Main.java

public class Main {

public static void main(String[] args) {

//You may test that your code works find here

//Please check that your code works and has no

//compilation problems before to submit

}

}

Student.java

import java.util.Date;

/\*\*

\* Class that stores information about the concrete student

\*

\* DO NOT REMOVE given methods, but you can add new methods/fields/constructor

\* and change the given methods implementation. For example you can change implementation for equals()

\* or hashCode() method

\*

\*/

public class Student implements Comparable {

/\*\*

\* student id

\*/

private int id;

/\*\*

\* student name and surname separated by the whitespace for example:

\* fullName = "David Luis";

\*/

private String fullName;

/\*\*

\* student date of birth in "yyyy-MM-dd" format

\*/

private Date birthDate;

/\*\*

\* student average mark

\*/

private double avgMark;

public Student(int id, String fullName, Date birthDate, double avgMark) {

this.id = id;

this.fullName = fullName;

this.birthDate = birthDate;

this.avgMark = avgMark;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getFullName() {

return fullName;

}

public void setFullName(String fullName) {

this.fullName = fullName;

}

public Date getBirthDate() {

return birthDate;

}

public void setBirthDate(Date birthDate) {

this.birthDate = birthDate;

}

public double getAvgMark() {

return avgMark;

}

public void setAvgMark(double avgMark) {

this.avgMark = avgMark;

}

@Override

public int hashCode() {

return super.hashCode();

}

@Override

public boolean equals(Object obj) {

return super.equals(obj);

}

/\*\*

\* DO NOT change this method it will be used during the task check

\*/

@Override

public int compareTo(Object o) {

Student other = (Student) o;

return (this.fullName.compareTo(other.fullName));

}

}

Studentgroup.java

import java.util.Date;

/\*\*

\* A fix-sized array of students

\* array length should always be equal to the number of stored elements

\* after the element was removed the size of the array should be equal to the number of stored elements

\* after the element was added the size of the array should be equal to the number of stored elements

\* null elements are not allowed to be stored in the array

\*

\* You may add new methods, fields to this class, but DO NOT RENAME any given class, interface or method

\* DO NOT PUT any classes into packages

\*

\*/

public class StudentGroup implements StudentArrayOperation {

private Student[] students;

/\*\*

\* DO NOT remove or change this constructor, it will be used during task check

\* @param length

\*/

public StudentGroup(int length) {

this.students = new Student[length];

}

@Override

public Student[] getStudents() {

// Add your implementation here

return null;

}

@Override

public void setStudents(Student[] students) {

// Add your implementation here

}

@Override

public Student getStudent(int index) {

// Add your implementation here

return null;

}

@Override

public void setStudent(Student student, int index) {

// Add your implementation here

}

@Override

public void addFirst(Student student) {

// Add your implementation here

}

@Override

public void addLast(Student student) {

// Add your implementation here

}

@Override

public void add(Student student, int index) {

// Add your implementation here

}

@Override

public void remove(int index) {

// Add your implementation here

}

@Override

public void remove(Student student) {

// Add your implementation here

}

@Override

public void removeFromIndex(int index) {

// Add your implementation here

}

@Override

public void removeFromElement(Student student) {

// Add your implementation here

}

@Override

public void removeToIndex(int index) {

// Add your implementation here

}

@Override

public void removeToElement(Student student) {

// Add your implementation here

}

@Override

public void bubbleSort() {

// Add your implementation here

}

@Override

public Student[] getByBirthDate(Date date) {

// Add your implementation here

return null;

}

@Override

public Student[] getBetweenBirthDates(Date firstDate, Date lastDate) {

// Add your implementation here

return null;

}

@Override

public Student[] getNearBirthDate(Date date, int days) {

// Add your implementation here

return null;

}

@Override

public int getCurrentAgeByDate(int indexOfStudent) {

// Add your implementation here

return 0;

}

@Override

public Student[] getStudentsByAge(int age) {

// Add your implementation here

return null;

}

@Override

public Student[] getStudentsWithMaxAvgMark() {

// Add your implementation here

return null;

}

@Override

public Student getNextStudent(Student student) {

// Add your implementation here

return null;

}

}

Student arrayoperation.java

import java.util.Date;

/\*\*

\* An interface that provide a number of operations for any array of students

\*

\*/

public interface StudentArrayOperation {

/\*\*

\* Returns the array of students

\*

\* @return the array of students

\*

\*/

Student[] getStudents();

/\*\*

\* Sets the array of students

\* if students == null method should throw IllegalArgumentException

\*

\* @param students

\*

\* @throws IllegalArgumentException

\*/

void setStudents(Student[] students);

/\*\*

\* if index lower than 0 or index higher/equal students.length method should

\* throw IllegalArgumentException

\*

\* @param index

\*/

Student getStudent(int index);

/\*\*

\* Replaces the element at the specified position in this array with the

\* specified element. if student == null method should throw

\* IllegalArgumentException if index lower than 0 or index higher/equal

\* students.length method should throw IllegalArgumentException

\*

\* @param student

\* the student to be stored at the specified position

\* @param index

\* the index of the element to replace

\*

\*

\* @throws IllegalArgumentException

\*/

void setStudent(Student student, int index);

/\*\*

\* Appends the specified element to the specified position of this array if

\* student == null method should throw IllegalArgumentException if index

\* lower than 0 or index higher/equal students.length method should throw

\* IllegalArgumentException

\*

\* @param student

\* the element to be appended to this array at the specified

\* position

\* @param index

\* the specified position

\*

\* @throws IllegalArgumentException

\*/

void add(Student student, int index);

