Профессиональное программирование на языке Java

Объектно-ориентированное программирование. Документирование программ

Автор курса

Александр Бабич

Microsoft CERTIFIED Trainer

Professional Advanced

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

Тема

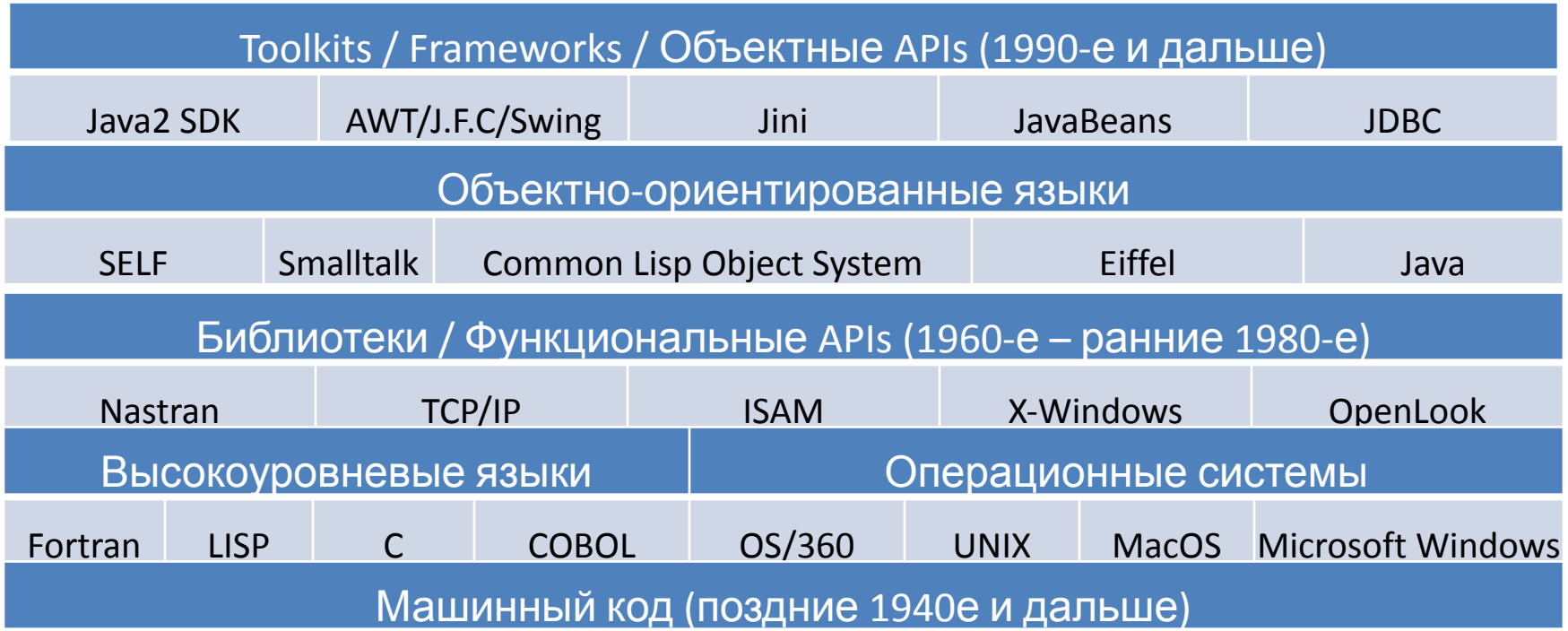
Объектно-ориентированное программирование. Документирование программ

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

Вместо предисловия



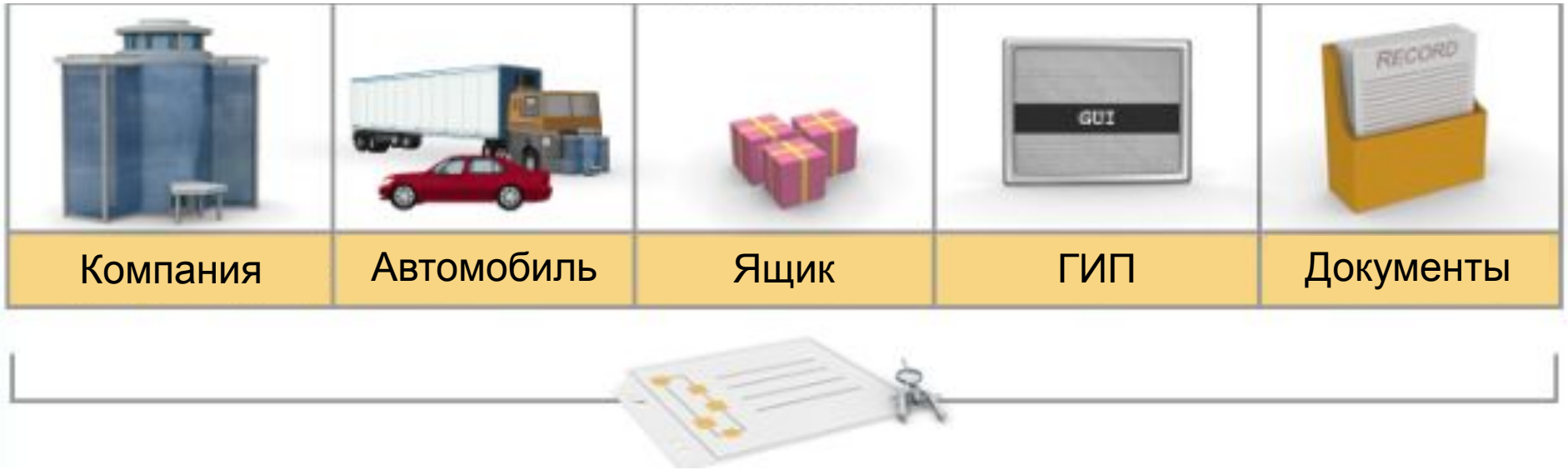
Вместо предисловия

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

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)

