Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Рефакторинг программного обеспечения”

Лабораторная работа №3

“Рефакторинг программного кода. Упрощение условных выражений”

Выполнил:

ст.гр. ИС/б-17-2

Черняев Н.Г.

Проверил:

Строганов В.А.

Севастополь

2020

1 ЦЕЛЬ РАБОТЫ

Исследовать эффективность рефакторинга программного кода путем упрощения условных выражений. Получить практические навыки применения приемов рефакторинга объектно-ориентированных программ.

2 ХОД РАБОТЫ

**1. Декомпозиция условного оператора**

Ниже представлен фрагмент кода на языке JavaScript, который реализует получение постов с сервера:

fetchStatuses = () => {

*this*.pending = true;

*if* (*this*.baseUrl) {

*if* (*this*.statuses.length !== 0) {

*if* (*this*.reverseOrder) {

                let url;

                const minId = *this*.statuses[*this*.statuses.length - 1].id;

*if* (*this*.sendLanguage) {

                    let language = localStorage.getItem("language");

*if* (language !== "en" && language !== "kr") {

                        language = "en";

                    }

                    url = `${*this*.baseUrl}?since\_id=${minId}&language=${language}`;

                } *else* {

                    url = `${*this*.baseUrl}?since\_id=${minId}`;

                }

                axiosInstance

                    .get(url)

                    .then(({ data }) => {

*if* (data.length !== 0) {

*this*.statuses.push(...data);

                        } *else* {

*this*.hasMore = false;

                        }

                    })

                    .finally(() => (*this*.pending = false));

            } *else* {

                let url;

                const maxId = *this*.statuses[*this*.statuses.length - 1].id;

*if* (*this*.sendLanguage) {

                    let language = localStorage.getItem("language");

*if* (language !== "en" && language !== "kr") {

                        language = "en";

                    }

                    url = `${*this*.baseUrl}?max\_id=${maxId}&language=${language}`;

                } *else* {

                    url = `${*this*.baseUrl}?max\_id=${maxId}`;

                }

                axiosInstance

                    .get(url)

                    .then(({ data }) => {

*if* (data.length !== 0) {

*this*.statuses.push(...data);

                        } *else* {

*this*.hasMore = false;

                        }

                    })

                    .finally(() => (*this*.pending = false));

            }

        } *else* {

            axiosInstance

                .get(*this*.baseUrl)

                .then(({ data }) => {

*if* (data.length !== 0) {

*this*.statuses.push(...data);

                    } *else* {

*this*.hasMore = false;

                    }

                })

                .finally(() => (*this*.pending = false));

        }

    }

};

Выполним декомпозицию метода, заменив фрагменты кода вызовом методов.

Код после рефакторинга представлен ниже:

fetchStatuses = () => {

*this*.pending = true;

*if* (*this*.baseUrl) {

*if* (*this*.statuses.length !== 0) {

*if* (*this*.reverseOrder) {

                fetchStatusesWithOrder("max\_id");

            } *else* {

                fetchStatusesWithOrder("since\_id");

            }

        } *else* {

            getStatuses(*this*.baseUrl);

        }

    }

};

fetchStatusesWithOrder = orderType => {

    let url;

    const id = *this*.statuses[*this*.statuses.length - 1].id;

*if* (*this*.sendLanguage) {

        let language = localStorage.getItem("language");

*if* (language !== "en" && language !== "kr") {

            language = "en";

        }

        url = `${*this*.baseUrl}?${orderType}=${id}&language=${language}`;

    } *else* {

        url = `${*this*.baseUrl}?${orderType}=${id}`;

    }

    getStatuses(url);

};

getStatuses = url => {

    axiosInstance

        .get(url)

        .then(({ data }) => {

*if* (data.length !== 0) {

*this*.statuses.push(...data);

            } *else* {

*this*.hasMore = false;

            }

        })

        .finally(() => (*this*.pending = false));

};

**2. Консолидация условного выражения**

В фрагменте кода ниже присутствует ряд условий, дающих одинаковый результат:

private void addButton\_Click(object sender, EventArgs e)

{

*if* (CheckIsEmpty())

    {

        MessageBox.Show("Необходимо заполнить все поля!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

*return*;

    }

*if* (CheckContainsNumbers())

    {

        MessageBox.Show("В поле 'Фамилия' или 'Имя' содержатся цифры!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

*return*;

    }

    DialogResult = DialogResult.OK;

