**Практична робота №3-4. Застосування методів рефакторінгу при розробці програмного забезпечення**

**Мета роботи**. Навчитися застосовувати на практиці прийоми рефакторингу та оптимізації коду.

**Завдання на лабораторну роботу**

1. Ознайомитись з теоретичними відомостями, необхідними для виконання роботи.

2. Використовуючи власні тексти програм виконати їх рефакторинг та оптимізацію коду.

3. Оформити звіт.

4. Відповісти на контрольні запитання.

**Зміст звіту**

1. Прізвище, назва групи, назва роботи.

2. Текст розробленого програмного забезпечення до проведення рефакторингу з коментарями щодо фрагментів програмного коду, які підлягають рефакторингу, з визначенням підстав для проведення рефакторингу та напрямок проведення рефакторингу.

3. Текст розробленого програмного забезпечення після проведення рефакторингу .

4. Висновки щодо цілей, які досягаються через проведення рефакторингу .

**Контрольні запитання**

1. Що таке рефакторинг?
2. Які підстави проведення рефакторингу?
3. Яким чином можна оптимізувати код?
4. Яким чином засоби рефакторингу впливають на ризики програмного забезпечення?
5. **Строк виконання цієї роботи ІПЗ-31 30.03.2023**

ТЕОРЕТИЧНІ ВІДОМОСТІ

**Рефакторинг** – підхід до розроблення програмного забезпечення з удосконаленням коду шляхом послідовності семантично коректних перетворень, зміна внутрішньої структури програмного забезпечення для полегшення розуміння коду і здешевлення модифікації.

**Підстави для проведення рефакторингу**:

– код дублюється;

– метод занадто довгий;

– цикл занадто довгий або рівень вкладеності тіла циклу занадто великий;

– клас має погану зв’язність;

– інтерфейс класу не формує узгоджену абстракцію;

– метод має занадто багато параметрів;

– окремі частини класу змінюються незалежно від інших частин;

– під час зміни програми потрібно паралельно змінювати декілька класів;

– доводиться паралельно змінювати декілька ієрархій наслідування;

– споріднені дані, які використовуються разом, не організовані в клас;

– назва класу чи методу має ім'я, яке недостатньо точно відповідає змісту;

– занадто поширене використання глобальних змінних;

– складний код пояснюється за допомогою коментарів;

– підклас використовує тільки деяку частину методів батьківського класу;

– об’єкт-посередник нічого не виконує;

– дані–члени класу відкриті.

**Напрямки проведення рефакторингу** можуть включати наступні:

– рефакторинг даних;

– рефакторинг на рівні окремих операторів;

– рефакторинг на рівні окремих методів;

– рефакторинг реалізації класів;

– рефакторинг інтерфейсів класів;

– рефакторинг на рівні системи.