Mittuniversitetet
MID SWEDEN UNIVERSITY

Assignment 5

Java for C ++ programmers, 7,5 hp

Objective:	To use classes in JAXB to save and read objects to and from XML files.
To read:	Lecture 6
Tasks:	1
Submission:	Inlämningslåda 5 at Moodle

Task 1

In this task, you will start with your solution from Lab 4.
Create a new Java project in Eclipse, which will be a copy from your previous solution. Your classes should belong to the package:
`dt062g.studentid.assignment5` where studentid is your username in the student portal / Moodle.
The program should be completely text-based (use System.in and System.out for input and output from / to the user).
<u>NOTE! Inputs and outputs may only be made in the class containing the main method (if it is not explicitly stated something else in the task).</u>

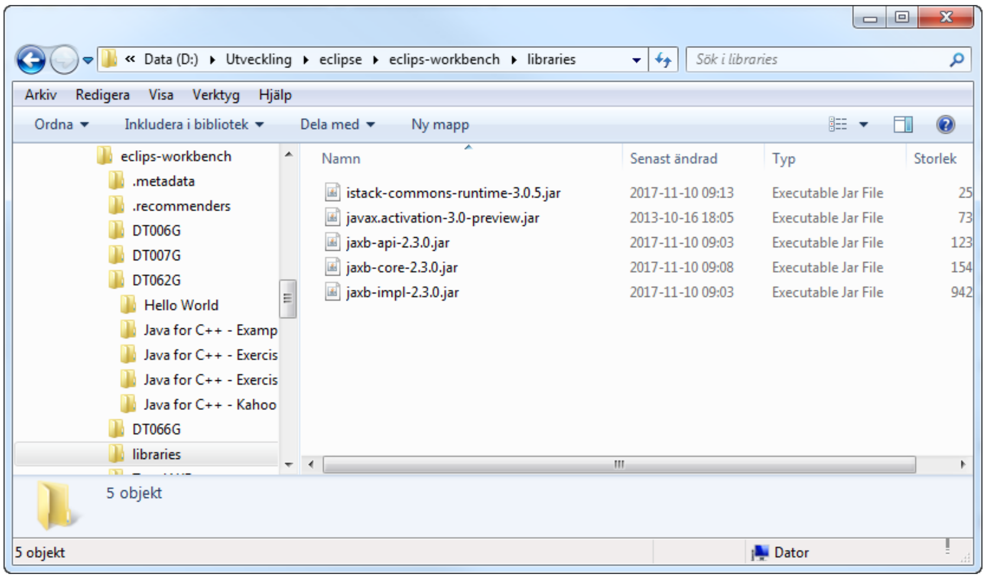
Each class and interface (interface) that you create must be documented with comments. Ex.

```
/**
* A short description (in Swedish or English) of the class.
*
* @author  Your Name (your student id)
* @version 1.0
* @since   yyyy-mm-dd (last edited)
*/
```

Preparation

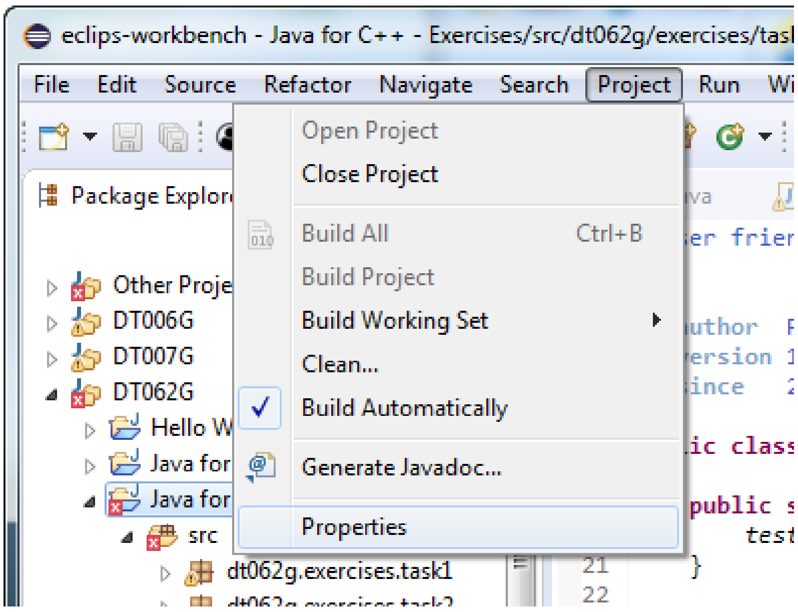
To use JAXB, you need to add some external libraries to your project. Along with lab description, a zip file is attached that contains the necessary libraries.

