





Software Systems Architecture


Lab 3 – Software Components


Goal 	Create software components with cohesion and decoupled from each other can be a challenge. Java 9 brought as a native language feature that increases the support for software components, allowing a precise definition of what is externally accessible and what are the component dependencies. The goal of this lab is to exercise the definition of components using this feature.
--	---


Task 	<p>Domain: We are creating a software for a company located in Javaland that receive products from all over the world. Depending on the country from which a shipment is received, the rules from Javaland for calculating the taxes are different. The goal is to combine components in a way that the software can read files with different formats with the shipments information and calculate the total tax owned.</p> <p>Component division: The modularity Java language feature to define modules should be used. The following describe the components that should be created:</p> <ul style="list-style-type: none">• ShipmentAPI: The interface ShipmentReader, the interface TaxesCalculator, and the domain class Shipment. This module cannot depend on any external module.• CSVReader: A class that implements ShipmentReader that gets data from a CSV file.• XMLReader: A class that implements ShipmentReader that gets data from a XML file.• ChinaTaxes: A class that implements TaxesCalculator for shipments that come from China.• BrazilTaxes: A class that implements TaxesCalculator for shipments that come from Brazil.• ItalyTaxes: A class that implements TaxesCalculator for shipments that come from Italy.• Main: The class ShipmentFileProcessor that integrates the other components to process a file with shipments information and return the total tax owned. <p>Testing: Unit tests should be created for classes that calculate taxes for each country. Integration tests should be created to test the ShipmentFileProcessor with different file formats.</p> <p>Dependencies: All modules can depend on the ShipmentAPI component; however, one component cannot depend directly on the classes of the other components. The ServiceLoader class should be used to get a concrete instance from the dependencies.</p>
--	---

<div data-bbox="245 315 400 528">  </div>	<p>ShipmentReader: This interface has the following methods:</p> <ul style="list-style-type: none"> • List<Shipment> readFile(String filepath) -> read the file and return the list of shipment contained • String getFileType() -> return a String with the file extension supported <p>TaxesCalculator: This interface has the following methods:</p> <ul style="list-style-type: none"> • double calculateTax(Shipment s) -> calculate the taxes for a given shipment • String getCountry() -> return a String with the country supported <p>Shipment: A domain class that should contain the following attributes:</p> <ul style="list-style-type: none"> • String productName • String productType • Integer amount • Double individualPrice • Double shipmentPrice • String country <p>CSV and XML: An example of XML and CSV files are provided.</p> <p>China: The shipments from China the taxes are 30% of the total price of the products. For the type ELETRONICS the percentage is 50%. For more than 1000 units or total a price of 100.000, a reduction of 20% of the total should be applied.</p> <p>Brazil: The shipment from Brazil the taxes are 40% of the products total price plus the shipment cost. For the type FOOD after the tax calculation, a reduction of 40% of what exceeded the value of 10.000, should be applied.</p> <p>Italy: The shipment from Italy the taxes are 30% of the products total price. For products with individual prices more than 500, an additional 5.00 of fixed cost should be applied. For the type CLOTHES after the tax calculation, a reduction of 20% of the total should be applied if the total price is below 50.000.</p> <p>ShipmentFileProcessor: This class should have a method that receives the file name and should return a double value with the total tax for all the shipments. This component should select the appropriate component for reading the file and the appropriate component to process each shipment.</p>
--	--

<div data-bbox="245 1852 400 2029">  </div>	<p>The delivery will be made by uploading to the assignment on Teams a .zip file with the complete source code developed. Don't forget to put your name on the file.</p>
--	--

<p>Tips</p> 	<p>Reading XML: There are many APIs and approaches for reading an XML file in Java. DOM can be used for processing the document as a tree, SAX as processing events and JAXB based on annotation-based mapping. However, the classes from the API module cannot depend on external APIs.</p> <p>Automated Tests: Create the automated test thinking about the different kinds of scenarios that can happen with that class. Create a unit test for each scenario. For the integration tests, create the files with meaningful data that allows you to verify if everything is working together correctly. Add tests about failed scenarios, such as when the file cannot be found or when the format is wrong.</p>
--	--

<p>Rules</p> 	<p>You cannot do: Copy any part of the code created by another student; Look at the code of another student before finishing.</p> <p>You can do: Ask help for debugging; help another student to debug after finishing your own.</p>
--	--

<p>More</p> 	<p>Tools for Building All Components: How about using a tool such as Maven or Gradle to configure the dependences and the build? Present how you did it for 15 stars! – only one student can present each tool!</p> <p>Code annotations: The methods <code>getFileType()</code> and <code>getCountry()</code> could be replaced by code annotations on the classes. Can you do that? – only one student can present for 10 stars!</p>
--	---