Лабораторні роботи ООП

Bohdan Shyiak

Зміст

Вступ

Даний посібник містить список лабораторних робіт по курсу Об’єктно-орієнтоване програмування для студентів 2 курсу спеціальності Прикладна математика. Від студента очікуються засвоєні знання з лінійної алгебри та математичного аналізу, а також здатність самостійно розібратись в алгоритмах деяких чисельних методів.

Загальні правила оцінювання

- Студент має сформувати достатній набір тестових сценаріїв. У разі відсутності важливих тестових сценаріїв робота може бути відправлена на доопрацювання або знижена оцінка, на розсуд викладача. Рекомендується мати покриття тестами не менше ніж 70%.
- Викладач може задавати теоретичні питання, за відсутність відповіді або неправильну відповідь оцінка може бути знижена.
- За здачу роботи після дедлайну оцінка буде знижена відповідно по РСО.

1. ТДД ката

1.1. Завдання

Реалізувати всі кроки String Calculator TDD ката (за основу взято версію [1])

Крок 1. Створити клас `StringCalculator` з публічним нестатичним методом, що має наступну сигнатуру:

```
int add(String numbers);
```

Метод має примати до 2 чисел, розділених комою, та повертати їх суму. Наприклад, для `""`, `"1"`, `"1,2"` метод має повертати 0, 1, 3 відповідно.

Поради

- Починайте з найпростішого тест кейсу (з пустим рядком), потім рухайтесь до одного числа, потім до пари
- Не забувайте розв'язувати завдання якомога простіше, щоб змусити себе писати тести, про які ви не думали
- Не забувайте про рефакторинг після кожного проходження тесту

Крок 2. Додайте можливість обробки довільної кількості чисел

Крок 3. Додайте можливість обробки символа нової строки замість коми. Наступні вхідні дані є допустимими: `"1\n2,3"` (результат 6) Наступні вхідні дані є некоректними та мають оброблятись відповідним чином: `"1,\n"`

Крок 4. Додайте підтримку роздільників, що задаються користувачем. Для задання роздільника використовуйте наступний шаблон `//[delimiter]\n[numbers…]`. Тобто перший рядок, що починається з `//` визначає роздільник. Наприклад, для вхідного тексту `"//;\n1;2"` роздільником має бути символ `;`, а результат обчислень — 3.

Важливо

Рядок з роздільником є опціональним, тобто всі попередні сценарії мають працювати без змін.

Крок 5. Виклик `add` з від'ємним числом має викликати виняток «недозволені від'ємні числа» — і переданий список від'ємних чисел. Тобто якщо є кілька від'ємних чисел, усі вони мають бути відображені в повідомленні про виняткову ситуацію.

Крок 6. Числа більше 1000 мають ігноруватись, тобто `"1000,999,1001"` дає результат 1999.

Крок 7. Роздільник може бути довільної довжини та задаватись наступним шаблоном: `//[delimiter]\n`. Таким чином `"//[]\n1*2**3"` дає в результаті 6.

Крок 8. Додайте можливість передавати декілька роздільників: `//[delimiter1][delimiter2]\n`.

Наприклад: `"//[*][%]\n1*2%3"` повертає 6.

Крок 9. Переконайтеся, що метод може обробляти кілька роздільників довжиною більше одного символу.

1.2. Правила оцінювання

- Перші 5 кроків (10 балів). Якщо хоча б 1 не виконано, робота не зараховується.
- Кроки 6-9 (10 балів). По 2 бали за крок + 2 бали за всі 4 реалізованих.

2. Моделювання математичних об’єктів

2.1. Завдання

Реалізувати клас матриця, що містить наступну функціональність. Кожен має загальну частину та одну задачу з блоку.

Загальна частина (**12 балів**):

1. Описати клас матриця (внутрішнє представлення має бути інкапсульоване).
2. Додати конструктори, що створюють пусту матрицю, матрицю заданого розміру та копію іншої матриці
3. Додати методи, що дозволяють заповнити матрицю значеннями
4. Додати методи, що дозволяють отримати заданий елемент, рядок чи стовпчик
5. Повертає розмірність матриці
6. Визначити методи equals/hashCode
7. Створити незмінний клас (immutable) з тим самим інтерфейсом, що і mutable клас.