Классы, как шаблоны (чертежи) для объектов

При производстве, чертеж – это описание (проект) внутренней
структуры и функциональности создаваемого объекта

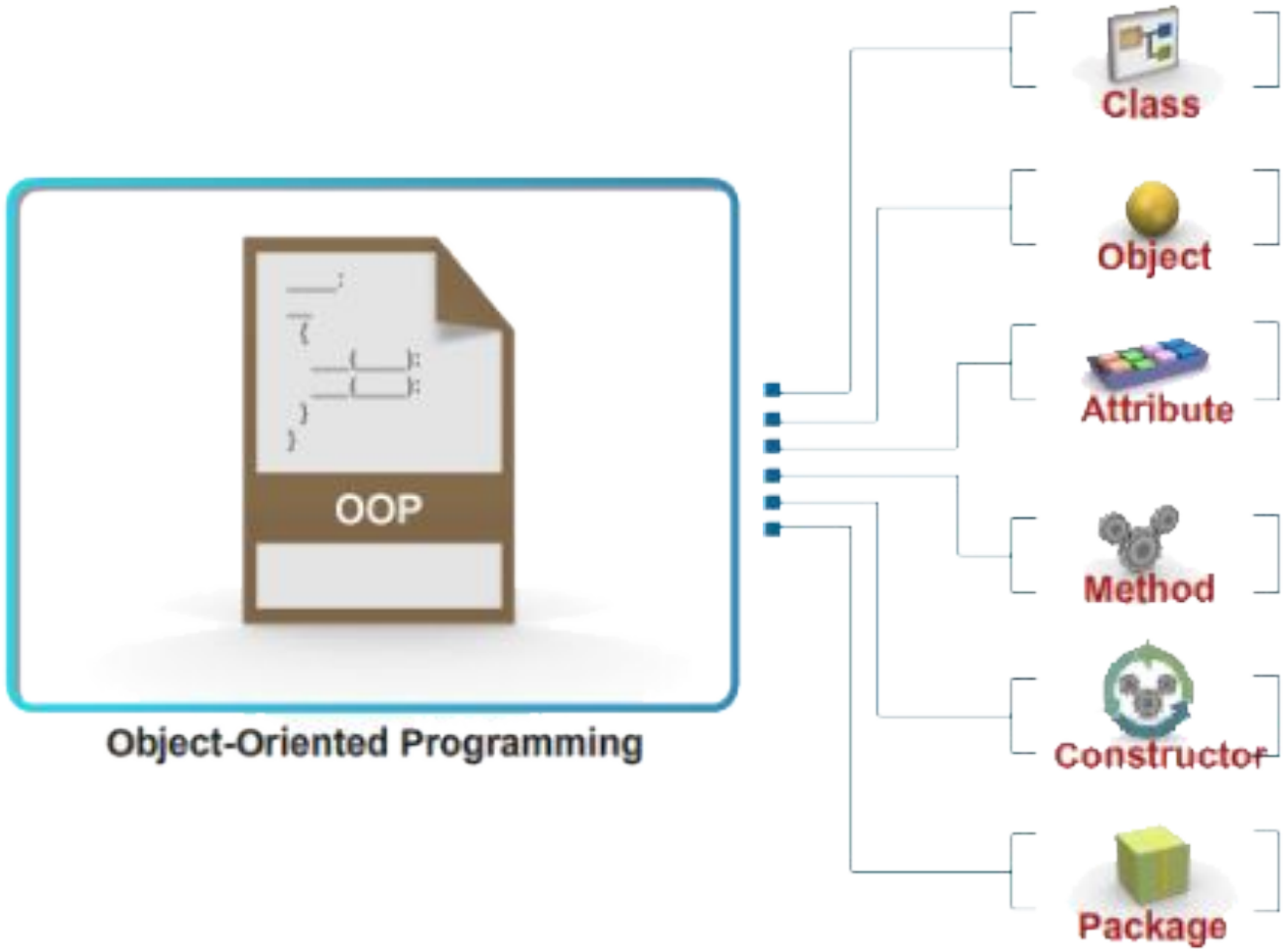
В программировании – класс – это описание объектов:

- 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

/ляция – свойство системы, позволяющее объединить работающие с ними, в рамках класса. C++, C#, Java капсуляцию

тием

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 Модула, 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

Повторное использование кода

Повторное использование кода (code reuse) – методология проектирования программных систем, при использовании которой система частично либо полностью составляется из частей, написанных ранее компонентов и/или частей другой системы, причем эти компоненты применяются более одного раза (возможно, в разных проектах)

Хороший пример модульности и принципа повторного использования – стандартные (и не только) классы 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";
   }
```

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)

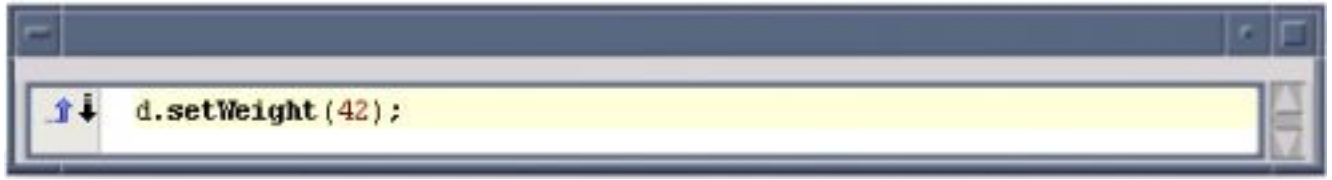
Набор операций является общим для всех объектов данного класса

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     }
}
```

