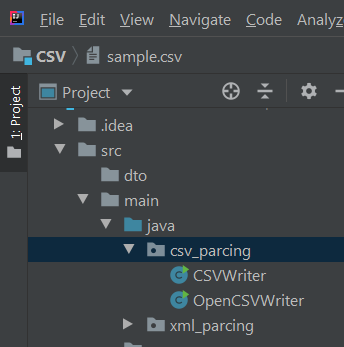
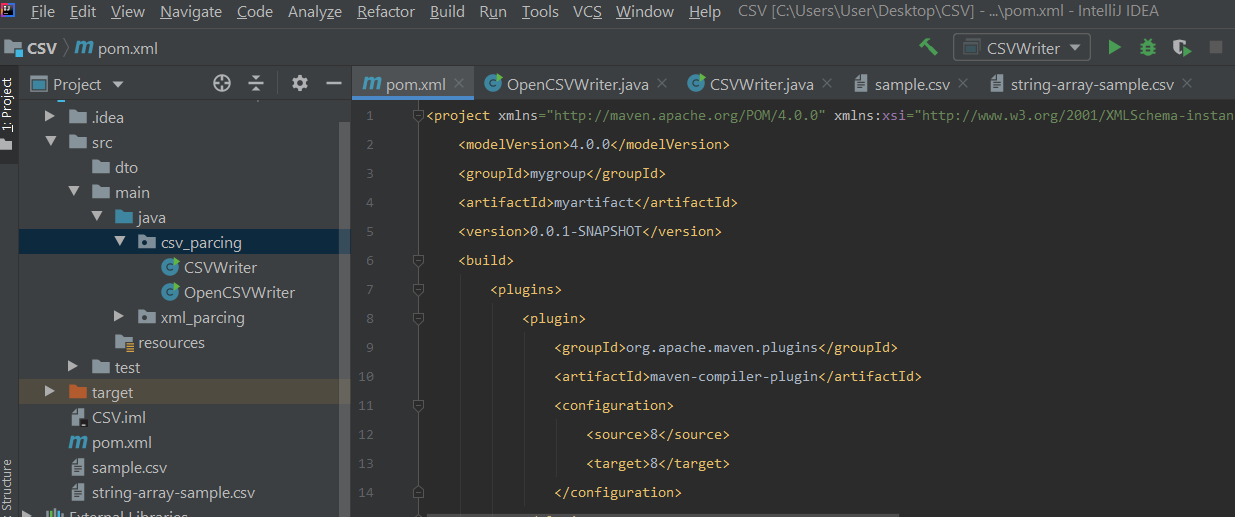
**HOMEWORK 3**

