

NOVIDADES DO JAVA 9 (E 10)

ALEXANDRE AQUILES

Instrutor e Desenvolvedor na Caelum

- Twitter: [@alex_aquiles](#)
- Blog: [alexandreaquiles.com.br](#)
- GitHub: [github.com/alexandreaquiles](#)
- Livro: [Controlando Versões com Git e GitHub](#)

TEMAS

- Interatividade com o JShell - Java 9
- Mudanças nas APIs - Java 9
- **Sistema de módulos** - Java 9
- **var** - Java 10

MUITAS NOVIDADES NO JAVA 9

Java 8: 55 JEPs entregues

Java 9: 91 JEPs entregues

<http://openjdk.java.net/projects/jdk9/>

Focaremos nas mais importantes

Java é chato para explorar as APIs.

Sempre aquela mesma coisa:

```
public class Teste {  
    public static void main (String[] args) {  
        //código aqui...  
    }  
}
```

JAVA INTERATIVO

O Java 9 introduz um REPL:

`jshell`

DEMO

/exit

/vars

/list

/drop

/edit

/set editor

/reset

/imports

/open

/save

MUDANÇAS NAS APIS DO JAVA 9

DEMO

List.of

Set.of

Map.of

LocalDate.datesUntil

Stream.iterate

Stream.takeWhile

Stream.dropWhile

Stream.ofNullable

Tem mais...

`Optional.or`

`Optional.stream`

`Optional.ifPresentOrElse`

`Collectors.flatMapping`

`Collectors.filtering`

BUSCA NO JAVADOC

<https://docs.oracle.com/javase/9/docs/api/overview-summary.html>

JAVA PLATFORM MODULE SYSTEM (JPMS)

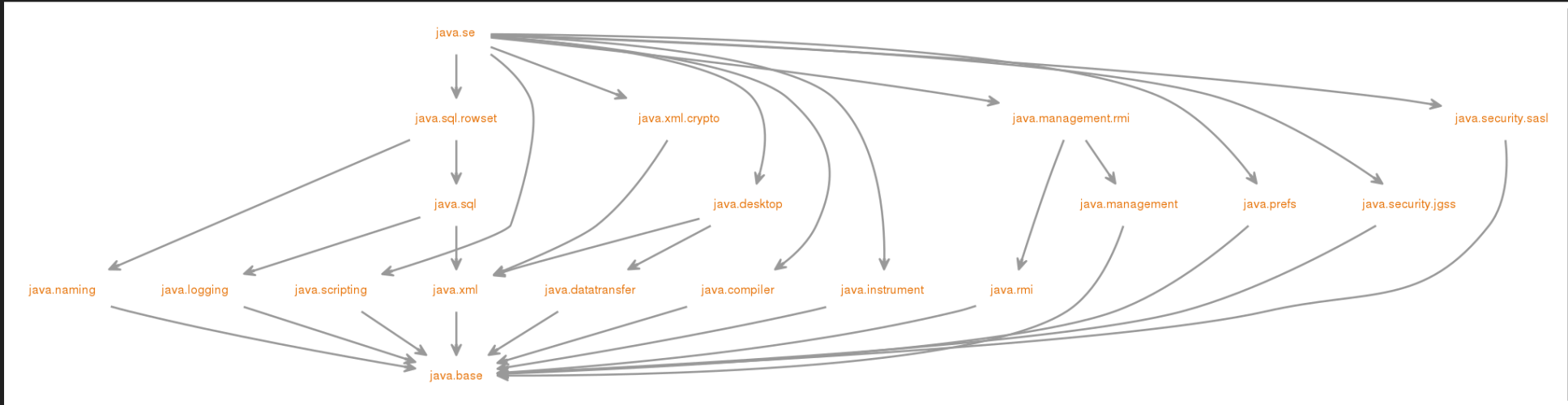
rt.jar do JDK 7 b65



Project Jigsaw: JDK modularization (mid 2009)

base dependia de **Logging** que dependia de **JMX**
que dependia de **JNDI** que dependia de **Applet**

JEP 200: THE MODULAR JDK



Módulo `java.base`

`java.lang`

`java.util`

`java.util.function`

`java.util.stream`

`java.time`

`java.math`

`java.text`

`java.io`

`java.nio`

`java.net`

Outros módulos

java.sql

java.desktop

java.logging

java.scripting

...