Доступ к атрибутам объекта из метода



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

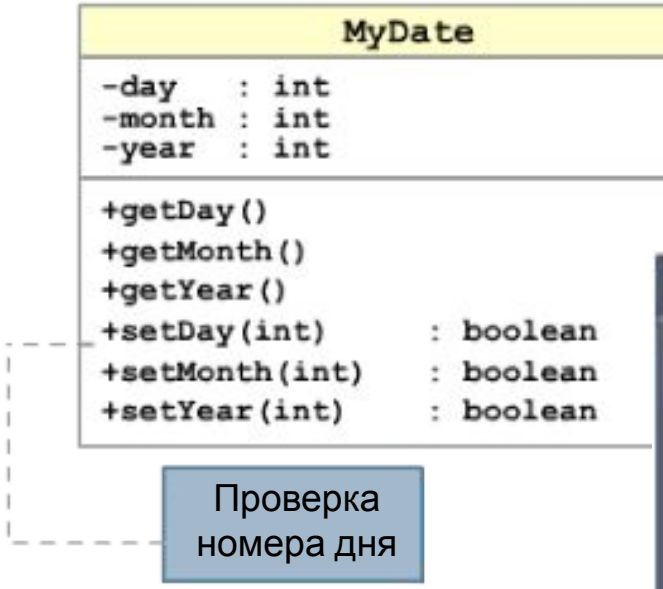
```
1  public class TestDog {
2      public static void main(String[] args) {
3          Dog d = new Dog();
4          System.out.println("Dog d's weight is "
5                             + d.getWeight());
6          d.setWeight(42);
7          System.out.println("Dog d's weight is "
8             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

Вспомним об инкапсуляции: «плохой» пример


```java
public class MyDate {
    public int day;
    public int month;
    public int year;
}
```

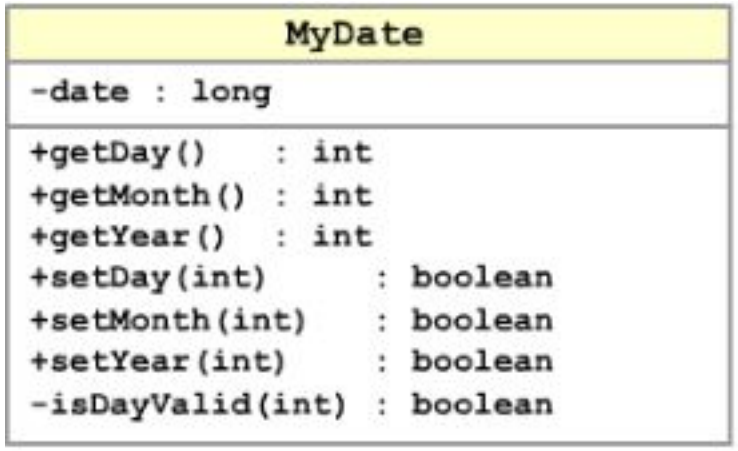
```java
d.day = 32;
// invalid day

d.month = 2; d.day = 30;
// plausible but wrong

d.day = d.day + 1;
// no check for wrap around
```

Вспомним об инкапсуляции: «правильный» пример




```
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);
   // this will return false if wrap around needs to occur
```

Подробнее об инкапсуляции



Позволяет скрыть подробности реализации класса

Заставляет использовать интерфейс класса для доступа к его данным

Делает код более управляемым

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;
       }
```

Конструктор по умолчанию

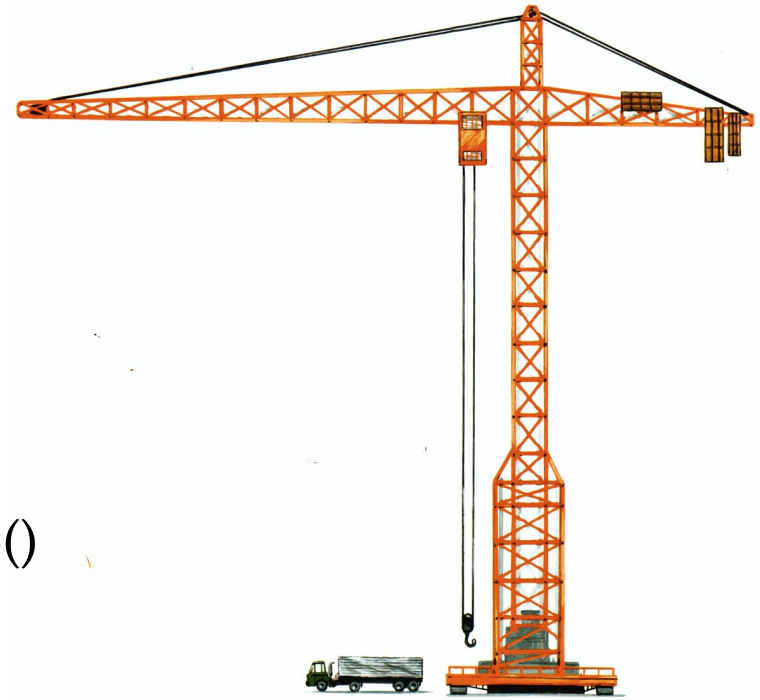
Каждый класс всегда имеет хотя бы один конструктор

