package podpunktA;  
  
import podpunktB.UniversityB;  
  
public class Main {  
 public static void main(String[] args) {  
 Student[] studentsA = new Student[5];  
 studentsA[0] = new Student(260387, "Bednarek", "Jakub", 4.0f);  
 studentsA[1] = new Student(260487, "Działowy", "Wojeciech", 2.0f);  
 studentsA[2] = new Student(260867, "Kowalski", "Tomasz", 2.5f);  
 studentsA[3] = new Student(262353, "Nowak", "Joanna", 3.5f);  
 studentsA[4] = new Student(263451, "Grabaż", "Karolina", 5.0f);  
  
 Student[] studentsB = studentsA.clone();  
  
 UniversityA uA = new UniversityA(studentsA);  
 UniversityB uB = new UniversityB(studentsB);  
  
 System.*out*.println("///TEST DLA ITERATORA Z PODPUNKUT A///");  
 uA.printStudents();  
 uA.correctMark(262353, 1.0f);  
 uA.correctMark(260867, 2.0f);  
 uA.correctMark(260387, 4.0f);  
  
 System.*out*.println("**\n**Po korekcie ocen: ");  
  
 uA.printStudents();  
  
 System.*out*.printf("%nSrednia ocen osób z oceną >= 3.0: %.2f%n", uA.getPositiveMarksAverage());  
  
 System.*out*.println("**\n**Studenci którzy nie zaliczyli: ");  
 uA.printStudentsWithNegativeMark();  
 System.*out*.println("**\n**Podzieleni studenci:**\n**" + uA.getSplittedArrays());  
  
 System.*out*.println("**\n\n**///TEST DLA ITERATORA Z PODPUNKTU B///");  
 uB.printStudents();  
  
 uB.correctMark(260867, 3.5f);  
 uB.correctMark(263451, 2.5f);  
  
 System.*out*.println("**\n**Po korekcie ocen: ");  
 uB.printStudents();  
  
 System.*out*.printf("**\n**Srednia ocen osob z oceną >= 3.0: %.2f%n", uB.getPositiveMarksAverage());  
 System.*out*.println("**\n**Studenci którzy nie zaliczyli: ");  
 uB.printStudentsWithNegativeMark();  
 System.*out*.println("**\n**Podzieleni studenci:**\n**" + uB.getSplittedArrays());  
 }  
}

package podpunktA;  
  
import java.util.Iterator;  
import java.util.NoSuchElementException;  
  
public class ArrayIteratorA<T> implements Iterator<T> {  
 private T[] array;  
 private int position = 0;  
  
 public ArrayIteratorA(T[] array){  
 this.array = array;  
 }  
  
 @Override  
 public boolean hasNext() {  
 return position < array.length;  
 }  
  
 @Override  
 public T next() throws NoSuchElementException{  
 if(hasNext()){  
 return array[position++];  
 } else {  
 throw new NoSuchElementException();  
 }  
 }  
  
 @Override  
 public void remove() throws UnsupportedOperationException{  
 throw new UnsupportedOperationException();  
 }  
}

package podpunktA;  
  
import java.util.Iterator;  
  
public class FilterArrayIteratorA<T> implements Iterator<T> {  
 private ArrayIteratorA<T> iterator;  
 private Predicate<T> predicate;  
  
 private boolean hasNext = true;  
 private T next = null;  
  
 public FilterArrayIteratorA(ArrayIteratorA<T> iterator, Predicate<T> predicate){  
 this.iterator = iterator;  
 this.predicate = predicate;  
  
 findNextValid();  
 }  
  
 private void findNextValid() {  
 while(iterator.hasNext()) {  
 next = iterator.next();  
  
 if(predicate.accept(next)){  
 return;  
 }  
 }  
  
 next = null;  
 hasNext = false;  
 }  
  
 @Override  
 public void remove() {  
 throw new UnsupportedOperationException();  
 }  
  
