МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ  
«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |  |
| --- | --- |
| Факультет | Физико-технический |
| Кафедра | Компьютерных технологий (КТ)  Зав. кафедрой КТ  Т. В. Ермоленко  (подпись)  «\_\_\_» 2017 г.  **Лабораторная работа №4** |
| Автор работы | Лямцев О. А. |
| Направление | 09.03.01 Информатика и вычислительная техника |

**Донецк**

**2021**

ГОУ ВПО «Донецкий национальный университет»  
кафедра Компьютерных технологий

Утверждаю

Зав. кафедрой КТ

подпись дата

ЗАДАНИЕ

*Краткая постановка задачи:*

1. Приведите определения рефакторинга. Что отличает рефакторинг от оптимизации производительности?

2. Укажите цели рефакторинга.

3. Когда следует проводить рефакторинг и когда он не нужен?

4. Приведите список признаков проблемного кода («запахи» кода).

5. Перечислите приемы рефакторинга, касающиеся составления методов. Опишите приемы «выделение метода» и «встраивание метода».

6. Перечислите приемы рефакторинга, касающиеся перемещения функций между объектами. Опишите приемы «перемещение метода» и «перемещение поля».

7. Перечислите приемы рефакторинга, касающиеся реорганизации данных.

8. Перечислите приемы рефакторинга, касающиеся упрощения вызовов методов.

9. Перечислите приемы рефакторинга, касающиеся реорганизации условных выражений.

10. Перечислите приемы рефакторинга, касающиеся решения задач обобщения.

**Ответы на контрольные вопросы**

1. **Рефакторинг** - изменение внутренней структуры [программы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), но без изменения внешнего поведения с целью облегчить понимание её работы.

Цель **оптимизации** — улучшение производительности программы, а **рефакторинга** — улучшение понятности кода. После **оптимизации** исходный код может стать сложнее для понимания. После **рефакторинга** программа может начать работать быстрее, но главное — её код становится проще и понятнее.

2. Цель рефакторинга — сделать код более легким для понимания.

3. **Когда не нужен:**

1) Необходимость переписать программу с нуля. Иногда имеющийся код настолько запутан, что подвергнуть его рефакторингу, конечно, можно, но проще начать все с самого начала.

2) Близость даты завершения проекта

**Когда нужен:**

1) Применяйте рефакторинг при добавлении новой функции

2) Применяйте рефакторинг, если требуется исправить ошибку

3) Применяйте рефакторинг при разборе кода

4.

|  |
| --- |
| Duplicated Code (Повторяющийся код) |
| Long Method (Длинный метод) |
| Conditional Complexity (Сложность условий) |
| Primitive Obsession (Зацикленность на примитивах) |
| Indecent Exposure (Неприличная демонстрация) |
| Solution Sprawl (Расплывшееся решение) |
| Alternative Classes with Different Interfaces (Альтернативные классы с различными интерфейсами) |
| Lazy Class (Ленивый класс) |
| Large Class (Большой класс) |
| Switch Statements (Операторы выбора вариантов) |
| Combinatorial Explosion (Комбинаторный взрыв) |
| Oddball Solution (Оригинальное решение) |

5. Извлечение и встраивание метода, извлечение и встраивание переменной, замена переменной вызовом метода, расщепление метода, удаление присваиваний параметрам, замена метода объектом метода, замена алгоритма

**Встраивание метода**: замените вызовы метода его содержимым и удалите сам метод.

**Выделение метода**: выделите участок кода в новый метод (или функцию) и вызовите этот метод вместо старого кода.

6. Перемещение метода и поля, извлечение и встраивание класса, сокрытие делегирования, удаление посредника, введение внешнего метода, введение локального расширения

**Перемещение метода**: создайте новый метод в классе, который использует его больше других, и перенесите туда код из старого метода. Код оригинального метода превратите в обращение к новому методу в другом классе либо уберите его вообще.

**Перемещение поля**: создайте поле в новом классе и перенаправьте к нему всех пользователей старого поля.

7. Замена значений ссылкой, замена ссылки значением, дублирование видимых данных, самоинкапсуляция поля, замена простого поля объектом, замена поля-массива объектом, замена однонаправленной связи двунаправленной, замена двунаправленной связи однонаправленной, инкапсуляция поля, инкапсуляция коллекции

8. Добавление параметра, удаление параметра, переименование метода, разделение запроса и модификатора, параметризация метода, замена параметров объектом, передача всего объекта, удаление сеттера, замена кода ошибки исключением, сокрытие метода

9. Объединение условных операторов, объединение дублирующихся фрагментов условных операторов, разбиение условного оператора, замена условного оператора полиморфизмом, удаление управляющего флага

10. Подъем поля, подъем метода, подъем тела конструктора, спуск поля, спуск метода, извлечение подкласса, извлечение суперкласса, извлечение интерфейса, свёртывание иерархии, замена делегирования наследованием