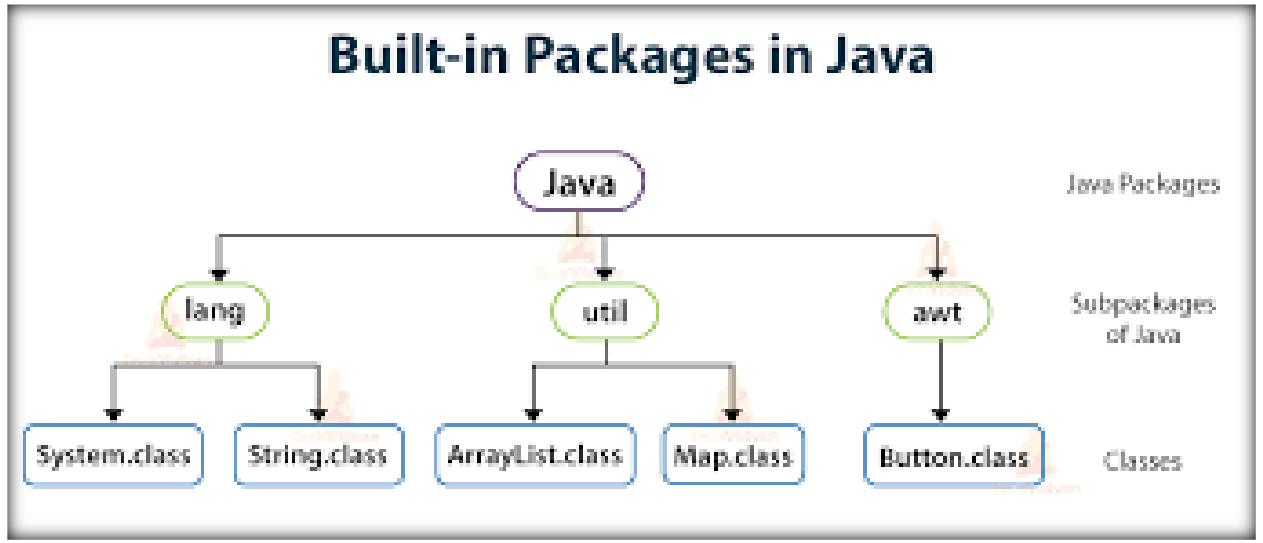
Why Packages?

- We use folders or directories in our computers for the classification and accessibility of various files;

- In Java, Packages are similar to folders, which are mainly used to organize classes and interfaces(we see interfaces later).

Why Packages?

- Packages help us to write better and manageable code by preventing naming conflicts. Java provides some built-in packages which we can use:



Why Packages?

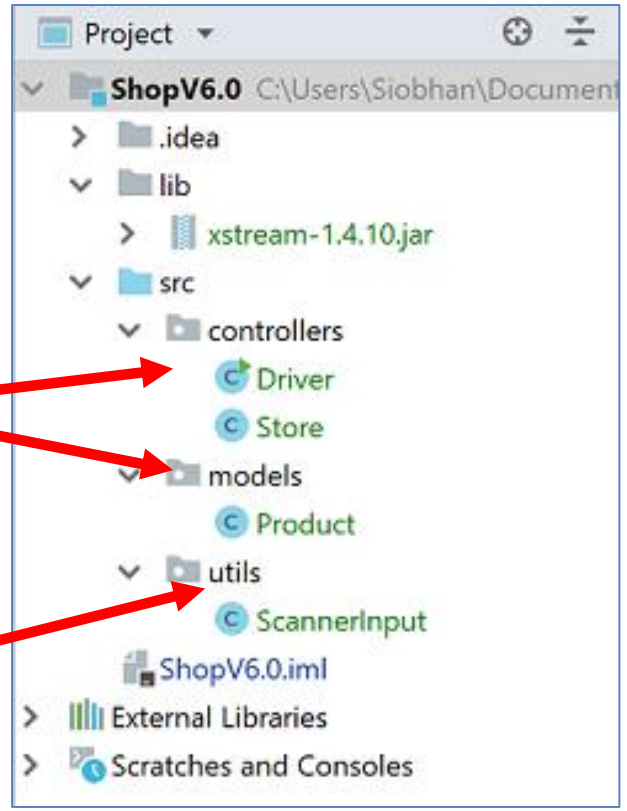
- We can similarly structure our Projects to categorise our classes.
- So how should we categorise our classes?
- Much work has been done on ‘Design Patterns’