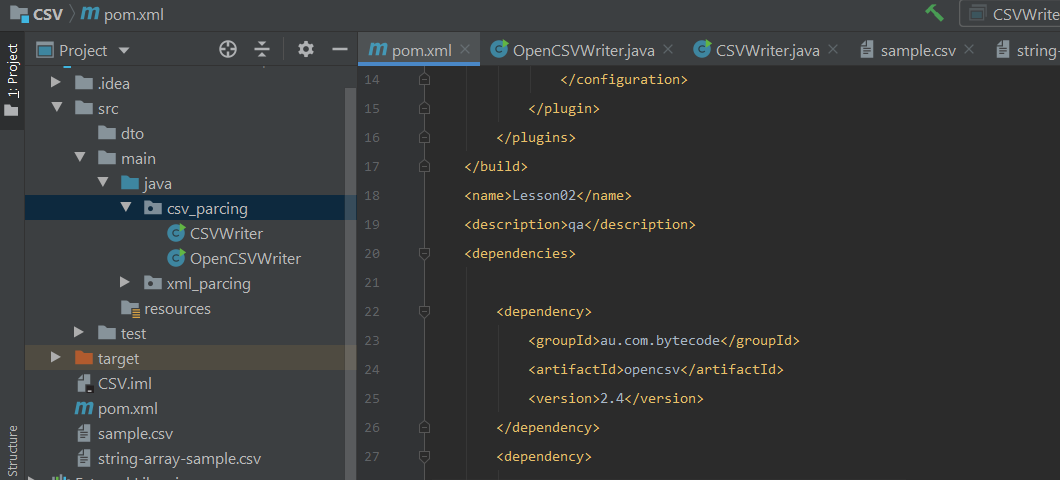
**CSV**

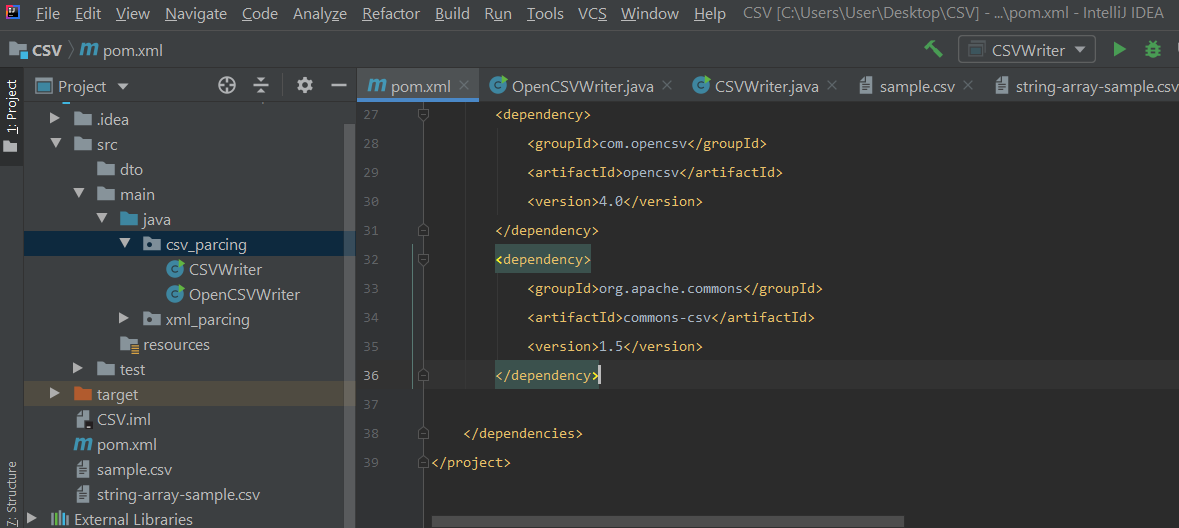
1. First created a Maven project, then the package.



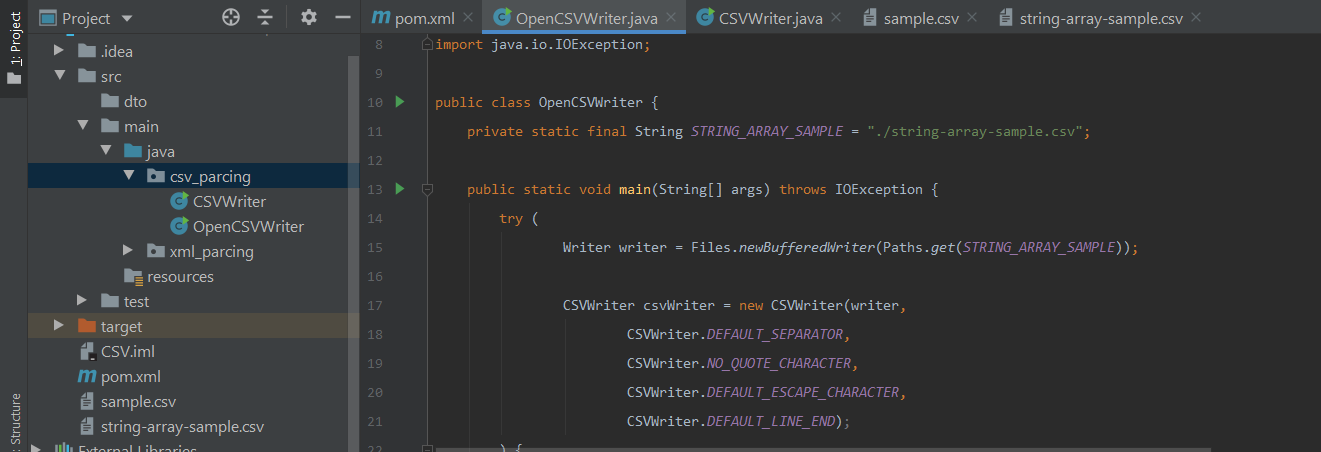
1. Secondly, to create a file with information about the students using each of the approaches, we need to add dependencies in our pom.xml file, that appears automatically, if we create an Maven project.