 @Override  
 public boolean hasNext() {  
 return hasNext;  
 }  
  
 @Override  
 public T next() {  
 T nextObject = next;  
 findNextValid();  
  
 return nextObject;  
 }  
}

package podpunktA;  
  
public class DoubleArray<T> {  
 private T[] array1;  
 private T[] array2;  
  
 public DoubleArray(T[] array1, T[] array2){  
 this.array1 = array1;  
 this.array2 = array2;  
 }  
  
 public T[] getArray1() {  
 return array1;  
 }  
  
 public T[] getArray2() {  
 return array2;  
 }  
  
 @Override  
 public String toString() {  
 StringBuffer buffer = new StringBuffer();  
  
 buffer.append("Array 1: ");  
 for(T object : array1){  
 buffer.append(object + " ");  
 }  
  
 buffer.append("**\n**Array 2: ");  
 for(T object : array2){  
 buffer.append(object + " ");  
 }  
  
 return buffer.toString();  
 }  
}

package podpunktA;  
  
public interface Predicate<T> {  
 boolean accept(T arg);  
}

package podpunktA;  
  
public class Student {  
 private int index;  
 private String lastName;  
 private String firstName;  
 private float mark;  
  
 public Student(int index, String lastName, String firstName, float mark) {  
 this.index = index;  
 this.lastName = lastName;  
 this.firstName = firstName;  
 this.mark = mark;  
 }  
  
 public int getIndex() {  
 return index;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public float getMark() {  
 return mark;  
 }  
  
 public void setMark(float mark){  
 this.mark = mark;  
 }  
  
 @Override  
 public String toString() {  
 return index + ", " + firstName + ", " + lastName + ", " + mark;  
 }  
}

package podpunktA;  
  
public class UniversityA {  
 private Student[] students;  
  
 public UniversityA(Student[] students){  
 this.students = students;  
 }  
  
 public void printStudents(){  
 ArrayIteratorA<Student> it = new ArrayIteratorA<>(students);  
  
 while(it.hasNext()) System.*out*.println(it.next());  
 }  
  
 public void correctMark(int index, float mark){  
 ArrayIteratorA<Student> it = new ArrayIteratorA<>(students);  
  
 while(it.hasNext()){  
 Student s = it.next();  
  
 if(s.getIndex() == index){  
 s.setMark(mark);  
 return;  
 }  
 }  
 }  
  
 public float getPositiveMarksAverage(){  
 FilterArrayIteratorA<Student> it = new FilterArrayIteratorA<>(new ArrayIteratorA<>(students), (student) -> { return student.getMark() >= 3.0f; });  
  
 float sum = 0.0f;  
 int marks = 0;  
 while(it.hasNext()){  
 sum += it.next().getMark();  
 marks++;  
 }  
  
 return sum / marks;  
 }  
  
 public void printStudentsWithNegativeMark(){  
 FilterArrayIteratorA<Student> it = new FilterArrayIteratorA<>(new ArrayIteratorA<>(students), (student) -> {return student.getMark() < 3.0f; });  
  
 while(it.hasNext()){  
 System.*out*.println(it.next());  
 }  
 }  
  
 public DoubleArray getSplittedArrays(){  
 Student[] arrayNegative = new Student[countNegativeStudents()];  
 Student[] arrayPositive = new Student[countPositiveStudents()];  
 FilterArrayIteratorA<Student> itP = new FilterArrayIteratorA<>(new ArrayIteratorA<>(students), (student) -> { return student.getMark() >= 3.0f; });  
 FilterArrayIteratorA<Student> itN = new FilterArrayIteratorA<>(new ArrayIteratorA<>(students), (student) -> { return student.getMark() < 3.0f; });  
  
 int i = 0;  
 while(itN.hasNext()){  
 arrayNegative[i++] = itN.next();  
 }  
  
 i = 0;  
 while(itP.hasNext()){  
 arrayPositive[i++] = itP.next();  
 }  
  
 return new DoubleArray(arrayNegative, arrayPositive);  
 }  
  
 private int countPositiveStudents(){  
 FilterArrayIteratorA<Student> it = new FilterArrayIteratorA<>(new ArrayIteratorA<>(students), (student) -> { return student.getMark() >= 3.0f; });  
  
