

# Core Java, весна

## 3 лекция

Модули

# Какие проблемы помогают решить java modules?


- 1) Избыточность подключаемых библиотек, для работы приложения. Что приводит к увеличению размеров итогового jar и всего jre.
- 2) Невозможность скрыть все желаемые части пакета, особенно с учетом рефлексии.

16.0.2-open Properties

Basic

Permissions

Local Network Share



Name:

16.0.2-open

Type:

Folder (inode/directory)

Contents:

491 items, totalling 317,2 MB

Parent folder:

/home/seralekseenko/.sdkman...

Modified:

cp, 04-cep-2021 12:34:19 +0300

Free space:


193,7 GB

myJRE Properties

Basic

Permissions

Local Network Share



Name:

myJRE

Type:

Folder (inode/directory)

Contents:

63 items, totalling 49,8 MB

Parent folder:

/mnt/share/academy/MODULES

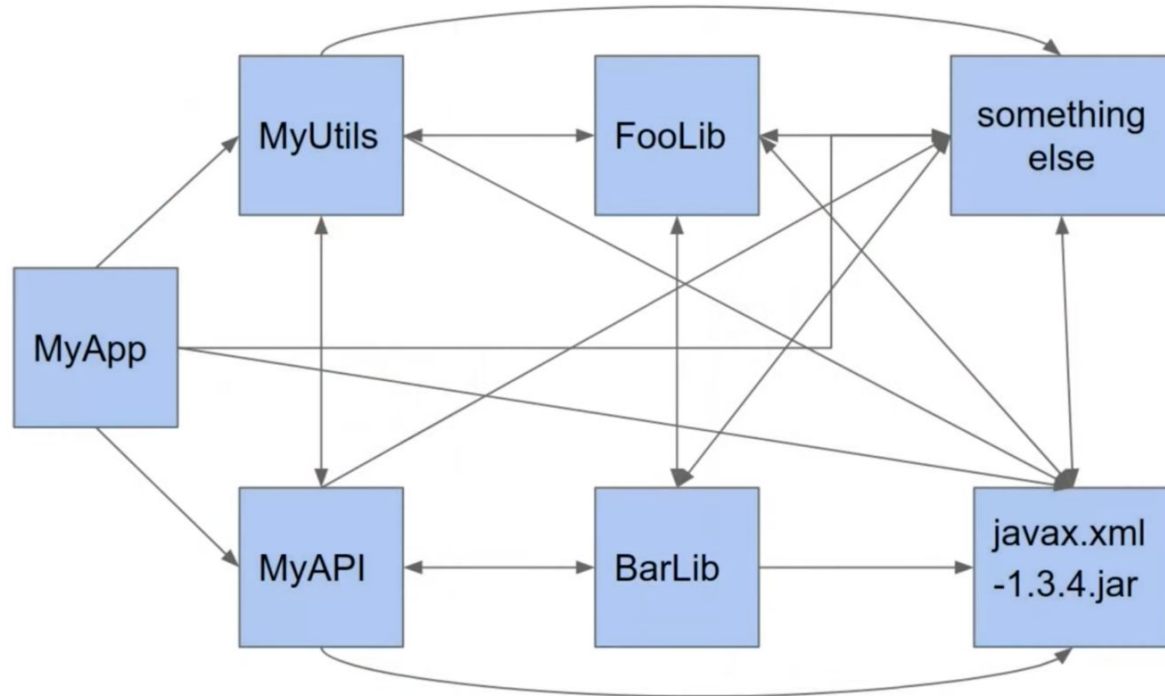
Modified:

пт, 03-вep-2021 21:12:00 +0300

Free space:

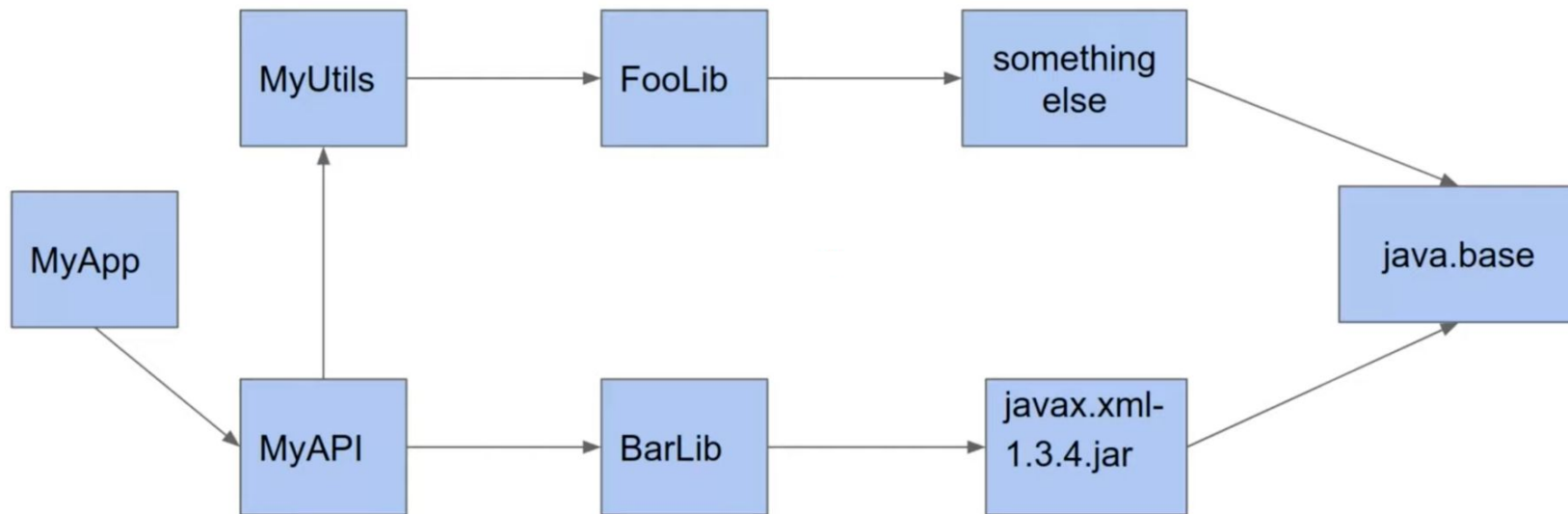
917,1 GB

Как раньше выглядели зависимости и classpath.



```
java -cp javax.xml-1.3.4.jar:MyUtils.jar:MyAPI.jar:Bar.jar:Foo.jar:MyApp.jar org.example.Main
```

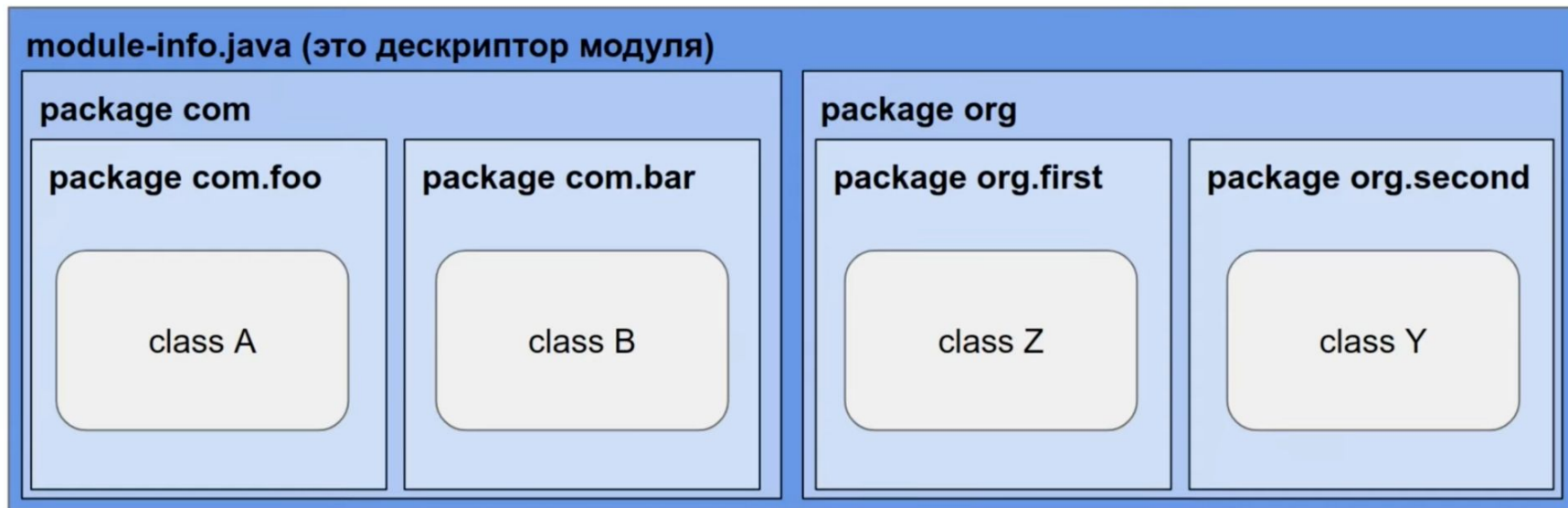
# А теперь?



```
java --module-path path/to/module --module path/to/module/org.example.Main
```

```
java -p path/to/module -m path/to/module/org.example.Main
```

