Звіт

До лабораторної роботи № 4

З дисципліни ”Сучасні середовища програмування”

Студента групи ПА-17-1

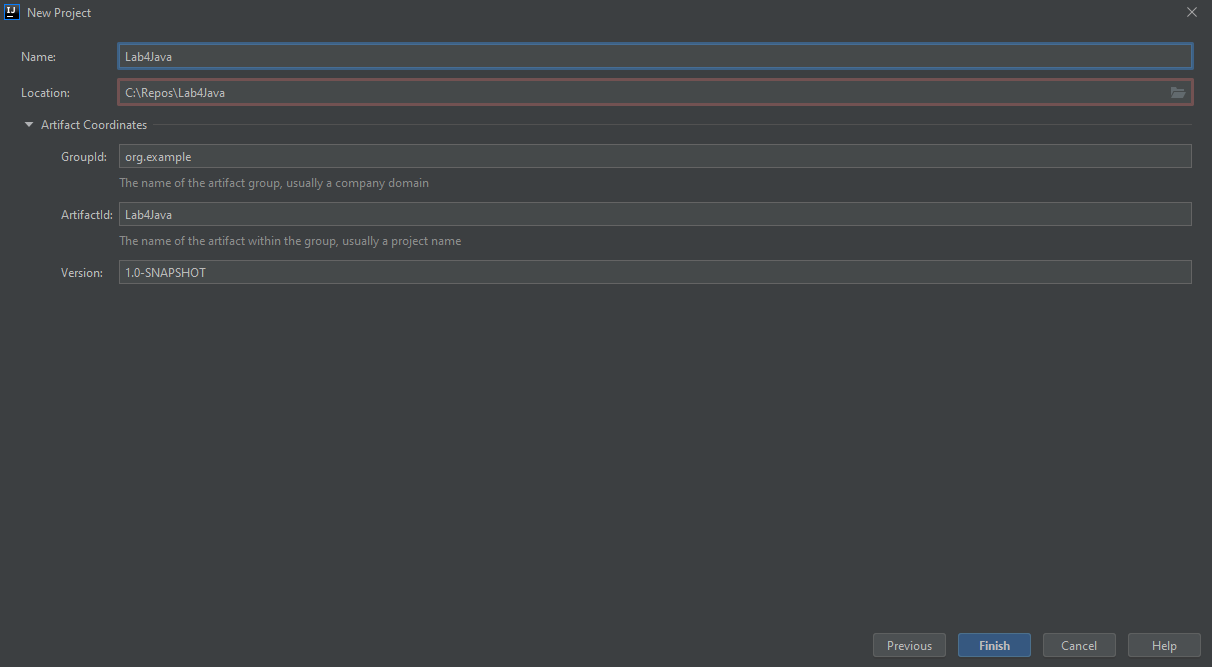
Кригіна Михайла

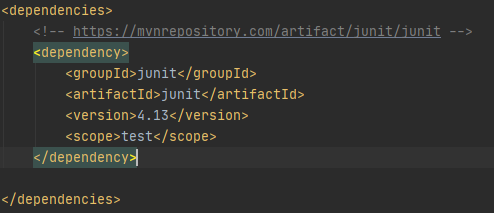
# Постановка задачі

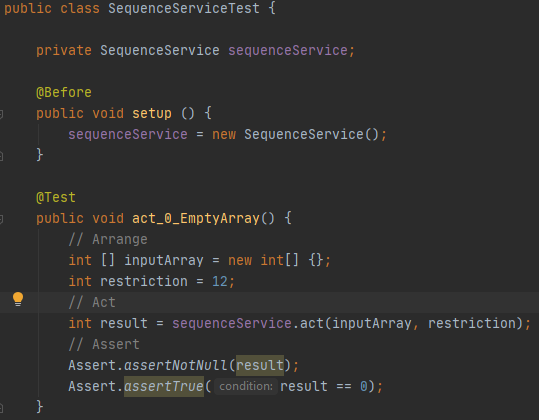
1. Необхідно ознайомитися з засобами модульного тестування програмного забезпечення та вивчити можливості засобу JUnit у середовищі IntelliJ IDEA (або в іншому середовищі програмування Java).
2. Додати до проекту лабораторної роботи № 3 бібліотеку JUnit.
3. Розробити набір unit-тестів для програми яка розв’язує задачу пакування рюкзака з однаковими вагами та цінністю предметів (value independent knapsack problem): для цього написати щонайменше по два тести для кожного власного методу з кожного класу, який було реалізовано в лабораторній роботі.
4. Тести мають бути лаконічними, доцільними та покривати весь код програми.
5. Продемонструвати необхідність використання перевантаження методу equals() для порівняння власних класів.
6. Продемонструвати скриншоти успішно пройдених тестів.
7. Додати реалізовані тести до віддаленого репозиторію