 int count = 0;  
 while(it.hasNext()){  
 it.next();  
 count++;  
 }  
  
 return count;  
 }  
  
 private int countNegativeStudents(){  
 FilterArrayIteratorA<Student> it = new FilterArrayIteratorA<>(new ArrayIteratorA<>(students), (student) -> { return student.getMark() < 3.0f; });  
  
 int count = 0;  
 while(it.hasNext()){  
 it.next();  
 count++;  
 }  
  
 return count;  
 }  
}

package podpunktB;  
  
public class ArrayIteratorB<T> implements Iterator<T> {  
 private T[] array;  
 private int pos = 0;  
  
 public ArrayIteratorB(T[] array){  
 this.array = array;  
 }  
  
 @Override  
 public void first() {  
 pos = 0;  
 }  
  
 @Override  
 public void next() {  
 if(!isDone()){  
 pos++;  
 }  
 }  
  
 @Override  
 public boolean isDone() {  
 return pos >= array.length;  
 }  
  
 @Override  
 public T currentItem() {  
 return array[pos];  
 }  
}

package podpunktB;  
  
import podpunktA.Predicate;  
  
public class FilterArrayIteratorB<T> implements Iterator<T> {  
 private ArrayIteratorB<T> iterator;  
 private Predicate<T> predicate;  
  
 private boolean hasNext = true;  
 private T next = null;  
  
 public FilterArrayIteratorB(ArrayIteratorB<T> iterator, Predicate<T> predicate){  
 this.iterator = iterator;  
 this.predicate = predicate;  
  
 findNextValid();  
 }  
  
 private void findNextValid() {  
 while(!iterator.isDone()) {  
 next = iterator.currentItem();  
  
 if(predicate.accept(next)){  
 return;  
 }  
  
 iterator.next();  
 }  
  
 next = null;  
 hasNext = false;  
 }  
  
 @Override  
 public void first() {  
 iterator.first();  
 findNextValid();  
 }  
  
 @Override  
 public void next() {  
 iterator.next();  
 findNextValid();  
 }  
  
 @Override  
 public boolean isDone() {  
 return !hasNext;  
 }  
  
 @Override  
 public T currentItem() {  
 return next;  
 }  
}

package podpunktB;  
  
public interface Iterator<T> {  
 void first();  
 void next();  
 boolean isDone();  
 T currentItem();  
}

package podpunktB;  
  
import podpunktA.ArrayIteratorA;  
import podpunktA.DoubleArray;  
import podpunktA.FilterArrayIteratorA;  
import podpunktA.Student;  
  
public class UniversityB {  
 private Student[] students;  
  
 public UniversityB(Student[] students){  
 this.students = students;  
 }  
  
 public void printStudents(){  
 ArrayIteratorB<Student> it = new ArrayIteratorB<>(students);  
  
 while(!it.isDone()){  
 System.out.println(it.currentItem());  
 it.next();  
 }  
 }  
  
 public void correctMark(int index, float mark){  
 ArrayIteratorB<Student> it = new ArrayIteratorB<>(students);  
  
 while(!it.isDone()){  
 Student s = it.currentItem();  
 it.next();  
  
 if(s.getIndex() == index){  
 s.setMark(mark);  
 return;  
 }  
 }  
 }  
  
 public float getPositiveMarksAverage(){  
 FilterArrayIteratorB<Student> it = new FilterArrayIteratorB<>(new ArrayIteratorB<>(students), (student) -> { return student.getMark() >= 3.0f; });  
  
 float sum = 0.0f;  
 int marks = 0;  
 while(!it.isDone()){  
 sum += it.currentItem().getMark();  
 marks++;  
 it.next();  
 }  
  
 return sum / marks;  
 }  
  
 public void printStudentsWithNegativeMark(){  
 FilterArrayIteratorB<Student> it = new FilterArrayIteratorB<>(new ArrayIteratorB<>(students), (student) -> {return student.getMark() < 3.0f; });  
  