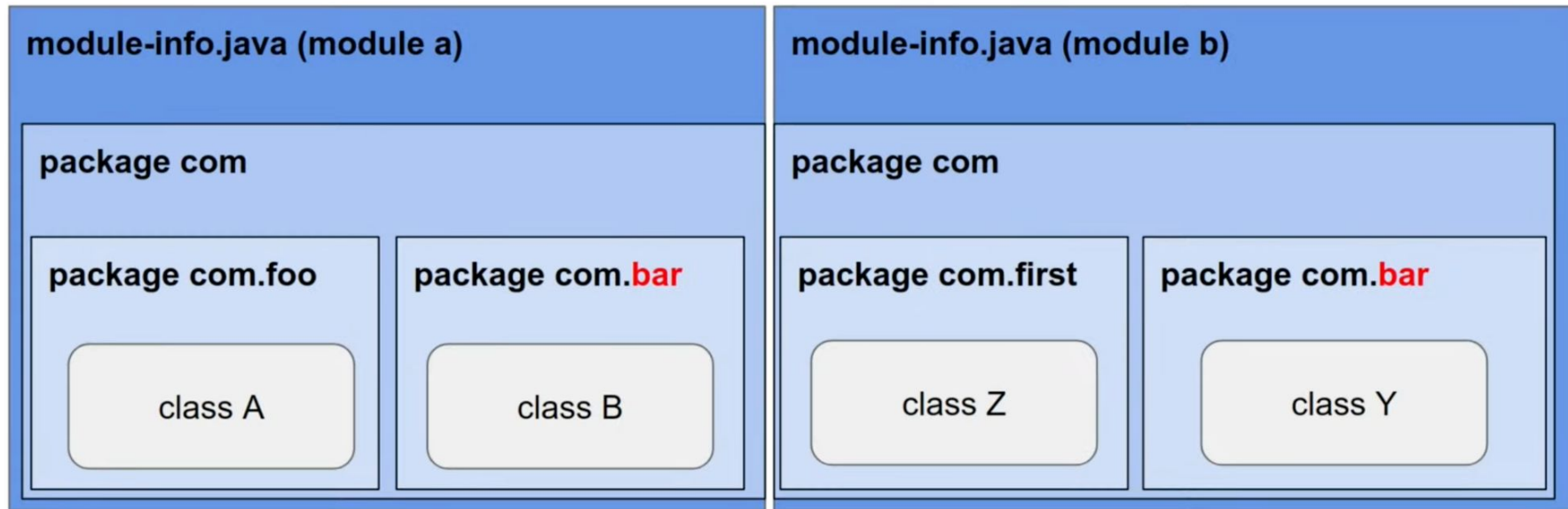
# Что такое модульность в java?



Модульность — это новая сущность в java, такая себе надстройка над стандартными пакетами, которая дает дополнительные инструменты инкапсуляции и управления зависимостями.