Design Patterns

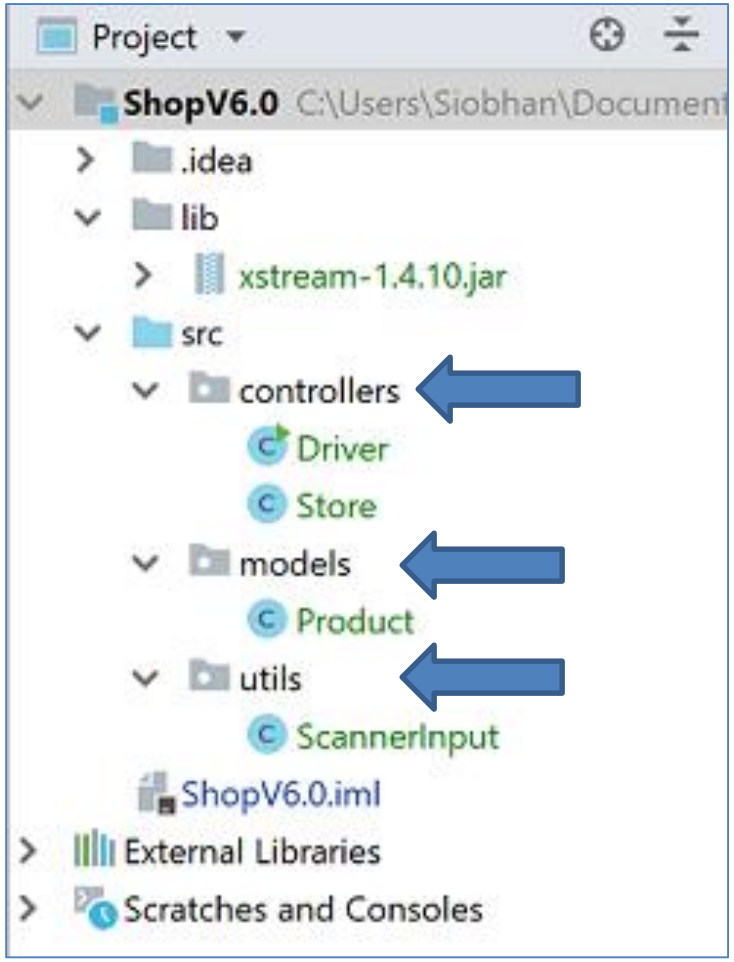
- A Design Pattern is an important concept in Computer Science.
- **A software design pattern** is a general, reusable solution to a commonly occurring problem within a given context in software design. (Wikipedia)
- Model-View Controller is an important one

Our Structure

- We break down our classes into the following sub-packages:
- **models** – has the 'bottom level' classes, e.g. Product
- **controllers** – this contains the classes that do the 'heavy lifting' and the user interface classes
- **utils** – this contains any Utility classes



Shop – packages



- We now look at the changes we need to make in the ‘packaged’ classes.

Accessing Packages or Classes from Another Package

If we want to access all the classes and interfaces of an existing package then we use the **import** statement. We can do it in three different ways:

- import package.*;
  - By using * after the import statement, we can access all the classes of the package but not the sub-packages.
- import package.classname;
- fully qualified name.