Додати статичний метод, що створює (**3 бали**):

1. Діагональну матрицю (на основі задано вектора)
2. Одиничну матрицю
3. Матрицю-строку, заповнену випадковими значеннями
4. Матрицю-стовпчик, заповнену випадковими значеннями

Додати методи, що (**5 балів**):

1. Перетворюють матрицю в нижню та верхню трикутну
2. Реалізують операції додавання матриць та множення на скаляр
3. Реалізують операцію множення матриць
4. Повертають транспоновану матрицю
5. Повертають обернену матрицю

Бонусні 2 бали:

1. Реалізувати матрицю, що містить generic елементи, будь-яку імплементацію мінімально необхідного інтерфейсу для елементу матриці. Інтерфейс визначається студентом.

2.2. Правила оцінювання

- Варіанти завдань формуються викладачем.
- Вагові бали за виконану роботу сказані вище.

3. Власний фреймворк впровадження залежностей

Інформацію про впровадження залежностей можна знайти в записах лекцій та [2].

3.1. Завдання та правила оцінювання

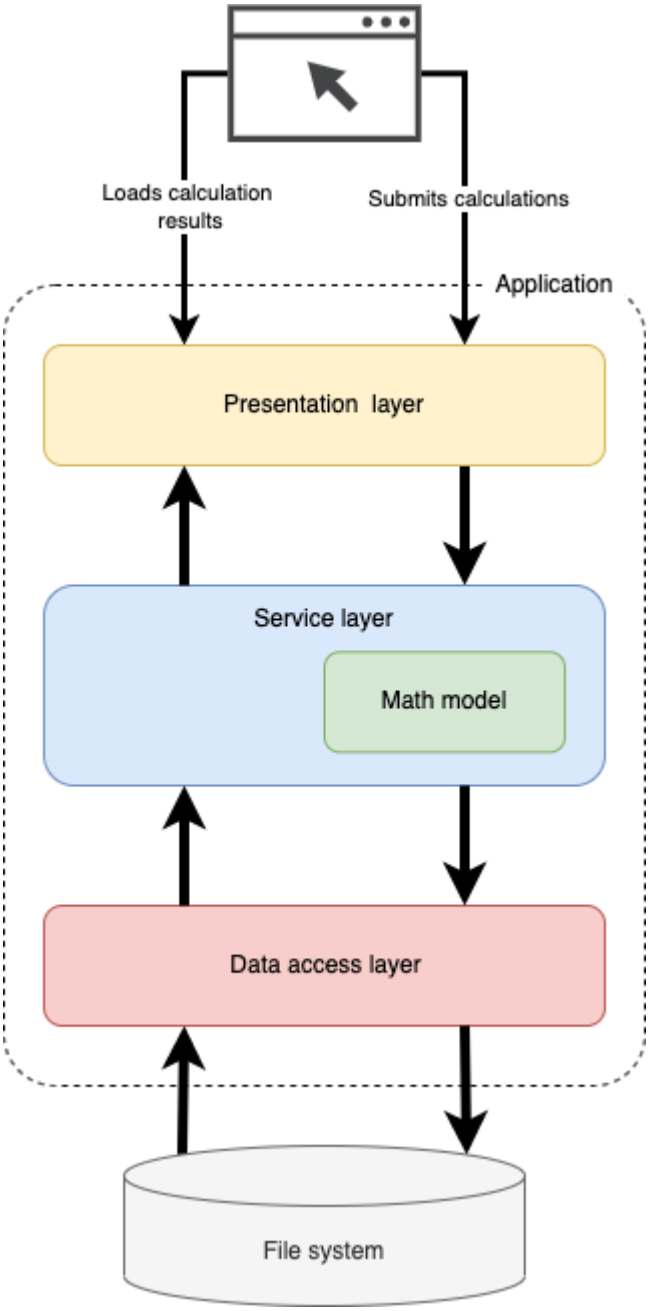
- Імплементувати бібліотеку, що дозволяє створювати екземпляри об'єктів з усіма залежностями схожу на PicoContainer. З загальними принципами роботи PicoContainer можна ознайомитись в [3]
- Студентам буде надано Gradle-проект, що містить всі необхідні публічні інтерфейси. Публічні інтерфейси змінювати без погодження з лектором не можна.
- Бібліотека має підтримувати анотації Inject та Singleton
- Проект-заглушка містить лише базові тестові сценарії, що мають на меті на прикладах продемонструвати очікуваний спосіб використання бібліотеки. **Студент має сформувати достатній набір тестових сценаріїв**. У разі відсутності важливих тестових сценаріїв робота може бути відправлена на доопрацювання або знижена оцінка, на розсуд викладача.

4. Побудова багатошарової архітектури

Сервіс на основі Spring Boot з 2 ендпоінти:

- Перший забезпечує виконання обчислень та збереження результату в файл у заданому форматі
- Другий читає результат за ідентифікатором з попереднього виклику

Вхідні та вихідні дані задаються у форматі JSON.