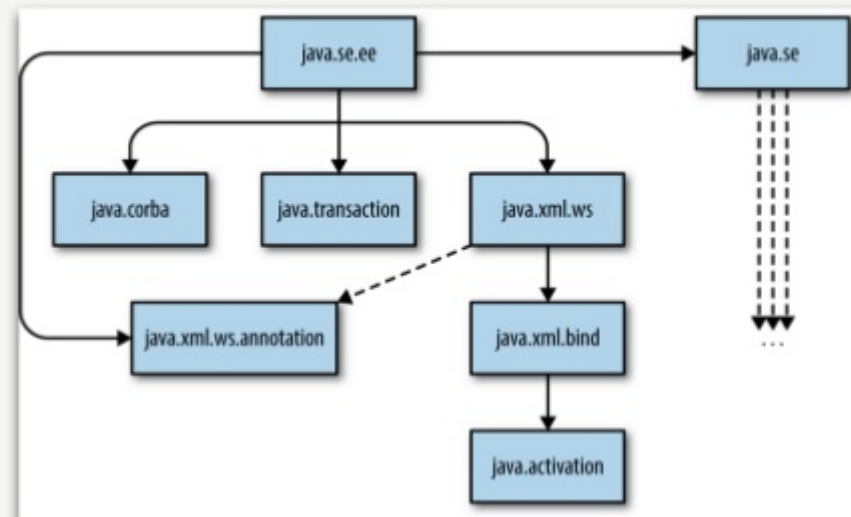
ERROS AO ATUALIZAR PARA JAVA 9 ☹️

DEMO

CLASS PATH X MODULE PATH

Módulo `java.se.ee`

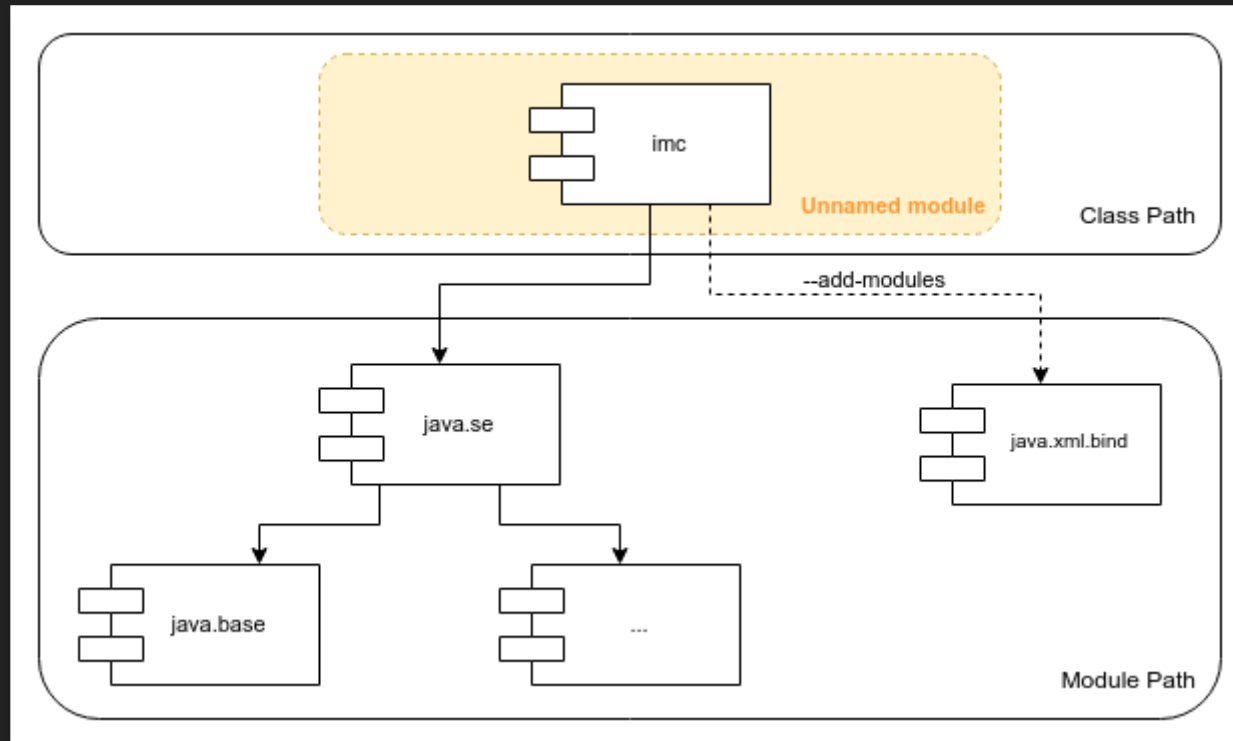
`java.se` vs `java.se.ee`



Fonte: [Migrating to Java 9 Modules](#), Paul Bakker

```
--add-modules java.xml.bind
```

UNNAMED MODULE



Run Configurations

Create, manage, and run configurations

Run a Java application



type filter text

▼ J2EE Preview

Java Applet

▼ Java Application

Teste

JUnit JUnit

JUnit Plug-in Test

Launch Group

▼ m2 Maven Build

Filter matched 22 of 25

Name: Teste

Main Arguments JRE

>> 4

Variables...

