



**Microsoft** Partner  
Silver Learning



# Профессиональное программирование на языке Java

Объектно-ориентированное программирование.  
Документирование программ



ITVVDN  
IT VIDEO DEVELOPERS NETWORK

# Профессиональное программирование на языке Java

Автор курса

Александр Бабич




**Microsoft**  
CERTIFIED

Trainer



MCT  
MCLC  
MCITP  
MCPD  
OCUP Advanced  
JAVA 8 programmer



  
более 20 лет  
преподавания

Intel, INTSPEI,  
Incom,  
Retratech,  
RUSSEE...



2 книги  
& более 50  
публикаций



  
Лауреат  
премии им  
Макаренко

[ProductivityBlog.com.ua](http://ProductivityBlog.com.ua)

## Тема

# Объектно-ориентированное программирование. Документирование программ

## Объектно-ориентированное программирование. Документирование программ

1. Абстракция, инкапсуляция, и пакетирование
2. Повторное использование Java-кода. Понятия класса, атрибутов, методов, конструкторов, пакетов
3. Использование модификаторов доступа (private и public)
4. Использование онлайн-документации по Java API
5. JavaDoc-комментарии. Генерация документации в Netbeans

## Абстракция, инкапсуляция, и пакетирование

# Профессиональное программирование на языке Java

## Вместо предисловия

Toolkits / Frameworks / Объектные APIs (1990-е и дальше)							
Java2 SDK	AWT/J.F.C/Swing		Jini	JavaBeans		JDBC	
Объектно-ориентированные языки							
SELF	Smalltalk	Common Lisp Object System		Eiffel	Java		
Библиотеки / Функциональные APIs (1960-е – ранние 1980-е)							
Nastran		TCP/IP		ISAM	X-Windows	OpenLook	
Высокоуровневые языки				Операционные системы			
Fortran	LISP	C	COBOL	OS/360	UNIX	MacOS	Microsoft Windows
Машинный код (поздние 1940е и дальше)							

# Профессиональное программирование на языке Java

## Вместо предисловия



## Вместо предисловия



**Анализ** – что должна делать система моделирование **реального мира**: действующие лица, задачи, объекты и поведение

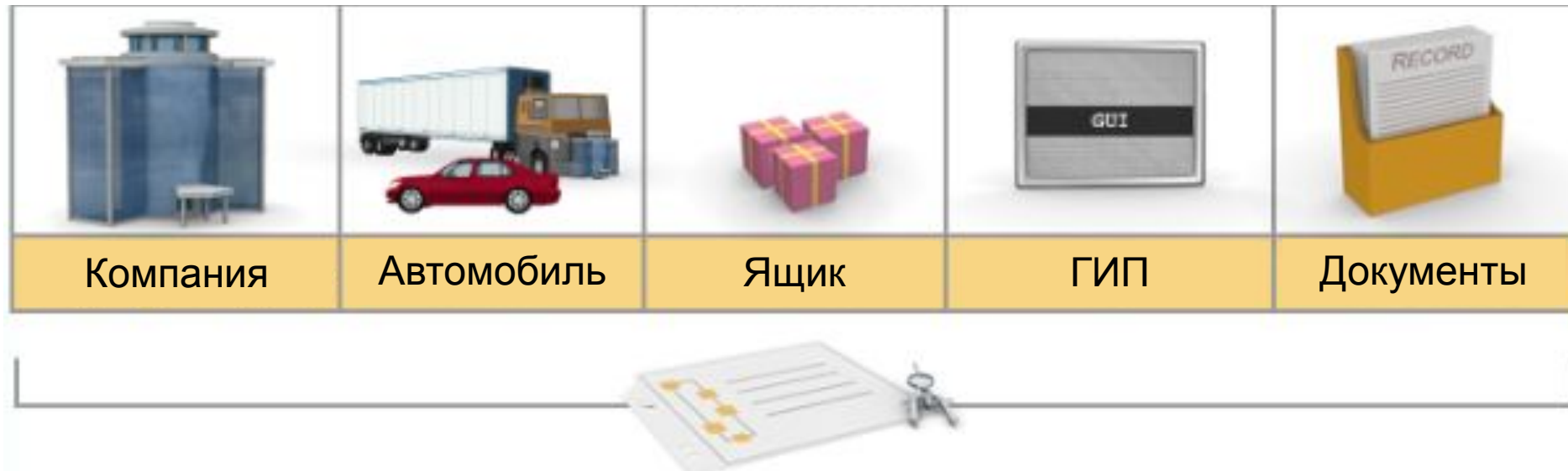


**Проектирование** описывает как система это делает моделирование **связей и взаимодействия** между объектами и действующими лицами в рамках системы **поиск подходящей абстракции** для упрощения проблемы или ее решения



# Профессиональное программирование на языке Java

## Пример анализа и проектирования



**Объекты** – компания, автомобили двух видов, ящики...

**Функциональные объекты** – экранные формы (ГИП),  
накладные и отчеты

## Упрощение и абстрагирование

Прячем  
детали  
внутренней  
реализации

Используем  
подпрограммы  
в языках  
высокого  
уровня



Абстракция

Создаем  
свои  
фреймворки  
и API

Группировка  
функций и  
данных  
внутри  
объектов

## Так что же такое абстракция?

**Абстракция** (лат. abstractio – отвлечение) – пренебрежение несущественными сторонами, свойствами, связями объекта (предмета или явления) с целью выделения существенных, закономерных признаков

**Абстрагирование** – обобщение, результат такого пренебрежения, выделение значимой информации и исключение из рассмотрения незначимой



## Абстракции в программировании

**Функции** – алгоритм, реализованный один раз и использующийся в разных ситуациях

**Объекты** – наборы атрибутов и операций (поведения), стандартных в рамках класса

**Фреймворки и API** – большие группы объектов, позволяющие решать сложные задачи (могут использоваться «как есть» или модифицироваться под конкретную задачу)

# Профессиональное программирование на языке Java

## Классы, как шаблоны (чертежи) для объектов

При производстве, **чертеж** – это описание (проект) внутренней

структуры и функциональности создаваемого объекта

В программировании – класс – **это описание объектов:**

**данные**, которые включает каждый объект

**поведение**, которое демонстрирует каждый объект

В JAVA классы поддерживают три принципа ООП:

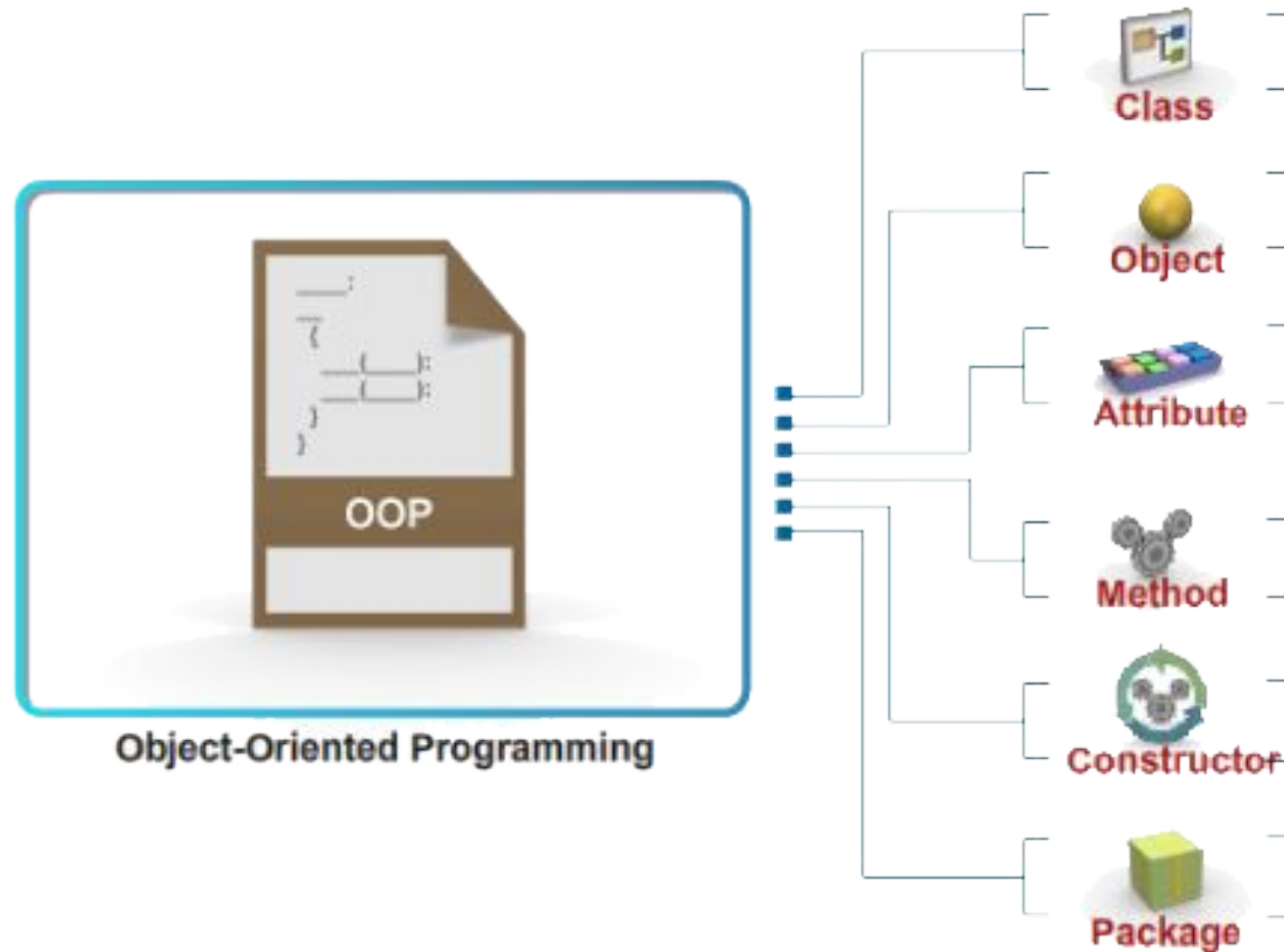
наследование

**инкапсуляция**

полиморфизм



## Абстракции в ООП



## Инкапсуляция, наследование, полиморфизм



**капсуляцией** – свойство системы, позволяющее объединить работающие с ними, в рамках класса. C++, C#, Java

Короче: **приватные переменные и публичные методы для работы с ними**

Наследование и полиморфизм мы рассмотрим чуть позже, потерпите!

## Пакетирование

**Пакеты Java** – механизм, позволяющий организовать набор Java-классов подобно **пространствам имен** в C# или **модулям** в языках Модуля, Pascal

Обычно в пакеты объединяют классы одной и той же категории, либо **предоставляющие сходную функциональность**

пакет предоставляет уникальное пространство имен для своего содержимого

допустимы вложенные пакеты

Пакеты могут содержаться в сжатом виде в JAR-файлах





## Повторное использование Java-кода. Понятия класса, атрибутов, методов, конструкторов, пакетов



# Профессиональное программирование на языке Java

## Понятие класса

Класс – **ключевое** понятие в ООП

Класс – именованное описание **совокупности объектов** с общими атрибутами, операциями, связями и семантикой

Класс описывает содержание и поведение некой совокупности **данных и действий** над этими данными

Объявление класса производится с помощью ключевого слова **class**



## Описание класса

```
<modifier>* class <class_name> {  
    <attribute_declaration>*  
    <constructor_declaration>*  
    <method_declaration>*  
}
```

```
public class Vehicle {  
    private double maxLoad;  
    public void setMaxLoad(double value) {  
        maxLoad = value;  
    }  
}
```

## Понятие атрибута

**Атрибутом** класса называется именованное свойство (переменная) класса, описывающее множество значений, которые могут принимать экземпляры этого свойства

Класс может иметь **любое число атрибутов**  
(в частности, не иметь ни одного атрибута)

Атрибут является **абстракцией** состояния объекта



## Описание атрибутов

```
<modifier>* <type> <name> [ = <initial_value>];
```

```
1 public class Foo {  
2     private int x;  
3     private float y = 10000.0F;  
4     private String name = "Bates Motel";  
5 }
```

## Понятие операции

**Операция** класса (метод) – это именованная услуга, которую можно запросить у любого объекта этого класса

Операция – это **абстракция** того, что можно делать с объектом

Класс может содержать **любое число** операций  
(в частности, не содержать ни одной операции)

Набор операций является **общим для всех объектов** данного класса



# Профессиональное программирование на языке Java

## Описание метода

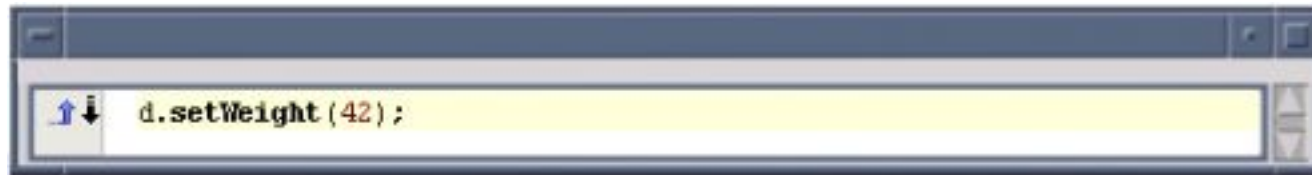
```
<modifier>* <return_type> <name> ( <argument>* ){  
<statement>*  
}
```

```
1 public class Dog {  
2     private int weight;  
3     public int getWeight() {  
4         return weight;  
5     }  
6     public void setWeight(int newWeight) {  
7         if ( newWeight > 0 ) {  
8             weight = newWeight;  
9         }  
10    }  
    }
```

