Edwin Maraví
emaravi@cjavaperu.com

CJAVA

Misión

Nuestro equipo trabaja para integrar la tecnología Java en la sociedad como solución a todas sus necesidades.

CJAVA

siempre para apoyarte

Visión

Poder aportar al desarrollo del País usando tecnología Java.

CJAVA
siempre para apoyarte

Servicios Académicos

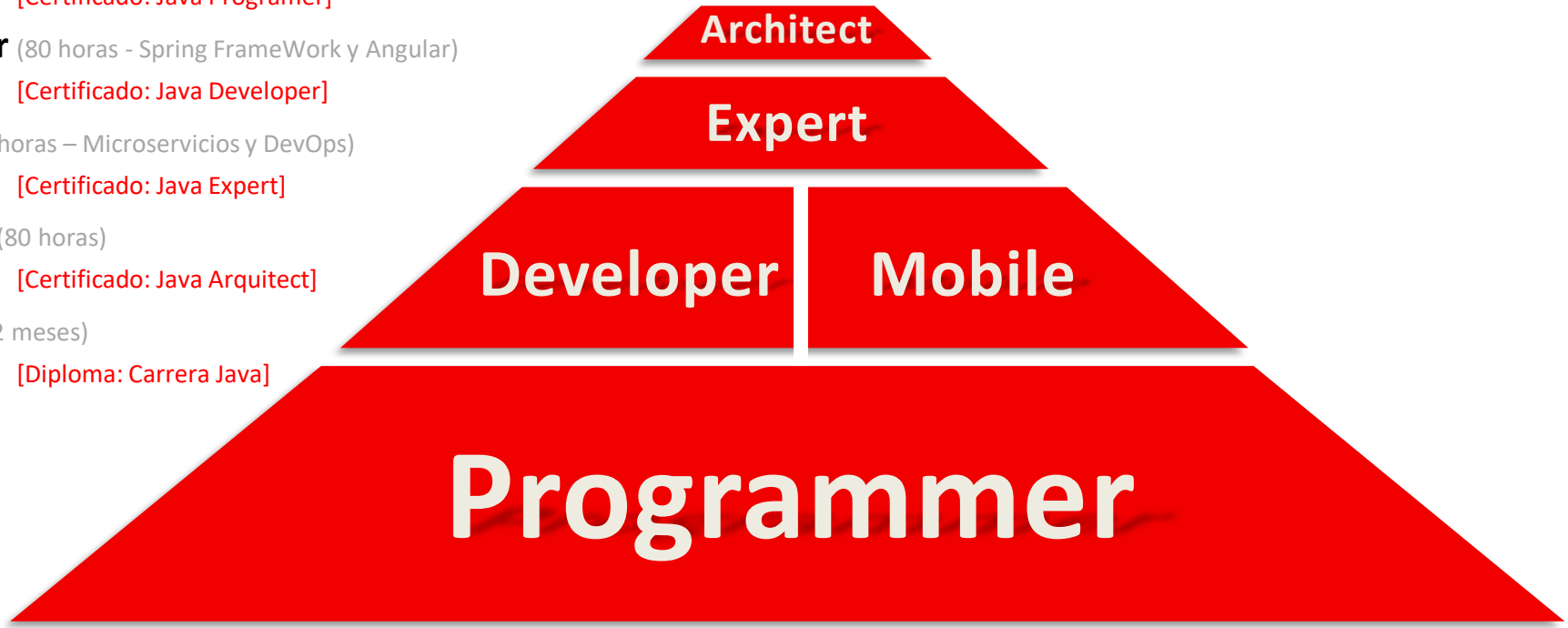
Programer (80 horas - Certificación Java 11)
[Certificado: Java Programer]

Developer (80 horas - Spring FrameWork y Angular)
[Certificado: Java Developer]

Expert (80 horas – Microservicios y DevOps)
[Certificado: Java Expert]

Architect (80 horas)
[Certificado: Java Arquitect]

Carrera (12 meses)
[Diploma: Carrera Java]



CJAVA
siempre para apoyarte

Quienes Somos

Somos una organización orientada a desarrollar, capacitar e investigar tecnología JAVA a través de un prestigioso staff de profesionales a nivel nacional.