VM arguments:

-add-modules java.xml.bind

Revert

Apply



Close

Run

COMPILAR COM O JAVA 9

```
<maven.compiler.source>9</maven.compiler.source>  
<maven.compiler.target>9</maven.compiler.target>
```

ou

```
<maven.compiler.release>9</maven.compiler.release>
```

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.7.0</version>
  <configuration>
    <compilerArgs>
      <arg>- -add-modules</arg>
      <arg>java.xml.bind</arg>
    </compilerArgs>
  </configuration>
</plugin>
```

```
<plugin>
  <artifactId>maven-surefire-plugin</artifactId>
  <version>2.21.0</version>
  <configuration>
    <argLine>
      --add-modules java.xml.bind
    </argLine>
  </configuration>
</plugin>
```







Module properties

Module related details for the library 'JRE System Library [JavaSE-9]'.

☒ Defines one or more modules (Checking/unchecking will move the entry to Modulepath/Classpath)






 Contents  Details

Available modules:

-  java.transaction
-  java.xml.bind
-  java.xml.ws
-  java.xml.ws.annotation
-  javafx.deploy








Explicitly included modules:

-  java.se
-  javafx.base
-  javafx.controls
-  javafx.fxml
-  javafx.graphics



Implicitly included modules:

-  java.base
-  java.compiler
-  java.datatransfer
-  java.desktop
-  java.instrument



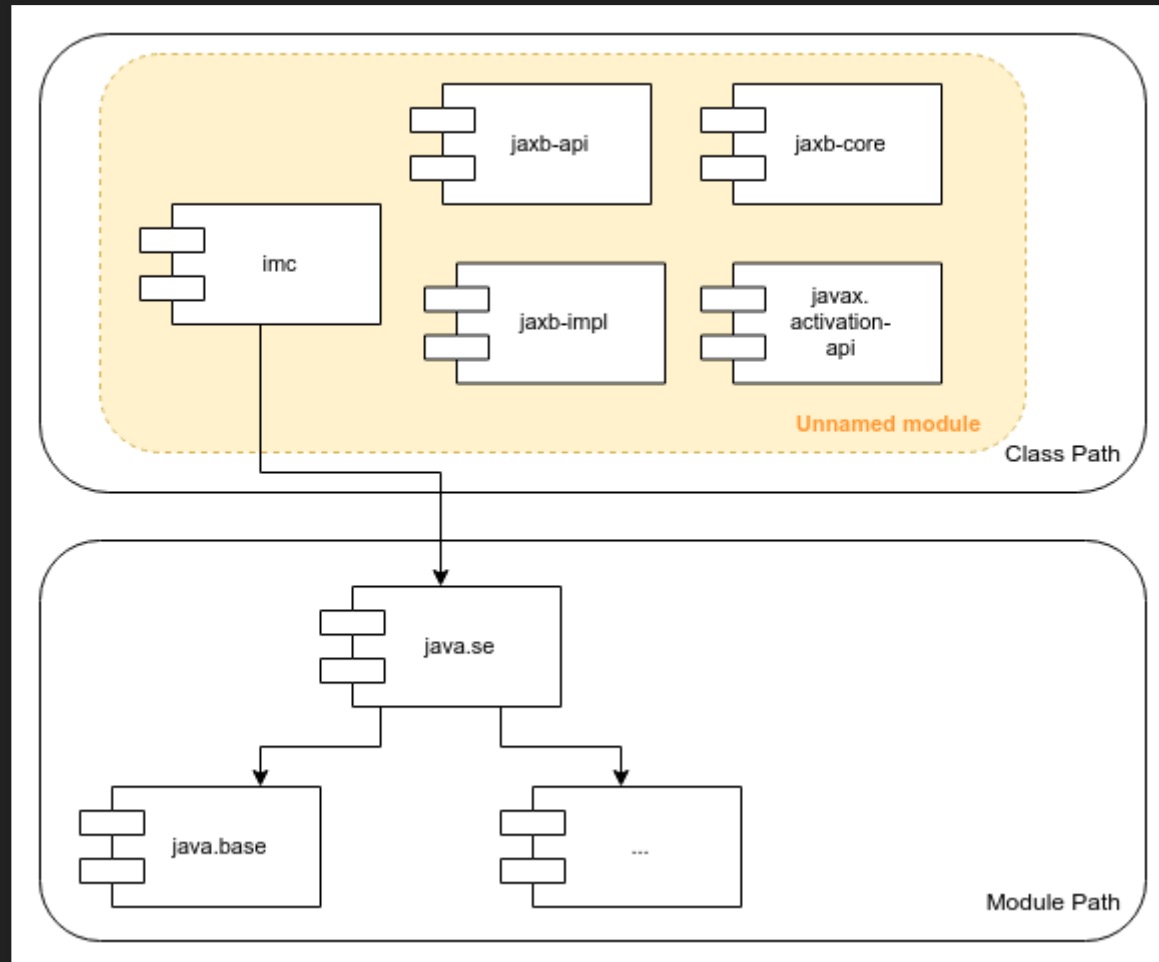
Cancel

OK

JEP 320: Remove the Java EE and CORBA Modules

previsto para o JDK 11 (Setembro/2018)