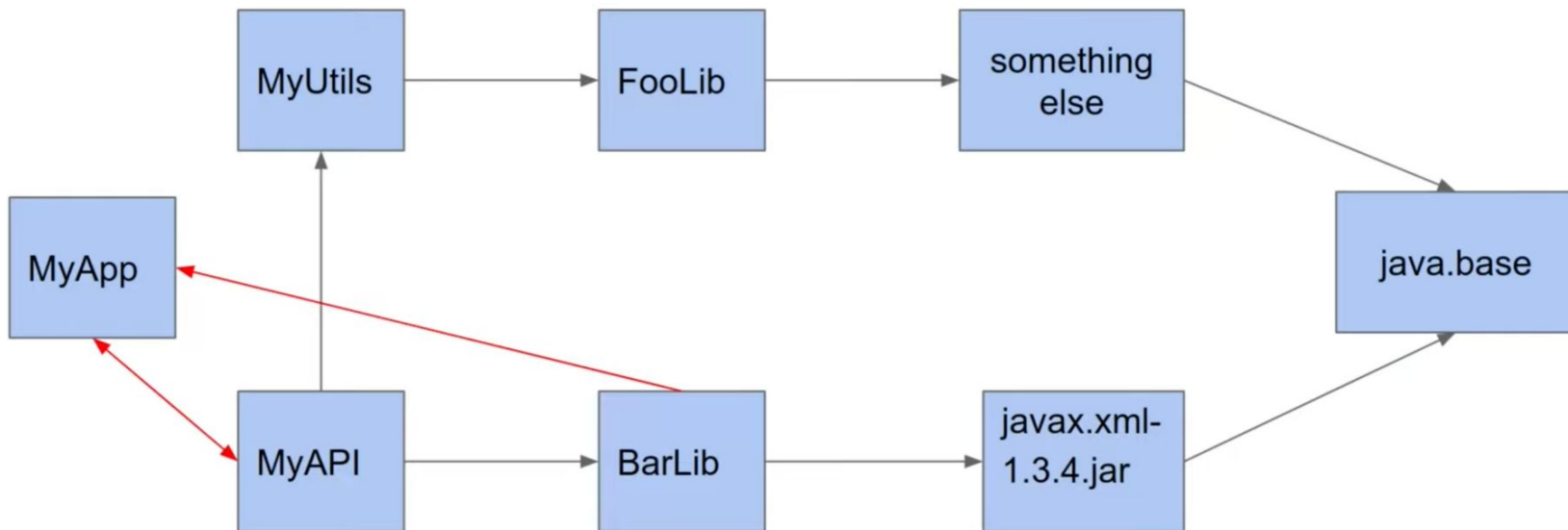
# В модулях запрещено:

A. Использовать одинаковые имена пакетов в нескольких модулях одного приложения.



# В модулях запрещено:

**В.** Создавать циклические зависимости между модулями





# Оговорка об этих запретах.

Это все не касается **unnamed** модулей, которые создаются автоматически из приложений и библиотек не являющихся модулями.

# Что может включать в себя модуль:

- the module's name (обязательно)
- the module's dependencies (не обязательно, вдруг модуль независим)
- the packages it explicitly makes available to other modules (не обязательно, если текущий модуль не будет использован другими модулями)
- the services it offers (не обязательно)
- the services it consumes (не обязательно)
- to what other modules it allows reflection (не обязательно)

# Новый уровень инкапсуляции.

java 8 и ранее:

- private
- <default>
- protected
- public

java 9 и далее:

- private
- <default>
- protected
- public (only in this module)
- public (for a specific module)
- public (for all)

```
module moduleName {  
    // Зависимости модуля  
    requires moduleName; // Зависимость от другого модуля  
  
    // Зависимость на модуль для компиляции и запуска (обычная зависимость)  
    requires transitive moduleName; // Зависимые модули автоматически доступны для пользователей данного модуля  
  
    // Опциональная зависимость, используется только во время выполнения  
    requires static moduleName; // Зависимость доступна только в рамках компиляции, позволяя использовать модуль при запуске при необходимости  
  
    // Экспортирование пакетов  
    exports packageName; // Делает пакет доступным для всех модулей  
  
    // Экспортирование пакета конкретному модулю  
    exports packageName to moduleName; // Делает пакет доступным только для указанного модуля  
  
    // Открытие пакетов  
    opens packageName; // Делает все типы в пакете доступными для deep reflection, использующих Reflection API для доступа к классам  
  
    // Открытие пакета конкретному модулю  
    opens packageName to moduleName; // Открывает пакет для deep reflection только для указанного модуля  
  
    // Использование сервисов  
    uses serviceName; // Указывает, что модуль использует конкретный сервис, определенный как интерфейс или абстрактный класс  
  
    // Предоставление реализаций сервисов  
    provides interfaceName with implementationName; // Предоставляет сервис с его реализацией другим модулям  
  
    provides interfaceName with implementationName1, implementationName2; // Можно указать несколько реализаций  
  
    // Произвольный доступ к рефлексии  
    open moduleName to moduleName; // Открывает модуль целиком для deep reflection операций из указанного модуля (доступно с Java 9)  
}
```

```
module com.example.myapplication {  
    requires java.sql;  
  
    requires transitive com.example.commons;  
  
    exports com.example.myapplication.api;  
  
    exports com.example.myapplication.internal to com.example.somemodule;  
  
    opens com.example.myapplication.config;  
  
    opens com.example.myapplication.data to com.fasterxml.jackson.databind;  
  
    uses com.example.myapplication.spi.ServiceInterface;  
  
    provides com.example.myapplication.spi.ServiceInterface with com.example.myapplication.impl.ServiceImpl;  
}
```