/\*\*

\* Inserts the specified element at the beginning of this array if student

\* == null method should throw IllegalArgumentException

\*

\* @param student

\* the element to add

\*

\* @throws IllegalArgumentException

\*/

void addFirst(Student student);

/\*\*

\* Appends the specified element to the end of this array. if student ==

\* null method should throw IllegalArgumentException

\*

\* @param student

\* the element to add

\*

\* @throws IllegalArgumentException

\*/

void addLast(Student student);

/\*\*

\* Removes the element at the specified position in this array if index

\* lower than 0 or index higher/equal students.length method should throw

\* IllegalArgumentException

\*

\* @param index

\* the index of the element to be removed

\*

\* @throws IllegalArgumentException

\*/

void remove(int index);

/\*\*

\* Removes the first occurrence of the specified element from this array, if

\* it is present. If this array does not contain the element, it is

\* unchanged and should throw IllegalArgumentException with "Student not

\* exist" message if student == null method should throw

\* IllegalArgumentException

\*

\* @param student

\* the element to remove

\*

\* @throws IllegalArgumentException

\*/

void remove(Student student);

/\*\*

\* Removes all elements from the specified index (except the element with

\* specified index) from this array if index lower than 0 or index

\* higher/equal students.length method should throw IllegalArgumentException

\*

\* @param index

\* the index elements from should be removed

\*

\* @throws IllegalArgumentException

\*/

void removeFromIndex(int index);

/\*\*

\* Removes all elements from the specified element (except the specified

\* element) from this array if student == null method should throw

\* IllegalArgumentException

\*

\* @param student

\* the element from elements should be removed

\*

\* @throws IllegalArgumentException

\*/

void removeFromElement(Student student);

/\*\*

\* Removes all elements to the specified index (except the element with the

\* specified index) from this array if index lower than 0 or index

\* higher/equal students.length method should throw IllegalArgumentException

\*

\* @param index

\* the index elements to should be removed

\*

\* @throws IllegalArgumentException

\*/

void removeToIndex(int index);

/\*\*

\* Removes all elements to the specified element (except the specified

\* element) from this array if student == null method should throw

\* IllegalArgumentException

\*

\* @param student

\* the element to elements should be removed

\*

\* @throws IllegalArgumentException

\*/

void removeToElement(Student student);

/\*\*

\* Performs Bubble Sort to this array

\*

\*/

void bubbleSort();

/\*\*

\* Returns all students with birthday to the specified date if date == null,

\* method should throw IllegalArgumentException

\*

\* @param date

\* the date that specify birthday to find students who was born

\* before that date and in that date

\*

\* @return the array of students who was born before/in specified date

\*

\* @throws IllegalArgumentException

\*/

Student[] getByBirthDate(Date date);

/\*\*

\* Returns all students with birthday to the specified range of dates

\* including specified dates if firstDate == null or lastDate == null,

\* method should throw IllegalArgumentException

\*

\* @param firstDate

\* the date that specify birthday to find students from

\* @param lastDate

\* the date that specify birthday to find students to

\*

\* @return the array of students who was born in specified range of dates

\*

\* @throws IllegalArgumentException

\*/

Student[] getBetweenBirthDates(Date firstDate, Date lastDate);

/\*\*

\* Returns all students with birthday to the specified date and in several

\* days after the specified date if date == null, method should throw

\* IllegalArgumentException

\*

\* @param date

\* the date that specify birthday to find students from

\* @param days

\* the number of days that specify the end date

\*

\* @return the array of students who has birthday in specified date and

\* several days after

\*

\* @throws IllegalArgumentException

\*/

Student[] getNearBirthDate(Date date, int days);

/\*\*

\* Find student at the specified position and calculate his age (total

\* years) if indexOfStudent == 0, method should throw

\* IllegalArgumentException

\*

\* @param indexOfStudent

\* the index to find student and calculate age

\*

\* @return student full years

\*

\* @throws IllegalArgumentException

\*/

int getCurrentAgeByDate(int indexOfStudent);

/\*\*

\* Returns all students with age equal the specified age

\*

\* @param age the age to find students

\*

\* @return the array of students of the specified age

\*/

Student[] getStudentsByAge(int age);

/\*\*

\* Finds the maximum average mark for all students in this array

\* and returns all students who has the maximum average mark

\*

\* @return the array of students who has maximum average mark

\*

\*/

Student[] getStudentsWithMaxAvgMark();

/\*\*

\* Finds element which is the next to the specified element

\* if student == null, method should throw IllegalArgumentException

\*

\* @param student the element to which take the next element

\*

\* @throws IllegalArgumentException

\*/

Student getNextStudent(Student student);

}

**The “Student Group” task description**

Your task – implement StudentGroup class. You need to add an implementation for all methods received from StudentArrayOperation interface. You need to add methods body, do not change the folder/package structure, do not rename Student, StudentGroup classes and StudentArrayOperation interface and do not rename StudentArrayOperation methods.

Keep in mind the deadline for task implementation. Do not try to submit the solution in the last 1 or 2 minutes, do it no less than in 10 minutes before the deadline.

Do not change the solution folder structure. You need to submit src folder with \*.java files. For more information read the instruction. https://epa.ms/padawans-java “Help ?” button

**The “Student Group” task requirements**

Every method in StudentArrayOperation interface has JavaDoc comments please read before starting to implement

Note that:

* array length should always be equal to the number of stored elements
* after the element was removed the size of the array should be equal to the number of stored elements
* after the element was added the size of the array should be equal to the number of stored elements
* null elements are not allowed to be stored in the array