```
<dependency>
  <groupId>javax.xml.bind</groupId>
  <artifactId>jaxb-api</artifactId>
  <version>2.3.0</version>
</dependency>
<dependency>
  <groupId>com.sun.xml.bind</groupId>
  <artifactId>jaxb-core</artifactId>
  <version>2.3.0</version>
</dependency>
<dependency>
  <groupId>com.sun.xml.bind</groupId>
  <artifactId>jaxb-impl</artifactId>
  <version>2.3.0</version>
</dependency>
<dependency>
  <groupId>javax.activation</groupId>
  <artifactId>javax.activation-api</artifactId>
  <version>1.2.0</version>
</dependency>
```



E o Java FX?

Será que funciona sem configurações a mais?

DEMO

```
java --show-module-resolution
```

UNNAMED MODULE

pode usar (por padrão)

java.se

javafx.base

javafx.controls

javafx.fxml

jdk.net

jdk.management

...

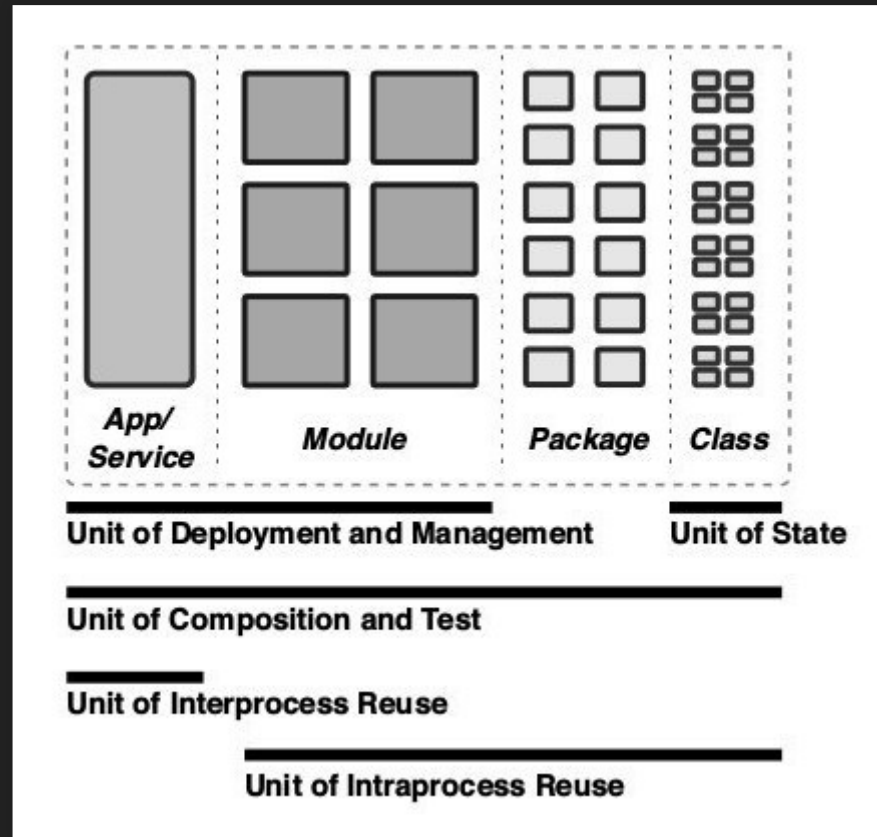
MODULARIZANDO O PROJETO

POR QUE MODULARIZAR?

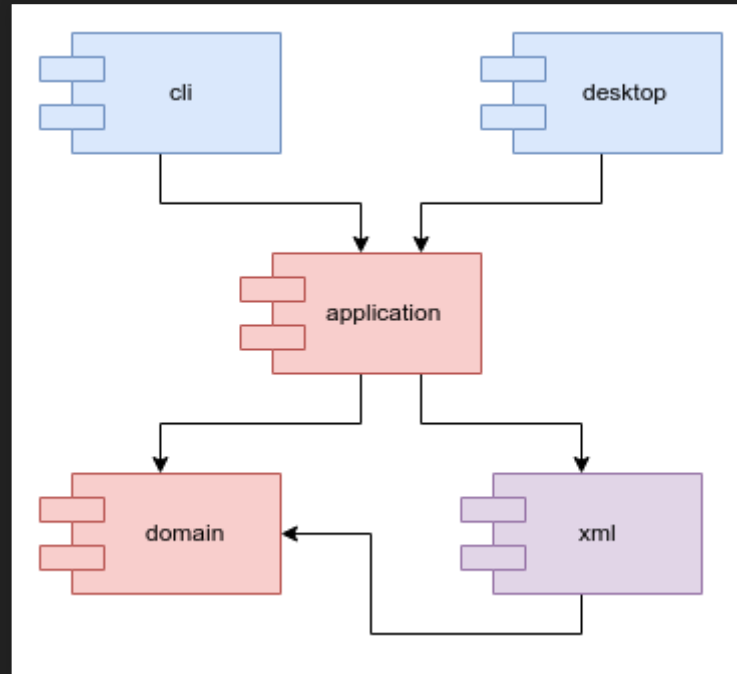
- Para controlar as dependências entre partes do nosso código
- Para tornar a arquitetura explícita (não só um diagrama)

MÓDULOS

- Unidade de implantação
- Unidade de composição
- Unidade de reuso (bibliotecas)



Java Application Architecture: Modularity Patterns, Kirk Knoernschild (2012)



