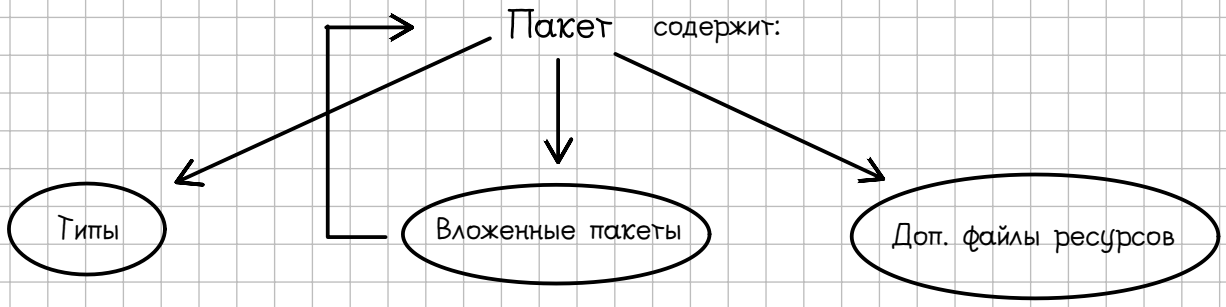
11.09.25

Понятие о пакетах

Пакет - логическая сущность

Все типы должны принадлежать какому-либо пакету

Типы - классы, интерфейсы, перечисления, записи



Назначения пакетов:

1. Группируют взаимосвязные типы
2. Избежание конфликтов имён
3. Доп. средства защиты элементов кода
4. Формируют иерархическую систему

Реализация пакетов

В виде структуры каталогов:

Каждый пакет - каталог

Каждый тип - файл (Например User.class)

В виде jar-архива

zip архив с байт-кодами

Указание пути к используемым пакетам

- Непосредственно при запуске JVM (ключ -cp)
- Через переменную окружения CLASSPATH
- Также используются пакеты "текущего каталога" из которого запускается JVM

Понятие имени

Пространства имён (namespaces)

- пакеты
- типы
- методы
- поля
- локальные переменные и параметры
- метки

Имена бывают

Простые

↓

Double

Составные

↓

java._____.Double

Понятие модуля компиляции

Модуль компиляции хранится в .java файле

Состоит из:

- объявление пакета

```
package mypackage;
```

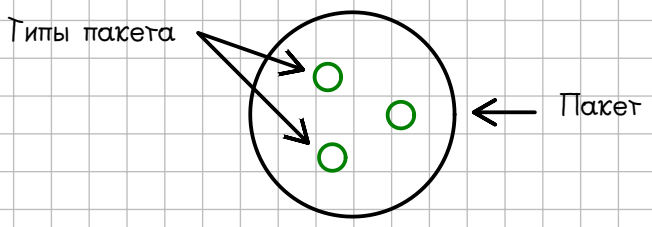
- выражений импортирования

```
import java.net.Socket;
```
→ Будем использовать Socket из пакета java.net

```
import java.io.*;
```
→ Будем использовать ВСЕ из пакета java.io

- объявлений верхнего уровня - типов

Объявление пакета




```
package java.lang;
```

При отсутствии объявления пакета модуль компиляции принадлежит безымянному пакету — На уровне JVM

Пакет доступен, если доступен модуль компиляции с объявлением пакета — Физически на диске!

Ограничения на доступ к пакетам нет — Только на типы пакета

Выражения импорта

Доступ к типу из данного пакета - по простому имени типа

Доступ к типу из других пакетов - по составному имени пакету и имени типа

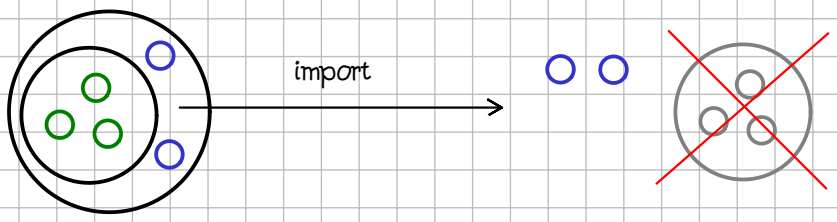

```
import java.net.Socket;
```
```
import java.io.*;
```