Contáctenos

Av. Arenales 395 oficina 405
Santa Beatriz - Lima 01 - Perú

Teléfono: 433-6948

RPC / WhatsApp: 932 656 459

Email: info@cjavaperu.com

Síguenos

/cjava.peru.1

/cjava_peru

/cjavaperu

Manejo de Cadenas

CJAVA
siempre para apoyarte

Objetivos

Al finalizar esta lección, debería estar capacitado para:

- Leer datos de la línea de comandos
- Buscar cadenas
- Analizar cadenas
- Crear cadenas mediante StringBuilder
- Buscar cadenas mediante el uso de expresiones regulares
- Analizar cadenas mediante el uso de expresiones regulares
- Sustituir cadenas mediante el uso de expresiones regulares

Argumentos de línea de comandos

- Todas las aplicaciones de tecnología Java pueden utilizar argumentos de línea de comandos.
- Estos argumentos de cadena se colocan en la línea de comandos para iniciar el intérprete de Java después del nombre de la clase:

```
java TestArgs arg1 arg2 "another arg"
```

- Cada argumento de la línea de comandos se coloca en la matriz args que se transfiere al método main estático:

```
public static void main(String[] args)
```

Argumentos de línea de comandos

```
public class TestArgs {
   public static void main(String[] args) {
     for ( int i = 0; i < args.length; i++ ) {
        System.out.println("args[" + i + "] is '" +
                           args[i] + "'");
     }
   }
}
```

Ejecución de ejemplo:

```
java TestArgs "Ted Baxter" 45 100.25
args[0] is 'Ted Baxter'
args[1] is '45'
args[2] is '100.25'
```

Propiedades

- La clase java.util.Properties se utiliza para cargar y guardar pares clave-valor en Java.
- Se pueden almacenar en un archivo de texto simple

```
hostName = www.example.com
userName = user
password = pass
```

- El nombre del archivo termina en .properties.
- El archivo puede estar en cualquier ubicación en la que el compilador pueda encontrarlo.

CJAVA
siempre para apoyarte

Carga y uso de un archivo de propiedades

```
1  public static void main(String[] args) {
2    Properties myProps = new Properties();
3    try {
4      FileInputStream fis = new FileInputStream("ServerInfo.properties");
5      myProps.load(fis);
6    } catch (IOException e) {
7      System.out.println("Error: " + e.getMessage());
8    }
9
10    // Print Values
11    System.out.println("Server: " + myProps.getProperty("hostName"));
12    System.out.println("User: " + myProps.getProperty("userName"));
13    System.out.println("Password: " + myProps.getProperty("password"));
14 }
```

Carga de propiedades desde la línea de comandos

- La información sobre propiedades también se puede transferir en la línea de comandos.
- Utilice la opción –D para transferir pares clave-valor:

```
java –Dpropertyname=value –Dpropertyname=value myApp
```

- Por ejemplo, transfiera uno de los valores anteriores:

```
java –Dusername=user myApp
```

- Obtenga los datos de Properties del objeto System:

```
String userName = System.getProperty("username");
```

PrintWriter y la consola

Si ya no desea utilizar System.out.println() para imprimir texto en la consola, existe una alternativa.

```
import java.io.PrintWriter;

public class PrintWriterExample {
    public static void main(String[] args){
        PrintWriter pw = new
  PrintWriter(System.out, true);
        pw.println("This is some output.");

    }
}
```

Formato printf

Java proporciona varias opciones para aplicar formato a las cadenas:

- printf y String.format

```
public class PrintfExample {
    public static void main(String[] args){
        PrintWriter pw = new PrintWriter(System.out, true);
        double price = 24.99; int quantity = 2; String color = "Blue";
        System.out.printf("We have %03d %s Polo shirts that cost
   $%3.2f.\n", quantity, color, price);
        System.out.format("We have %03d %s Polo shirts that cost
   $%3.2f.\n", quantity, color, price);
        String out = String.format("We have %03d %s Polo shirts that cost
   $%3.2f.", quantity, color, price);
        System.out.println(out);
        pw.printf("We have %03d %s Polo shirts that cost $%3.2f.\n",
   quantity, color, price);
    }
}
```

CJAVA
siempre para apoyarte

Procesamiento de cadenas

- StringBuilder para construir la cadena
- Métodos de cadena incorporados
  - Búsqueda
  - Análisis
  - Extracción de subcadenas
- Análisis con StringTokenizer

CJAVA
siempre para apoyarte

StringBuilder y StringBuffer

- StringBuilder y StringBuffer son las herramientas preferidas cuando la concatenación de cadenas no es trivial.
  - Más eficaces que “+”
- Simultaneidad
  - StringBuilder (sin protección de thread)
  - StringBuffer (con protección de thread)
- Definición de la capacidad con el tamaño realmente necesario.
  - El cambio continuo de tamaño del buffer también puede producir problemas de rendimiento.