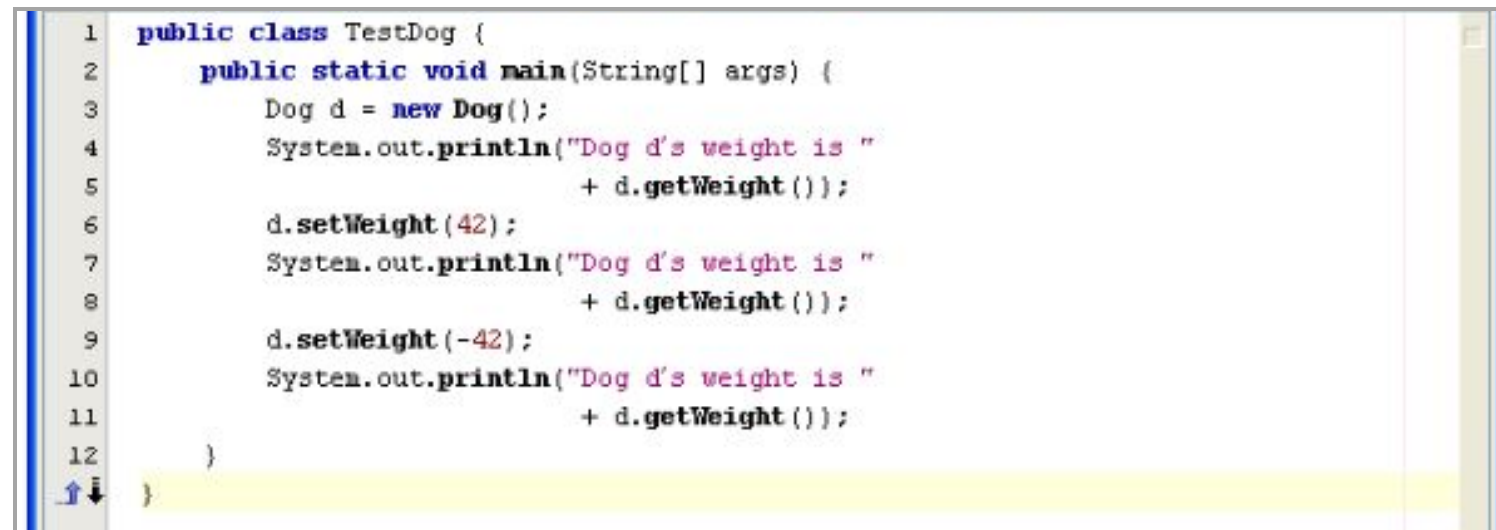


# Профессиональное программирование на языке Java

## Доступ к атрибутам объекта из метода



```
d.setWeight(42);
```



```
1 public class TestDog {  
2     public static void main(String[] args) {  
3         Dog d = new Dog();  
4         System.out.println("Dog d's weight is "  
5                             + d.getWeight());  
6         d.setWeight(42);  
7         System.out.println("Dog d's weight is "  
8                             + d.getWeight());  
9         d.setWeight(-42);  
10        System.out.println("Dog d's weight is "  
11                            + d.getWeight());  
12    }  
}
```

# Профессиональное программирование на языке Java

## Вспомним об инкапсуляции: «плохой» пример

MyDate	
+day	: int
+month	: int
+year	: int

```
1 public class MyDate {
2     public int day;
3     public int month;
4     public int year;
}
```

```
1 d.day = 32;
2 // invalid day
3
4 d.month = 2; d.day = 30;
5 // plausible but wrong
6
7 d.day = d.day + 1;
8 // no check for wrap around
```

# Профессиональное программирование на языке Java

## Вспомним об инкапсуляции: «правильный» пример

MyDate	
-day	: int
-month	: int
-year	: int
+getDay()	
+getMonth()	
+getYear()	
+setDay(int)	: boolean
+setMonth(int)	: boolean
+setYear(int)	: boolean

Проверка  
номера дня

```
1 MyDate d = new MyDate();
2
3 d.setDay(32);
4 // invalid day, returns false
5
6 d.setMonth(2); d.setDay(30);
7 // plausible but wrong, setDay returns false
8
9 d.setDay(d.getDay() + 1);
10 // this will return false if wrap around needs to occur
```

## Подробнее об инкапсуляции

MyDate	
-date	: long
+getDay()	: int
+getMonth()	: int
+getYear()	: int
+setDay(int)	: boolean
+setMonth(int)	: boolean
+setYear(int)	: boolean
-isDayValid(int)	: boolean

Позволяет **скрыть** подробности реализации класса

Заставляет использовать **интерфейс класса** для доступа к его данным

Делает код более **управляемым**

# Профессиональное программирование на языке Java

## Понятие конструктора

Конструктор – это специальный **метод**

Конструктор – это метод, который **инициализирует** новый объект после его создания (присваивает значения атрибутам)

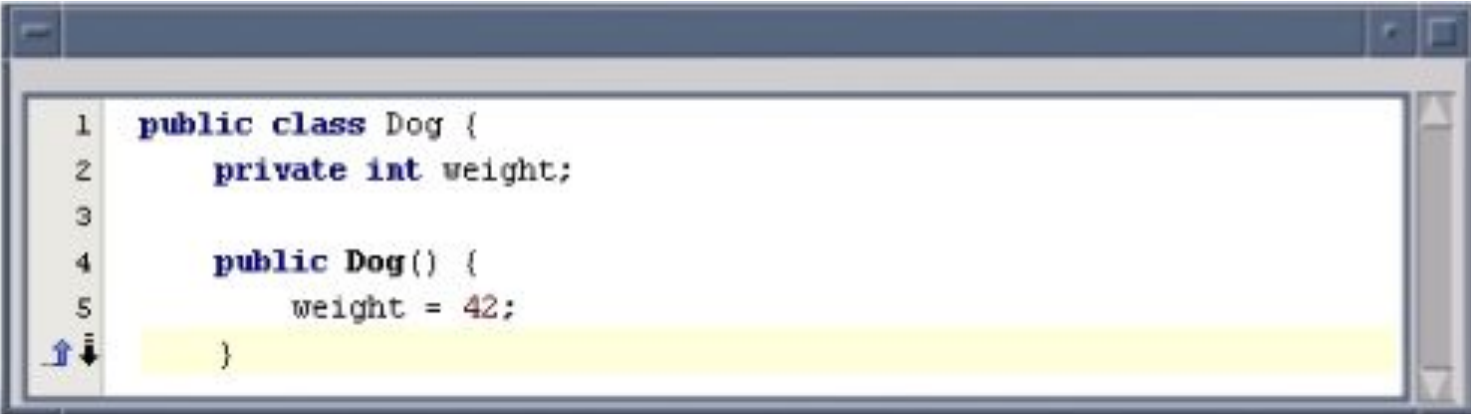
Имя конструктора всегда совпадает с **именем класса**, в котором он расположен

У конструкторов **нет типа возвращаемого результата**, даже void



## Описание конструктора

```
[<modifier>] <class_name> ( <argument>* ) {  
  <statement>*  
}
```



```
1 public class Dog {  
2     private int weight;  
3  
4     public Dog() {  
5         weight = 42;  
6     }  
}
```

# Профессиональное программирование на языке Java

## Конструктор по умолчанию

Каждый класс **всегда** имеет хотя бы один конструктор

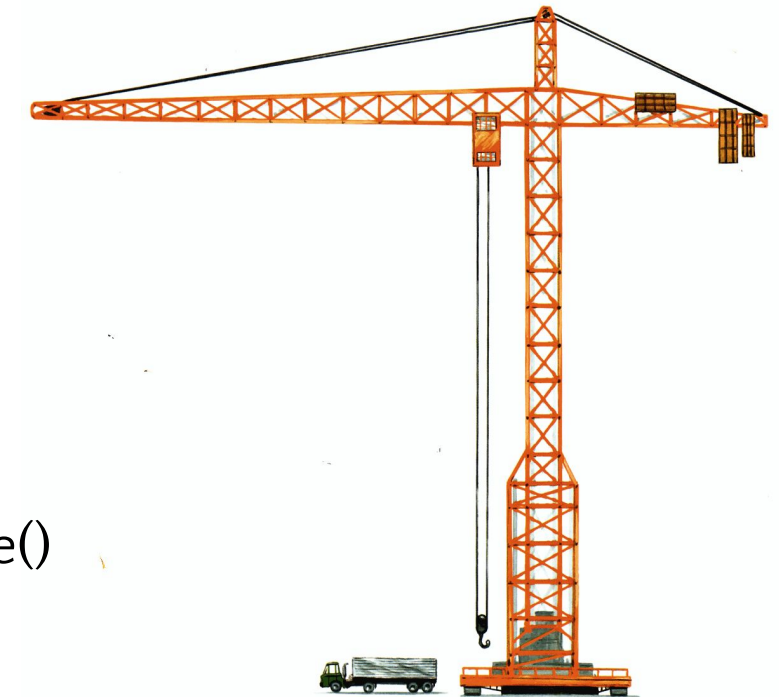
Конструктор по умолчанию создается **автоматически**

без параметров

без тела

Позволяет создавать объекты с помощью `new ClassName()`

**без необходимости писать код инициализации**



## Структура файла с исходным кодом

```
[<package_declaration>]  
<import_declaration>*  
<class_declaration>+
```

```
1  package shipping.reports;  
2  
3  import shipping.domain.*;  
4  import java.util.List;  
5  import java.io.*;  
6  
7  public class VehicleCapacityReport {  
8      private List vehicles;  
9      public void generateReport(Writer output) {  
10         // code to generate the report  
11     }  
}
```