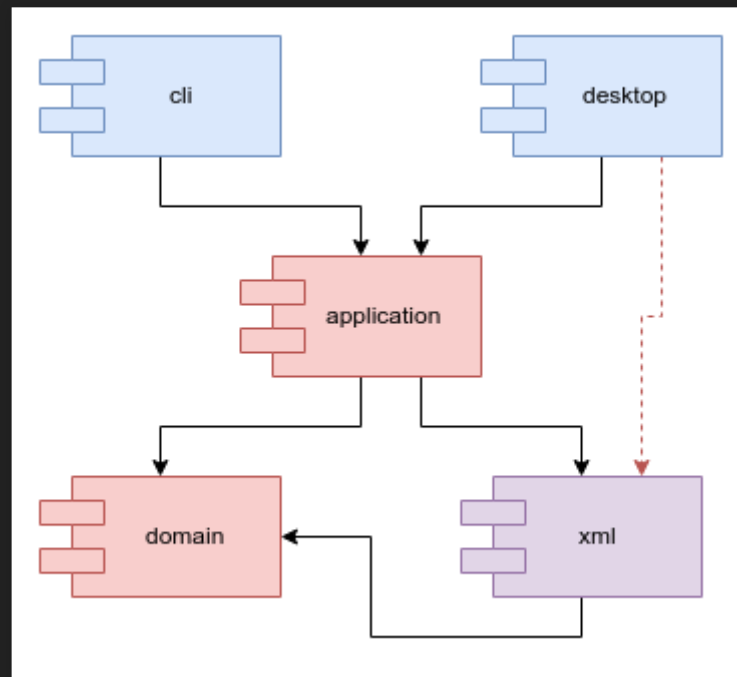
MÓDULOS MAVEN?

DEMO

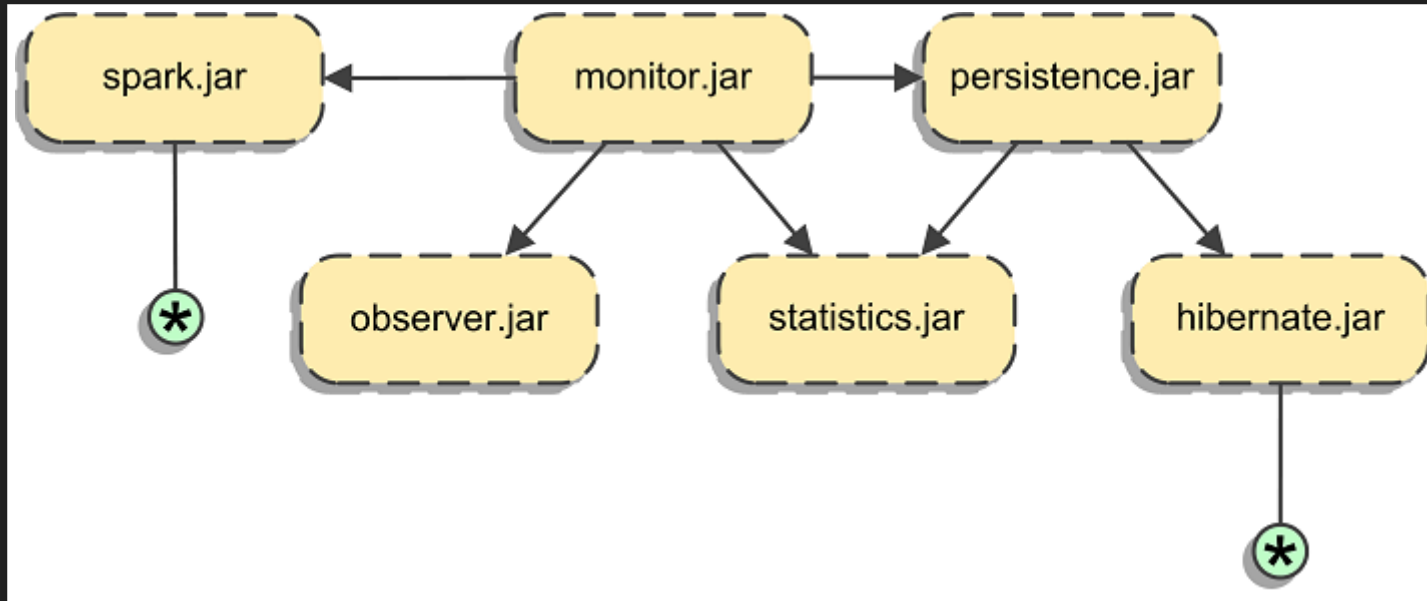
MÓDULOS MAVEN

Funcionam com Java 8 ✓

Encapsulamento ✗

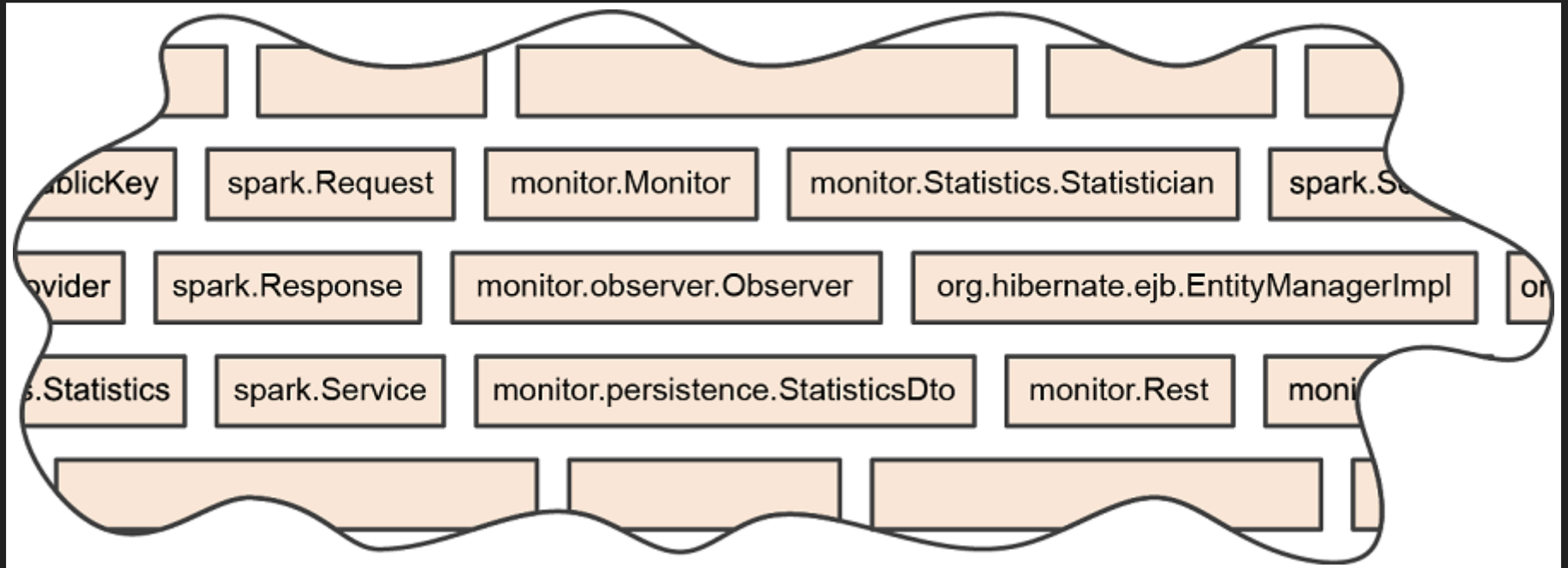


MÓDULOS MAVEN E JARS ☺



The Java Module System, Nicolai Parlog (2018, est.)

CLASS PATH NO RUNTIME ☹️



The Java Module System, Nicolai Parlog (2018, est.)

MODULE ERASURE

- Class Path é apenas uma lista de classes
- JAR é apenas um ZIP
- Dependências entre JARs não são mantidas em runtime

JSR 376/JEP 261: MODULE SYSTEM

module-info.java