Start by creating a directory/folder called libraries in your directory for the workspace you are using in Eclipse. If you are unsure of which directory it is, you can find it by going to Eclipse menu option File -> Switch Workspace -> Other ... Copy all files from libraries.zip to the directory libraries you created.

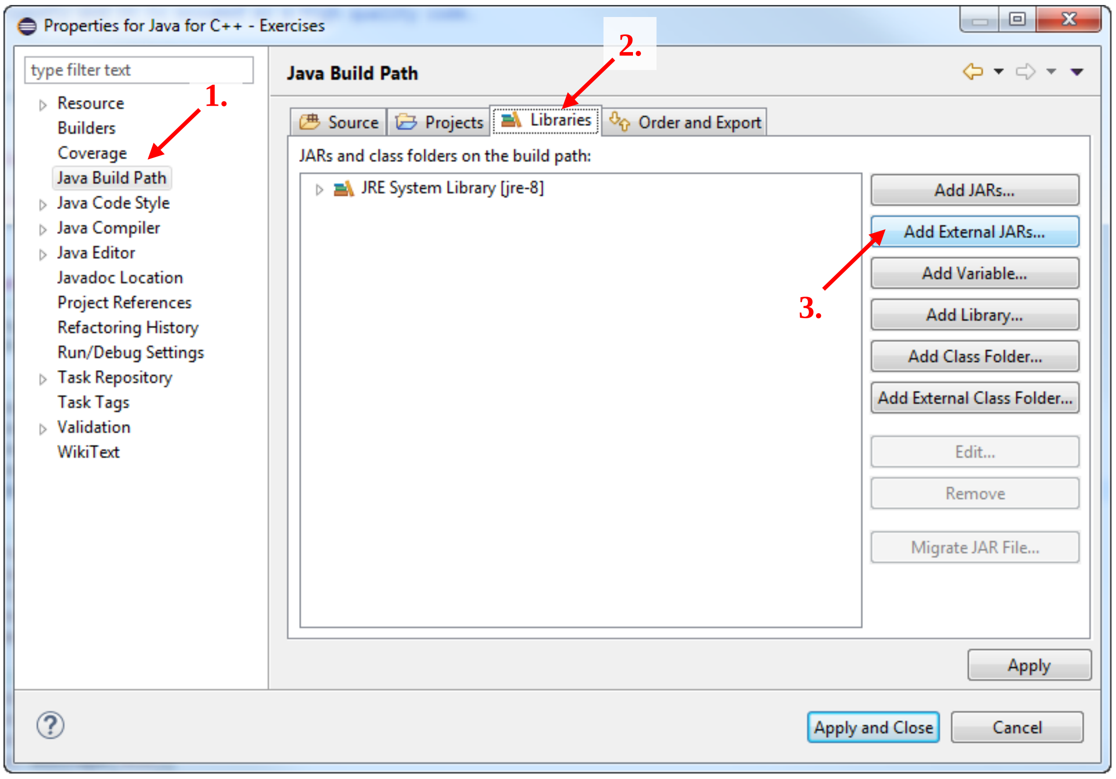


Namn	Senast ändrad	Typ	Storlek
istack-commons-runtime-3.0.5.jar	2017-11-10 09:13	Executable Jar File	25
javax.activation-3.0-preview.jar	2013-10-16 18:05	Executable Jar File	73
jaxb-api-2.3.0.jar	2017-11-10 09:03	Executable Jar File	123
jaxb-core-2.3.0.jar	2017-11-10 09:08	Executable Jar File	154
jaxb-impl-2.3.0.jar	2017-11-10 09:03	Executable Jar File	942