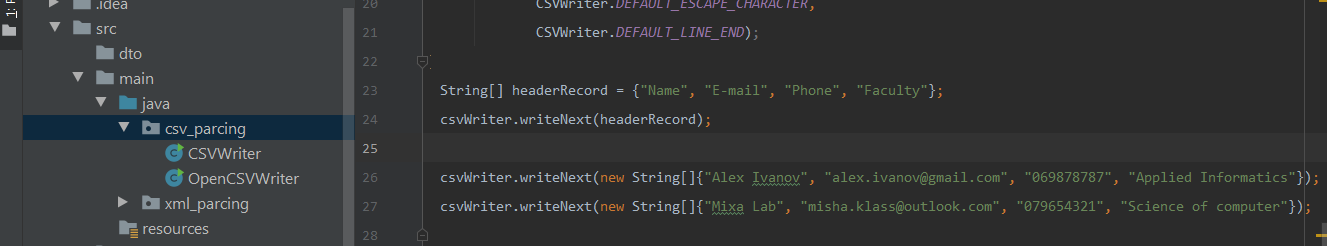




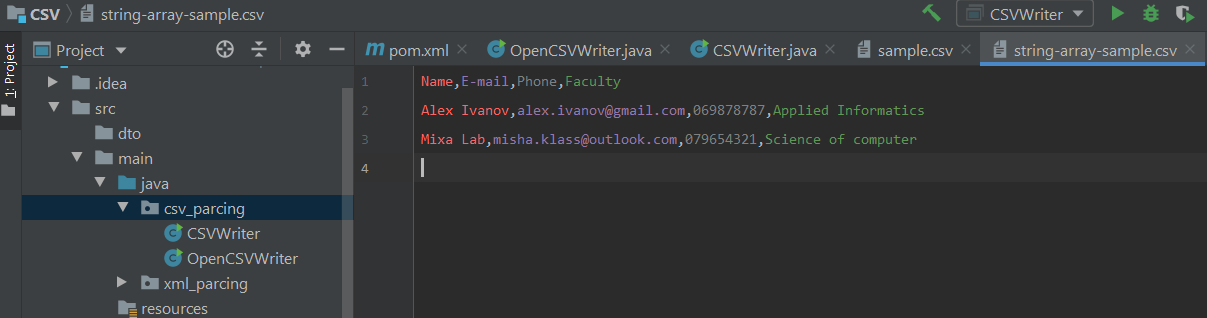


1. So we can start parsing. The first approach was **OpenCSV.** The file consists information about two students (their personal data). We created csv\_package in in src/main/java, then a Java Class.

