

Java project

Jan Martin Jurie Daniel de Waard



Neptun: ap7n11

Table of contents

[1. Introduction 2](#_Toc103266185)

[2. XML structure 3](#_Toc103266186)

[3. What the application looks like 4](#_Toc103266187)

[3.1 Showing student information 4](#_Toc103266188)

[3.2 Editing/removing a student 4](#_Toc103266189)

[3.3 Adding grades 5](#_Toc103266190)

[3.4 Adding a new student 6](#_Toc103266191)

[4. The code and how it works 7](#_Toc103266192)

[4.1 The main method 8](#_Toc103266193)

[4.2 Adding grades (main) 9](#_Toc103266194)

[4.3 Editing students (main) 9](#_Toc103266195)

[4.4 Creating students (Main) 10](#_Toc103266196)

[4.5 Showing students (Main) 10](#_Toc103266197)

[4.6 The getXmlDocument method 11](#_Toc103266198)

[4.7 Showing students (xml) 11](#_Toc103266199)

[4.8 The getStudent method 12](#_Toc103266200)

[4.9 Getting student grades (xml) 12](#_Toc103266201)

[4.10 Creating a new student (xml) 13](#_Toc103266202)

[4.11 Editing a student (xml) 14](#_Toc103266203)

[4.12 The writeXml method 14](#_Toc103266204)

[4.13 Adding grades (xml) 15](#_Toc103266205)

[4.14 Removing students (xml) 15](#_Toc103266206)

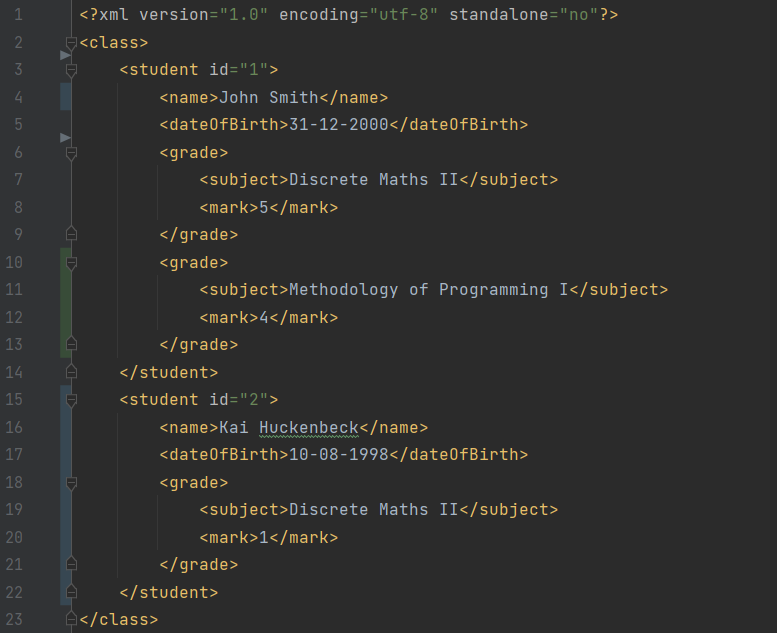
# Introduction

For the Methodology of Programming I project, I’ve made an application with education as the topic. The application can manage students and their grades.

There are 3 parts in this document: the explanation of the XML-structure, a manual of how the application works/what it can do, and finally technical information about the actual code.

# XML structure

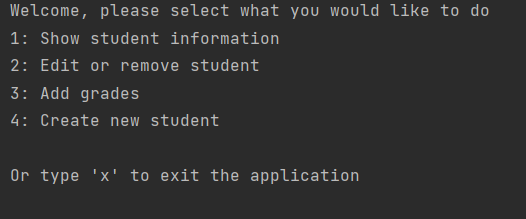
Below is an example of what the XML file looks like.



The root element is “class” (as in a school class). The “class” element contains a number of “student” child elements, which all have an id property. This id is used to find the correct student when doing things like editing a student, or adding a grade to a particular student.

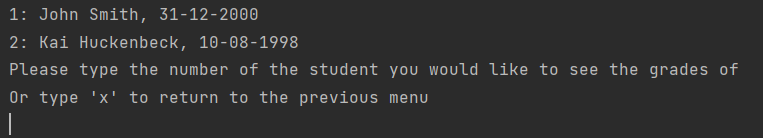
Each student has exactly 1 “name” and “date of birth” child element, and furthermore the student can have a finite number of ‘grade” elements. These “grade” elements have exactly 2 child elements: “subject” and “mark”.