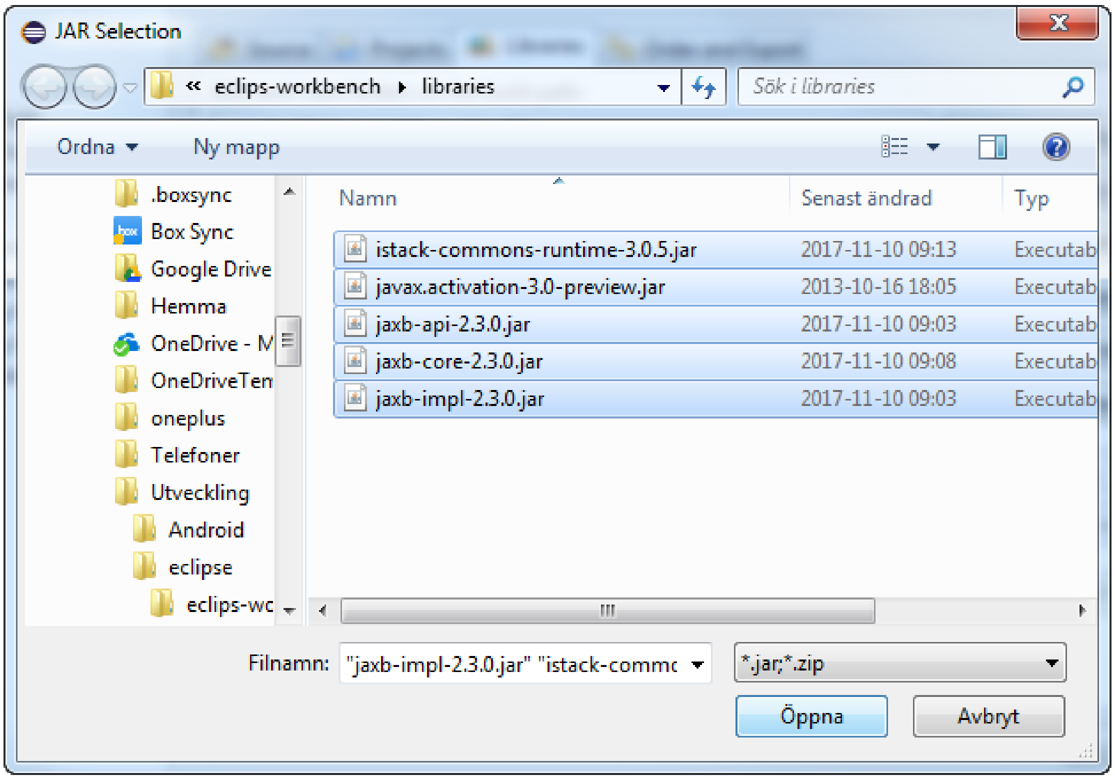
Now select your project in the Project Explorer and select the menu option Project -> Properties.



Click on Java Build Path, followed by the Libraries tab and then the Add External Jars



Go to the directory libraries that you created. Then select all five jar files and click the Open button.