## Структура файла с исходным кодом

Оператор импорта должен быть указан **перед** объявлениями классов

Имя файла должно **совпадать** с именем публичного класса, который в нем содержится

Классов в файле может быть несколько, **но публичный – только один!**

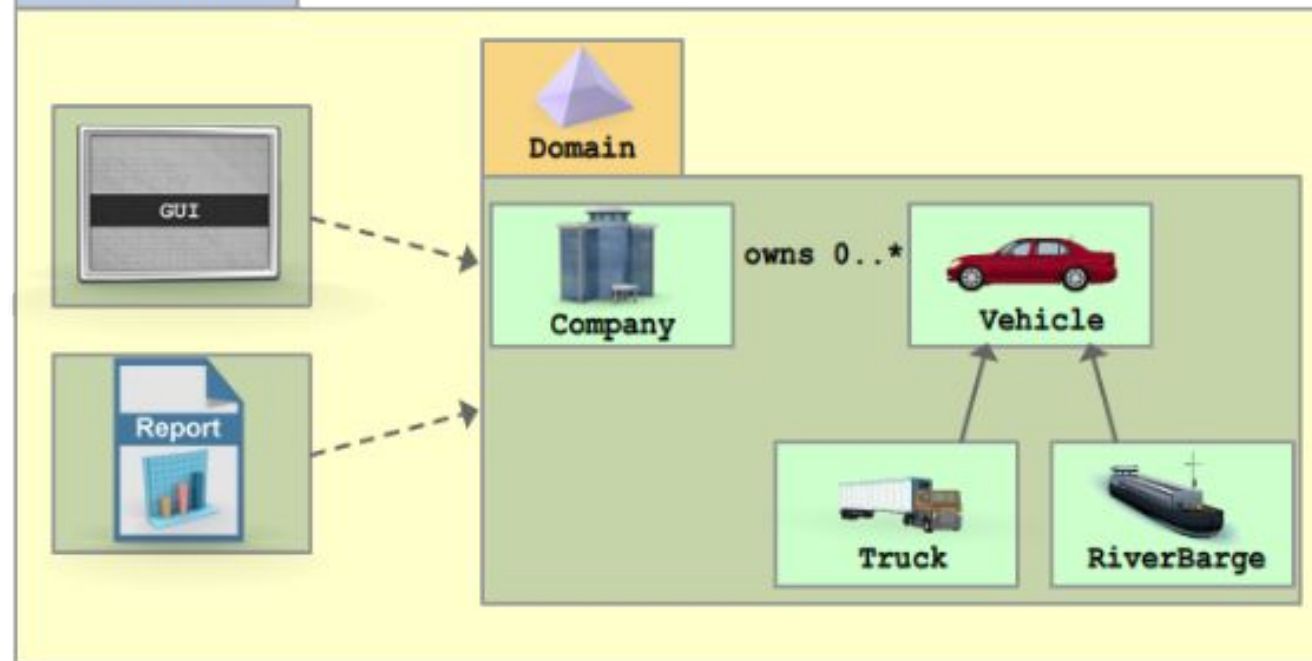
О модификаторах видимости – чуть позже...

## Программные пакеты



Делают большие системы **управляемыми**

Могут содержать классы и **подпакеты**



## Программные пакеты

```
package <top_pkg_name>[.<sub_pkg_name>]*;
```

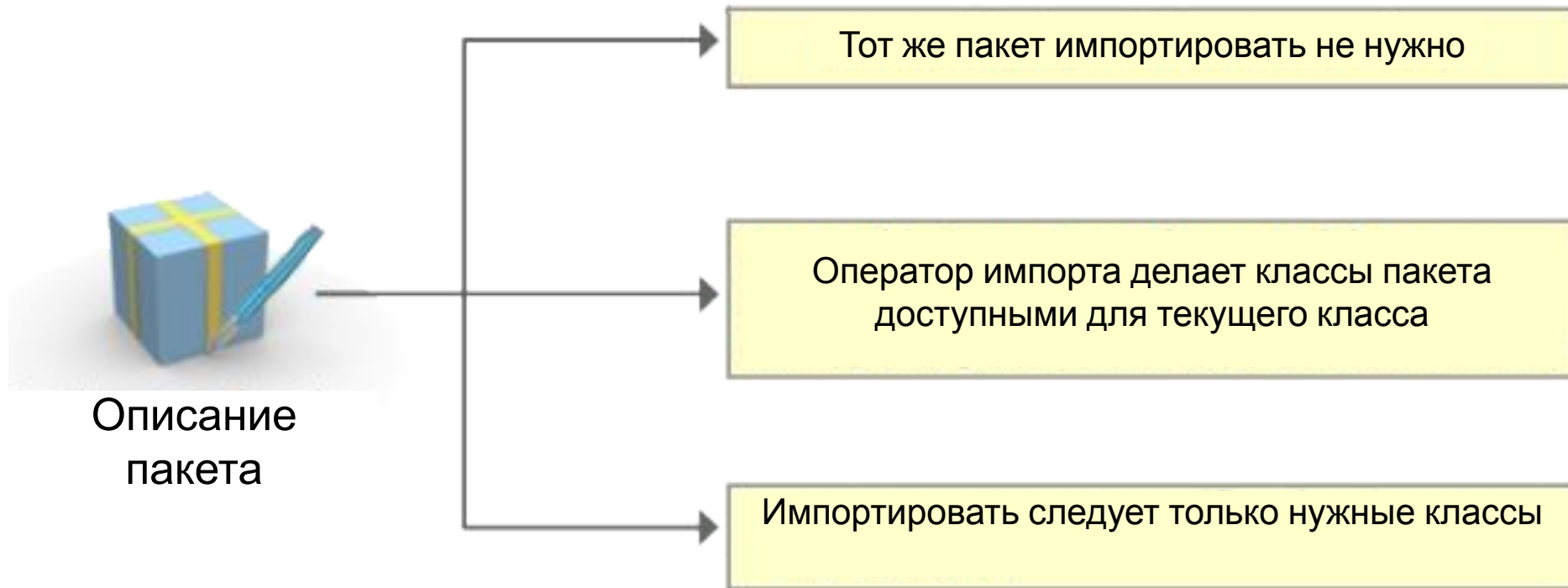
```
1  package shipping.domain;  
2  
3  // Class Vehicle of the 'domain' sub-package within  
4  // the 'shipping' application package.  
5  public class Vehicle {  
6      ...  
7  }
```

## Оператор импорта

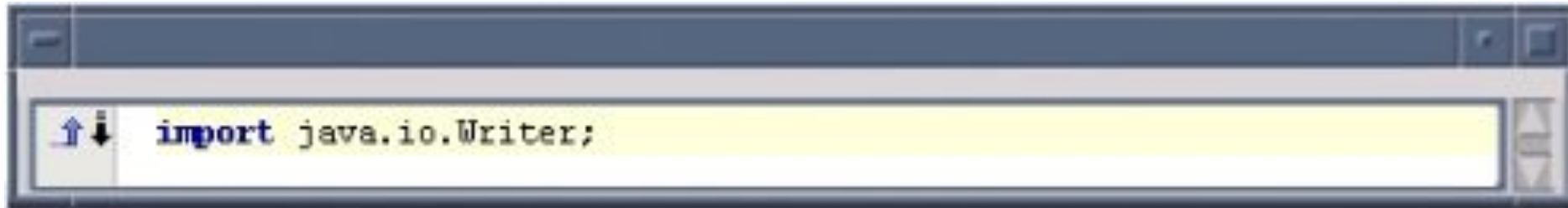
```
import <pkg_name>[.<sub_pkg_name>].<class_name>;  
or  
import <pkg_name>[.<sub_pkg_name>].*;
```

```
1  package shipping.reports;  
2  
3  import shipping.domain.*;  
4  import java.util.List;  
5  import java.io.*;  
6  
7  public class VehicleCapacityReport {  
8      private Company companyForReport;  
9      ...  
10 }
```

