Методичні вказівки до лабораторних робіт з курсу «Інженерія програмного забезпечення»

Лабораторна робота №4

Тема: Структурні шаблони проектування ПЗ - 2. Шаблони Flyweight, Adapter, Bridge, Facade.

Мета: Вивчення структурних шаблонів. Отримання базових навичок з застосування шаблонів Flyweight, Adapter, Bridge, Facade.

Завдання

1. Закріпити призначення шаблонів проектування ПЗ, їх класифікацію. Знати назву і коротку характеристику кожного з шаблонів, що відносяться до певного класу.

2. Повторити структурні шаблони проектування ПЗ. Знати загальну характеристику структурних шаблонів та призначення кожного з них.

3. Детально вивчити структурні шаблони проектування Flyweight, Adapter, Bridge, Facade. Для кожного з них:

- вивчити Шаблон, його призначення, альтернативні назви, мотивацію, випадки коли його застосування є доцільним та результати такого застосування;
- знати особливості реалізації Шаблону, споріднені шаблони, відомі випадки його застосування в програмних додатках;
- вільно володіти структурою Шаблону, призначенням його класів та відносинами між ними;
- вміти розпізнавати Шаблон в UML діаграмі класів та будувати сирцеві коди Java-класів, що реалізують шаблон.

4. В підготованому проекті (ЛР1) створити програмний пакет com.lab111.labwork4. В пакеті розробити інтерфейси і класи, що реалізують завдання (згідно варіанту) з застосуванням одного чи декількох шаблонів (п.3). В розроблюваних класах повністю реалізувати методи, пов'язані з функціюванням Шаблону. Методи, що реалізують бізнес-логіку закрити заглушками з виводом на консоль інформації про викликаний метод та його аргументи. Приклад реалізації бізнес-методу:

```
void draw(int x, int y){
    System.out.println("Метод  draw  з  параметрами
```

```
x=”+x+” y=”+y);
}
```

5. За допомогою автоматизованих засобів виконати повне документування розроблених класів (також методів і полів), при цьому документація має в достатній мірі висвітлювати роль певного класу в загальній структурі Шаблону та особливості конкретної реалізації.

Варіанти (№зк **mod** 11)

0. Визначити специфікації класів та реалізацію методів для об'єктів-гліфів латинського алфавіту та об'єктів-строк. Об'єкти-гліфи мають атрибути координат та використовуються в якості складових при побудові композитних об'єктів-строк. Забезпечити ефективне використання пам'яті при роботі з великою кількістю об'єктів-гліфів. Реалізувати метод виводу строки.
1. Визначити специфікації класів, які подають графічні об'єкти у редакторі растрової графіки - примітиви (точка) та їх композиції (прямокутне зображення). Забезпечити ефективне використання пам'яті при роботі з великою кількістю графічних об'єктів. Реалізувати метод рисування графічного об'єкту.
2. Визначити специфікації класів, які подають графічні об'єкти у редакторі векторної графіки - примітиви (лінія) та їх композиції (прямокутник, трикутник). Забезпечити ефективне використання пам'яті при роботі з великою кількістю графічних об'єктів. Реалізувати метод рисування графічного об'єкту.

3. Визначити специфікації класів, які подають об'єкти-іконки для зображення елементів файлової системи при побудові графічного інтерфейсу користувача (GUI) - примітиви (файли) та їх композиції (директорії). Забезпечити ефективне використання пам'яті при роботі з великою кількістю графічних об'єктів. Реалізувати метод рисування графічного об'єкту.

4. Визначити специфікації класів, які подають графічні об'єкти у редакторі векторної графіки - примітиви (точка) та їх композиції (коло). Примітиви мають атрибути координат в декартовій системі, а об'єкти-композиції — в полярній. Відповідно інтерфейс точки містить методи setX(int) та setY(int), а метод малювання кола може оперувати лише методами setRo(double), setPhi(double) примітива (які відсутні в класі точки). Забезпечити можливість використання функціональності точки при малюванні кола без зміни інтерфейсу точки та методу малювання кола.

