ЛАБОРАТОРНАЯ РАБОТА «ДОКУМЕНТИРОВАНИЕ КОДА НА ЯЗЫКЕ JAVA С ИСПОЛЬЗОВАНИЕМ УТИЛИТЫ JAVADOC»

Цель работы: Изучить технологию документирования исходных кодов программ с помощью Javadoc.

Теоретический материал

Наибольшая проблема, связанная с документированием кода – поддержка этой документации. Если документация и код разделены, возникают трудности, связанные с необходимостью внесения изменений в соответствующие разделы сопроводительной документации всякий раз при изменении программного кода. Среда разработки предлагает решение – связать код с документацией, поместив всё в один файл.

Java комментарии необходимы для комментирования программы и для составления или оформления документации. Существует специальный синтаксис для оформления документации в виде комментариев и инструмент для выделения этих комментариев в удобную форму. Инструмент называется javadoc. Обрабатывая файл с исходным текстом программы, он выделяет помеченную документацию из комментариев и связывает с именами соответствующих классов или методов. Таким образом, затратив минимум усилий на оформления комментариев, можно получить хорошую документацию к программе.

На выходе javadoc получается HTML файл, который можно просмотреть любым веб-обозревателем. Этот инструмент позволяет создавать и поддерживать файлы с исходным текстом программы и, при необходимости, генерировать сопроводительную документацию. Библиотеки Java обычно документируются именно таким способом, именно поэтому при разработке программ удобно использовать JDK с комментированным для javadoc исходным текстом библиотек вместо JRE, где исходники отсутствуют.

Java имеет три типа комментариев. Первые два типа: //... и /*...*/. Третий тип называется комментарием документации. Такой комментарий начинается с последовательности символов /** и заканчивается последовательностью */. Комментарии документации позволяют добавлять в программу информацию о ней самой. С помощью утилиты javadoc (входящей в состав JDK) эту информацию 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!

Пример:

```
/**
* Creates new form GUI_application
*/
```

Средством обработки внедренных в исходный код комментариев и создания для класса справочных *HTML*-файлов является инструмент *javadoc*, входящий в состав *JDK*. Но в среде IntelliJ IDEA удобнее пользоваться вызовом через главное *меню*: **Tools/Generate Javadoc… .** Документационные комментарии бывают для:

- Пакетов (пока не функционируют).
- Классов.
- Интерфейсов.
- *Пользовательских типов*-перечислений (на уровне пакетов пока не функционируют, но можно использовать для типов, заданных в классах).
- Методов.
- Переменных.

Документационные комментарии пишутся непосредственно перед заданием соответствующей конструкции – пакета, класса, интерфейса, типа-перечисления, метода или переменной. Следует учитывать, что *по* умолчанию документация создается только для элементов, имеющих уровень видимости public или protected. **Пример фрагмента кода с документационными комментариями**:

Имеется два типа кода внутри блока документационного комментария – *HTML*-текст и *метаданные* (команды документации, начинающиеся с символа @). Если пишется обычный текст, он рассматривается как *HTML*-текст, поэтому все пробелы и переносы на новую строку при показе приводятся к одному пробелу. Для того, чтобы очередное предложение при показе начиналось с новой строки, следует вставить последовательность символов

, называющуюся тегом *HTML*. Возможно использование произвольных тегов *HTML*, а не только тега переноса на новую строку: теги неупорядоченного списка <ul> и <li>, теги гиперссылок, изображений и т.д. В то же время не рекомендуется использовать *заголовки* и фреймы, поскольку это может привести к проблемам – *javadoc* создает на основе документационного кода собственную систему заголовков и фреймов. Кроме того, при преобразовании в *HTML*-документ из документационного кода удаляются символы "*", если они стоят на первом значимом месте в строке (символы пробелов не являются значимыми).

Для более подробного изучения тегов *HTML* следует читать справочную или учебную литературу *по* этому языку разметки документов. Соответствующие ссылки и документы можно найти, например, на сайте http://barsic.spbu.ru/ www/comlan/html_r.html.