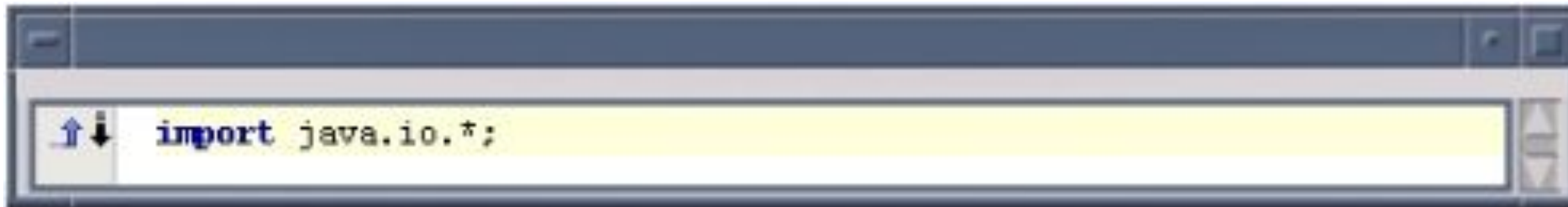
## Оператор импорта



## Оператор импорта



```
import java.io.Writer;
```

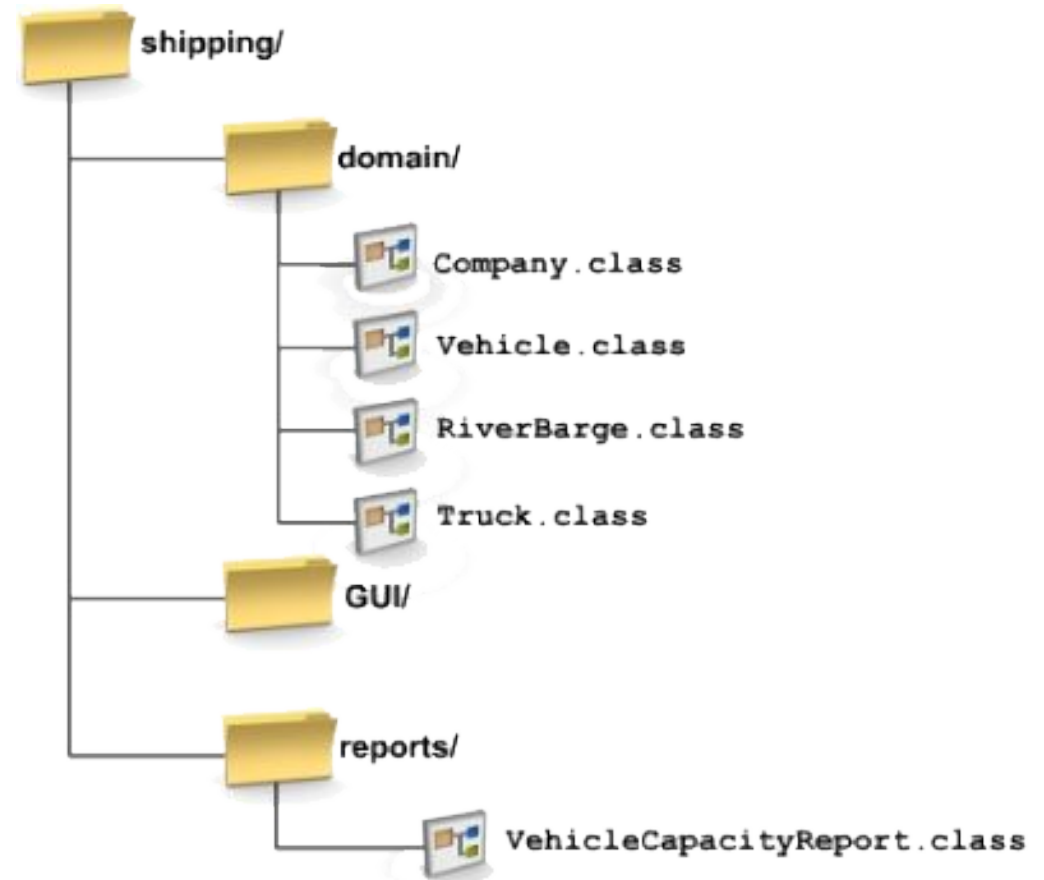


```
import java.io.*;
```

# Профессиональное программирование на языке Java

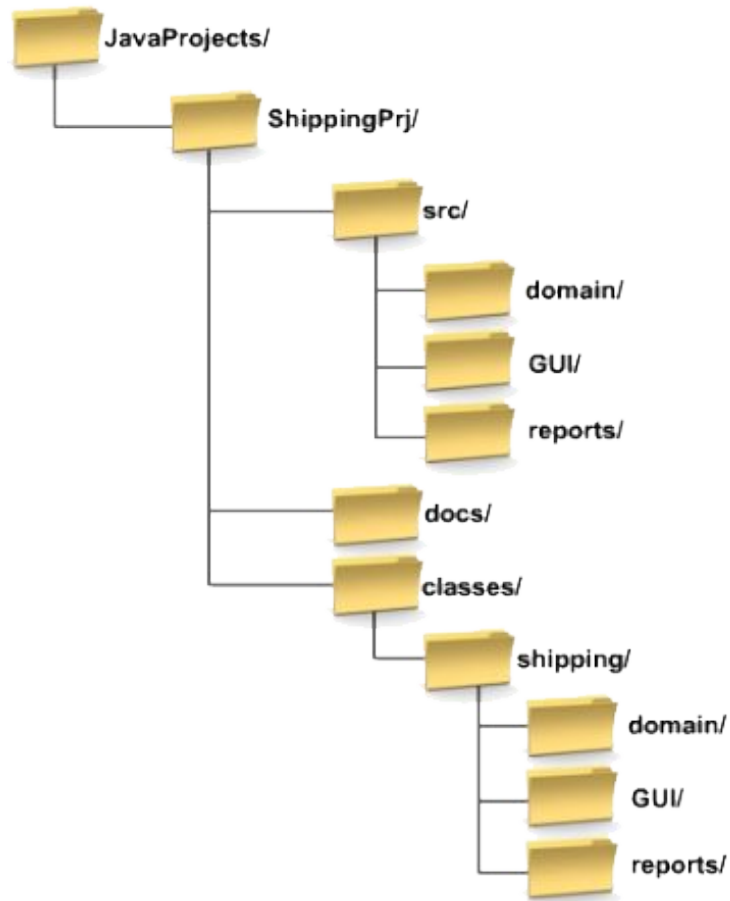
## Папки и пакеты

Файлы классов, входящих в состав пакета, хранятся в папке с именем пакетом



# Профессиональное программирование на языке Java

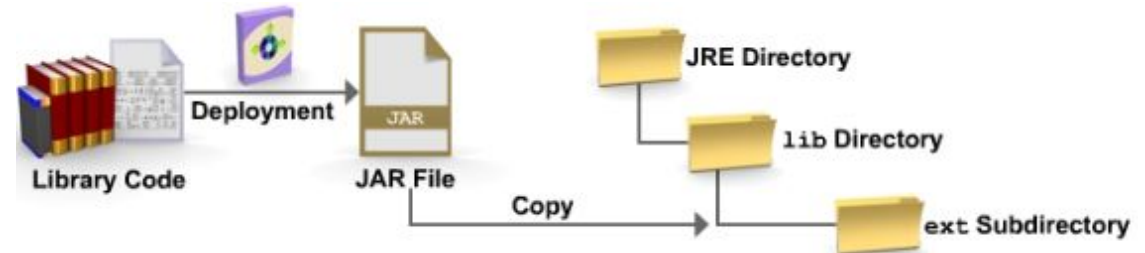
## Папки и пакеты



Откомпилированные классы можно собрать в **JAR-архив**

JAR-файл можно поместить папку **ext** внутри папки **lib** в главной папке JRE

Имена ваших классов не должны конфликтовать с классами JDK!





