Завдання 3
SmartArray
(with Decorators and Strategy patterns)

В архіві знаходиться шаблон проекту, який містить інтерфейси та набір класів, які необхідно реалізувати.

Інструкція:

1. Скачайте архів з шаблоном проекту *SmartArray* та розархівуйте його
2. Додайте його в локальний *Git* репозиторій
    a. Перейдіть в командному рядку до директорії *SmartArray*
    b. Виконайте команди:
    > git init
    *> git add **
    > git commit -m "SmartArray initial commit"
3. Створіть на своєму GitHub новий репозиторій з іменем за наступним шаблоном *apps19*<surname>-hw3
4. Знову перейдіть в директорію *SmartArray* і виконайте наступну команду
    *> git remote add origin https://github.com/**<user_name>/apps19<surname>-hw3.git***
5. Виконайте команду
    > git push -u origin master

Після цього може відкрити проект в IDE і почати виконувати завдання

Важливо:

- не змінюйте структуру шаблоного проекту та не видаляйте файли *checkstyle.xml, pom.xml*
- при коміті проекту в локальний репозиторій (та на GitHub) необхідно щоб виключно були закомічені: *src/, checkstyle.xml, pom.xml*. Директорію *target/,* яка буде створена під час компіляції комітити не треба.

Завдання:

1. Реалізуйте класи

BaseArray - базовий клас, який є “обгорткою” для початкового масиву та основою для декорування

SmartArrayDecorator SmartArrayDecorator - абстрактний клас від якого успадковуються усі декоратори

DistinctDecorator - декоратор, який видаляє повтори елементів з масиву (перевірка має йти на основі методу equals())

FilterDecorator - декоратор, який видаляє елементи, які не підпадають під умови предикату. Предикат реалізує інтерфейс MyPredicate, та передає його у конструктор класу (шаблон стратегія)

MapDecorator - декоратор, який відображає кожен елемент (об'єкт) SmartArray в інший об'єкт, шляхом застосування до них функції - MyFunction.

Реалізація MyFunction передається в конструктор класу MapDecorator (шаблон стратегія)

SortDecorator - декоратор, який сортує елементи SmartArray на основі MyComparator. Реалізація MyComparator передається в конструктор класу SortDecorator (шаблон стратегія). Для сортування можна використовувати вже існуючи методу класу Arrays.sort

Приклад використання декораторів для фільтрації від'ємних значень масиву, їх сортування та помноження на 2, наведено нижче:

```
MyPredicate pr = new MyPredicate() {
      @Override
      public boolean test(Object t) {
            return ((Integer) t) > 0;
      }
};

MyComparator cmp = new MyComparator() {
      @Override
      public int compare(Object o1, Object o2) {
            return ((Integer) o1) - ((Integer) o2);
      }
};

MyFunction func = new MyFunction() {
      @Override
      public Object apply(Object t) {
            return 2 * ((Integer) t);
      }
};

Integer[] integers = {-1, 2, 0, 1, -5, 3};
SmartArray sa = new BaseArray(integers);

sa = new FilterDecorator(sa, pr); // Result: [2, 1, 3];
sa = new SortDecorator(sa, cmp); // Result: [1, 2, 3]
sa = new MapDecorator(sa, func); // Result: [2, 4, 6]
```

2. Після коректної реалізації перелічених вище класів має почати проходити модульний тест
   SmartArrayAppTest.testFilterPositiveIntegersSortAndMultiplyBy2
   для методу
   SmartArrayApp.filterPositiveIntegersSortAndMultiplyBy2

3. Напишіть тіло методу

*SmartArrayApp.*findDistinctStudentNamesFrom2ndYearWithGPAgt4AndOrderedBySurname

так, щоб він проходив модульний тест *SmartArrayAppTest.testFindDistinctStudentNamesFrom2ndYearWithGPAgt4AndOrderedBySurname*

Зазначений метод приймає масив студентів, видаляє повтори, відбирає тих хто вчиться на другому курсі та чий середній бал вище 4-х, сортує їх по прізвищу та повертає масив рядків в якому міститься Прізвище та Ім'я студента.

Приклад вхідних даних:

```
Student[] students = {
        new Student("Ivar", "Grimstad", 3.9, 2),
        new Student("Ittai", "Zeidman", 4.5, 1),
        new Student("Antons", "Kranga", 4.0, 2),
        new Student("Burr", "Sutter", 4.2, 2),
        new Student("Philipp", "Krenn", 4.3, 3),
        new Student("Tomasz", "Borek", 4.1, 2),
        new Student("Ittai", "Zeidman", 4.5, 1),
        new Student("Burr", "Sutter", 4.2, 2)};
```

Очикуваний результат:

```
String[] expectedStudentNames = {"Borek Tomasz", "Kranga Antons", "Sutter Burr"};
```

4. Аналогічно до попереднього завдання має бути створена нова Build Job на системі **CI Hudson/Jenkins** з іменем ***apps19<surname>-hw3*** та виконані вимоги до якості коду та його покриття тестами