Команды документации (символы метаданных):

- @see ("смотри") - применяется для создания в документе гиперссылок на другие комментарии. Можно использовать для любых конструкций (классов, методов и т.д.). Формат использования: @see ИмяКласса - для класса; @see ИмяКласса.ИмяПеречисления- для типа-перечисления, заданного в классе; @see ИмяКласса#ИмяЧлена - для метода или переменной; для интерфейса - аналогично классу. При этом имя класса или интерфейса может быть либо коротким, либо квалифицировано именем пакета.
- @version ("версия") - информация о версии. Используется для классов и интерфейсов. Формат использования: @version Информация о версии в произвольной форме.
- @author ("автор") - Информация об авторе. Используется для классов и интерфейсов. Формат использования: @author Информация об авторе в произвольной форме. Может включать не только имя, но и данные об авторских правах, а также об электронной почте автора, его сайте и т.д.
- @since ("начиная с") - Информация о версии *JDK*, начиная с которой введен или работоспособен класс или интерфейс. Формат использования: @since Информация в произвольной форме.
- @param (сокращение от parameter -"параметр") - информация о параметре метода. Комментарий /** @param … */ ставится в месте декларации метода в списке параметров перед соответствующим параметром. Формат использования: @param ИмяПараметра Описание.
- @return ("возвращает") - информация о возвращаемом методом значении и его типе. Формат использования: @return Информация в произвольной форме.

- @throws ("возбуждает исключение") - информация об исключительных ситуациях, которые могут возбуждаться методом. Формат использования: @throws ИмяКлассаИсключения Описание.
- *@deprecated* ("устаревшее") - информация о том, что данный метод устарел и в последующих версиях будет ликвидирован. При попытке использования таких методов компилятор выдает программисту предупреждение (*warning*) о том, что метод устарел, хотя и компилирует проект. Формат использования: *@deprecated* Информация в произвольной форме.

Признаком окончания команды документации является начало новой команды или окончание комментария.

Пример. Требуется описать и задокументировать класс для работы с объектом «Автомобиль».

Описание класса выглядит следующим образом:

```
/**
* Класс Автомобиль - базовый класс для объектов транспорта
* @author Слива М.В.
*/
public class Auto {
/**Поле для хранения названия фирмы автомобиля */
private String firm;
/**Поле для хранения максимальной скорости автомобиля */
private int maxSpeed;
/**
* Устанавливает значение поля {@link Auto#firm}
* @param firma - название фирмы автомобиля */
public void setFirm(String firma){
firm=firma;
}
/**
* Устанавливает значение поля {@link Auto#maxSpeed}
* @param speed - значение максимальной скорости автомобиля */
public void setMaxSpeed(int speed){
maxSpeed=speed;
}
/**
* Возвращает значение поля {@link Auto#maxSpeed}
*@return целое значение максимальной скорости автомобиля */
public int getMaxSpeed(){
return maxSpeed;
}
/**
* Возвращает значение поля {@link Auto#firm}
*@return строку с названием фирмы автомобиля */
public String getFirm(){
return firm;
```

```
}
/**
* Создает автомобиль с фирмой "Без названия" и максимальной скоростью,
равной 0*/
public Auto(){
firm="Без названия";
maxSpeed=0;
}
/**
* Создает автомобиль с заданными значениями фирмы и максимальной скорости
* @param firma - название фирмы автомобиля
* @param speed - значение максимальной скорости автомобиля*/
public Auto(String firma, int speed){
firm=firma;
maxSpeed=speed;
}
}
```

Созданные описания можно предварительно посмотреть (рис. 1), нажав сочетание клавиш CTRL+Q в среде IntelliJ IDEA или выбрав пункт меню View / Quick documentation. Для этого нужно установить курсор на названии описанного метода, поля или класса.