```
module br.com.empresa.modulo {  
    exports br.com.empresa.pacote;  
    requires br.com.empresa.outro.modulo;  
}
```

module-info.java

```
module br.com.empresa.modulo {  
    exports br.com.empresa.pacote  
        to br.com.empresa.outro.modulo;  
    requires br.com.empresa.outro.modulo;  
}
```

DEMO

MANIFEST.MF

```
Automatic-Module-Name: br.com.empresa.modulo
```



```
jar --file arquivo.jar --describe-module
```

~~java -cp diretorio ClassePrincipal~~

java -p diretorio --module modulo/ClassePrincipal

OU

java --module-path diretorio
--module modulo/ClassePrincipal

DEPENDÊNCIAS TRANSITIVAS

```
module br.com.empresa.modulo {  
  //...  
  requires transitive br.com.empresa.outro.modulo;  
}
```

REFLECTION

```
open module br.com.empresa.modulo {  
  //...  
}
```

REFLECTION

```
module br.com.empresa.modulo {  
  //...  
  opens br.com.empresa.pacote;  
}
```

REFLECTION

```
module br.com.empresa.modulo {  
  //...  
  opens br.com.empresa.pacote  
    to br.com.empresa.outro.modulo;  
}
```

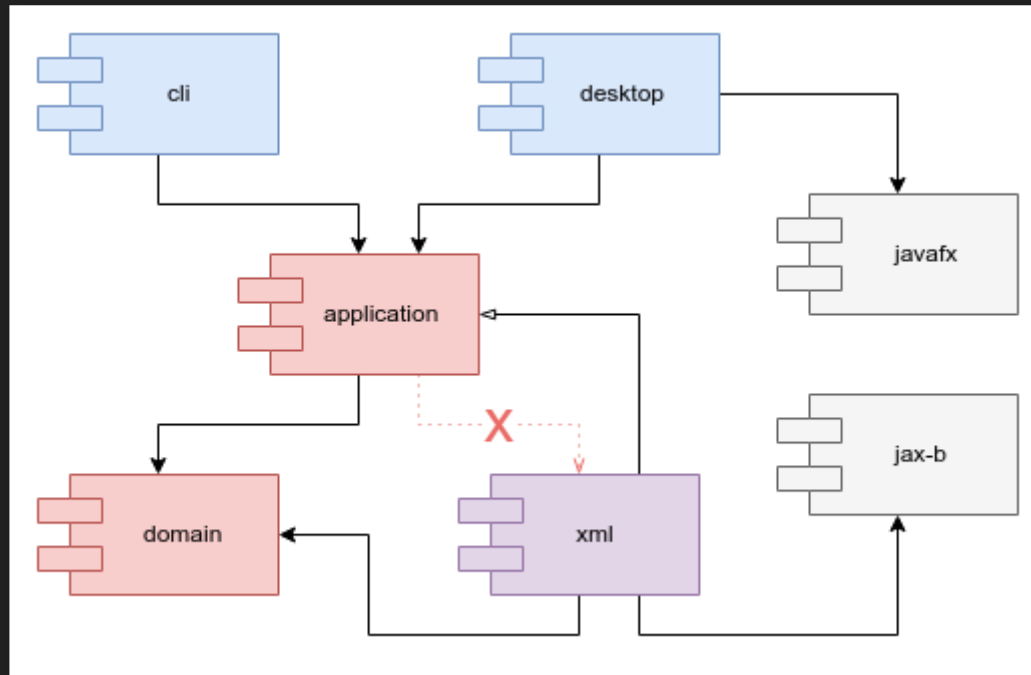
JPMS

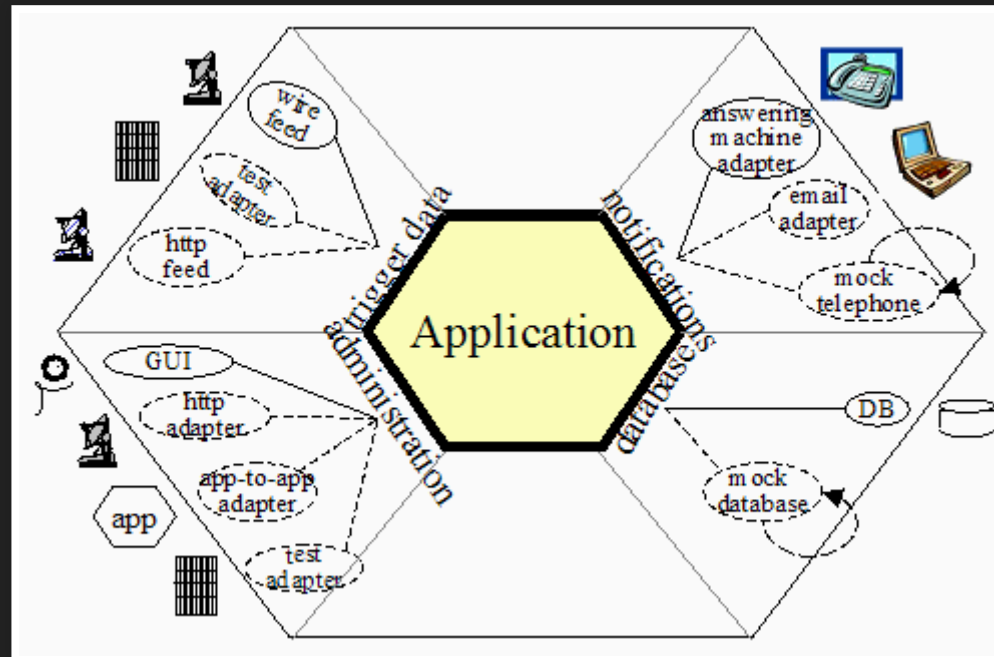
- Encapsulamento forte
- Configuração confiável

A BOA MODELAGEM

- Proteger Regras de Negócio de mudanças
 - na Tela