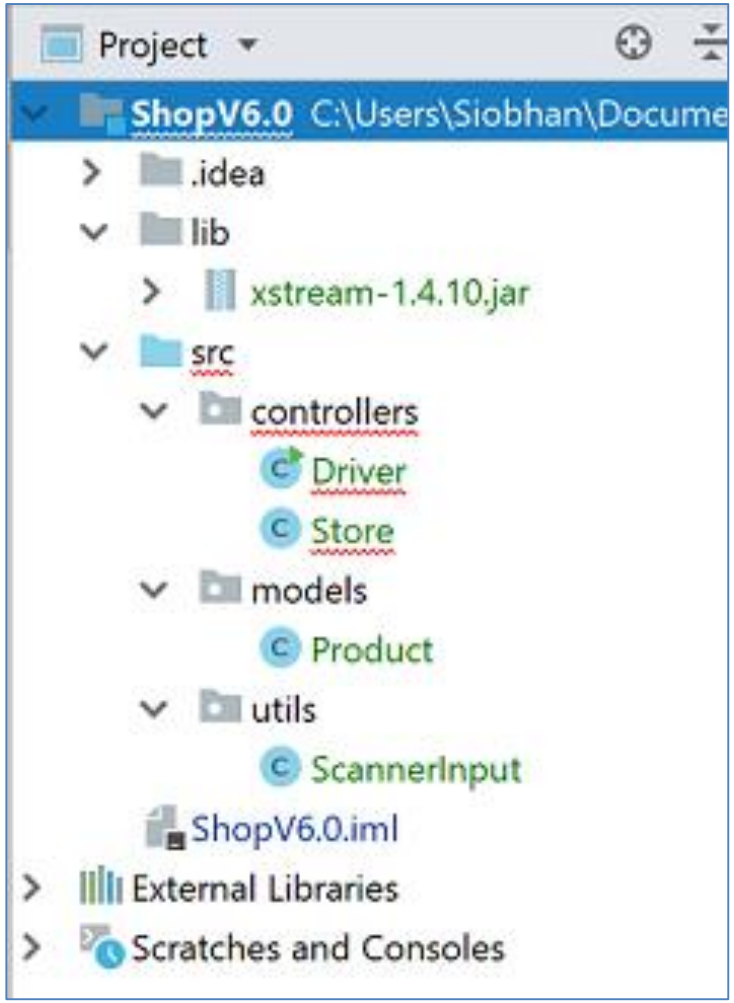
Shop – utilities and packages

- Refactor your Shop project to create this package structure.

  - Right-click on the **src** folder and select New → Package. Enter “models” as the package name.

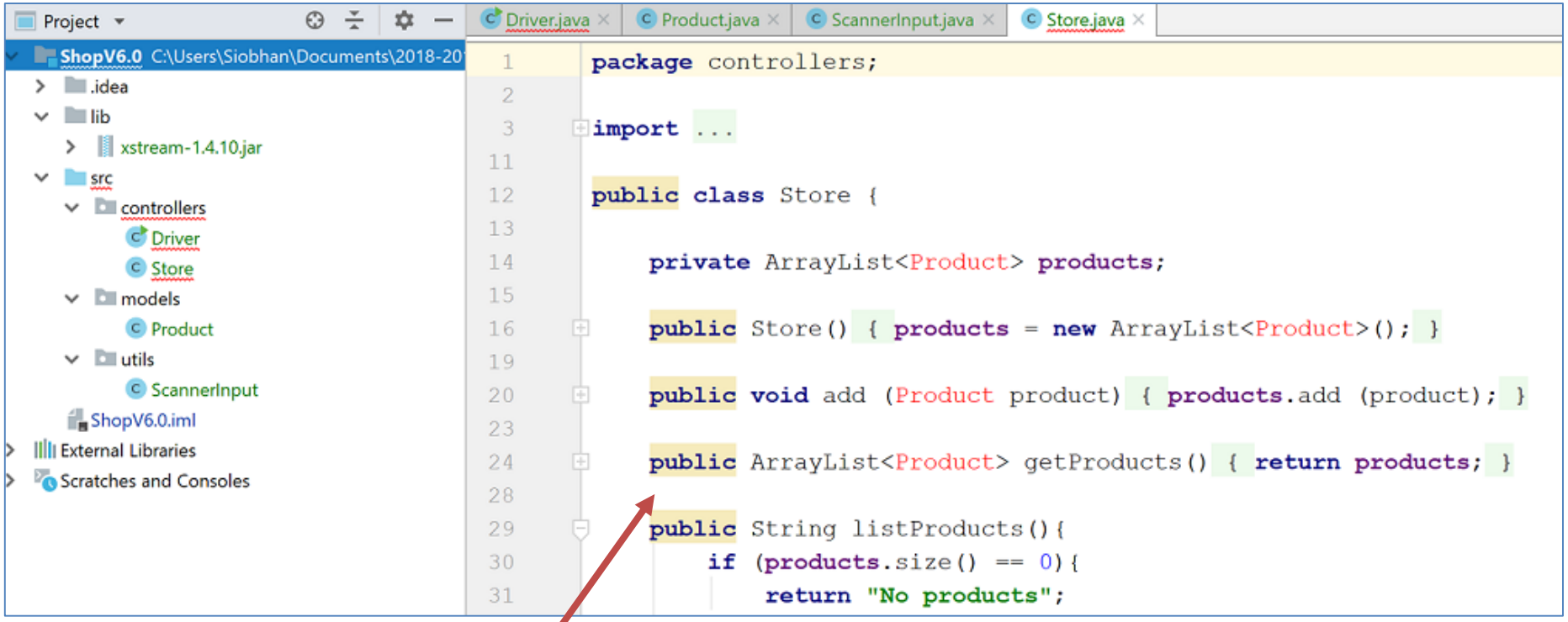
  - Repeat this process and create two other packages called “controllers” and “utils”.