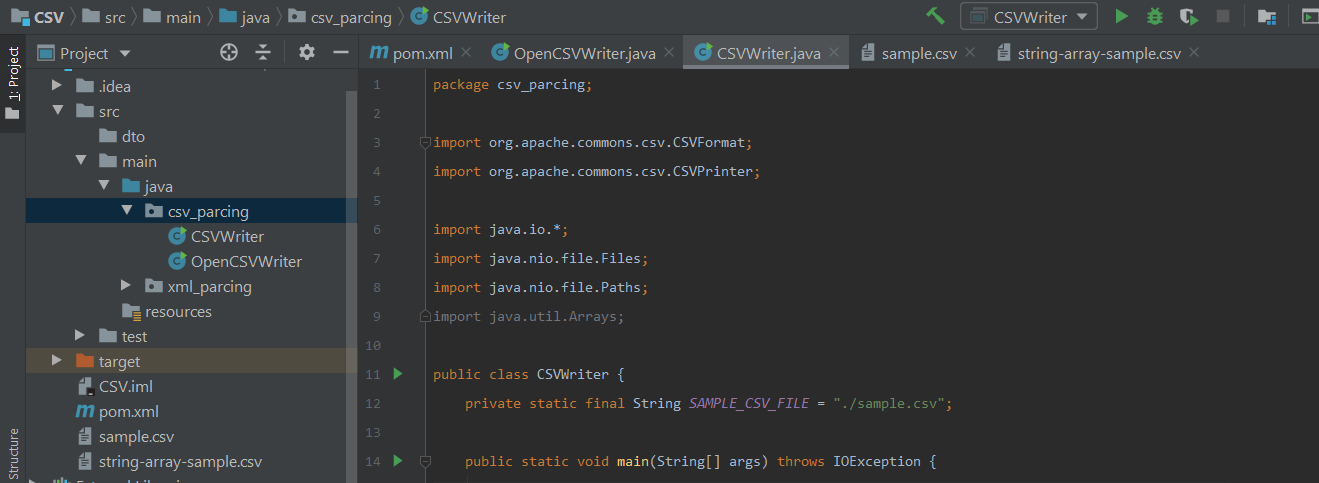


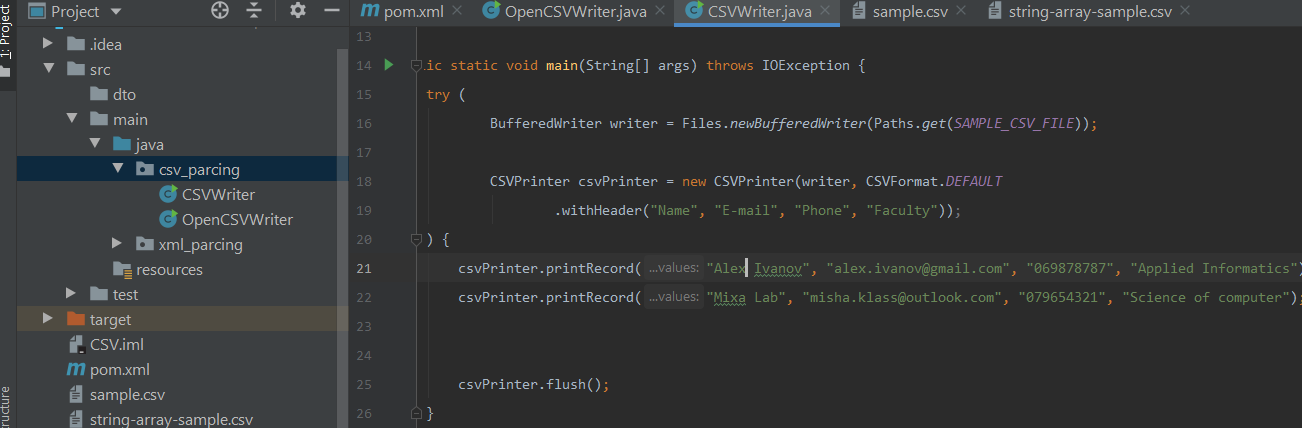


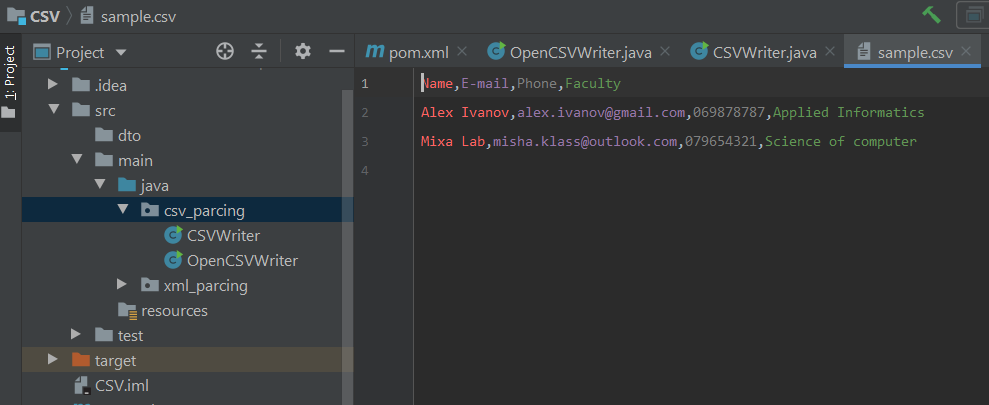
1. We can see that under the pom.xml there was created string-array-sample.csv file. There we can see the info about that two students. (there was created an constant)



1. The second approach was **common-csv**. So we created new Java Class named CSVWriter with the same data as in the previous file. So in the same way there was created file .csv