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

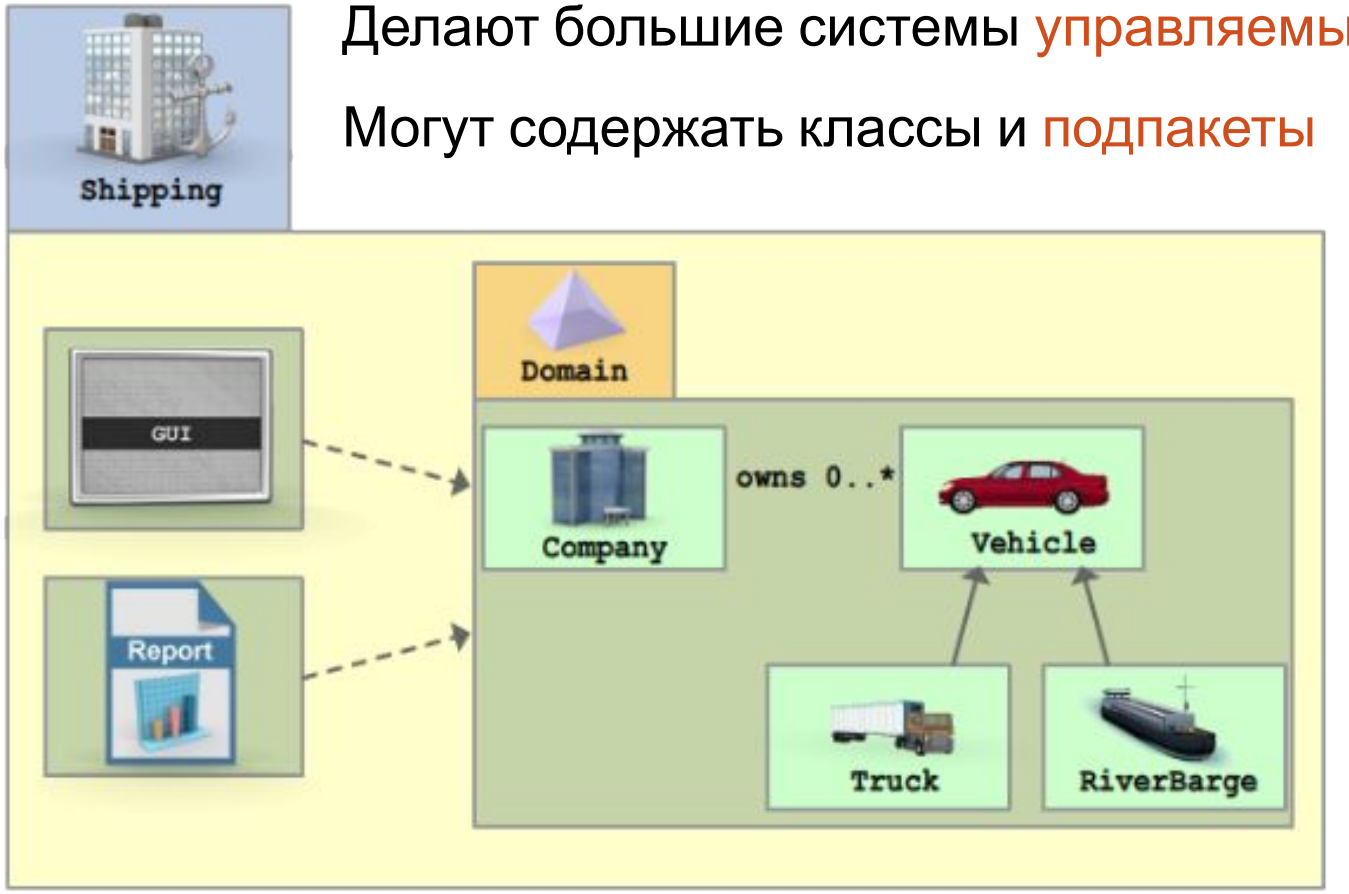
Оператор импорта должен быть указан перед объявлениями классов

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      ...
   }
```

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      ...
   }
```

Оператор импорта

Описание пакета

- Тот же пакет импортировать не нужно
- Оператор импорта делает классы пакета доступными для текущего класса
- Импортировать следует только нужные классы