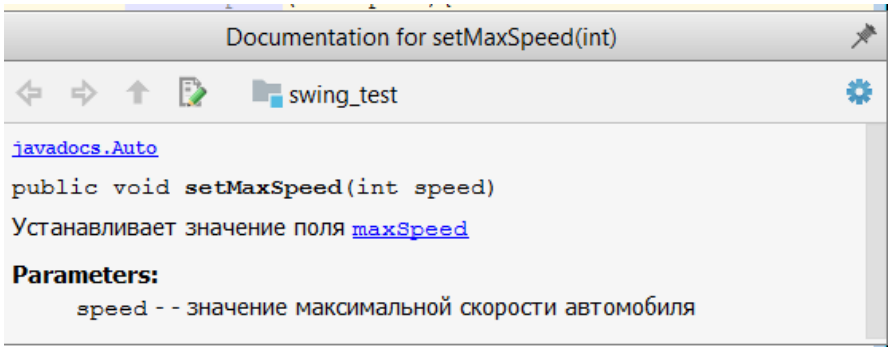

Рис.1. Предварительный просмотр созданной документации

Чтобы сгенерировать документацию по классу в виде html-страниц, нужно выбрать команды меню Tools / Generate Javadoc… в среде IntelliJ IDEA. Появится диалоговое окно (рис. 2), в котором нужно выбрать область генерации документации (весь проект, конкретный файл и др.) и путь для сохранения документации. Для того, чтобы информация по закрытым членам класса также отображалась в документации, нужно установить движок на значении private. Примерное содержание каталога с документацией представлено на рис.3. Для просмотра созданной документации в браузере запустите файл index.html (рис.4).