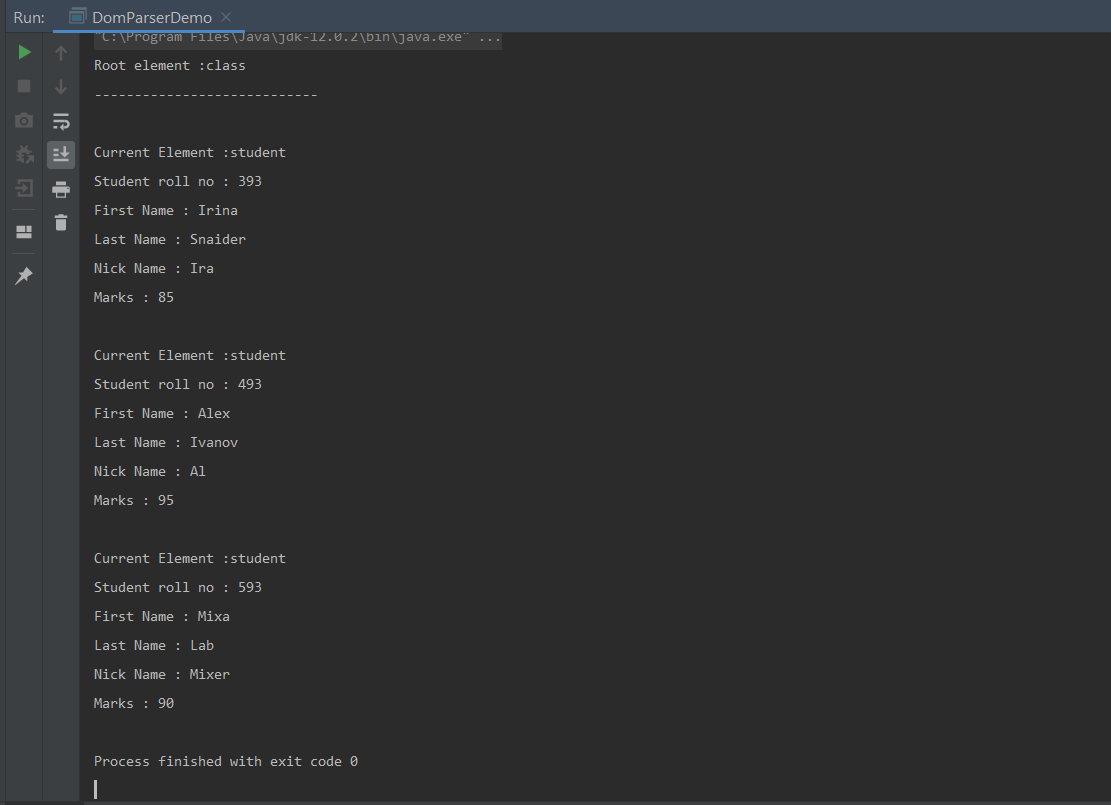


**XML**

DOM.

1. First created a txt file about the students by the instructions.
2. Then parsed via DOM. Code is in GitHub with the comments.

(By the results of the compiler you can see that I succeed. No need to screenshot the code.)



The difference between other parsers is that The DOM is a common interface for manipulating document structures. One of its design goals is that Java code written for one DOM-compliant parser should run on any other DOM-compliant parser without having to do any modifications. Kind of simple in understanding. No need to add special dependencies what makes your work less complicated.

SAX.

1. Starting to parse the same as in the previous example. The file we already have in our directory.
2. Need to create a new Java class.

The same result in the compiler. Data is available as soon as it is seen by the parser, so SAX works well for an XML document that arrives over a stream. We have no random access to an XML document since it is processed in a forward-only manner. If you need to keep track of data that the parser has seen or change the order of items, you must write the code and store the data on your own. The document is not deeply nested.

Xpath.

1. As we did in previous method of parsing we first import XML-related packages.
2. The document we already have.
3. The new Java class will be now created.
4. Create an Xpath object and an XPath path expression.

XPath uses a path expression to select node or list of nodes from an XML document. Following is a list of useful paths and expression to select any node/list of nodes from an XML document. Also we can use predicates. The difference in Xpath was to add iteration over the list of nodes, Examine attributes.

Examine sub-elements. Because of this the code was much more big than another classes.