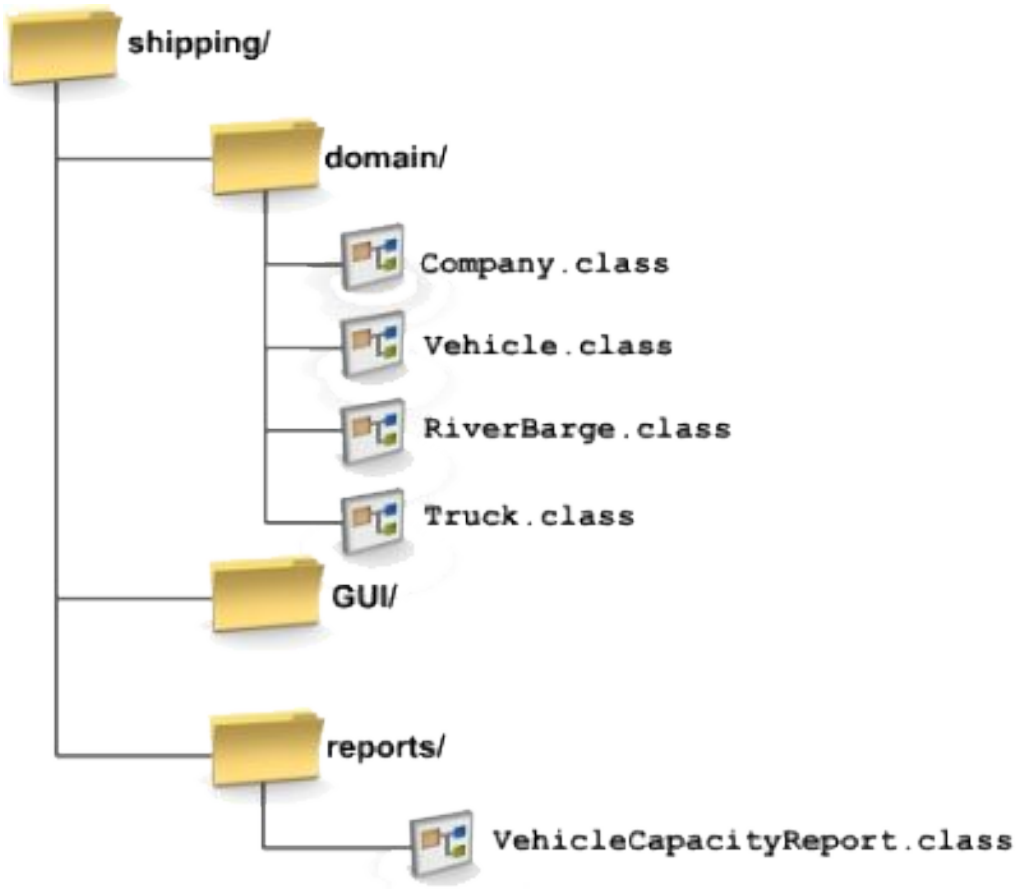
Оператор импорта

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

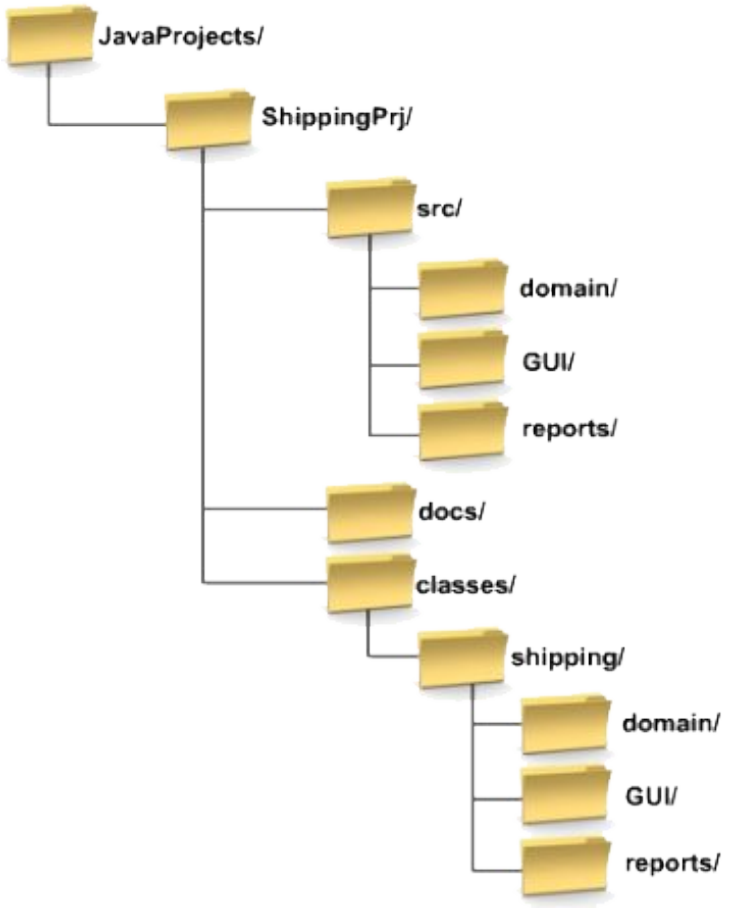
```
import java.io.*;
```

Папки и пакеты

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



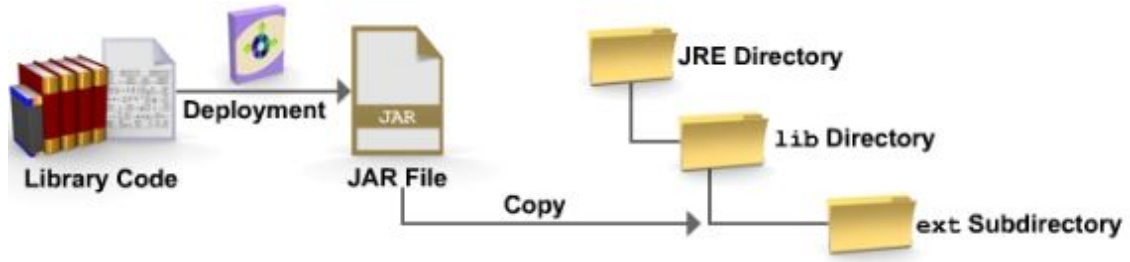
Папки и пакеты



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

Повторяем: модификаторы видимости

private – члены класса доступны только внутри класса

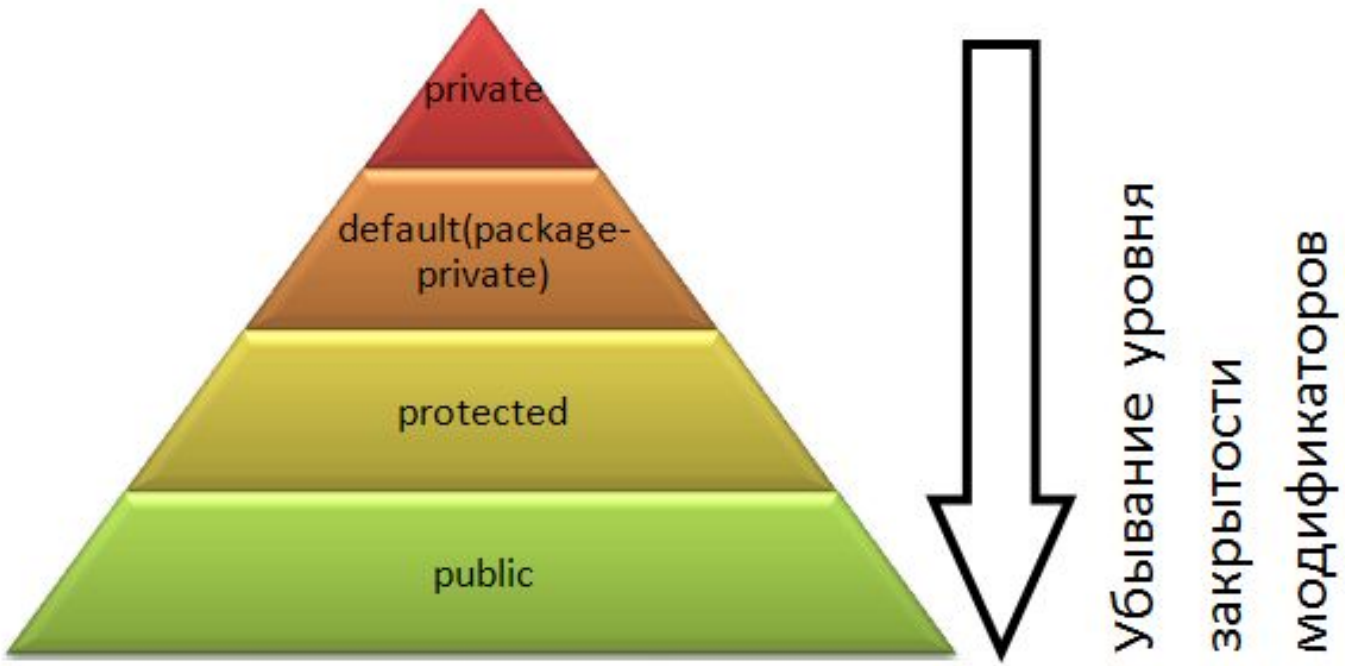
default (package-private) (по-умолчанию) – члены класса видны только внутри пакета