Малюнок 1. Схема шарів

4.1. Варіанти завдань

4.1.1. Загальні варіанти

1. Визначити взаємне розташування прямих у просторі. На вході у система має приймати об’єкти, що описують прямі у канонічному рівнянні. Результатом обчислень має бути канонічне рівняння кожної з прямих та текстовий опис їх взаємного розташування. При отриманні результату користувачу має надаватись можливість вказати мову (наприклад англійська чи українська), якою результати обчислень будуть відображені користувачу.
2. Визначити взаємне розташування прямої та площини у просторі. На вході у система має приймати об’єкти, що описує пряму у канонічному рівнянні та площину. Результатом обчислень має бути канонічне рівняння прямої, загальне рівняння площини та текстовий опис їх взаємного розташування (перпендикулярність має відображатись як окремий тип розташування). При отриманні результату користувачу має надаватись можливість вказати мову (наприклад англійська чи українська), якою результати обчислень будуть відображені користувачу.
3. Чисельне інтегрування заданої функції методом правих або лівих прямокутників (задається користувачем). Функція видається викладачем. У системі має бути передбачена легка заміна функції *на етапі компіляції* без внесення змін у код, що не пов’язаний з заданням функції. На вході у система має приймати алгоритм інтегрування, інтервал та крок. Результатом роботи має бути інтегрована функція у текстовому представленні, алгоритм інтегрування, інтервал, крок та результат інтегрування. При отриманні результату користувачу має надаватись можливість вказати мову (наприклад англійська чи українська), якою результати обчислень будуть відображені користувачу.
4. Чисельне інтегрування заданої функції методом трапеції. Функція видається викладачем. У системі має бути передбачена легка заміна функції *на етапі компіляції* без внесення змін у код, що не пов’язаний з заданням функції. На вході у система має приймати інтервал та крок. Результатом роботи має бути інтегрована функція у текстовому представленні, алгоритм інтегрування, інтервал, крок та результат інтегрування. При отриманні результату користувачу має надаватись можливість вказати мову (наприклад англійська чи українська), якою результати обчислень будуть відображені користувачу.
5. Чисельне інтегрування заданої функції методом Сімпсона. Функція видається викладачем. У системі має бути передбачена легка заміна функції *на етапі компіляції* без внесення змін у код, що не пов’язаний з заданням функції. На вході у система має приймати інтервал та крок. Результатом роботи має бути інтегрована функція у текстовому представленні, алгоритм інтегрування, інтервал, крок та результат інтегрування. При отриманні результату користувачу має надаватись можливість вказати мову (наприклад англійська чи українська), якою результати обчислень будуть відображені користувачу.
6. Пошук НСД двох чисел за алгоритмом Евкліда. Користувач задає 2 числа. При отриманні результату, користувач може обрати чи отримати лише результат обчислень, чи повний список кроків, що привели до результату.

7. Імплементувати ділення 2 многочленів в стовпчик. На вході у система має приймати об’єкти, що описують многочлени. При отриманні результату, користувач може обрати чи отримати лише результат обчислень, чи повний список кроків, що привели до результату.
8. Ортогоналізувати систему векторів довільної розмірності. На вході у система має приймати об’єкти, що описують систему векторів. При отриманні результату, користувач може обрати чи отримати лише результат обчислень, чи повний список кроків, що привели до результату.
9. Перевірити знаковизначеність квадратичної форми. При отриманні результату, користувач може обрати чи отримати лише результат обчислень, чи повний список кроків, що привели до результату.

4.1.2. За бажанням студента

1. Визначення типу поверхні другого порядку. На відміну від основних завдань, для даного завдання студент має побудувати веб-інтерфейс, що дозволяє задавати поверхні другого порядку та відображати результати для попередньо оцінених поверхонь. У користувача має бути можливість переглянути список вже визначених поверхонь. Коли користувач обирає детальний перегляд результатів, система має відображати пояснення, на основі який фактів було прийняте рішення про той чи інший тип поверхні. **Розробка веб-інтерфейсів не розглядається в рамках даного курсу.**

Посилання та джерела

[1] R. Osherove, “TDD Kata 1 - String Calculator,” *Roy Osherove*. [Online]. Available: https://osherove.com/tdd-kata-1.

[2] M. Fowler, “Inversion of Control Containers and the Dependency Injection pattern,” *martinfowler.com*. Jan. 2004, [Online]. Available: https://martinfowler.com/articles/injection.html.

[3] A. Hellesoy and J. Tirsen, “PicoContainer - Introduction.” Sep. 2017, [Online]. Available: http://picocontainer.com/introduction.html.