## Учреждение образования

## Белорусский государственный технологический университет

Кафедра полиграфического оборудования и

системы обработки информации

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

по дисциплине «Стандартизация и сертификация

информационных систем и технологий»

Тема

Изучение соглашение по оформлению кода - Java Code Conventions

Выполнил студент:

Миневич Кристина

ф-та ИТ 4 к. 4 гр.

Проверил

кандидат технических наук

Сулим П.Е.

Отчет по лабораторной работе

защищен с отметкой баллов

Минск 2024

**Лабораторная работа № 1**

**ИЗУЧЕНИЕ СОГЛАШЕНИЕ ПО ОФОРМЛЕНИЮ КОДА - JAVA CODE CONVENTIONS**

***Цель работы:*** написать Java-code по правилам Java Code Conventions.

***Описание работы:***

Необходимо ознакомится с Java Code Conventions, привести примеры Java-кода с пояснениями по оформлению.

1. **Обеспечение доступа к экземпляру и переменным класса**

Не стоит делать какие-либо переменные экземпляра или класса общедоступными без уважительной причины. Часто встречаются случаи, когда поля класса не должны устанавливаться или считываться напрямую - их чтение и запись должны происходить лишь при вызове методов.

Одним из примеров уместного использования public полей может быть случай, когда класс описывает лишь структуру данных, без какого-либо поведения. Другими словами, если бы вы могли использовать struct вместо class (если бы Java поддерживал struct), тогда можно сделать переменные экземпляра класса public.

1. **Обращение к переменным и методам класса**

Избегайте использование объекта для доступа к статическим полям и методам класса. Вместо этого используйте имя класса. Например:

classMethod(); //OK

AClass.classMethod(); //OK

anObject.classMethod(); //ИЗБЕГАЙТЕ!

Вот пример Java-кода, который следует Java Code Conventions и включает рекомендацию по обращению к переменным и методам класса:

public class MyClass {

private static int staticVariable;

public static void main(String[] args) {

// Использование имени класса для доступа к статическим методам и полям

MyClass.staticMethod();

MyClass.staticVariable = 10;

// Избегание использования объекта для доступа к статическим методам и полям

// Желательно избегать такого использования

// MyClass myObject = new MyClass();

// myObject.staticMethod();

// myObject.staticVariable = 5;

}

public static void staticMethod() {

System.out.println("This is a static method.");

}

}

1. **Константы**

Численные константы (литералы) не должны кодироваться напрямую, за исключением **-1**, **0** и **1**, которые могут использоваться в циклах для управления счетчиком.

Вот пример Java-кода, следующего рекомендации Java Code Conventions относительно использования численных констант:

public class ConstantsExample {

// Пример числовой константы для использования в коде

private static final int MAX\_ATTEMPTS = 3;

public static void main(String[] args) {

int counter = 0;

// Пример использования числовой константы в цикле

while (counter < MAX\_ATTEMPTS) {

System.out.println("Attempt " + (counter + 1));

counter++;

}

}

}

Вместо использования числового литерала напрямую в цикле, мы использовали эту константу, что делает код более читаемым и понятным.

Кроме того, рекомендуется использовать ключевое слово final для определения констант, чтобы предотвратить их изменение после инициализации.

1. **Примеры присваивания значение переменным**

Избегайте присваивания значения некоторым переменным в одном выражении. Не используйте оператор присваивания в местах, где он может быть легко спутан с оператором сравнения. Не используйте вложенные присваивания, пытаясь ускорить скорость выполнения программы. Это работа компилятора, и, кроме того, на самом деле редко помогает.

|  |
| --- |
| fooBar.fChar = barFoo.lchar = 'c'; //ИЗБЕГАЙТЕ!  if (c++ = d++) { //ИЗБЕГАЙТЕ! В Java запрещено  ...  }  //лучше написать так:  if ((c++ = d++) != 0) {  ...  }  d = (a = b + c) + r; //НЕПРАВИЛЬНО!  //должно быть написано так:  a = b + c;  d = a + r; |

1. **Круглые скобки**

Как правило, хорошей практикой является использование круглые скобки в выражениях, содержащих различные операторы, чтобы избегать проблем с приоритетом операторов. Даже если приоритет оператора вам кажется очевидным, то для других это может быть не так - вы не должны предполагать, что другие программисты знают приоритет, как и вы.

if (a == b && c == d) // ИЗБЕГАЙТЕ!

if ((a == b) && (c == d)) // ПРАВИЛЬНЕЙ ЗАПИСАТЬ ТАК

1. **Примеры кода**

Следующий пример показывает, как форматировать файл с исходным кодом Java, содержащим отдельный класс. Интерфейсы форматируются отдельно. Для более подробного изучения прочтите "[Объявление классов и интерфейсов](https://www.magnumblog.space/java/131-translating-java-code-conventions#class_and_interface_declarations)" и "[Документирующие комментарии](https://www.magnumblog.space/java/131-translating-java-code-conventions#documentation_comments)".

package java.blah;

import java.blah.blahdy.BlahBlah;

/\*\*

\* Описание класса здесь.

\*

\* @version 1.10 04 Oct 1996

\* @author Имя Фамилия

\*/

public class Blah extends SomeClass {

/\* Здесь может идти комментарий реализации класса. \*/

/\*\* Комментарий, документирующий classVar1 \*/

public static int classVar1;

/\*\*

\* Документирующий комментарий к classVar2, который бывает

\* более чем одна строка

\*/

private static Object classVar2;

/\*\* комментарий, документирующий поле instanceVar1\*/

public Object instanceVar1;

/\*\* комментарий, документирующий поле instanceVar2 \*/

protected int instanceVar2;

/\*\* комментарий, документирующий поле instanceVar3 \*/

private Object[] instanceVar3;

/\*\*

\* ...комментарий, документирующий метод Blah...

\*/

public Blah() {

//...здесь идет реализация...

}

/\*\*

\* ...комментарий, документирующий метод doSomething...

\*/

public void doSomething() {

//...здесь идет реализация...

}

/\*\*

\* ...комментарий, документирующий метод doSomethingElse...

\* @param someParam описание

\*/

public void doSomethingElse(Object someParam) {

// ...здесь идет реализация...

}

}

**Выводы:**

При выполнение лабораторной работы №1 «Изучение соглашение по оформлению кода – Java Code Conventions» получены базовые знания о оформлении программного кода на языке программирования JAVA. Были изучены основные правила оформления операций начального уровня по рекомендуемой документации Java Code Conventions. Многочисленные исследования показывают, что программист проводит гораздо больше времени за чтением кода, а не его написанием. Читать код сложнее, чем писать его. Особенно если речь идет о чужом программном коде. При следовании правилам вы можете серьезно облегчить работу себе и коллегам, если ваш код будет качественным, понятным. “Качество кода” довольно обширный термин, включающий в себя достаточно много разных аспектов, среди которых проектирование интерфейсов классов и методов, реализация методов, правильное взаимодействие классов и методов друг с другом, конвенции по форматированию и оформлению кода, именованию классов, методов и переменных, принятые в конкретной корпоративной среде. Соблюдение этих правил достаточно облегчит работу с чужим кодом.

**ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ:**

1. https://www.magnumblog.space/java/translating-java-code-conventions