protected – члены класса доступны внутри пакета и в его наследниках

public – члены класса доступны всем

Пирамида модификаторов видимости

По убыванию уровня закрытости: private, default ,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

Онлайновая документация

Большой набор HTML- фалов 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

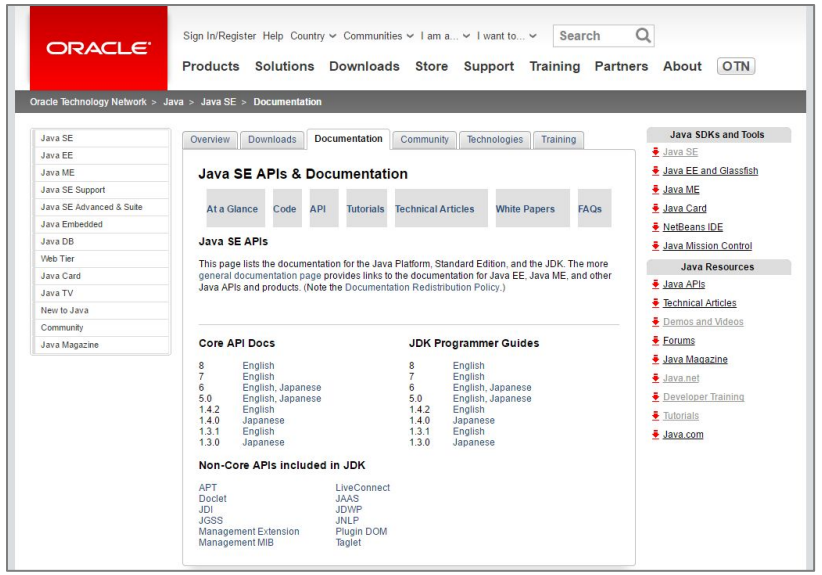
Выбор нужной документации

Раздел Java SE APIs & Documentation

- примеры кода
- обучающие материалы
- статьи и т.п.

