Лабораторная работа №5. Документирование javadoc

Требование к отчету.

Отчет должен содержать:

- Титульный лист согласно образца с указанным номером и названием лабораторной работы.
- Оглавление с указанием номера страницы для каждого раздела.
- Цель работы.
- Теоретическое обоснование. Пишется самостоятельно и должно охватывать вопросы, затрагиваемые в лабораторной работе.
- Задание.
- Документированные листинги программ. После каждого листинга программы должен приводится скриншот с результатом её работы.
- Выводы.

Теоретическое обоснование.

При документировании приложения необходима также поддержка документации программы. Если документация и код разделены, то непроизвольно создаются сложности, связанные с необходимостью внесения изменений в соответствующие разделы сопроводительной документации при изменении программного кода.

Как правило, все существующие среды разработки IDE приложений Java предлагают решение по связыванию кода с документацией в процессе разработки с использованием **javadoc**. Для этого необходимо соответствующим образом написать комментарий к коду, т.е. документировать. Java комментарии необходимы как для комментирования программы, так и для составления или оформления документации.

Разработан специальный синтаксис для оформления документации в виде комментариев и инструмент для создания из комментариев документации. Этим инструментом является **javadoc**, который обрабатывая файл с исходным текстом программы, выделяет помеченную документацию из комментариев и связывает с именами соответствующих классов, методов и полей. Таким образом, при минимальных усилиях создания комментариев к коду, можно получить хорошую документацию к программе.

javadoc — это генератор документации в HTML-формате из комментариев исходного кода Java и определяет стандарт для документирования классов Java. Для создания доклетов и тэглетов, которые позволяют программисту анализировать структуру Java-приложения, **javadoc** также предоставляет API. В каждом случае комментарий должен находиться перед документируемым элементом.

При написании комментариев к кодам Java используют три типа комментариев :

```
// однострочный комментарий;
/* многострочный комментарий */
/** комментирование документации */
```

С помощью утилиты **javadoc**, входящей в состав JDK, комментарий документации можно извлекать и помещать в HTML файл. Утилита **javadoc** позволяет вставлять HTML тэги и использовать специальные ярлыки (дескрипторы) документирования. HTML тэги

заголовков не используют, чтобы не нарушать стиль файла, сформированного утилитой.

Дескрипторы **javadoc**, начинающиеся со знака @, называются автономными и должны помещаться с начала строки комментария (лидирующий символ * игнорируется). Дескрипторы, начинающиеся с фигурной скобки, например **{@code}**, называются встроенными и могут применяться внутри описания.

Комментарии документации применяют для документирования классов, интерфейсов, полей (переменных), конструкторов и методов. В каждом случае комментарий должен находиться перед документируемым элементом.

javadoc дескрипторы : @author, @version, @since, @see, @param, @return

Дескриптор	Применение	Описание
@author	Класс, интерфейс	Автор
@version	Класс, интерфейс	Версия. Не более одного дескриптора на класс
@since	Класс, интерфейс, поле, метод	Указывает, с какой версии доступно
@see	Класс, интерфейс, поле, метод	Ссылка на другое место в документации
@param	Метод	Входной параметр метода
@return	Метод	Описание возвращаемого значения
@exception имя_класса описание	Метод	Описание исключения, которое может быть послано из метода
@throws имя_класса описание	Метод	Описание исключения, которое может быть послано из метода
@deprecated	Класс, интерфейс, поле, метод	Описание устаревших блоков кода
{@link reference}	Класс, интерфейс, поле, метод	Ссылка
{@value}	Статичное поле	Описание значения переменной

Форма документирования кода

Документирование класса, метода или переменной начинается с комбинации символов /** , после которого следует тело комментариев; заканчивается комбинацией символов */.

В тело комментариев можно вставлять различные дескрипторы. Каждый дескриптор, начинающийся с символа '@' должен стоять первым в строке. Несколько дескрипторов одного и того же типа необходимо группировать вместе. Встроенные дескрипторы (начинаются с фигурной скобки) можно помещать внутри любого описания.

```
/**
 * Класс продукции со свойствами <b>maker</b> и <b>price</b>.
 * @autor Киса Воробьянинов
 * @version 2.1
*/
class Product
{
```

```
    /** Поле производитель */
    private String maker;

    /** Поле цена */
    public double price;

    /**
     * Конструктор - создание нового объекта
     * @see Product#Product(String, double)
     */
    Product()
    {
        setMaker("");
        price=0;
    }

    /**
     * Конструктор - создание нового объекта с определенными значениями
     * @param maker - производитель
     * @param price - цена
     * @see Product#Product()
     */
    Product(String maker,double price){
        this.setMaker(maker);
        this.price=price;
    }

    /**
     * Функция получения значения поля {@link Product#maker}
     * @return возвращает название производителя
     */
    public String getMaker() {
        return maker;
    }

   /**
     * Процедура определения производителя {@link Product#maker}
     * @param maker - производитель
     */
    public void setMaker(String maker) {
        this.maker = maker;
    }
}
```

Для документирования кода можно использовать HTML теги. При использовании ссылочных дескрипторов **@see** и **@link** нужно сначала указать имя класса и после символа "#" его метод или поле.

Утилита **javadoc** в качестве входных данных принимает файл с исходным кодом программы, для которого генерируется HTML файл. Документация для каждого класса содержится в отдельном HTML файле. Кроме этого, создается дерево индексов и иерархии. Могут быть сгенерированы и другие HTML файлы.

Задание.

1. Прокомментировать программу из предыдущей работы.
2. С помощью Jvadoc создать HTML-файл, содержащий исходный код программы.