Shop – utilities and packages



Copy the Shop classes into package locations specified in the screen shot.

Shop – utilities and packages



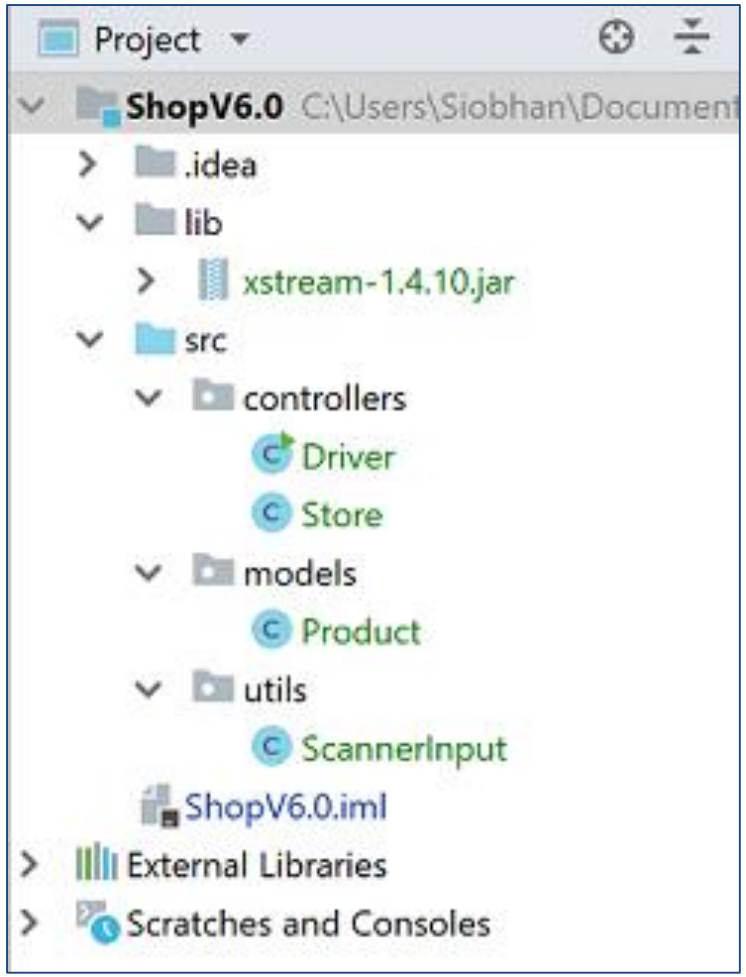
This move caused the following error: The Product class can't be found in the Driver and Store class.

Shop – utilities and packages

To fix this error, use IntelliJ's Alt + Enter to:

- import **models.Product;** into Driver and Store classes.
- import **utils.ScannerInput;** into Driver.

Shop – utilities and packages



- The errors are now gone.
- Test the app to make sure it is running as expected.
- We will use packages in this way for the remainder of the module.

Any Questions?