# What the application looks like

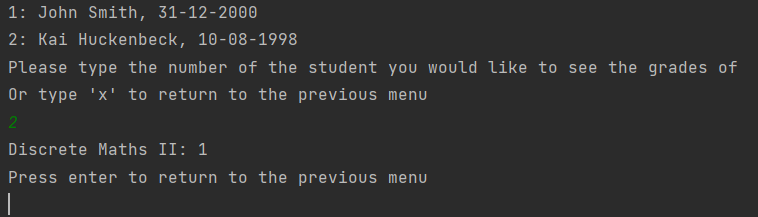


When started up, the user will first see the main menu. From here there are 4 different options, with another option ‘x’ to leave the application.

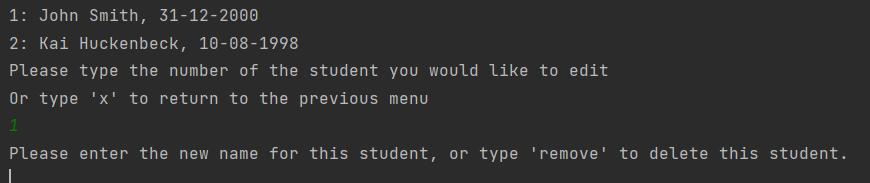
## Showing student information



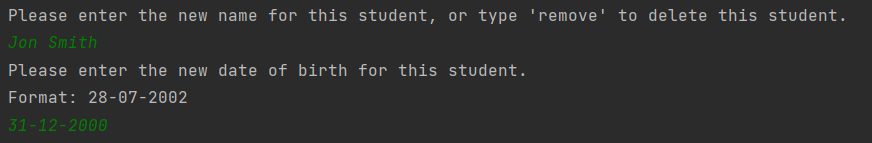
This option will display all the students in the database. We can select one of these students to see their grades.

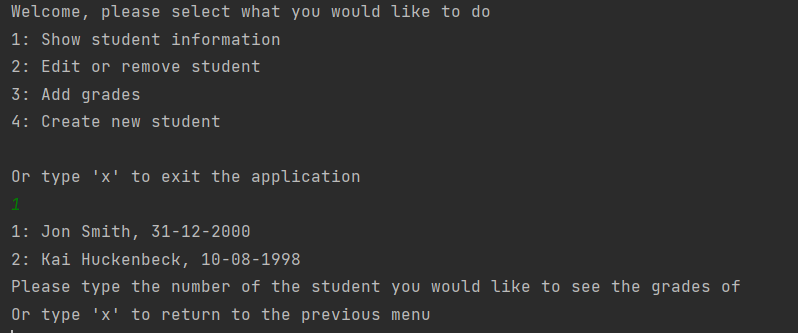


## Editing/removing a student

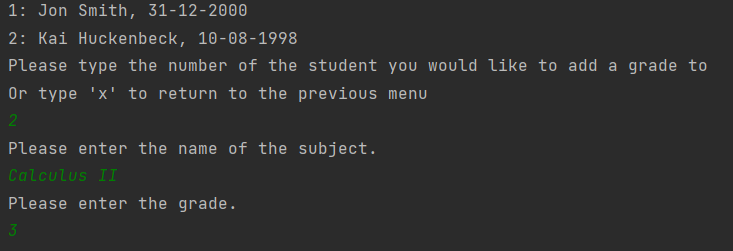


Once again we see the list of students, but now we can either enter a new name for the student, or we can type “remove” to delete the student.