Можно выбрать версию Java

Документация по другим API, включенным в JDK



http://www.oracle.com/technetwork/java/javase/documentation/api-jsp-136079.html

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) {…}
```

Специальные теги javadoc для параметров методов

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

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 ArgumentValueException 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#mathodA()}

преобразуется в гиперссылку на метод класса

Что в остатке?

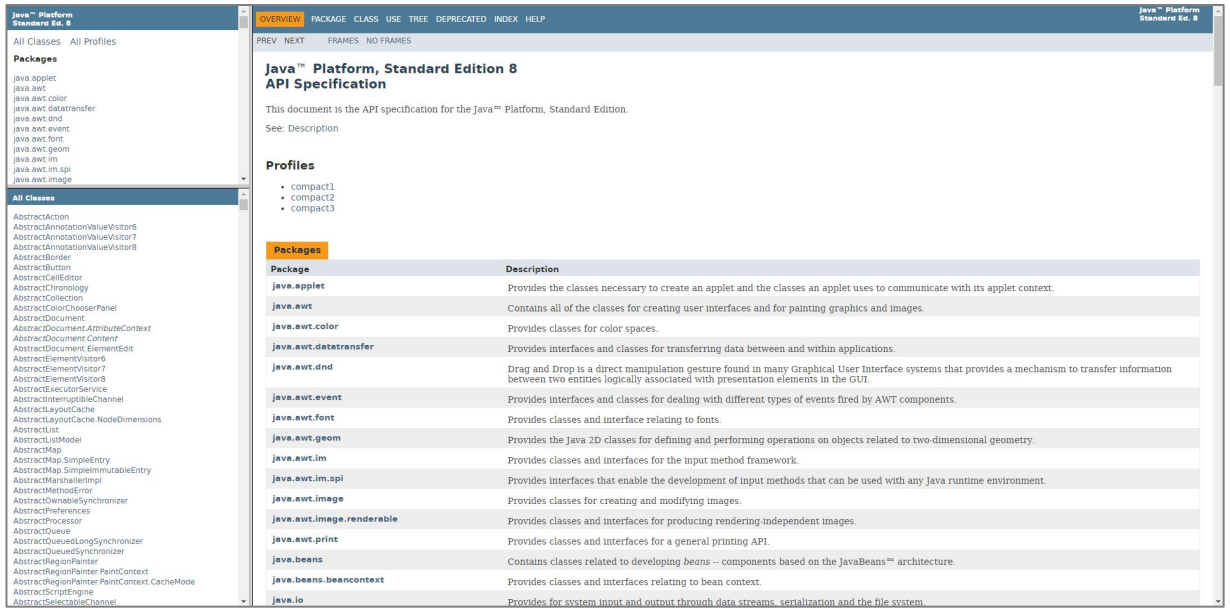
Доступность, простота, стимул комментировать код☺

Создаем документацию не покидая исходный код

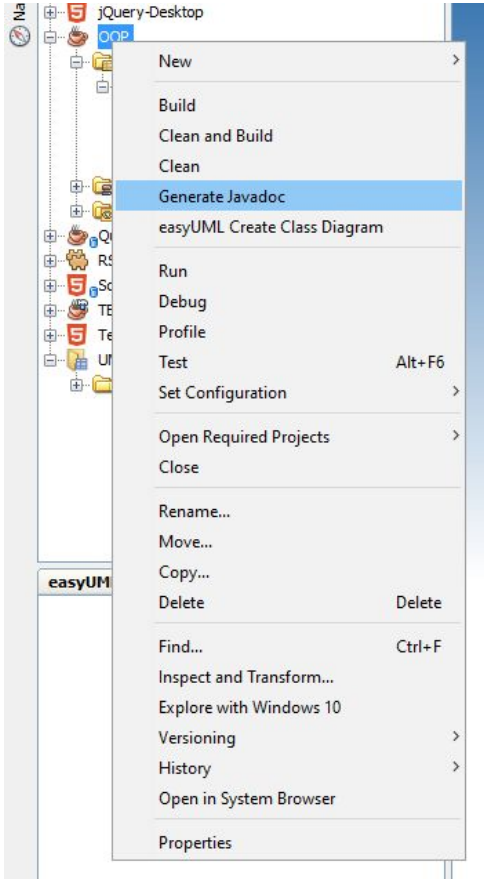
Имеем:

стандартное оформление

удобную навигацию



Генерация документации в Netbeans



Простой вызов javadoc прямо из контекстного меню проекта

Результат помещается в папку dist › javadoc внутри папки проекта

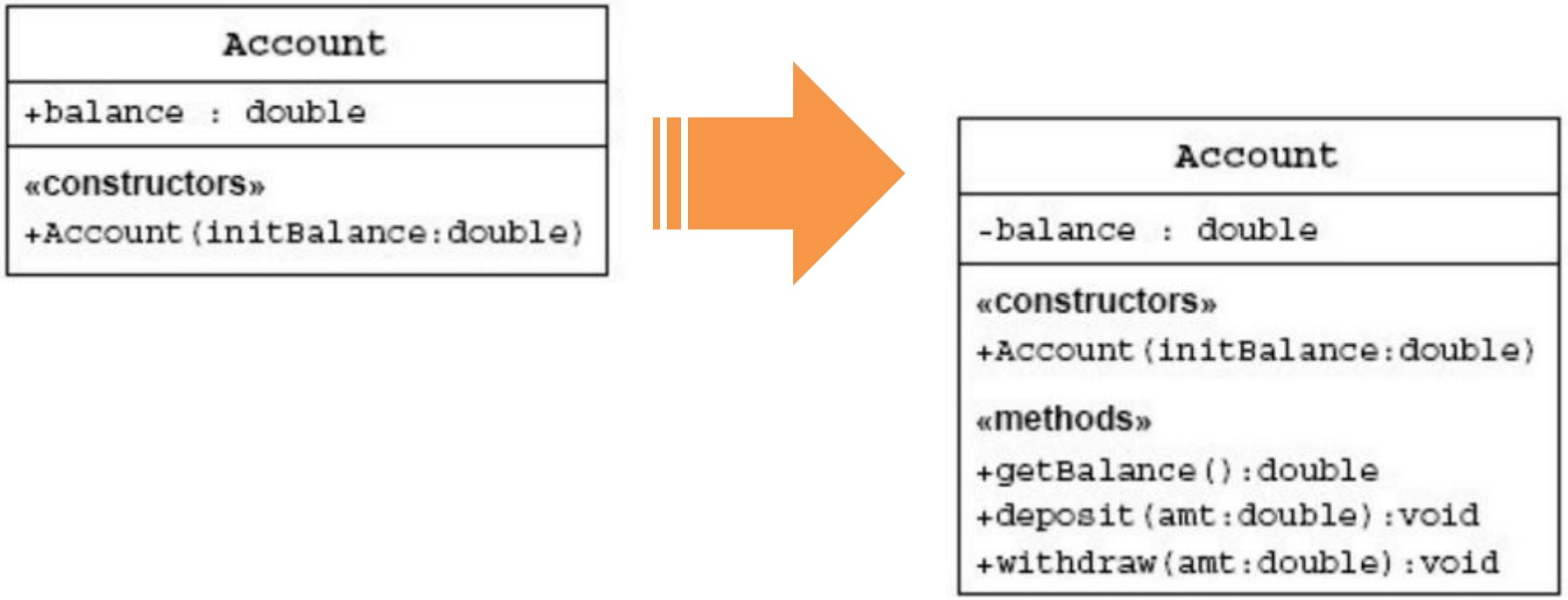