## Повторим и закрепим терминологию



**Класс** – код-шаблон для создания объектов

**Объект** – экземпляр класса

**Метод (операция)** – поведенческий элемент объекта

**Атрибут** – элемент, описывающий состояние объекта

**Конструктор** – метод для инициализации нового объекта

**Пакет** – группа классов и/или подпакетов

# Использование модификаторов доступа

# Профессиональное программирование на языке Java

## Повторяем: модификаторы видимости

**private** – члены класса доступны только внутри класса

**default** (package-private) (по-умолчанию) – члены класса видны только внутри пакета

**protected** – члены класса доступны внутри пакета и в его наследниках

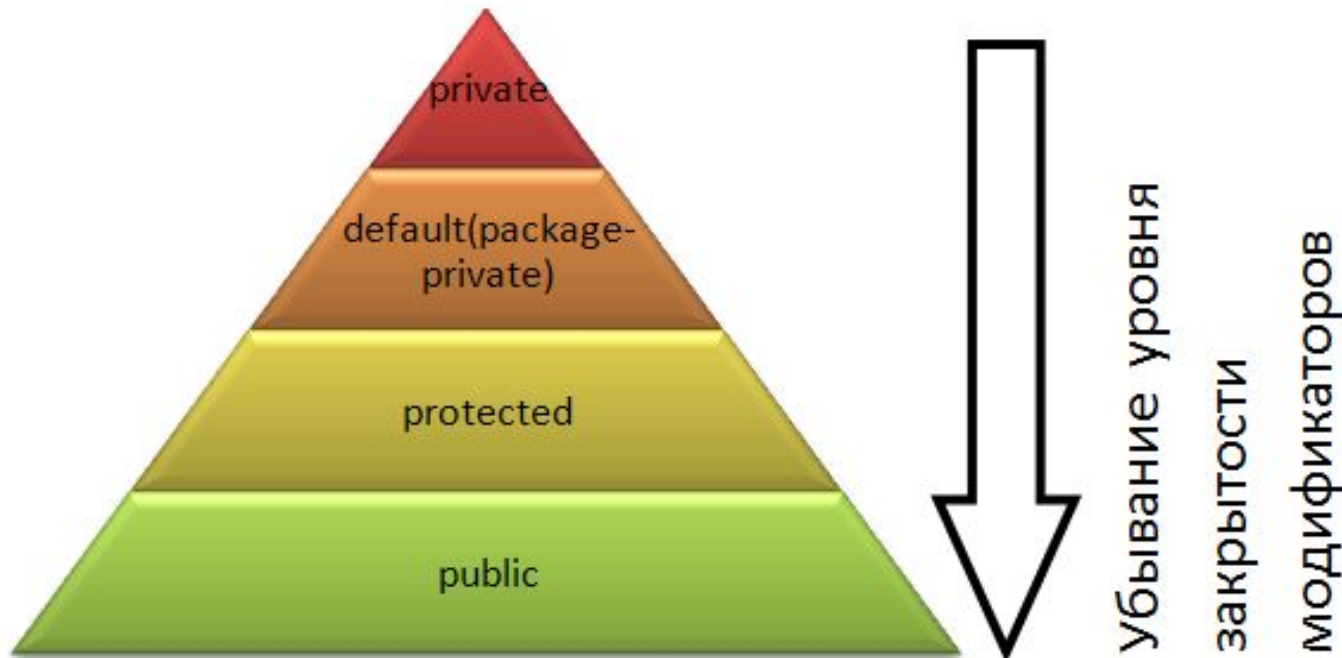
**public** – члены класса доступны всем



# Профессиональное программирование на языке Java

## Пирамида модификаторов видимости

По убыванию уровня закрытости: private, default (package-private), protected, public



Во время наследования  
возможно изменения  
модификаторов доступа в  
сторону большей видимости!

## Подытожим

	Класс	Внутренний класс	Переменная	Метод	Конструктор	Логический блок
public	Да	Да*	Да	Да	Да	Нет
protected	Нет	Да	Да	Да	Да	Нет
default	Да	Да	Да**	Да	Да	Да
private	Нет	Да	Да	Да	Да	Нет

\* кроме локальных и анонимных классов

\*\* и для локальной переменной

## Использование онлайн- документации по Java API

# Профессиональное программирование на языке Java

## Онлайновая документация

Большой набор **HTML- файлов** с подробной информацией о классах API

иерархия наследования

описание

члены класса и т.п.



**Быстрый вызов из Netbeans, поиск по документации**

<https://docs.oracle.com/javase/8/docs/api/>





## Использование онлайновой документации



# Профессиональное программирование на языке Java

## Выбор нужной документации

Раздел Java SE APIs & Documentation

примеры кода

обучающие материалы

статьи и т.п.

Можно выбрать версию Java

Документация по другим API, включенным в JDK

The screenshot shows the Oracle Java SE APIs & Documentation page. The page has a navigation bar with tabs for Overview, Downloads, Documentation, Community, Technologies, and Training. The main content area is titled 'Java SE APIs & Documentation' and includes a table of 'Core API Docs' and 'JDK Programmer Guides'. The table lists various Java versions and their corresponding documentation links. A sidebar on the right lists 'Java SDKs and Tools' and 'Java Resources'.

<http://www.oracle.com/technetwork/java/javase/documentation/api-jsp-136079.htm>



[ml](#)

## JavaDoc-комментарии. Генерация документации в Netbeans

## Документирование кода

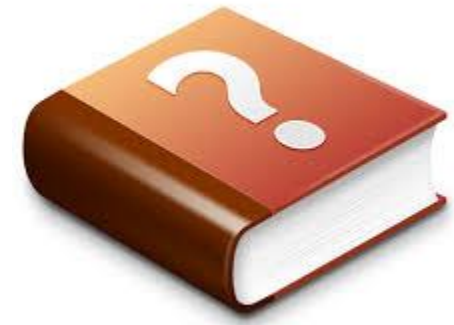
Нужно поддерживать и код, и документацию, так почему бы их **не совместить?**

простота

единый стиль

структура

навигация



Хорошо бы **автоматизировать** создание документации!

## Виды комментариев

//однострочный комментарий

/\* Это обычный комментарий, который распространяется на несколько строк \*/

/\*\* Это комментарий, понятный javadoc\*/

## Что такое javadoc?

**Консольная утилита**, входящая в состав JDK, которая формирует стандартную документацию на основе исходных кодов

```
javadoc [options] [packagenames] [sourcefiles] [@files]
```

GUI для javadoc:

JDocEditor для Eclipse

DocFlex/Javadoc

**Netbeans IDE**

## Начнем с простого

```
/** Компонент - класс */
public class DocumentPrinter {
    /** Компонент - переменная */
    public int id;
    //в документацию по умолчанию включаются
    //только public и protected члены
    private String excludeFromSpec;
    /** Компонент - метод */
    public void print() {}
}
}
```

## HTML-теги в javadoc-комментариях

```
/**  
 *<pre>  
 *System.out.println(newDate());  
 *</pre>  
 */
```

```
/**  
 Вы можете вставить список:  
 * <ol>  
 * <li> Первый элемент  
 * <li> Второй элемент  
 * <li> Третий элемент  
 * </ol>  
 */
```

## Специальные теги javadoc

`@see` – ссылка на другой класс, метод или поле.

Например:

```
@see Window
```

```
@see java.awt.Window
```

```
@see java.awt.Window#isActive
```



## Специальные теги javadoc для класса

`@version` – указание версии класса.

`@author` – автор класса.