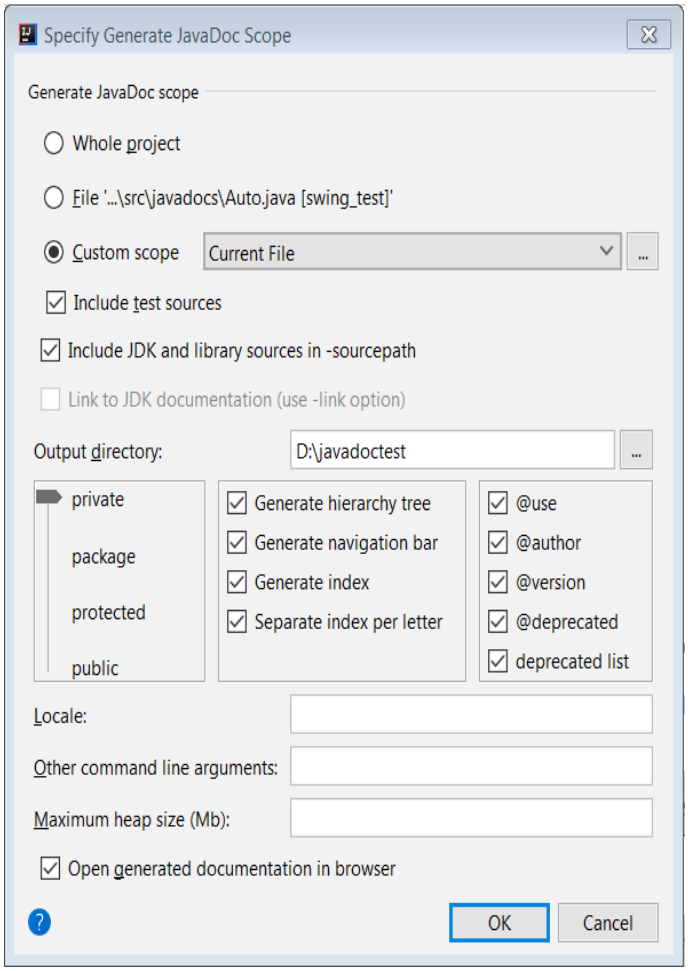

Рис. 2. Диалоговое окно Tools\Generate Javadoc…

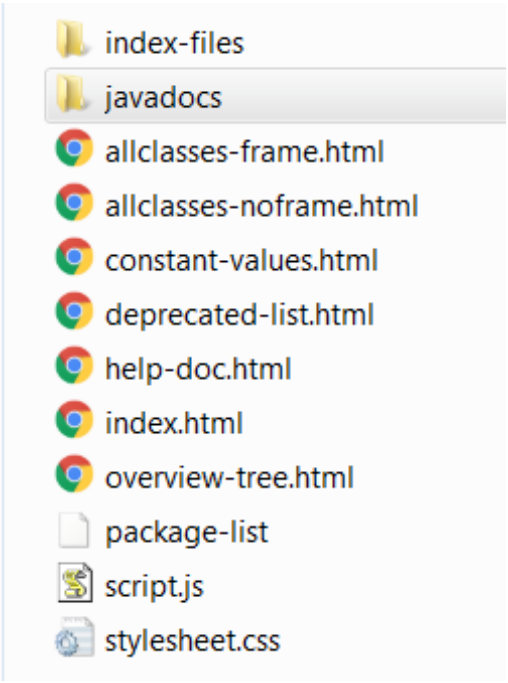

Рис.3. Каталог с созданной документацией

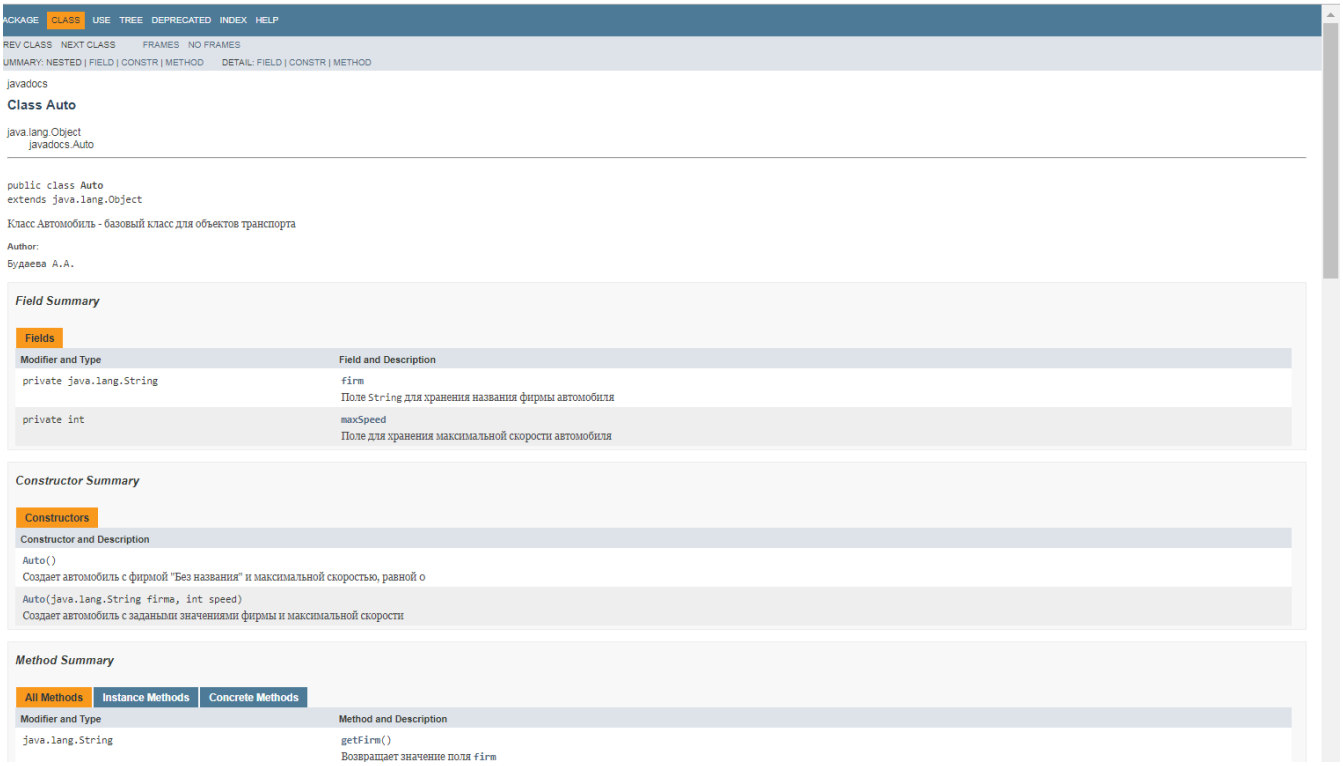

Рис. 4. Документация в сгенерированном файле

Задание на лабораторную работу

Требуется создать документацию по всем созданным в предыдущей лабораторной работе классам и интерфейсам.

Контрольные вопросы

1. Назначение утилиты Javadoc
2. Правила оформления документации
3. Команды документации и их назначение
4. Правила создания и просмотра документации в среде IntelliJ IDEA