StringBuilder: ejemplo

```
public class StringBuilding {
    public static void main(String[] args){
        StringBuilder sb = new StringBuilder(500);

        sb.append(", the lightning flashed and the thunder
   rumbled.\n");
        sb.insert(0, "It was a dark and stormy night");

        sb.append("The lightning struck...\n").append("[  ");
        for(int i = 1; i < 11; i++){
            sb.append(i).append("  ");
        }
        sb.append("] times");

        System.out.println(sb.toString());
    }
}
```

CJAVA
siempre para apoyarte

Métodos de cadena de ejemplo

```
1  public class StringMethodsExample {
2      public static void main(String[] args){
3          PrintWriter pw = new PrintWriter(System.out, true);
4          String tc01 = "It was the best of times";
5          String tc02 = "It was the worst of times";
6
7          if (tc01.equals(tc02)){
8              pw.println("Strings match..."); }
9          if (tc01.contains("It was")){
10             pw.println("It was found"); }
11         String temp = tc02.replace("w", "zw");
12         pw.println(temp);
13         pw.println(tc02.substring(5, 12));
14     }
15 }
```

CJAVA
siempre para apoyarte

Uso del método split()

```
1  public class StringSplit {
2      public static void main(String[] args){
3          String shirts = "Blue Shirt, Red Shirt, Black
   Shirt, Maroon Shirt";
4
5          String[] results = shirts.split(", ");
6          for(String shirtStr:results){
7              System.out.println(shirtStr);
8          }
9      }
10 }
```

CJAVA
siempre para apoyarte

Análisis con StringTokenizer

```
1  public class StringTokenizerExample {
2      public static void main(String[] args){
3          String shirts = "Blue Shirt, Red Shirt, Black Shirt,
   Maroon Shirt";
4
5          StringTokenizer st = new StringTokenizer(shirts, ", ");
6
7          while(st.hasMoreTokens()){
8              System.out.println(st.nextToken());
9          }
10     }
11 }
```

Scanner

Una clase Scanner puede convertir en un token una cadena o un flujo.

```
1       public static void main(String[] args) {
2           Scanner s = null;
3           StringBuilder sb = new StringBuilder(64);
4           String line01 = "1.1, 2.2, 3.3";
5           float fsum = 0.0f;
6
7           s = new Scanner(line01).useDelimiter(", ");
8           try {
9               while (s.hasNextFloat()) {
10                  float f = s.nextFloat();
11                  fsum += f;
12                  sb.append(f).append(" ");
13              }
14              System.out.println("Values found: " + sb.toString());
15              System.out.println("FSum: " + fsum);
16          } catch (Exception e) {
17              System.out.println(e.getMessage());
18          }
```

CJAVA
siempre para apoyarte

Expresiones regulares

- Lenguaje para coincidencias de cadenas de texto
  - Vocabulario muy detallado
  - Búsqueda, extracción o búsqueda y sustitución
- Con Java, el uso de la barra invertida (\) es importante.
- Objetos Java
  - Pattern
  - Matcher
  - PatternSyntaxException
  - java.util.regex

CJAVA
siempre para apoyarte

Pattern y Matcher

- Pattern: define una expresión regular
- Matcher: especifica una cadena de búsqueda

```
1  import java.util.regex.Matcher;
2  import java.util.regex.Pattern;
3
4  public class PatternExample {
5      public static void main(String[] args){
6          String t = "It was the best of times";
7
8          Pattern pattern = Pattern.compile("the");
9          Matcher matcher = pattern.matcher(t);
10
11         if (matcher.find()) { System.out.println("Found match!");
   }
12     }
13 }
```

CJAVA
siempre para apoyarte

Clases de caracteres

Carácter	Descripción
.	Coincide con cualquier carácter único (letra, dígito o carácter especial), salvo marcadores de final de línea.
`[abc]`	Coincidiría con “a”, “b” o “c” en esa posición.
`[^abc]`	Coincidiría con cualquier carácter que no fuera “a”, “b” o “c” en esa posición.
`[a-c]`	Rango de caracteres (en este caso, “a”, “b” y “c”).
|	Alternancia; básicamente un indicador “or”.

CJAVA
siempre para apoyarte

Clase de caracteres: ejemplos

Cadena de destino	It was the best of times

Patrón	Descripción	Texto de coincidencia
`w.s`	Cualquier secuencia que empiece por “w” seguida de cualquier carácter seguido de “s”.	It was the best of times
`w[abc]s`	Cualquier secuencia que empiece por “w” seguida de “a”, “b” o “c” y, a continuación, “s”.	It was the best of times
`t[^aeo]mes`	Cualquier secuencia que empiece por “t” seguida de cualquier carácter que no sea “a”, “e” u “o” seguida de “mes”.	It was the best of times

Patrones: ejemplos

```
1  public class CustomCharClassExamples {
2      public static void main(String[] args) {
3          String t = "It was the best of times";
4
5          Pattern p1 = Pattern.compile("w.s");
6          Matcher m1 = p1.matcher(t);
7          if (m1.find()) { System.out.println("Found: " + m1.group());
8          }
9
10         Pattern p2 = Pattern.compile("w[abc]s");
11         Matcher m2 = p2.matcher(t);
12         if (m2.find()) { System.out.println("Found: " + m2.group());
13         }
14
15         Pattern p3 = Pattern.compile("t[^eou]mes");
16         Matcher m3 = p3.matcher(t);
17         if (m3.find()) { System.out.println("Found: " + m3.group());
18         }
```