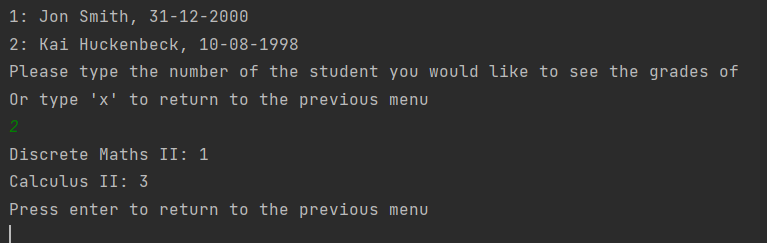




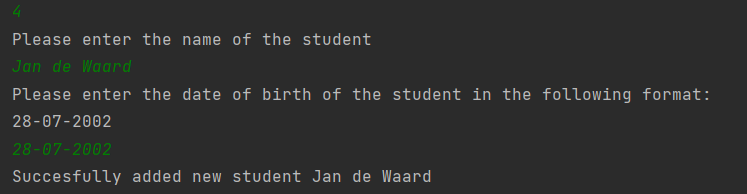
## Adding grades



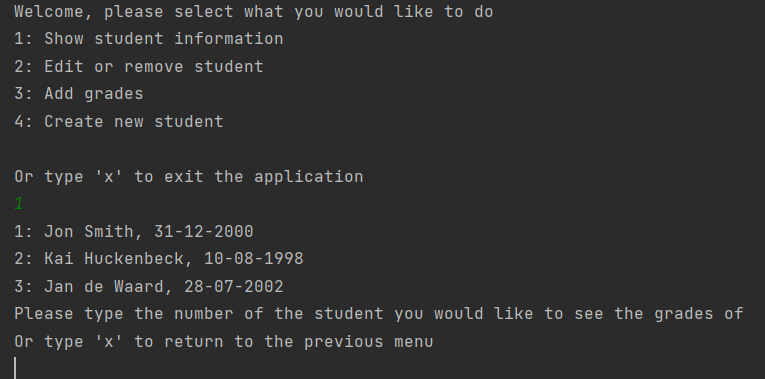
We are shown the list of students, and after selecting a student we can enter the name of the subject, as well as the grade that the student received.



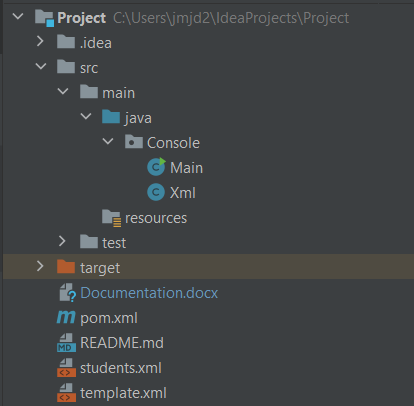
## Adding a new student



When selecting this option, the user gets asked to enter the name and date of birth of the student. This student will then be added to the xml file.



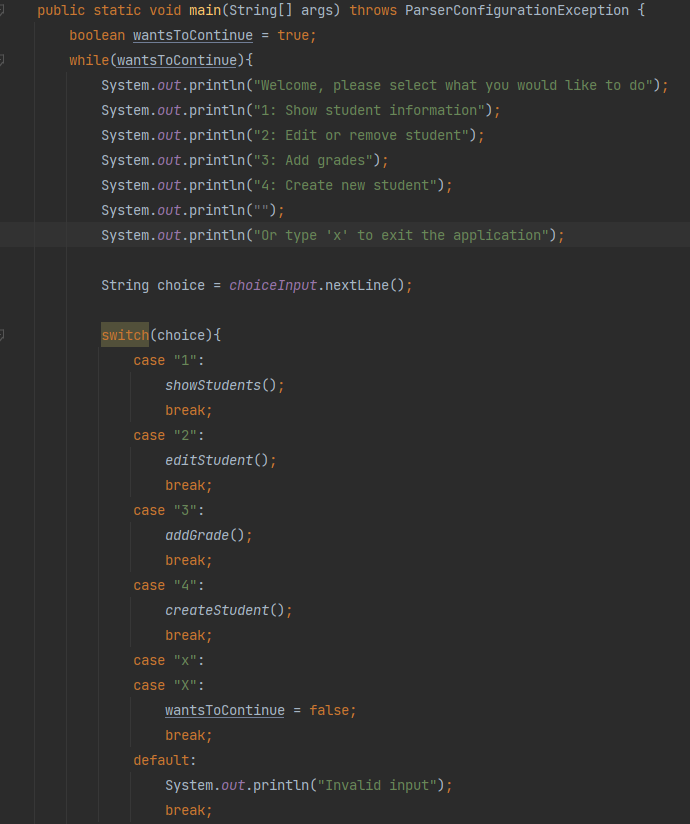
# The code and how it works



There are 3 important files for this project. First of all, in the root of the project folder there is the **students.xml**, which is the xml file where all the students/grades will be stored.

