**Пакеты позволяют организовать связанные между собой типы в отдельные пространства имён. Это позволяет типам (классам, интерфейсам и перечислениям) в разных пакетах иметь одинаковые имена, но при этом не порождать конфликтов и неопределённостей.**

**Причины использования пакетов:**

**1)-Можно легко понять, что эти типы в одном пакете связаны между собой.**

**2)-Легче искать классы с определённым функционалом, так как они находятся в одном пакете.**

**3)-Имена типов не будут конфликтовать с именами типов, созданными другими разработчиками.**

**4)-Можно дать классам внутри пакета полный доступ к членам друг друга, но при этом запретить доступ к некоторым из них снаружи (будет объяснено в главе про классы).**

Дальше идёт актуальная для консольной работы информация, в условиях современных IDE это мало актуально.

Чтобы указать пакет, к которому относится класс нужно использовать ключевое слово package в самом верху файла с классом. Пример:

Main.javaJava

package ru.urvanov.javaexamples.packageexamples;

public class Main {

public static void main(String[] args) {

System.out.println("Hello, World!");

}

}

Большинство реализаций Java рассчитывают на иерархическую файловую систему при организации файлов исходников и файлов .class , хотя спецификация Java НЕ требует этого.

Чтобы успешно запустить (речь идёт об консоли) скомпилированный Main.class из примера выше нужно создать структуру каталогов, повторяющую структуру пакета:

-> ru

-> urvanov

-> javaexamples

-> packageexamples

-> Main.java

-> Main.class

Запускать нужно из каталога, в котором находится каталог «ru» (в случае Linux нужно использовать знак «/» вместо «\»):

javac ru\urvanov\javaexamples\packageexamples\Main.java

java ru.urvanov.javaexamples.packageexamples.Main

Обратите внимание на используемое имя пакета: «ru.urvanov.javaexamples.packageexamples». Согласно соглашению о кодировании имена пакетам даются в соответствии с правилами:

1)-**Пакеты записываются строчными буквами**, чтобы избежать конфликтов с именами классов и интерфейсов.

2)-**Используется обратная запись домена организации**, например для urvanov.ru получится ru.urvanov.

3)-После обратной записи домена добавляется имя пакета или имя проекта с именем пакета внутри проекта. В примере выше имя проекта javaexamples, а имя пакета внутри проекта packageexamples.

4)-Чтобы избежать конфликтов имён внутри организации можно добавлять между именем домена и именем проекта имя региона или отдела. Например: ru.urvanov.moscow.javaexamples.packageexamples.

5)-Пакеты внутри Java начинаются с java или javax.

Дефис в имени домена заменяется на символ подчёркивания.

Если имя домена содержит ключевое слово, то к имени пакета добавляется подчёркивание в конце.

Если имя домена начинается с цифры, то добавляется подчёркивание в начало имени пакета.

Примеры:

Имя домена Префикс имени пакета

hyphenated-name.example.org org.example.hyphenated\_name

example.int int\_.example

123name.example.com com.example.\_123name

Если же пакет не указывать, то используется пакет по умолчанию. Пакет по умолчанию можно использовать только для очень простых проектов.

Обратиться к членам пакета внутри самого пакета можно по имени самого члена, имя пакета указывать не обязательно:

Main main = new Main();

Обратиться к членам пакета из другого пакета можно по полному имени, включающему имя пакета. Например, для класса Main выше:

ru.urvanov.javaexamples.packageexamples.Main

Создать экземпляр класса Main из другого пакета можно так:

ru.urvanov.javaexamples.packageexamples.Main main = new ru.urvanov.javaexamples.packageexamples.Main();

Если пакет используется часто, то постоянное написание полного имени пакета усложнит понимание кода. В таких случаях можно использовать операторы import , которые добавляются между package и объявлениями классов:

package ru.urvanov.javaexamples.otherpackage;

import ru.urvanov.javaexamples.packageexamples.Main;

// другие операторы import

class OtherClass {

public void method1() {

Main main = new Main();

}

}

С помощью оператора import можно также импортировать всё содержимое пакета с помощью символа «\*», а не только один класс:

OtherClass.javaJava

package ru.urvanov.javaexamples.otherpackage;

import ru.urvanov.javaexamples.packageexamples.\*;

class OtherClass {

public void method1() {

Main main = new Main();

}

}

С помощью звёздочки можно импортировать только всё содержимое пакета, но нельзя импортировать лишь какую-то его часть.

По умолчанию в Java в каждый файл уже импортированы два пакета: java.lang и пакет текущего файла.

Не смотря на то что пакеты кажутся иерархическими, они такими не являются. Пакет ru.urvanov.javaexamples не включает в себя пакет ru.urvanov.javaexamples.packageexamples. Это два совершенно разных пакета.

Строка

import ru.urvanov.javaexamples.\*;

Импортирует только члены пакета ru.urvanov.javaexamples , но не члены пакета ru.urvanov.javaexamples.packageexamples.

Если два пакета содержат член (класс, интерфейс или перечисления) с одинаковым именем, и оба этих пакета импортированы, то для доступа к этим членам нужно использовать полное имя с указанием пакета:

ru.urvanov.javaexamples.packageexamples.Main main;

Если вы часто используете константы из какого-либо класса, то постоянное указание имени класса делает код более запутанным. В таких случаях можно использовать оператор import static. Класс java.lang.Math, например, содержит константу PI. Для более простого доступа к ней можно импортировать эту константу:

Java

import static java.lang.Math.PI;

Слова import и static менять местами нельзя! Нельзя писать static import, только import static!

Теперь можно обращаться к PI только по имени:

Java

double x = PI \* 2;

Можно импортировать все константы из класса с помощью звёздочки:

Java

import static java.lang.Math.\*;

Не стоит лишний раз использовать import со звёздочкой и import static. Лучше включать в файл только нужное. Современные IDE позволяют буквальное парой кнопок организовать операторы import в файле.