`@since` – с какой версии продукта или библиотеки появился класс.

## Специальные теги javadoc для атрибутов

```
public class DocumentPrinter {  
    /** Идентификатор класса.  
     * @see #print()  
     */  
    public int id;  
    public void print() {}  
}
```

## Специальные теги javadoc для операций

```
/** Посылает документ на печатающее устройство. Возвращает значение
{@code true} если документ успешно отослан.
 * @see Printer#getDefaultPrinter()
 * @param document - документ, предназначенные для печати.
 * @return {@code true} если документ успешно отослан;
 *         {@code false} в противном случае.
 * @throws IOException при возникновении ошибки ввода-вывода.
 * @throws PrinterException при возникновении ошибки
печатающего
 * устройства.
 * @deprecated рекомендуется использовать метод
 * {@code print(Document, Printer)}.
 */
public boolean print(Document document) {...}
```

# Профессиональное программирование на языке Java

## Специальные теги javadoc для параметров методов

```
/**  
 * @param studID – уникальный идентификатор студента  
 * @param discID – уникальный идентификатор  
 ДИСЦИПЛИНЫ  
 */  
public boolean isLearned(int studID, int discID){  
    ...  
}
```

# Профессиональное программирование на языке Java

## Специальные теги javadoc для результатов методов

```
/**
 * @return - {@code true} если дисциплина была изучена
 студентом;
 *           {@code false} в противном случае.
 */
public boolean isLearned(int studID, int discID){
    ...
}
```

## Специальные теги javadoc для исключений

```
/**
 * @throws ArgumentException при некорректных входных
 * параметрах (идентификатор меньше или равен нулю).
 */
public boolean isLearned(int studID, int discID) {
    ...
}
```



## Законченный пример

```
/**
 * Простой класс для демонстрации
 * @author dav
 */
public class SimpleClass {
    /**Текущее значение экспериментальной переменной */
    int value = 0;
    /**
     * Метод, демонстрирующий особенности
     * приведения примитивных типов
     */
    public void simpleMethod() {...}
}
```

## Гиперссылки

`{@link ClassName}`

преобразуется в гиперссылку на класс ClassName

`{@link ClassName#methodA()}`

преобразуется в гиперссылку на метод класса





# Профессиональное программирование на языке Java

## Что в остатке?

Доступность, простота, **стимул комментировать код** 😊

Создаем документацию не покидая исходный код

Имеем:

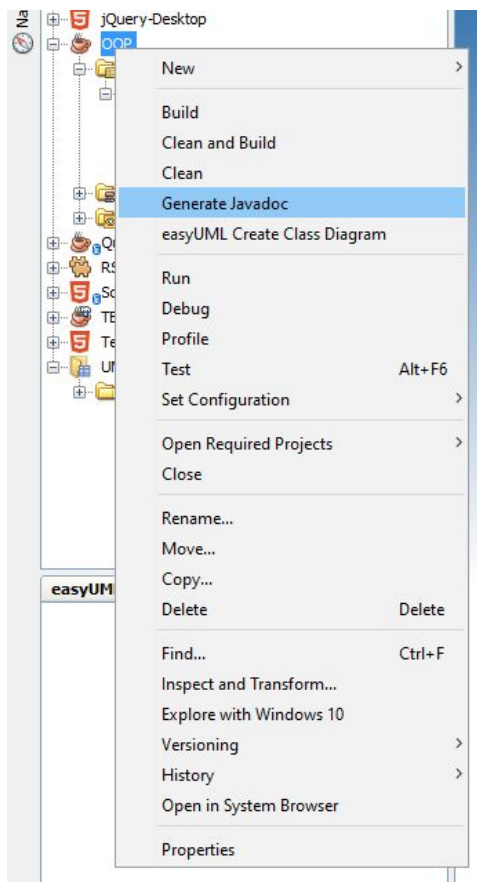
**стандартное** оформление  
удобную навигацию

The screenshot shows the Java Platform Standard Edition 8 API Specification website. The left sidebar lists various packages and classes. The main content area displays the 'Overview' page for the API Specification, including a table of packages and their descriptions.

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing beans - components based on the JavaBeans™ architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.

# Профессиональное программирование на языке Java

## Генерация документации в Netbeans



Простой вызов javadoc прямо из контекстного меню проекта

Результат помещается в папку `dist > javadoc` внутри папки проекта





## Генерация документации в Netbeans

# Лабораторная работа

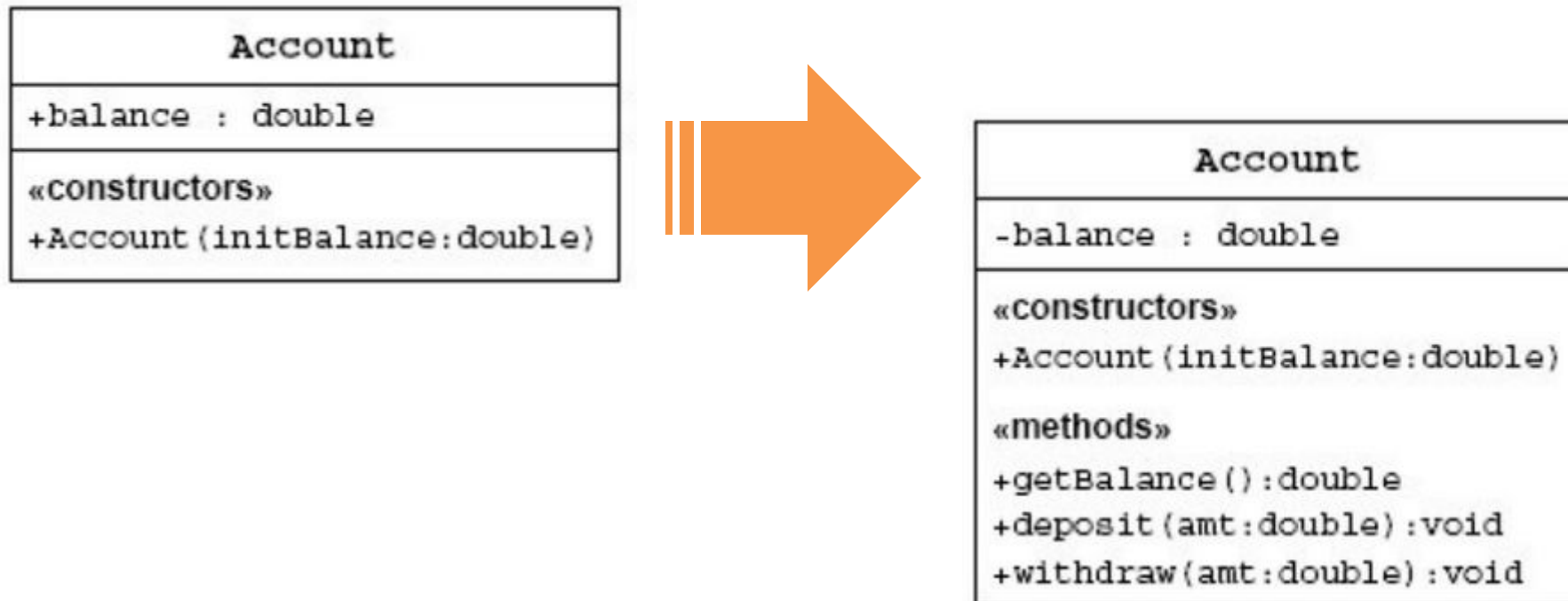
# Профессиональное программирование на языке Java

## ЛР: Создание простых программ в IDE Netbeans

1. Использование онлайн-документации
2. Исследование инкапсуляции
3. Использование пакетов для организации классов