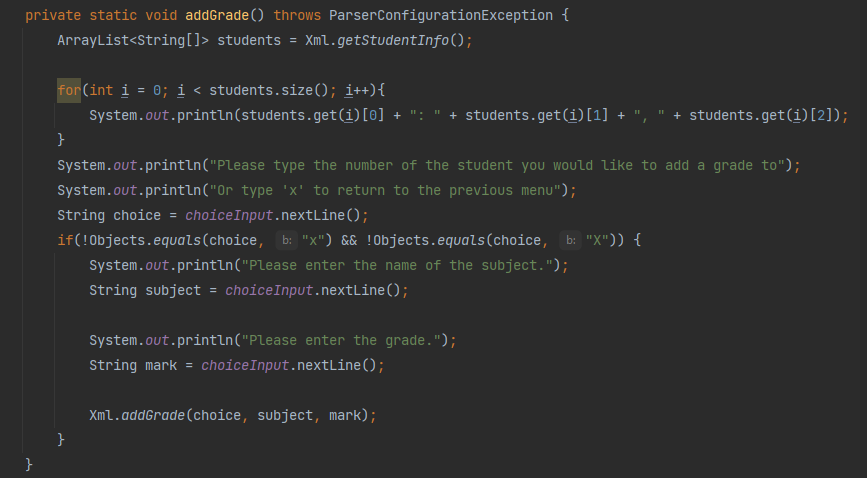
Then there are 2 java class files: Main and Xml. **Main.java** contains the main method (which controls the menu), and methods for certain tasks (create new student, add grades, edit student etc.). All these methods will call other methods in the **Xml.java** file, and these methods are used for reading from and writing to the xml file.

## The main method



The main method prints out all the menu options and asks the user for a choice. Based on the input, the corresponding method will be called, or in the case of ‘x’, the variable that is controlling the loop will be set to false, which will end the loop and because of that the whole application.

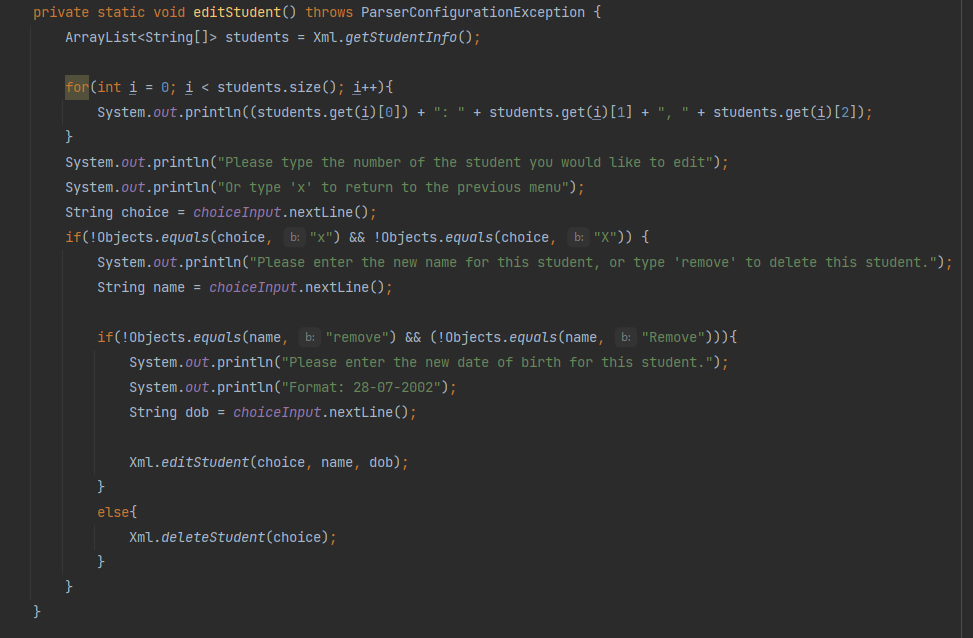
## Adding grades (main)



This method first gets the list of students by calling the getStudentInfo method from the xml class (this will be explained later in the documentation), and then prints out the students.

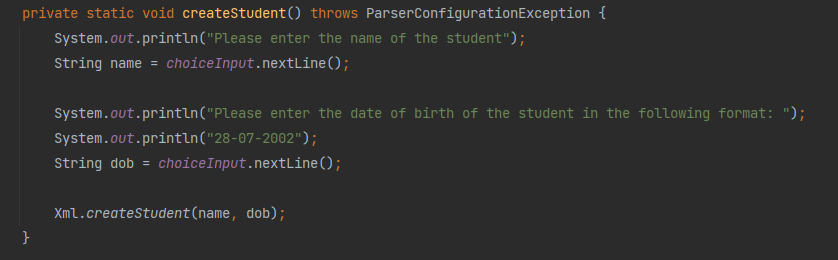
After selecting the id of the student, the user will be asked for the subject and grade, and after this the addGrade method from the xml class will be called, with the student id and the chosen subject/mark as parameters.