 while(!it.isDone()){  
 System.out.println(it.currentItem());  
 it.next();  
 }  
 }  
  
 public DoubleArray getSplittedArrays(){  
 Student[] arrayNegative = new Student[countNegativeStudents()];  
 Student[] arrayPositive = new Student[countPositiveStudents()];  
 FilterArrayIteratorB<Student> itP = new FilterArrayIteratorB<>(new ArrayIteratorB<>(students), (student) -> { return student.getMark() >= 3.0f; });  
 FilterArrayIteratorB<Student> itN = new FilterArrayIteratorB<>(new ArrayIteratorB<>(students), (student) -> { return student.getMark() < 3.0f; });  
  
 int i = 0;  
 while(!itN.isDone()){  
 arrayNegative[i++] = itN.currentItem();  
 itN.next();  
 }  
  
 i = 0;  
 while(!itP.isDone()){  
 arrayPositive[i++] = itP.currentItem();  
 itP.next();  
 }  
  
 return new DoubleArray(arrayNegative, arrayPositive);  
 }  
  
 private int countPositiveStudents(){  
 FilterArrayIteratorB<Student> it = new FilterArrayIteratorB<>(new ArrayIteratorB<>(students), (student) -> { return student.getMark() >= 3.0f; });  
  
 int count = 0;  
 while(!it.isDone()){  
 it.next();  
 count++;  
 }  
  
 return count;  
 }  
  
 private int countNegativeStudents(){  
 FilterArrayIteratorB<Student> it = new FilterArrayIteratorB<>(new ArrayIteratorB<>(students), (student) -> { return student.getMark() < 3.0f; });  
  
 int count = 0;  
 while(!it.isDone()){  
 it.next();  
 count++;  
 }  
  
 return count;  
 }  
}

Wyniki powyższego programu:

///TEST DLA ITERATORA Z PODPUNKUT A///

260387, Jakub, Bednarek, 4.0

260487, Wojeciech, Działowy, 2.0

260867, Tomasz, Kowalski, 2.5

262353, Joanna, Nowak, 3.5

263451, Karolina, Grabaż, 5.0

Po korekcie ocen:

260387, Jakub, Bednarek, 4.0

260487, Wojeciech, Działowy, 2.0

260867, Tomasz, Kowalski, 2.0

262353, Joanna, Nowak, 1.0

263451, Karolina, Grabaż, 5.0

Srednia ocen osób z oceną >= 3.0: 4,50

Studenci którzy nie zaliczyli:

260487, Wojeciech, Działowy, 2.0

260867, Tomasz, Kowalski, 2.0

262353, Joanna, Nowak, 1.0

Podzieleni studenci:

Array 1: 260487, Wojeciech, Działowy, 2.0 260867, Tomasz, Kowalski, 2.0 262353, Joanna, Nowak, 1.0

Array 2: 260387, Jakub, Bednarek, 4.0 263451, Karolina, Grabaż, 5.0

///TEST DLA ITERATORA Z PODPUNKTU B///

260387, Jakub, Bednarek, 4.0

260487, Wojeciech, Działowy, 2.0

260867, Tomasz, Kowalski, 2.0

262353, Joanna, Nowak, 1.0

263451, Karolina, Grabaż, 5.0

Po korekcie ocen:

260387, Jakub, Bednarek, 4.0

260487, Wojeciech, Działowy, 2.0

260867, Tomasz, Kowalski, 3.5

262353, Joanna, Nowak, 1.0

263451, Karolina, Grabaż, 2.5

Srednia ocen osob z oceną >= 3.0: 3,75

Studenci którzy nie zaliczyli:

260487, Wojeciech, Działowy, 2.0

262353, Joanna, Nowak, 1.0

263451, Karolina, Grabaż, 2.5

Podzieleni studenci:

Array 1: 260487, Wojeciech, Działowy, 2.0 262353, Joanna, Nowak, 1.0 263451, Karolina, Grabaż, 2.5

Array 2: 260387, Jakub, Bednarek, 4.0 260867, Tomasz, Kowalski, 3.5