**Lambda and streams**

**Exercise 1**

Create a new project in your corresponding GitHub-folder (according to the instructions in the document on Canvas “How to create a project in IntelliJ) and add a class **PrimeCalculator** (in a package called *model*)according to the UML-diagram below:

Afbeelding met tafel

Automatisch gegenereerde beschrijving

Don’t forget to commit and push regularly!

Implement the methods below without using any plain iteration(!). We advise you to do this in the following sequence:

* The method *getListOfAllNumbersUntil* creates a list of integers starting from the start-value until the end-value (inclusive). Tip: use the InStream-class which has an interesting static method called “range(start, end)”. Be careful! This method generates a list of integers (the end-value exclusive) as an InStream-object. To make a list of integers, you need to use the method “boxed()” and the “toList()” method
* The method *isPrime* has to implement the prime logic: if a number is a prime, this method returns true. In all other cases this method returns false. This method makes use of the previous method…
* The method *getAllPrimesUntil* generates all prime numbers starting from 0 until the value of the number-value (inclusive). This method makes use of the previous methods…

Write tests with 100% coverage for this class. Take into account in your tests that “1” is not a prime…

Now create the web application. In **index.html** you can enter a number, e.g. 80 like in the example below:

Afbeelding met tafel

Automatisch gegenereerde beschrijving

After submitting this number your web application has to take you to a page where all the primes are shown until the number entered in the index-page (inclusive):

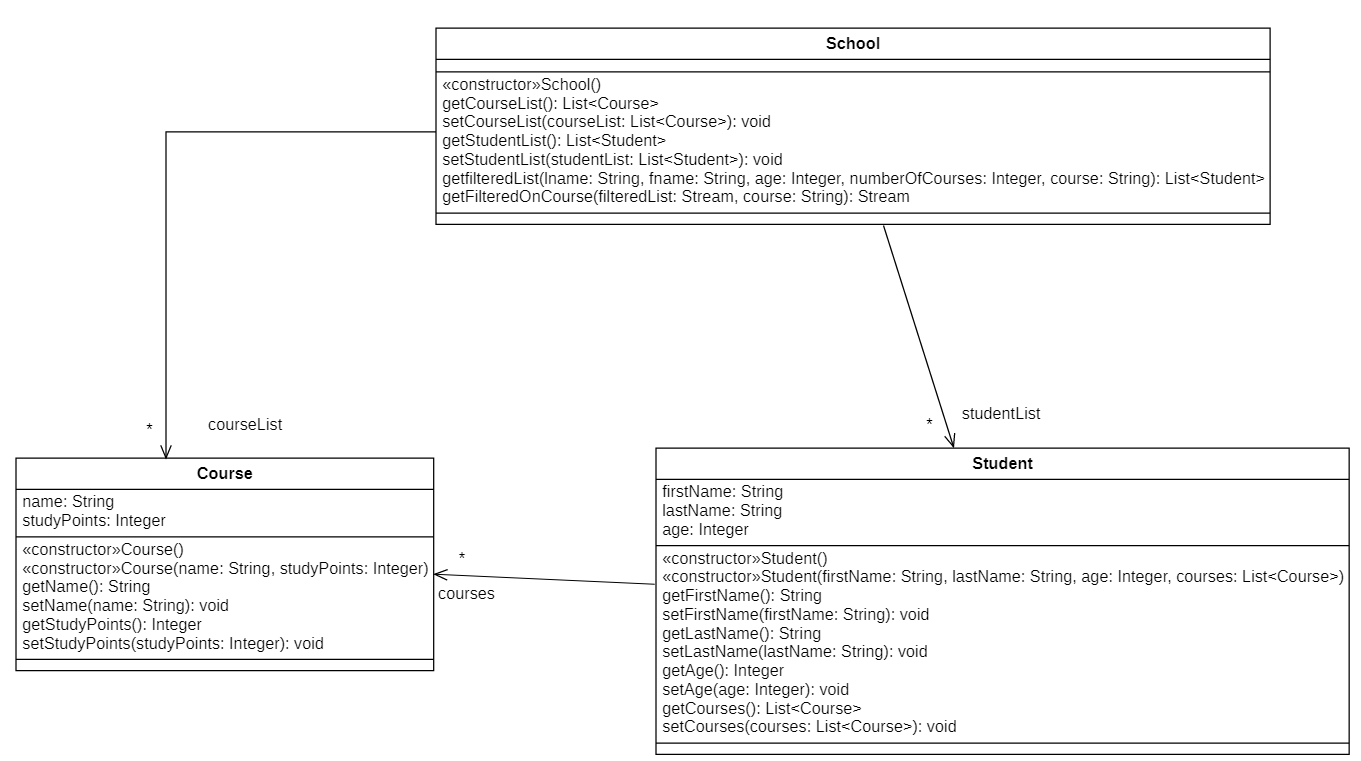
Afbeelding met tafel

Automatisch gegenereerde beschrijving

In order to generate all prime numbers, make optimal use of the methods in the PrimeCalculator-class.

