**Лабораторная работа № 1**

**«Рефакторинг программного кода. Составление методов»**

**Цель работы**

Исследовать эффективность составления методов при рефакторинга программного кода. Получить практические навыки применения приемов рефакторинга методов.

**Постановка задачи**

1. Выбрать фрагмент программного кода для рефакторинга.

2. Выполнить рефакторинг программного кода, применив не менее 7 приемов, рассмотренных в разделе 2.2.

3. Составить отчет, содержащий подробное описание каждого модифицированного фрагмента программы и описание использованного метода рефакторинга.

**Ход работы**

1. Выделение метода (Extract Method)

Участок кода до рефакторинга:

class Board

{

private:

char board[3][3];

public:

Board()

{

**for (int i = 0; i < 3; i++)**

**{**

**for (int j = 0; j < 3; j++)**

**{**

**board[i][j] = '-';**

**}**

**}**

}

Получившийся код после рефакторинга:

class Board

{

private:

char board[3][3];

**void initializeBoard()**

**{**

**for (int i = 0; i < 3; i++)**

**{**

**for (int j = 0; j < 3; j++)**

**{**

**board[i][j] = '-';**

**}**

**}**

**}**

public:

Board()

{

**initializeBoard();**

}

2. Встраивание метода (Inline Method)

Код до рефакторинга:

class Student : public Youth, public Studying

{

public:

...

**bool Student::KickOut()**

**{**

**if (AverageMark < 60)**

**{**

**cout << "Student " << FIO << " probably be expelled from the university" << endl;**

**return true;**

**}**

**else**

**{**

**cout << "Student " << FIO << " probably not be expelled from the university" << endl;**

**return false;**

**}**

**}**

...

case 3:

{

system("clear");

Yth.Input();

Std.Input();

Stud.Inhert(Yth, Std);

Stud.Output();

**Stud.KickOut();**

cout << "~~~~~~~~~~~~~~~~~~~~~" << endl;

}

break;

Код после рефакторинга

...

case 3:

{

system("clear");

Yth.Input();

Std.Input();

Stud.Inhert(Yth, Std);

Stud.Output();

**if (Stud.AverageMark < 60)**

**{**

**cout << "Student " << Stud.FIO << " probably be expelled from the university" << endl;**

**return true;**

**}**

**else**

**{**

**cout << "Student " << Stud.FIO << " probably not be expelled from the university" << endl;**

**return false;**

**}**

cout << "~~~~~~~~~~~~~~~~~~~~~" << endl;

}

break;

...

3. Встраивание временной переменной (Inline Temp)

Код до рефакторинга:

int calculate\_cur\_column(int rank, int i)

{

**int cur\_column = (4 - i + rank) % 4;**

return cur\_column;

}

Код после рефакторинга:

int calculate\_cur\_column(int rank, int i)

{

**return (4 - i + rank) % 4;**

}

4.Замена временной переменной вызовом метода (Replace Temp with Query)

Код до рефакторинга:

class ShoppingCart

{

public:

void addItem(std::string name, double price, int quantity)

{

items.push\_back({name, price, quantity});

}

double calculateTotal()

{

double total = 0.0;

for (const auto &item : items)

{

**double itemTotal = item.price \* item.quantity;**

double discount = getDiscount(itemTotal);

itemTotal -= discount;

total += itemTotal;

}

return total;

}

Код после рефакторинга:

class ShoppingCart

{

public:

void addItem(std::string name, double price, int quantity)

{

items.push\_back({name, price, quantity});

}

double calculateTotal()

{

double total = 0.0;

for (const auto &item : items)

{

**double discount = getDiscount(getItemTotal(item));**

**total -= discount;**

**total += getItemTotal(item);**

}

return total;

}

private:

struct Item

{

std::string name;

double price;

int quantity;

};

std::vector<Item> items;

**double getItemTotal(const Item &item) const**

**{**

**return item.price \* item.quantity;**

**}**

5. Введение поясняющей переменной (Introduce Explaining Variable)

Код до рефакторинга:

bool checkWin(char player)

{

for (int i = 0; i < 3; i++)

{

if (board[i][0] == player && board[i][1] == player && board[i][2] == player)

{

**return true;**

}

if (board[0][i] == player && board[1][i] == player && board[2][i] == player)

{

**return true;**

}

}

if (board[0][0] == player && board[1][1] == player && board[2][2] == player)

{

**return true;**

}

if (board[0][2] == player && board[1][1] == player && board[2][0] == player)

{

**return true;**

}

**return false;**

}

Код после рефакторинга

bool checkWin(char player)

{

**bool isWin = false;**

for (int i = 0; i < 3; i++)

{

if (board[i][0] == player && board[i][1] == player && board[i][2] == player)

{

**isWin = true;**

}

if (board[0][i] == player && board[1][i] == player && board[2][i] == player)

{

**isWin = true;**

}

}

if (board[0][0] == player && board[1][1] == player && board[2][2] == player)

{

**isWin = true;**

}

if (board[0][2] == player && board[1][1] == player && board[2][0] == player)

{

**isWin = true;**

}

**return isWin;**

}

6. Расщепление временной переменной (Split Temporary Variable)

7. Удаление присваиваний параметрам (Remove Assignments to Parameters)

8. Замена метода объектом методов (Replace Method with Method Object)

9. Замещение алгоритма (Substitute Algorithm)

**Выводы**

В ходе выполнения лабораторной работы были изучены и применены на практике методы рефакторинга программного обеспечения. В соответствии с методами рефакторинга были изменены исходные части кода, затем была проверена корректность выполнения программ после рефакторинга.