МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХФРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра програмної інженерії

ЗВІТ

З практичної роботи № 2

з дисципліни «Аналіз та рефакторинг коду»

з теми: «Методи рефакторингу коду програмного забезпечення»

Виконала
ст. гр. ПЗПІ-22-7
Великотрав Віолетта

Перевірив
ст. викладач кафедри ПІ
Сокорчук І. П.

Харків 2024

1.1 Мета роботи

Вивчити основні методи рефакторингу коду на основі реальних прикладів. Навчитись ідентифікувати проблеми в коді та використовувати відповідні методи рефакторингу для покращення його якості.

2.1 Хід роботи

Обрати три методи рефакторингу коду із книги Мартіна Фаулера «Refactoring. Improving the Design of Existing Code». Створити презентацію на тему «Методи рефакторингу коду програмного забезпечення».

Було обрано методи Extract Method (Виділення методу), Rename Method (Перейменування методу), Simplify Conditional Expression (Спрощення умовних виразів).

Висновки

Було вивчено основні методи рефакторингу коду на основі реальних прикладів. Набуто навичок ідентифікування проблем в коді та використання відповідних методів рефакторингу для покращення його якості.

Відео-презентація: https://youtu.be/qI919VIlF10

Додаток А

Програмний код, використаний як приклад у презентації:

```
// Extract Method
// Код до рефакторингу

function calculateOrder(cartItems) {
  let total = 0;

  for (let item of cartItems) {
    total += item.price * item.quantity;

    if (item.category === "electronics") {
      console.log(`Applying electronics discount for ${item.name}`);
      total -= item.price * 0.1;
    }
  }

  console.log(`Total order price: ${total}`);
  return total;
}

// Код після рефакторингу

function calculateOrder(cartItems) {
  let total = 0;

  for (let item of cartItems) {
    total += calculateItemPrice(item);
    logDiscount(item);
  }

  console.log(`Total order price: ${total}`);
  return total;
}

function calculateItemPrice(item) {
  let price = item.price * item.quantity;
  if (item.category === "electronics") {
    price -= item.price * 0.1;
  }
  return price;
}

function logDiscount(item) {
  if (item.category === "electronics") {
    console.log(`Applying electronics discount for ${item.name}`);
  }
```

```
}
```

```
// Rename Method
// Код до рефакторингу

function calc(cart) {
  let sum = 0;
  for (let item of cart) {
    sum += item.price * item.quantity;
  }
  return sum;
}

// Код після рефакторингу

function calculateCartTotal(cart) {
  let sum = 0;
  for (let item of cart) {
    sum += item.price * item.quantity;
  }
  return sum;
}
```

```
// simplify conditional expression
// Код до рефакторингу

function getShippingCost(order) {
  if (order.total > 1000) {
    return 0;
  } else if (order.total > 500) {
    return 10;
  } else {
    return 20;
  }
}

// Код після рефакторингу

function getShippingCost(order) {
  if (isFreeShipping(order)) return 0;
  if (isMediumShipping(order)) return 10;
  return 20;
}

function isFreeShipping(order) {
```

```
  return order.total > 1000;
}

function isMediumShipping(order) {
  return order.total > 500;
}
```

Додаток Б

Презентація на тему «Правила оформлення програмного коду»

МЕТОДИ РЕФАКТОРИНГУ КОДУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Великотрав Віолетта ПЗПІ-22-7

РЕФАКТОРИНГ

Рефакторинг – це процес удосконалення коду без зміни його зовнішньої поведінки. Це не лише інструмент покращення якості коду, а й стратегія забезпечення довготривалої ефективності розробки. Він дозволяє уникнути хаосу в проєкті та створює фундамент для майбутніх оновлень.

Мета рефакторингу — покращення якості коду без зміни його функціональності

EXTRACT METHOD

Призначення: виділення фрагментів коду у нові методи для покращення читабельності

```
function calculateOrder(cartItems) {
  let total = 0;

  for (let item of cartItems) {
    total += item.price * item.quantity;

    if (item.category === "electronics") {
      console.log(`Applying electronics discount for ${item.name}`);
      total -= item.price * 0.1;
    }
  }

  console.log(`Total order price: ${total}`);
  return total;
}
```

Код до рефакторингу

Код після рефакторингу:

```
function calculateOrder(cartItems) {
  let total = 0;

  for (let item of cartItems) {
    total += calculateItemPrice(item);
    logDiscount(item);
  }

  console.log(`Total order price: ${total}`);
  return total;
}

function calculateItemPrice(item) {
  let price = item.price * item.quantity;
  if (item.category === "electronics") {
    price -= item.price * 0.1;
  }
  return price;
}

function logDiscount(item) {
  if (item.category === "electronics") {
    console.log(`Applying electronics discount for ${item.name}`);
  }
}
```

Переваги:

- Читабельність: Логіка розбита на менші частини, що легше зрозуміти.
- Повторне використання: Можна повторно використовувати окремі функції.
- Тестування: Тести стають простішими і точнішими, оскільки можна тестувати кожну частину окремо.

RENAME VARIABLE

Призначення: зміна імен для поліпшення розуміння коду

```
function calc(cart) {
  let sum = 0;
  for (let item of cart) {
    sum += item.price * item.quantity;
  }
  return sum;
}
```

Код до рефакторингу

Код після рефакторингу:

```
function calculateCartTotal(cart) {
  let sum = 0;
  for (let item of cart) {
    sum += item.price * item.quantity;
  }
  return sum;
}
```

Переваги:

- Зрозумілість: Метод має чітку назву, що описує його функціональність.
- Читабельність: Легше зрозуміти, що робить метод без необхідності заглядати в його код.
- Підтримка: Розробникам легше працювати з кодом, оскільки вони одразу розуміють, як використовувати метод.

SIMPLIFY CONDITIONAL EXPRESSIONS

Призначення: зменшення складності умовних конструкцій

```
function getShippingCost(order) {
  if (order.total > 1000) {
    return 0;
  } else if (order.total > 500) {
    return 10;
  } else {
    return 20;
  }
}
```

Код до рефакторингу

Код після рефакторингу:

```
function getShippingCost(order) {
  if (isFreeShipping(order)) return 0;
  if (isMediumShipping(order)) return 10;
  return 20;
}

function isFreeShipping(order) {
  return order.total > 1000;
}

function isMediumShipping(order) {
  return order.total > 500;
}
```

Переваги:

- Читабельність: Логіка умов стала зрозумілішою завдяки розділенню на окремі функції.
- Зменшення складності: Умови більше не вкладені, і їх легше змінювати.
- Підтримка: Легко додавати нові умови, оскільки кожну умову можна окремо модифікувати без впливу на інші частини коду.

ВИСНОВКИ

Рефакторинг є важливим процесом для підтримки високої якості коду. Він дозволяє покращити читабельність, знизити складність та підвищити підтримуваність коду.

Методи рефакторингу слід застосовувати, коли код стає важким для розуміння або підтримки, при додаванні нових функцій, під час тестування або для покращення продуктивності. Це дозволяє зберігати код чистим і зрозумілим на всіх етапах розробки.

СПИСОК ДЖЕРЕЛ

1) Fowler, Martin. Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional, 1999. – 464 p.

2) Martin, Robert C. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008. – 464 p.

3) McConnell, Steve. Code Complete: A Practical Handbook of Software Construction. Microsoft Press, 2004. – 960 p.