 - no BD
 - nos frameworks
- Isolar Mecanismos de Entrega

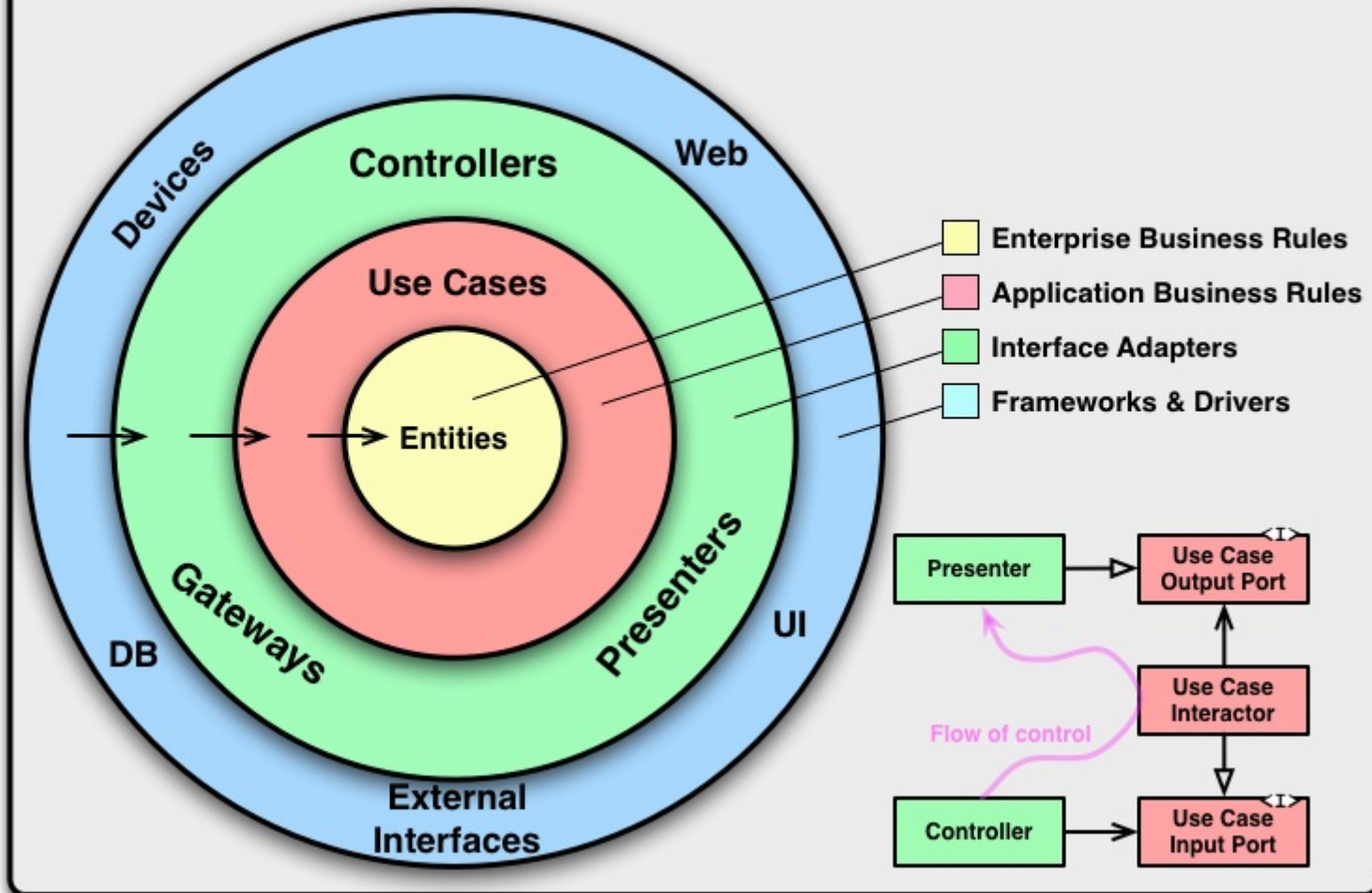
INVERSÃO DE DEPENDÊNCIAS





Hexagonal Architecture, Alistair Cockburn (2005)

The Clean Architecture



The Clean Architecture, Uncle Bob (2012)

VÁRIAS OUTRAS NOVIDADES NO JAVA 9

JEP 282: jlink: The Java Linker

JEP 266: More Concurrency Updates (Reactive Streams)

JEP 248: Make G1 the Default Garbage Collector

JEP 254: Compact Strings `byte[]`

JEP 295: Ahead-of-Time Compilation (Gaal)

JEP 264: Platform Logging API and Service

JEP 259: Stack-Walking API

...

NOVIDADES DO JAVA 10

Java 8: 55 JEPs entregues

Java 9: 91 JEPs entregues

Java 10: 12 JEPs entregues

<http://openjdk.java.net/projects/jdk/10/>

JEP 286: LOCAL-VARIABLE TYPE INFERENCE

DEMO

JEP 322: TIME-BASED RELEASE VERSIONING

- Nova versão a cada 6 meses (Março/Setembro)
- Time-box fechado, escopo aberto

JAVA IMPROVEMENTS FOR DOCKER CONTAINERS

[JDK-8146115](#): Improve docker container detection and resource configuration usage

[JDK-8186248](#): Allow more flexibility in selecting Heap % of available RAM

[JDK-8179498](#): attach in Linux should be relative to /proc/pid/root and namespace aware

DÚVIDAS?

<https://github.com/alexandreaquiles/workjava>