## Editing students (main)



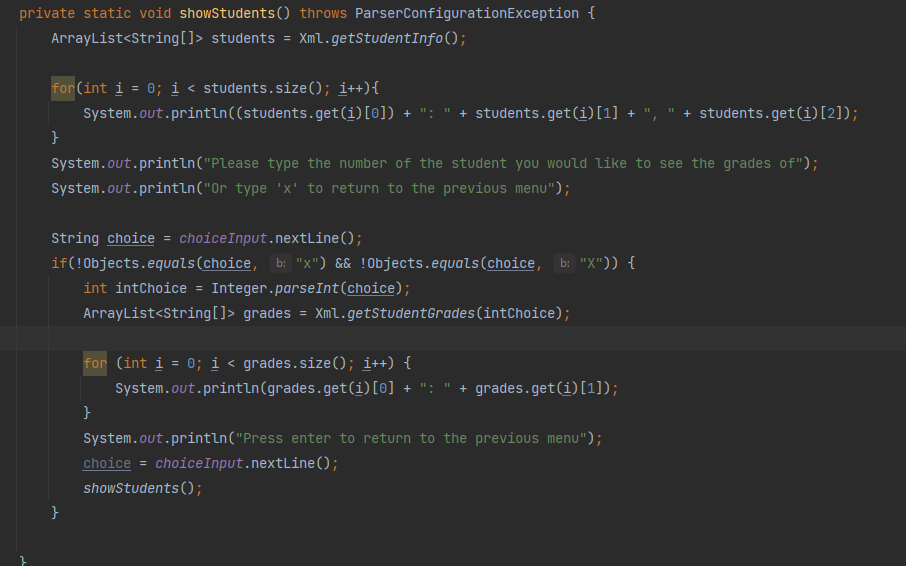
The method gets the list of students from the getStudentInfo method, and then prints out all the students and asks the user which student they would like to edit. Then, the user will be asked for a new name or to type ‘remove’, and if the user does the latter then the deleteStudent method will be called. If not, the user will also be asked for a new date of birth, and the editStudent method will then be called with all the needed parameters.

## Creating students (Main)



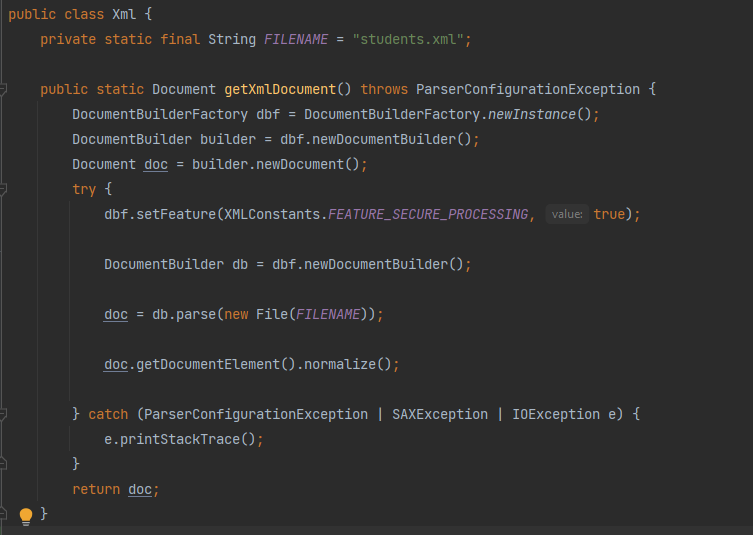
The user will be asked both the name and date of birth of the student, and then the createStudent method of the xml class will be called.