5. Визначити специфікації класів, які подають графічні об'єкти у редакторі векторної графіки - примітиви (точка) та їх композиції (лінія). Примітиви мають цілочислові атрибути координат (в пікселях), а об'єкти-композиції — раціональні (в сантиметрах). Відповідно інтерфейс точки містить методи setX(int) та setY(int), а метод малювання лінії може оперувати лише методами setX(double), setY(double) примітива (які відсутні в класі точки). Забезпечити можливість використання функціональності точки при малюванні лінії без зміни інтерфейсу точки та методу малювання лінії.

6. Визначити специфікації класів, які подають графічні об'єкти у редакторі векторної графіки - примітиви (лінія) та їх композиції (прямокутник). Примітиви мають атрибути координат в системі з центром координат в лівому верхньому куті екрана з інверсною віссю абсцис, а об'єкти-композиції — з центром координат посередині екрана і стандартним напрямком вісі абсцис. Забезпечити можливість використання функціональності лінії при малюванні прямокутника без зміни методів точки та методу малювання лінії.

7. Визначити специфікації класів, які подають елементи графічного інтерфейсу користувача (GUI) — кнопка, вікно, тощо. Забезпечити розділення абстракції і реалізації таким чином, щоб елементи інтерфейсу могли мати реалізації для різних бібліотек (наприклад Qt та GTK) прозорі для користувача. Реалізувати метод малювання елементу.

8. Визначити специфікації класів, які подають графічні об'єкти у редакторі векторної графіки (прямокутник) через різні інтерфейси API1 та API2. Забезпечити прозору для користувача можливість заміни реалізації графічних об'єктів. Реалізувати метод малювання елементу.

9. Визначити специфікації класів, які подають об'єкти для маніпулювання елементами файлової системи - файлами та директоріями. Інтерфейс файлу містить методи open(String path, boolean createIfNotExist), close() та delete(String path) для відкриття, закриття та видалення файлу (при createIfNotExist==true файл буде створений, якщо він не існує або обрізаний до нульової довжини, якщо існує). Інтерфейс директорії містить методи create(String path), та rmdir(String path) для створення та видалення директорії. Задати підсистему з 3-ох файлів та 2-х директорій. Забезпечити можливість створення та видалення такої підсистеми через методи create(), destroy() та зміни структури

підсистеми без впливу на її користувача.

10. Визначити специфікації класів, які подають графічні об'єкти у редакторі векторної графіки — лінія (Line) та растрове зображення (Image). Інтерфейс лінії містить метод setOpacity(double op), що регулює рівень її непрозорості між повністю непрозорою (op == 1.0) та невидимою (op == 0.0). Інтерфейс растрового зображення містить метод setTransparency (double tr), що регулює рівень її прозорості між повністю прозорою (tr == 1.0) та повністю непрозорою (tr == 0.0). Задати підсистему з зображення та ліній, які його обрамляють. Забезпечити можливість вмикання/вимикання відображення підсистеми через метод show(boolean vis), та зміни типу обрамлення (горизонтальне, вертикальне, повне, тощо) без впливу на користувача такої підсистеми.

Протокол

Протокол має містити титульну сторінку (з номером залікової книжки), завдання, роздруківку діаграми класів, розроблений сирцевий код та згенеровану документацію в форматі JavaDoc.

Матеріали

Підготовка до лабораторної роботи здійснюється за допомогою книги:

Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес Приемы объектно-ориентированного проектирования. Паттерны проектирования = Design Patterns: Elements of Reusable Object-Oriented Software. — СПб: «Питер», 2007. — С. 366. — ISBN 978-5-469-01136-1 (также ISBN 5-272-00355-1)

За необхідності додаткової інформації можливо використання матеріалів з мережі Інтернет, наприклад:

Шаблони проектування програмного забезпечення

- Шаблони проектування програмного забезпечення
- Шаблон проектирования
- Обзор паттернов проектирования
- Объектно-ориентированное проектирование, паттерны проектирования (Шаблоны)
- David Gallardo. Шаблоны проектирования Java
- Design pattern (computer science)
- Подготовка к собеседованию по Java/J2EE

Структурні шаблони

- Структурні шаблони
- Structural pattern
- Структурные шаблоны
- Шаблоны проектирования: структурные паттерны
- Структурные шаблоны проектирования