**Exercise 2**

Create a new project in your corresponding GitHub-folder (according to the instructions in the document on Canvas “How to create a project in IntelliJ) and add the following classes (in a package called model) according to the UML-diagram below:



Implement the methods below without using any plain interation.

* In the constructor School() fill the courseList and studentList with the following objects:
* Course programmerI = **new** Course(**"Programmer I"**, 5);  
  Course sqlDatabases = **new** Course(**"SQL Databases"**, 3);  
  Course linux = **new** Course(**"Linux"**, 8);  
  **courseList** = Arrays.*asList*(programmerI, sqlDatabases, linux);  
  **studentList** = Arrays.*asList*(  
   **new** Student(**"Amber"**, **"Akkermans"**, 21, Arrays.*asList*(programmerI, sqlDatabases, linux)),  
   **new** Student(**"Bert"**, **"Bergmans"**, 18, Arrays.*asList*(linux)),  
   **new** Student(**"Carly"**, **"Coopman"**, 25, Arrays.*asList*(programmerI, linux)),  
   **new** Student(**"Dirk"**, **"Dieltjens"**, 19, Arrays.*asList*(programmerI, sqlDatabases)),  
   **new** Student(**"Erik"**, **"Eyken"**, 17, Arrays.*asList*(programmerI, sqlDatabases, linux)),  
   **new** Student(**"Fré"**, **"Frederickx"**, 23, Arrays.*asList*(linux)),  
   **new** Student(**"Gust"**, **"Grevers"**, 20, Arrays.*asList*(sqlDatabases, linux)),  
   **new** Student(**"Hans"**, **"Hooimans"**, 22, Arrays.*asList*(programmerI, sqlDatabases, linux)),  
   **new** Student(**"Isis"**, **"Ilsen"**, 19, Arrays.*asList*(sqlDatabases, linux)),  
   **new** Student(**"Joris"**, **"Jonkers"**, 25, Arrays.*asList*(programmerI, linux)),  
   **new** Student(**"Kim"**, **"Keuppens"**, 31, Arrays.*asList*(programmerI)),  
   **new** Student(**"Laura"**, **"Liekens"**, 21, Arrays.*asList*(sqlDatabases)),  
   **new** Student(**"Merel"**, **"Meulemans"**, 18, Arrays.*asList*(programmerI, sqlDatabases, linux)),  
   **new** Student(**"Nele"**, **"Nauwelaers"**, 19, Arrays.*asList*(linux)),  
   **new** Student(**"Oscar"**, **"Oppens"**, 18, Arrays.*asList*(programmerI, sqlDatabases, linux)),  
   **new** Student(**"Pieter"**, **"Peters"**, 21, Arrays.*asList*(programmerI, sqlDatabases, linux)),  
   **new** Student(**"Quinten"**, **"Qwards"**, 22, Arrays.*asList*(sqlDatabases, linux)),  
   **new** Student(**"Rik"**, **"Rubens"**, 20, Arrays.*asList*(programmerI, sqlDatabases)),  
   **new** Student(**"Stef"**, **"Stekker"**, 20, Arrays.*asList*(programmerI)),  
   **new** Student(**"Toon"**, **"Timmers"**, 21, Arrays.*asList*(programmerI, linux)),  
   **new** Student(**"Ursula"**, **"Uyterhoeven"**, 22, Arrays.*asList*(programmerI, sqlDatabases, linux)),  
   **new** Student(**"Viktor"**, **"Van der Veken"**, 20, Arrays.*asList*(programmerI, linux)),  
   **new** Student(**"Wim"**, **"Winkers"**, 21, Arrays.*asList*(programmerI)),  
   **new** Student(**"Xanty"**, **"Xanders"**, 19, Arrays.*asList*(programmerI, sqlDatabases, linux)),  
   **new** Student(**"Yael"**, **"Ysebaert"**, 18, Arrays.*asList*(linux)),  
   **new** Student(**"Zuster"**, **"Zulma"**, 22, Arrays.*asList*(sqlDatabases, linux)));
* The method *getfilteredList(String lname, String fname, Integer age, Integer numberOfCourses, String course)* returns a list of students that are filtered according to the arguments that are given. This method has to call the method *getFilteredOnCourse(Stream<Student> filteredList, String course)* only if a choice is made for a course. This method returns the students with the course that is given in the parameter.

Write tests with 100% coverage for all classes in the UML diagram.

The MainController has 1 private attribute *school* of the class School. When starting the application, the MainController has to instantiate the attribute causing the *studentList* and *courseList* of the object *school* to be filled. create the web application.

Now create the web application and start with the page **index.html**:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

After filling in the desired criteria, the web application should show the students that meet alle the criteria.

Hereafter you see some examples to check whether your implementation is correct:

**Example 1**

Students with lastname contains “r” and firstname contains “a”:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Afbeelding met tekst

Automatisch gegenereerde beschrijving

**Example 2:**

Students older than 20 years and more than 2 courses:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Afbeelding met tekst

Automatisch gegenereerde beschrijving

**Example 3:**

Students with Lastname containing a “u” and following the SQL Databases course:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Afbeelding met tekst

Automatisch gegenereerde beschrijving