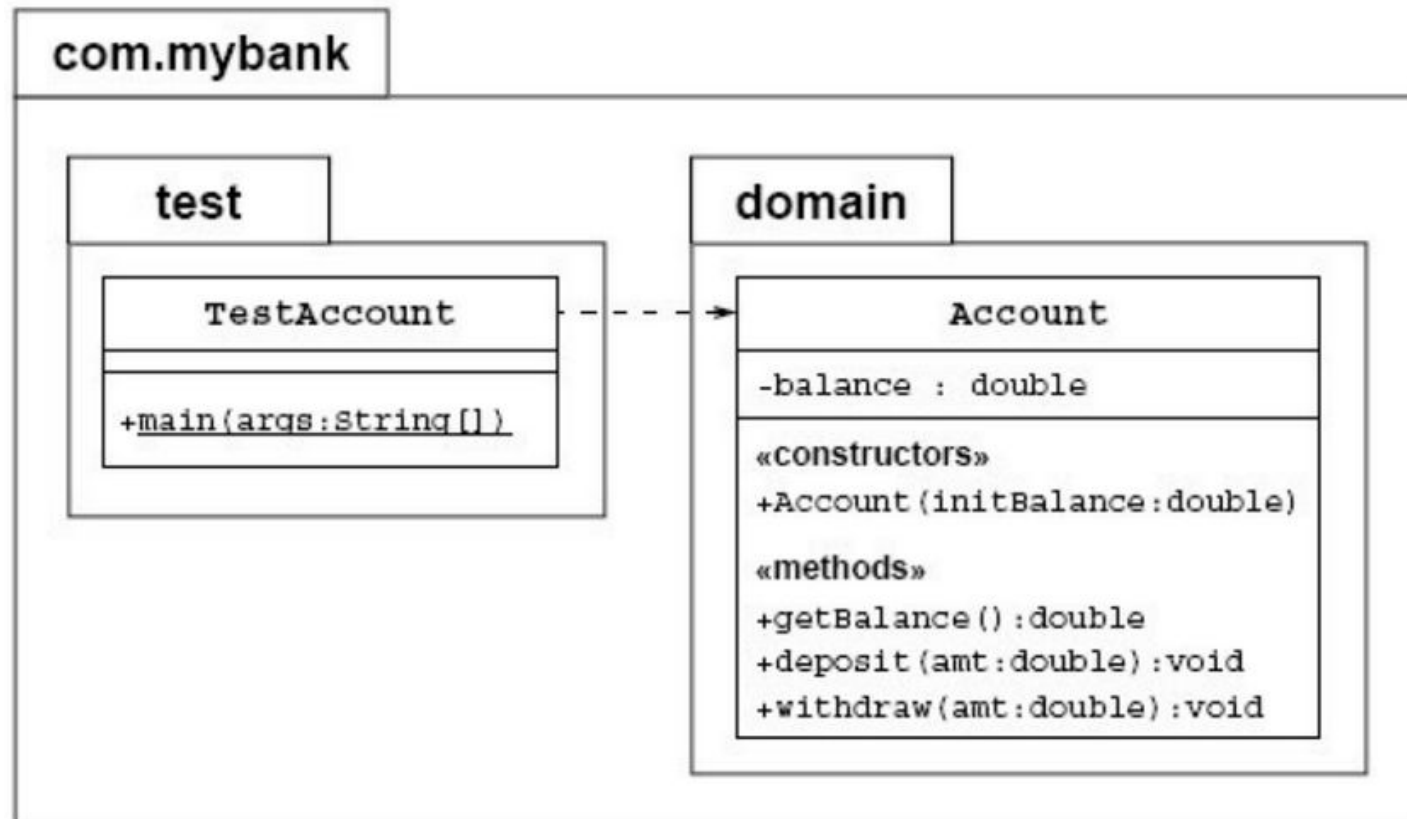


## ЛР: Создание простых программ в IDE Netbeans



# Профессиональное программирование на языке Java

## ЛР: Создание простых программ в IDE Netbeans



## Подведение итогов

- Обсуждение лабораторной работы
- О чем мы узнали в этом уроке
- Вопросы для размышлений
- Рекомендации





## Вопросы для самоконтроля

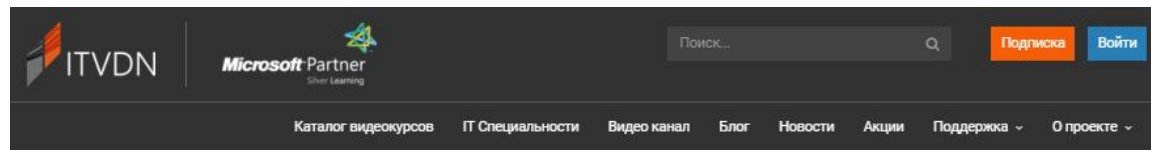
1. Что такое абстракция?
2. Как в Java обеспечивается повторное использование кода?
3. Что такое инкапсуляция?
4. Что такое класс, атрибут, операция, конструктор, пакет?
5. Какие модификаторы видимости есть в Java?
6. Что такое javadoc?
7. Правда ли, что в документацию автоматически попадают все комментарии в коде?

Следующий урок:

Проектирование иерархии классов. Знакомство с UML



# Смотрите наши уроки в видео формате ITVDN.com



**ITVDN 2015. Наши награды**

**ITVDN 2015. Итоги года**

В марте 2015 года ITVDN стал победителем конкурса IT Education Awards, который проходил в рамках IT Jam 2015 и награжден как лучший образовательный ресурс в номинации Online Education. Экспертное жюри, в состав которого вошли представители ведущих IT компаний, отметили такие преимущества ITVDN, как системный подход в обучении, позволяющий удаленно получить качественное образование по наиболее популярным специальностям, высокий профессионализм авторов видео курсов и использование современных методик оценки знаний.

## Новые видео

Исключения	0
Итераторы и генераторы	0

## Популярные видео курсы

Видео курс C# Стартовый (для начинающих)	9 уроков (16 ч. 3 мин.)
Видео курс по шаблону проектирования	20 уроков (16 ч. 7 мин.)

## Теги

.NET Developer
Frontend Developer

Посмотрите этот урок в видео формате на образовательном портале [ITVDN.com](http://ITVDN.com) для закрепления пройденного материала.

Все курсы записаны сертифицированными тренерами, которые работают в учебном центре CyberBionic Systematics



# Проверка знаний

## TestProvider.com

TestProvider | Мы поможем людям оценить себя

Регистрация Войти

Главная Каталог Сертификация Microsoft Поддержка О нас

Тестирование

Языки программирования и информационные технологии

Microsoft

C# ASP.NET MVC JavaScript Patterns Of Design SQL Architecture Guide WCF HTML&CSS XML SEO WPF HTML5&CSS3 JQuery XNA SharePoint GUI for Android Windows Azure Platform Microsoft Patterns&Practices TFS SCRUM ReSharper TDD WWF LINQ Entity Framework Windows Forms Refactoring Microsoft Expression Blend 4 Windows Phone 8 Windows 8 AppStore Visual Studio Tips&Tricks MSF MEF SilverLight AJAX MEF Service Oriented Architecture

Пройти тест

Наши партнеры

Microsoft Partner CyberBionic ITVDN PROMETRIC TEST CENTER PEARSON VUE Authorized Test Center Windows Azure Cloud Partner EBA

Дополнительные ресурсы:

Очное обучение On-line обучение Видео обучение

TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и общей оценки знаний IT специалиста.

После каждого урока проходите тестирование для проверки знаний на [TestProvider.com](http://TestProvider.com)

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат



# Профессиональное программирование на языке Java

После каждого урока обязательно



Повторите этот урок в видео формате на [ITVDN.com](http://ITVDN.com)

Доступ можно получить через руководство вашего учебного центра



Проверьте как Вы усвоили данный материал на [TestProvider.com](http://TestProvider.com)

Спасибо за внимание! До новых встреч!

# Информационный видеосервис для разработчиков программного обеспечения