}

private bool CheckIsEmpty()

{

*if* (lastNameTextBox.Text == "") *return* true;

*if* (firstNameTextBox.Text == "") *return* true;

*if* (markTextBox.Text == "") *return* true;

*if* (questionTextBox.Text == "") *return* true;

*return* false;

}

private void CheckContainsNumbers()

{

*if* (MainForm.IsInt(lastNameTextBox.Text)) *return* true;

*if* (MainForm.IsInt(firstNameTextBox.Text)) *return* true;

*return* false;

}

Заменим последовательность условий одним условным предложением с помощью логических операторов:

private bool CheckIsEmpty()

{

*return* lastNameTextBox.Text == "" || firstNameTextBox.Text == "" || markTextBox.Text == "" || questionTextBox.Text == "";

}

private void CheckContainsNumbers()

{

*return* MainForm.IsInt(lastNameTextBox.Text) || MainForm.IsInt(firstNameTextBox.Text);

}

**3. Консолидация дублирующихся условных фрагментов**

В данном фрагменте присутствует одинаковый код во всех ветвях условного выражения:

...

reaction(

    () => *this*.user,

    user => {

*if* (user) {

*if* (*this*.userFollowersStore.id !== user.id) {

*this*.userFollowersStore.reset();

*this*.userStatusesStore.setBaseUrl(`/api/v1/accounts/${user.id}/statuses`);

*this*.userStatusesStore.fetchStatuses();

            }

*if* (*this*.userFollowingStore.id !== user.id) {

*this*.userFollowingStore.reset();

*this*.userStatusesStore.setBaseUrl(`/api/v1/accounts/${user.id}/statuses`);

*this*.userStatusesStore.fetchStatuses();

            }

        }

    }

);

...

Вынесем повторяющийся код за пределы выражения:

...

reaction(

    () => *this*.user,

    user => {

*if* (user) {

*if* (*this*.userFollowersStore.id !== user.id) {

*this*.userFollowersStore.reset();

            }

*if* (*this*.userFollowingStore.id !== user.id) {

*this*.userFollowingStore.reset();

            }

*this*.userStatusesStore.setBaseUrl(`/api/v1/accounts/${user.id}/statuses`);

*this*.userStatusesStore.fetchStatuses();

        }

    }

);

...

**4. Удаление управляющего флага**

В фрагменте кода ниже имеется переменная, действующая как управляющий флаг для ряда булевых выражений:

...

bool exist = false;

*for* (int i = 0; i < n; i++) {

*if* (!exist) {

*if* (arr[i] == 0) {

            Console.WriteLine("Строка содержит цифру 0!");

            exist = true;

        }

*if* (people[i] == 1) {

            Console.WriteLine("Строка содержит цифру 1!");

            exist = true;

        }

    }

}

...

Уберем управляющий флаг и воспользуемся оператором break:

...

*for* (int i = 0; i < n; i++) {

*if* (arr[i] == 0) {

        Console.WriteLine("Список содержит цифру 0!");

*break*;

    }

*if* (people[i] == 1) {

        Console.WriteLine("Список содержит цифру 1!");

*break*;

    }

}

...

**5. Замена вложенных условных операторов граничным оператором**

Метод из кода ниже использует условное поведение, из которого неясен нормальный путь выполнения программы:

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

    bool result;

*if* (filesave) {

        result = false;

    } *else* *if* (fileName == "" && textBox1.Text != "") {

        result = MessageBox.Show("Вы хотите выйти не сохранив?", "Выход", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.No;

    } *else* *if* (fileName != "" && textBox1.Text != File.ReadAllText(fileName)) {

        result = MessageBox.Show("Вы хотите выйти не сохранив?", "Выход", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.No;

    }

*return* result;

}

Фрагмент кода после рефакторинга:

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

*if* (filesave) {

*return* false;

    }

*if* (fileName == "" && textBox1.Text != "") {

*return* MessageBox.Show("Вы хотите выйти не сохранив?", "Выход", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.No;

    }

*if* (fileName != "" && textBox1.Text != File.ReadAllText(fileName)) {

*return* MessageBox.Show("Вы уверены?", "Выход", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.No;

    }

*return* true;

}

ВЫВОД

В ходе выполнения лабораторной работы была исследована эффективность рефакторинга программного кода путем упрощения условных выражений. Получены практические навыки применения приемов рефакторинга объектно-ориентированных программ.