## 4.5 Showing students (Main)



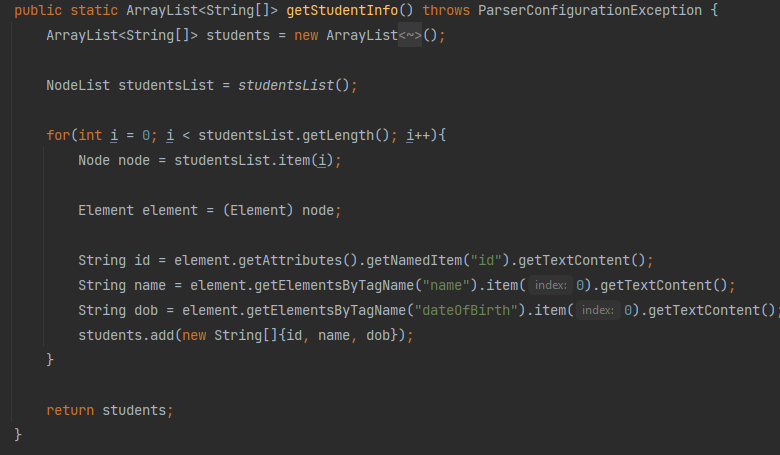
The list of students will be displayed, and then the user will be asked to select a student id to see their grades. If the user does this, the getStudentGrades method from the xml class will look for the grades, and these will then be printed out.

## 4.6 The getXmlDocument method



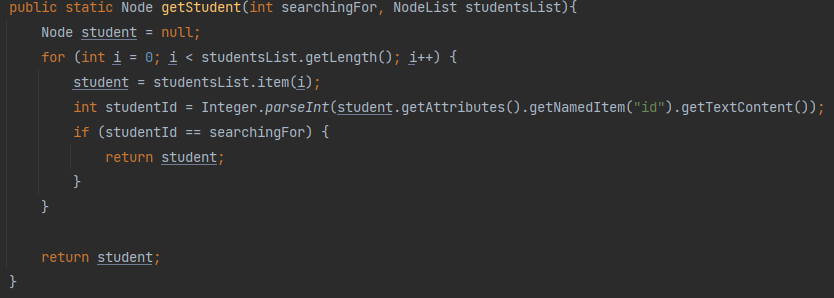
This method is called a lot of times in the Xml class. It is used for turning the xml data into a Java document. First, the document builder is created, and then the students.xml file will be parsed and returned.

## 4.7 Showing students (xml)



The “student” nodes will be selected, and a for loop will run for the length of this list. Inside the loop, we will get certain information from the student at index of variable i (id, the name & date of birth child nodes) and push these to the array list, which will be returned.

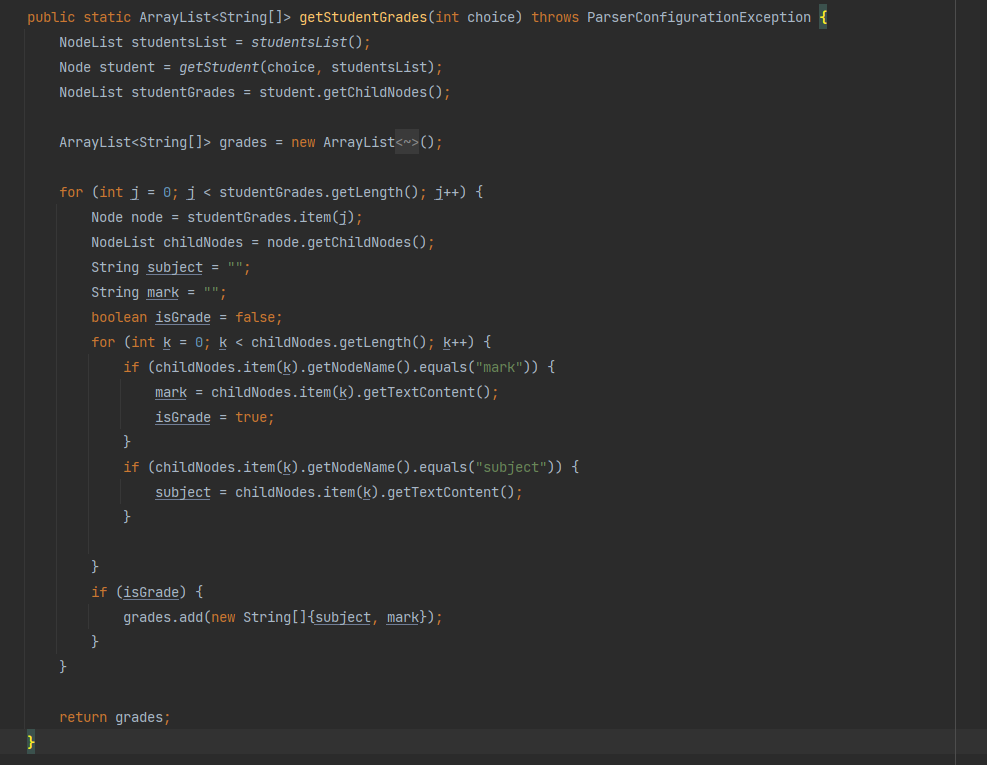
## 4.8 The getStudent method



This method is called whenever we need to find a student based on their id (examples are when editing a student or adding a grade to a student).