# Хід роботи

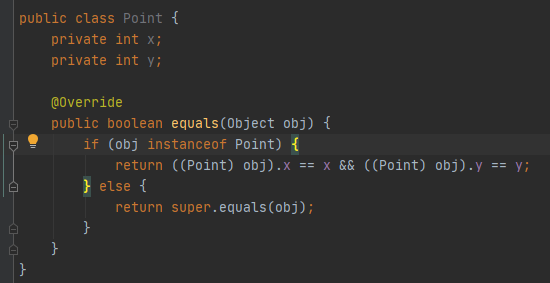
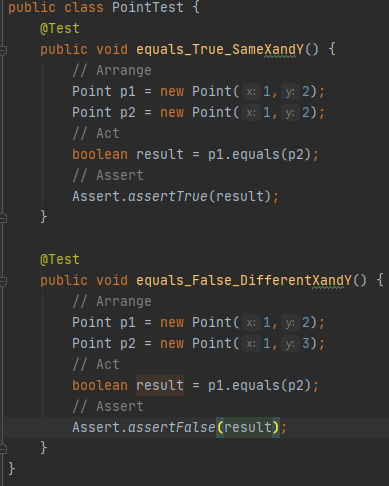
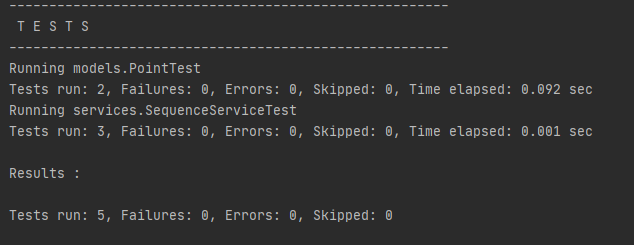
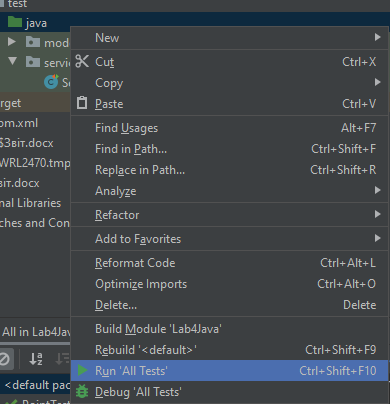
1. Створимо новий проект у Intellij IDEA з використанням Maven як засобу для керування процесу збірки проекту:



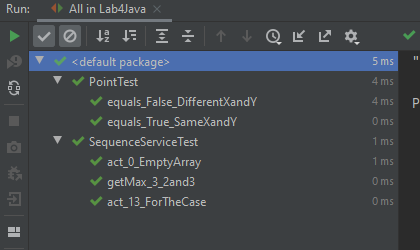
1. Розв’яжемо проблему пакування рюкзаку, застосувавши метод динамічного програмування
2. Підключимо бібліотеку JUnit, додавши вказану нижче конфігурацію до головного файлу pom проекту: 
3. Напишемо тести застосовуючи методологію ААА (Arrange, Act, Assert). Приклад:



Метод позначається як тест за допомогою анотації **@Test**. Анотація **@Before** декларується над методом, який треба викликати кожний раз перед запуском метода з анотацією **@Test**. В нашому випадку ми створюємо новий екземпляр сервайсу перед кожним запуском тесту. Зроблено це для того, щоб результати минулого тесту не вплинули на подальші тести. Слід зауважити, що тести мають певну структуру назви у форматі <метод>\_<очікуваний результат>\_<вхідні умови>. Назва «act\_0\_EmptyArray» говорить про те, що ми тестуємо метод «act», очікуємо отримати «0», та подаємо на вхід порожній масив

1. Задля перевірки перевантаження методу equals новий клас Point, який будемо вважати однаковими, якщо його параметри x та y будуть однаковими: 
2. Напишемо тести:  
   
3. Задля запуску всіх тестів, скористаємося командою mvn:test та отримаємо:   
   Або використаємо внутрішні засоби Intellij IDEA:  
   

Та отримаємо результати:

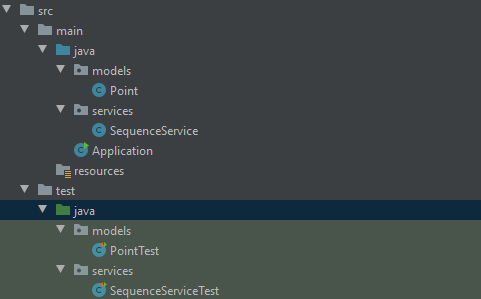


# Висновки

Завдяки засобам Intellij IDEA та бібліотеці модульного тестування Junit можна гарантувати коректну роботу окремих методів найбільш важливий класів в будь-якому програмному продукті, який був написаний за допомогою java. Основною перевагою тестування в JUinit є відсутність необхідності повторної побудови проекту для запуску одного тесту. Такий підхід гарантує підвищену швидкість тестування на великих проектах, де процес компіляції може перевищувати декілька хвилин. В цілому основною перевагою модульного тестування є можливість миттєво реагувати на помилки у коді, та швидко їх позбавлятись. У додаток слід зауважити, що JUnit э однією з найбільш поширених бібліотек тестування, тому ознайомлення х її можливостями є необхідним кроком у вивчені java.

# Додатки

Структура проекту:



<https://github.com/Noderoid64/Lab4Java/network> - посилання на сторінку github