Генерация документации в Netbeans

Лабораторная работа

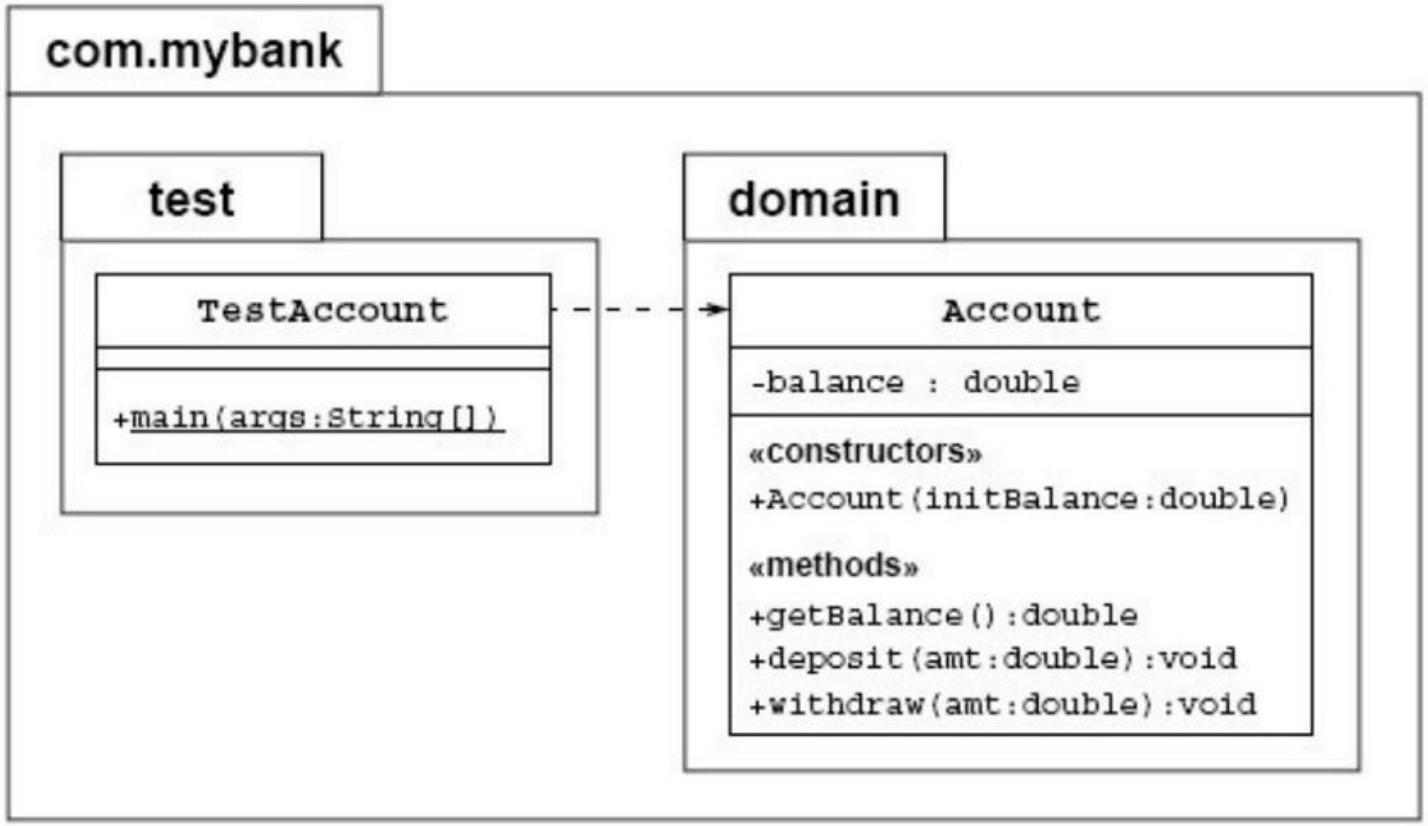
ЛР: Создание простых программ в IDE Netbeans

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

ЛР: Создание простых программ в IDE Netbeans



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 видимость 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

Посмотрите этот урок в видео формате на образовательном портале ITVDN.com для закрепления пройденного материала.

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

ITVDN
IT VIDEO DEVELOPERS NETWORK

Проверка знаний

TestProvider.com

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

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

TestProvider

После каждого урока обязательно

Повторите этот урок в видео формате на ITVDN.com

Доступ можно получить через руководство вашего учебного центра

Проверьте как Вы усвоили данный материал на TestProvider.com

Спасибо за внимание! До новых встреч!