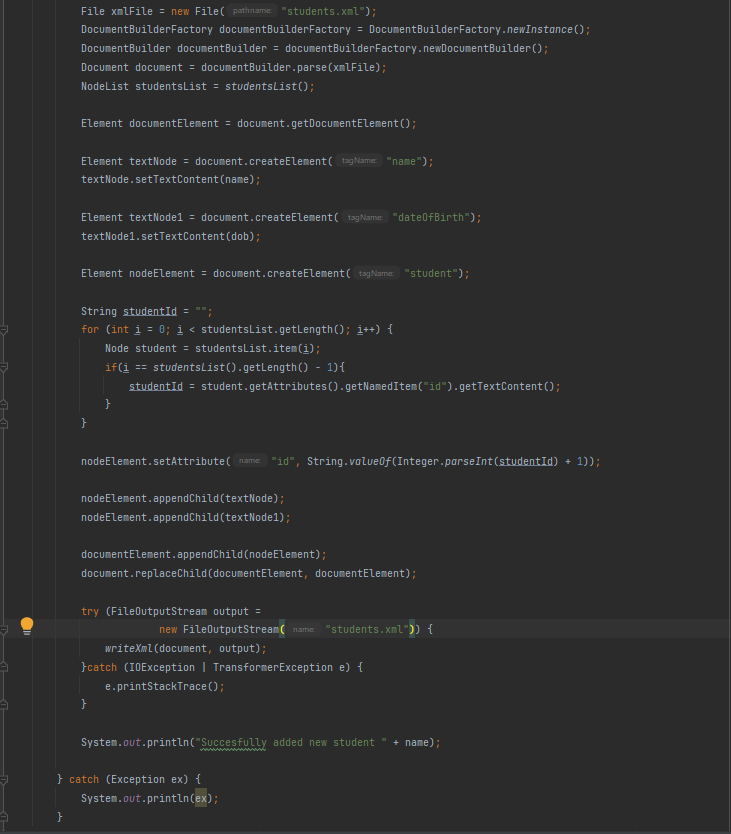
There is a loop that runs for the length of the list of students, and if the id of the student at index i is the id we are looking for, we will return this student.

## 4.9 Getting student grades (xml)



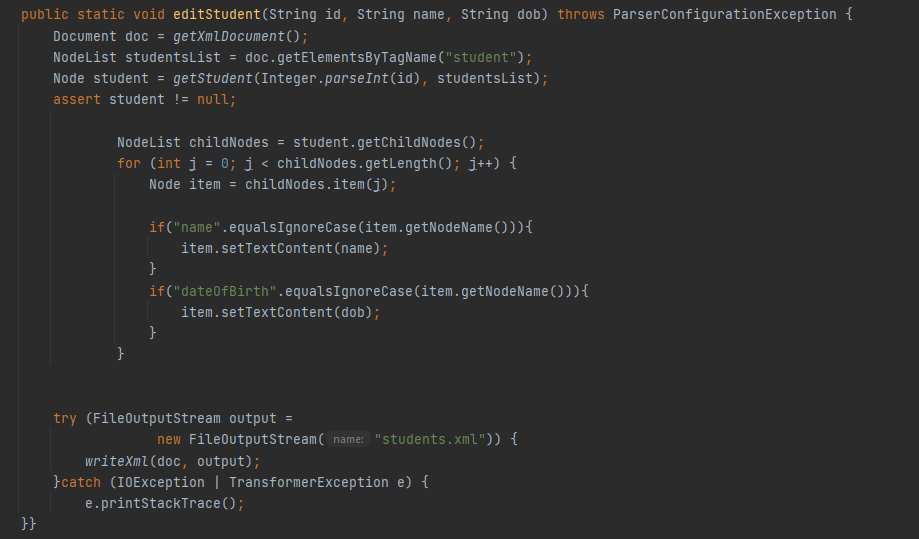
We use several of the previously mentioned methods to get the document and the particular student that we are looking for, and we will get all the child nodes from this student, and loop through them. If we find a “mark” child node, we will get its value and we will confirm that this child node is a grade. We will then also get the subject, and add this grade to the grades array list, which is returned at the end of the method.