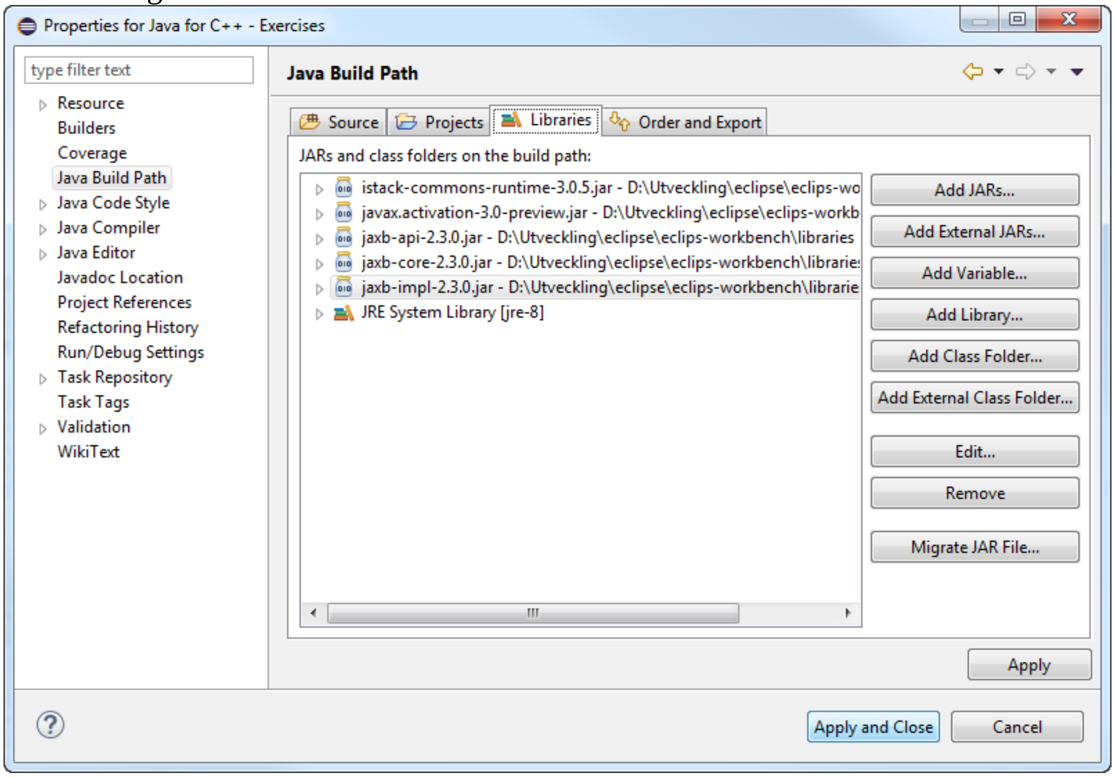


Namn	Senast ändrad	Typ
istack-commons-runtime-3.0.5.jar	2017-11-10 09:13	Executab
javax.activation-3.0-preview.jar	2013-10-16 18:05	Executab
jaxb-api-2.3.0.jar	2017-11-10 09:03	Executab
jaxb-core-2.3.0.jar	2017-11-10 09:08	Executab
jaxb-impl-2.3.0.jar	2017-11-10 09:03	Executab

Filnamn: "jaxb-impl-2.3.0.jar" "istack-commc

.jar;.zip

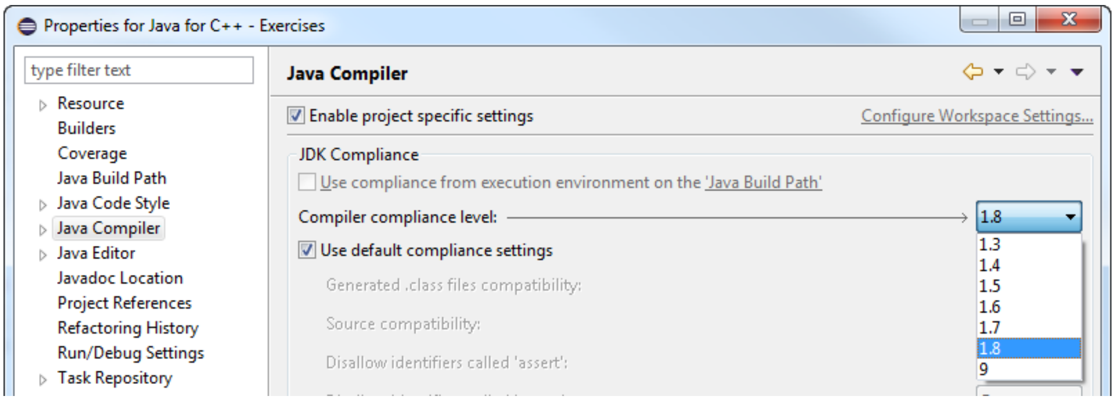
Your settings should now look like that.