Попытка импорта пакета, недоступного на момент компиляции вызовет ошибку

Дублирование импорта игнорируется

Нельзя импортировать вложенный пакет по имени пакета — `import java.net;` ✗

При импортировании типов из пакета, вложенные пакеты НЕ импортируются



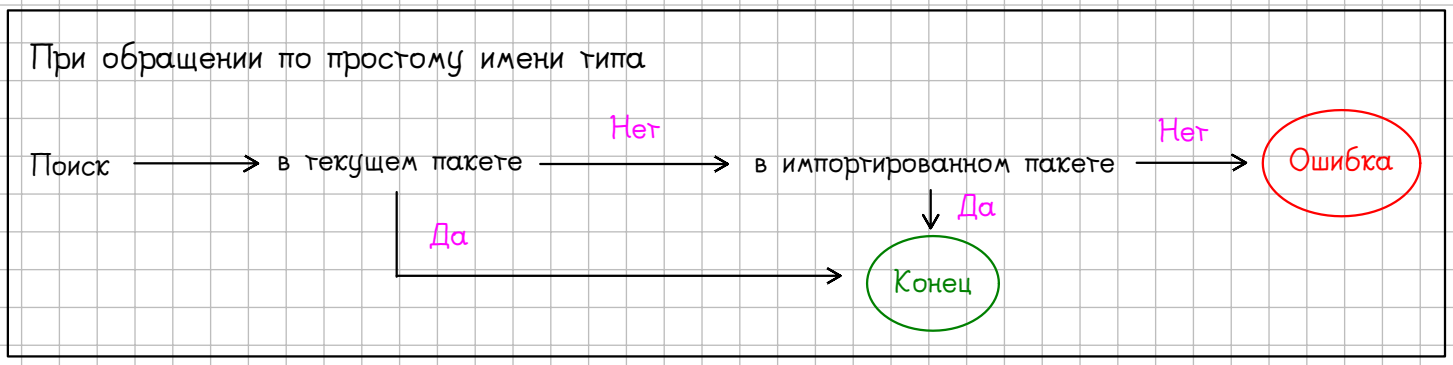
Алгоритм компилятора при анализе типов:

- выражения, импортирующие типы → `import p1.MyClass;`
- выражения, импортирующие пакеты → `import p2.*;`
- другие объявленные типы → `class MyClass {}`

Создавать тип с именем, идентичным импортированному типу НЕЛЬЗЯ

Во всех 3х случаях импортируется MyClass

Последний импортированный тип переопределяет предыдущие



Статический импорт

Статический импорт - импорт статических элементов (константы, статич. методы и т.д.)

```
import static java.lang.Math.sqrt;      Явно
import static java.lang.Math.*;         Неявно
```

Особенности статического импорта:

- Уменьшает объём кода
- Уменьшает удобство чтения программ
- Приводит к конфликтам имён

Мораль: использовать, когда действительно необходим

Объявление верхнего уровня

```
package first;
import …;
class MyFirstClass { … }
interface MyFirstInterface { … }
enum MyFirstEnumeration { … }
record MyFirstRecord { … }
```

Только ОДИН может быть public

Тогда имя модуля компиляции должны совпадать

Следовательно - 1 класс - 1 файл

(но можно добавлять не-public типы)

Правила именования

Пакеты с маленькой буквы

java.lang, javax.swing, ru.ssau.infokom

Типы

Student, ArrayIndexOf..., Cloneable, Runnable, Serializable

Обозначаются интерфейсы

Поля camelCase

value, enabled, distanceFromShop

Методы camelCase

getValue, setValue, isEnabled, length, toString

Поля-константы

PI, SIZE_MIN, SIZE_MAX, SIZE_DEF

Локальные переменные camelCase

i, value, isFound