## 4.10 Creating a new student (xml)



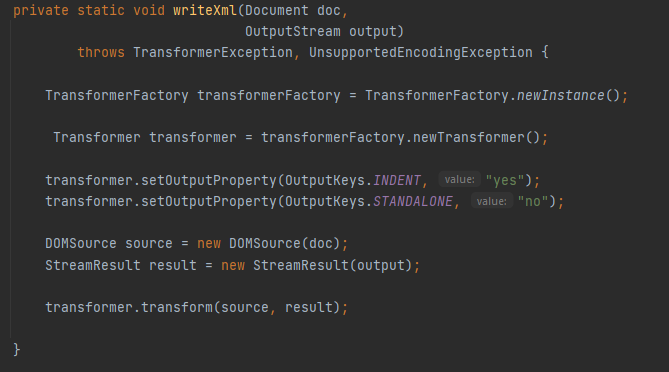
We create a new node and create the name and date of birth child nodes, which we give values that are passed to this method by the main class. We then assign the id, which will be the highest current id + 1. After this, we write the xml.

## 4.11 Editing a student (xml)



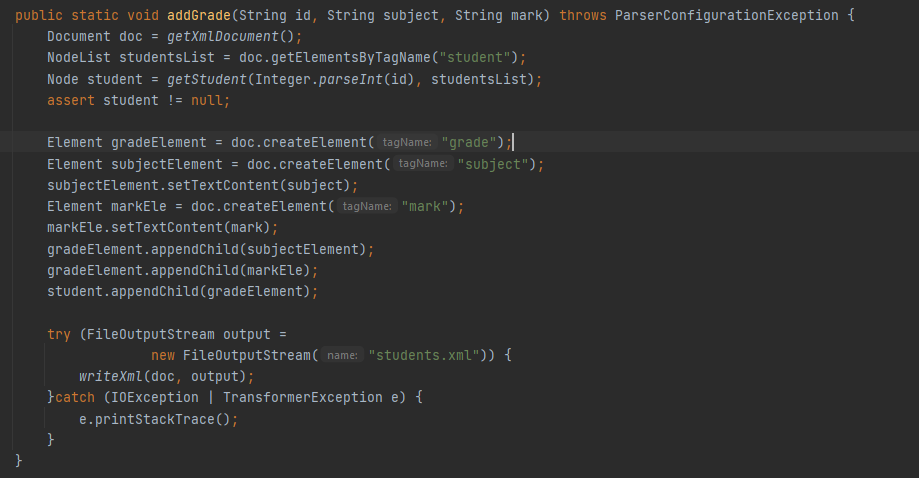
We get the student based on the id, and then we loop through its child nodes to find the name and date of birth, which we replaced with the values that have been passed to this method. After this, we call the writeXml method.

## 4.12 The writeXml method



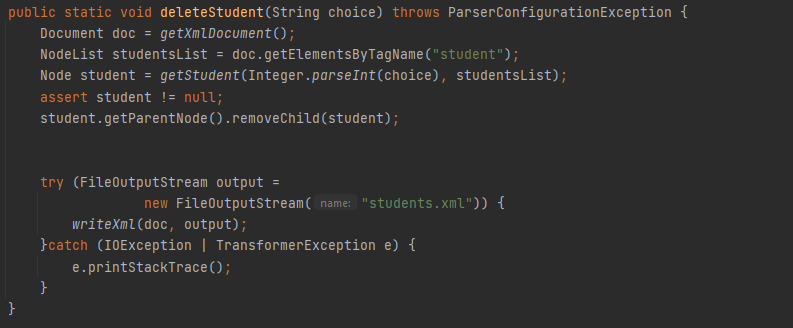
This method is called every time we need to write to the xml file.

## 4.13 Adding grades (xml)



After getting the student, we create new elements for grade and subject, and set the value based on values passed to the method. We then append these to a new “mark” element, which then gets appended to the student. After this, we call the writeXml method.

## 4.14 Removing students (xml)



After getting the student, we remove this student node and call the writeXml method.