JARs and class folders on the build path:

```
istack-commons-runtime-3.0.5.jar - D:\Utveckling\eclipse\eclips-wo
javax.activation-3.0-preview.jar - D:\Utveckling\eclipse\eclips-workb
jaxb-api-2.3.0.jar - D:\Utveckling\eclipse\eclips-workbench\libraries
jaxb-core-2.3.0.jar - D:\Utveckling\eclipse\eclips-workbench\librarie
jaxb-impl-2.3.0.jar - D:\Utveckling\eclipse\eclips-workbench\librarie
JRE System Library [jre-8]
```

I found that Java 9 could create some strange problems and error messages. If you encounter such problems, please change the project settings so that Java 8 is used instead (click on Java Compiler).


Properties for Java for C++ - Exercises

Java Compiler

[✓] Enable project specific settings

JDK Compliance

[ ] Use compliance from execution environment on the 'Java Build Path'

Compiler compliance level: 1.8

[✓] Use default compliance settings

Generated .class files compatibility:

Source compatibility:

Disallow identifiers called 'assert':

Press the Apply and Close button when you are finished with your settings.

Assignment

Now you can start with the actual task. The idea is that an object of the `Drawing` class and all other classes used in this, together with all values of the instance variables, should be able to be saved to an XML file. It should also be possible to load data from an existing XML file and create a `Drawing`-object of that.

All the class that uses the `Drawing`, which you have created, you must annotate with `@XmlRootElement`. In all of these classes, you must also add an empty/default constructor without arguments. This is because JAXB call this constructor when the objects to be loaded from file.
In a class that has instance variables whose values are to be saved/read to XML, must be annotated with `@XmlAccessorType(XmlAccessType.FIELD)`. This indicates that we will annotate the instance variables and not the methods for specifying what to save. All instance variables to be saved are then annotated with `@XmlElement` or `@XmlAttribute`. The difference is if data is to be saved as a separate element or as an attribute. This is because JAXB uses set methods to re-create the objects and their values, and the GET method is used to retrieve the objects' values when they are saved to a file.

For an abstract super class like Shape, the class must also be annotated with `@XmlSeeAlso({MySubClass1.class, MySubClasss2.class})` so that JAXB will know which subclasses are available.

Point

Annotate the class with `@XmlRootElement` and add an empty/default constructor without arguments.
Annotate the class with `@XmlAccessorType(XmlAccessType.FIELD)` and then annotate the instance variables x and y with `@XmlElement`. Make sure that there are get and set methods for each instance variables.

Circle

Annotate the class with `@XmlRootElement` and add an empty/default constructor without arguments. If you declared pi as an instance variable, you may need to annotate it with `@XmlTransient` to prevent it from becoming an element when saving to XML.

Rectangle

Annotate the class with `@XmlRootElement` and add an empty/default constructor without arguments.

Shape

Start by annotating the class with `@XmlRootElement` and add an empty/default constructor without arguments.
Annotate the class even with `@XmlAccessorType(XmlAccessType.FIELD)` and then annotate instance variables `color` and `points` with `@XmlElement`. For instances variable `points`, you must "rename" the item to points. Make sure that there are get and set methods for each instance variables.
Annotate the class even with `@XmlSeeAlso` and add the both subclasses to a comma-separated list.

Drawing

Start by annotating the class with `@XmlRootElement` and add an empty/default constructor without arguments.
Annotate the class even with `@XmlAccessorType(XmlAccessType.FIELD)` and then annotate instance variables `name`, `author` and `shapes` with `@XmlElement`. For instances variable `shapes`, you must "rename" the item to shape. Make sure that there are get and set methods for all three instance variables.

Finally add a method in this class called `clear`. When this method is called, the list `shapes` should be set to empty and `name` and the `author` will be set to an empty string ("").

FileHandler

Create a new class called `FileHandler`. This class will save and load `Drawing`-object using JAXB. The class should have the following static and public methods:

- `saveToXML(Drawing drawing, String fileName)`

  The method should begin by checking that `fileName` is a string that ends with `.xml`. If this is not the case, `.xml` should be added last in the `fileName`.

  The method should use `JAXBContext` and `Marshaller` to save the argument `drawing` to the file `fileName`. Catch any errors that may be thrown and print an appropriate error message with `System.err`.

- `saveToXML(Drawing drawing)`

  The method should function as the above method with the difference that the filename should be generated automatically based on the name and author of the drawing according to:
  `name by author.xml`

  Remember to avoid repeating code in these methods. Instead, let one method call the other and thus avoid any form of repetition.

- `loadFromXML(String fileName)` which have return type `Drawing`. The method should use `JAXBContext` and `Unmarshaller` to load a Drawing-object from the file `fileName`. Catch any errors that may be thrown and print an appropriate error message with `System.err`.

Assignment5

This class is attached to the task description. The class is meant to demonstrate the use of the `FileHandler`. However, the class is not fully completed. Your task is to complete the class by adding code only under the following comments:

```
// Create a drawing with a name and author.
// Create at least 5 shapes with end points
// Add shapes to the drawing
```

In addition, you will give the `fileName` variable a suitable filename to save your drawing. Also change the information about package, author, version, and date as usual.
Mona Lisa.xml file is attached with the task. Try if your solution can load this file and provide the same prints as in the example on the next page. If you have the possibility, you can also borrow XML files from your classmates.

The next page shows an example of printout while running the test program. The important thing is that your solution should give the same printouts of the drawing (call to `d1.draw ()`) before and after you save and upload your drawing to/from file.

```
Create a drawing...

A drawing by L. da Vincis called Mona Lisa
Circle[start=100.0, 100.0; end=175.0, 100.0; radius=75.0;
color=ffe0bd]
Circle[start=75.0, 75.0; end=85.0, 75.0; radius=10.0; color=0000ff]
Circle[start=125.0, 75.0; end=135.0, 75.0; radius=10.0; color=0000ff]
Rectangle[start=95.0, 100.0; end=105.0, 115.0; width=10.0;
height=15.0; color=000000]
Rectangle[start=55.0, 130.0; end=145.0, 140.0; width=90.0;
height=10.0; color=ff0000]

Save the drawing to Mona Lisa.xml...

Clearing the drawing and then draw it....
A drawing by  called

Load drawing from Mona Lisa.xml...

A drawing by L. da Vincis called Mona Lisa
Circle[start=100.0, 100.0; end=175.0, 100.0; radius=75.0;
color=ffe0bd]
Circle[start=75.0, 75.0; end=85.0, 75.0; radius=10.0; color=0000ff]
Circle[start=125.0, 75.0; end=135.0, 75.0; radius=10.0; color=0000ff]
Rectangle[start=95.0, 100.0; end=105.0, 115.0; width=10.0;
height=15.0; color=000000]
Rectangle[start=55.0, 130.0; end=145.0, 140.0; width=90.0;
height=10.0; color=ff0000]
```

The content of Mona Lisa.xml is shown below. Your solution should create XML files with the same structure. The same name for elements as below should be used.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<drawing>
    <name>Mona Lisa</name>
    <author>L. da Vincis</author>
    <shape xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="circle">
        <color>ffe0bd</color>
        <point>
            <x>100.0</x>
            <y>100.0</y>
        </point>
        <point>
            <x>175.0</x>
            <y>100.0</y>
        </point>
    </shape>
    <shape xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="circle">
        <color>0000ff</color>
        <point>
            <x>75.0</x>
            <y>75.0</y>
        </point>
        <point>
            <x>85.0</x>
            <y>75.0</y>
        </point>
    </shape>
    <shape xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="circle">
        <color>0000ff</color>
        <point>
            <x>125.0</x>
            <y>75.0</y>
        </point>
        <point>
            <x>135.0</x>
            <y>75.0</y>
        </point>
    </shape>
    <shape xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="rectangle">
        <color>000000</color>
        <point>
            <x>95.0</x>
            <y>100.0</y>
        </point>
        <point>
            <x>105.0</x>
            <y>115.0</y>
        </point>
    </shape>
    <shape xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="rectangle">
        <color>ff0000</color>
        <point>
            <x>55.0</x>
            <y>130.0</y>
        </point>
        <point>
            <x>145.0</x>
            <y>140.0</y>
        </point>
    </shape>
</drawing>
```

In the element shape, you'll see the following:
`xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`

This will be added automatically by using the annotation `XmlSeeAlso` in the `Shape` class. xsi:Type is then used to specify what kind of subclass to Shape it is.

When submitting your solution, you should also include the XML file you created.