CJAVA
siempre para apoyarte

Clases de caracteres predefinidas

Carácter predefinido	Clase de caracteres	Carácter negado	Clase negada
\d (digit)	[0-9]	\D	[^0-9]
\w (word char)	[a-zA-Z0-9_]	\W	[^a-zA-Z0-9_]
\s (white space)	[\r\t\n\f\0XB]	\S	[^ \r\t\n\f\0XB]

Clases de caracteres predefinidas: ejemplos

Cadena de destino	Jo told me 20 ways to San Jose in 15 minutes.

Patrón	Descripción	Texto de coincidencia
`\\d\\d`	Buscar dos dígitos.**	Jo told me 20 ways to San Jose in 15 minutes.
`\\sin\\s`	Buscar “in” entre dos espacios y, a continuación, los tres caracteres siguientes.	Jo told me 20 ways to San Jose in 15 minutes.
`\\Sin\\S`	Buscar “in” entre dos caracteres que no sean de espacio y, a continuación, los tres caracteres siguientes.	Jo told me 20 ways to San Jose in 15 minutes.

CJAVA
siempre para apoyarte

Cuantificadores

Cuantificador	Descripción
`*`	El carácter precedente se repite cero o más veces.
`+`	El carácter precedente se repite una o más veces.
`?`	El carácter precedente debe aparecer una vez o ninguna.
`{n}`	El carácter precedente aparece exactamente *n* veces.
`{m,n}`	El carácter precedente aparece de *m* a *n* veces.
`{m,}`	El carácter precedente aparece *m* o más veces.
`(xx){n}`	Este grupo de caracteres se repite *n* veces.

CJAVA
siempre para apoyarte

Cuantificador: ejemplos

Cadena de destino	Longlonglong ago, in a galaxy far far away

Patrón	Descripción	Texto de coincidencia
ago.*	Buscar “ago” y 0 o todos los caracteres restantes en la línea.	Longlonglong ago, in a galaxy far far away
gal.{3}	Coincidir con “gal” y los tres caracteres siguientes. Esto sustituye a “…” como se utiliza en un ejemplo anterior.	Longlonglong ago, in a galaxy far far away
(long){2}	Buscar “long” repetido dos veces.	Longlonglong ago, in a galaxy far far away

Voracidad

- Una expresión regular intenta recuperar siempre tantos caracteres como sea posible.
- Utilice el operador ? para limitar la búsqueda al menor número de coincidencias posible.

Cadena de destino	**Longlonglong ago, in a galaxy far far away.**

Patrón	Descripción	Texto de coincidencia
`ago.*far`	Una expresión regular recupera siempre el mayor número de caracteres posible.	Longlonglong ago, in a galaxy far far away.
`ago.*?far`	El carácter “?” básicamente desactiva la voracidad.	Longlonglong ago, in a galaxy far far away.

CJAVA
siempre para apoyarte

Coincidencias de límite

Fijación	Descripción
`^`	Coincide con el principio de una línea.
`$`	Coincide con el final de una línea.
`\b`	Coincide con el inicio o el final de una palabra.
`\B`	*No* coincide con el principio o el final de una palabra.

Límite: ejemplos

Cadena de destino	it was the best of times or it was the worst of times

Patrón	Descripción	Texto de coincidencia
`^it.*?times`	Secuencia que empieza una línea con “it” seguido de algunos caracteres y “times”, con la voracidad desactivada.	it was the best of times or it was the worst of times
`\\sit.*times$`	Secuencia que empieza con “it” seguido de algunos caracteres y termina la línea con “times”.	it was the best of times or it was the worst of times
`\\bor\\b.{3}`	Buscar “or” entre límites de palabras y los tres caracteres siguientes.	it was the best of times or it was the worst of times

CJAVA
siempre para apoyarte

Coincidencia y grupos

Cadena de destino	**george.washington@example.com**

Coincidencia de 3 partes	**(george).(washington)@(example.com)**

Números de grupo	**(1).(2)@(3)**

Patrón	**(\\S+?)\\.(\\S+?)@(\\S+)**

CJAVA
siempre para apoyarte

Uso del método replaceAll

Con el método replaceAll puede buscar y sustituir elementos.

```
public class ReplacingExample {
    public static void main(String[] args){
        String header = "<h1>This is an H1</h1>";

        Pattern p1 = Pattern.compile("h1");
        Matcher m1 = p1.matcher(header);
        if (m1.find()){
            header = m1.replaceAll("p");
            System.out.println(header);
        }
    }
}
```

CJAVA
siempre para apoyarte

Resumen

En esta lección, debe haber aprendido a hacer lo siguiente:

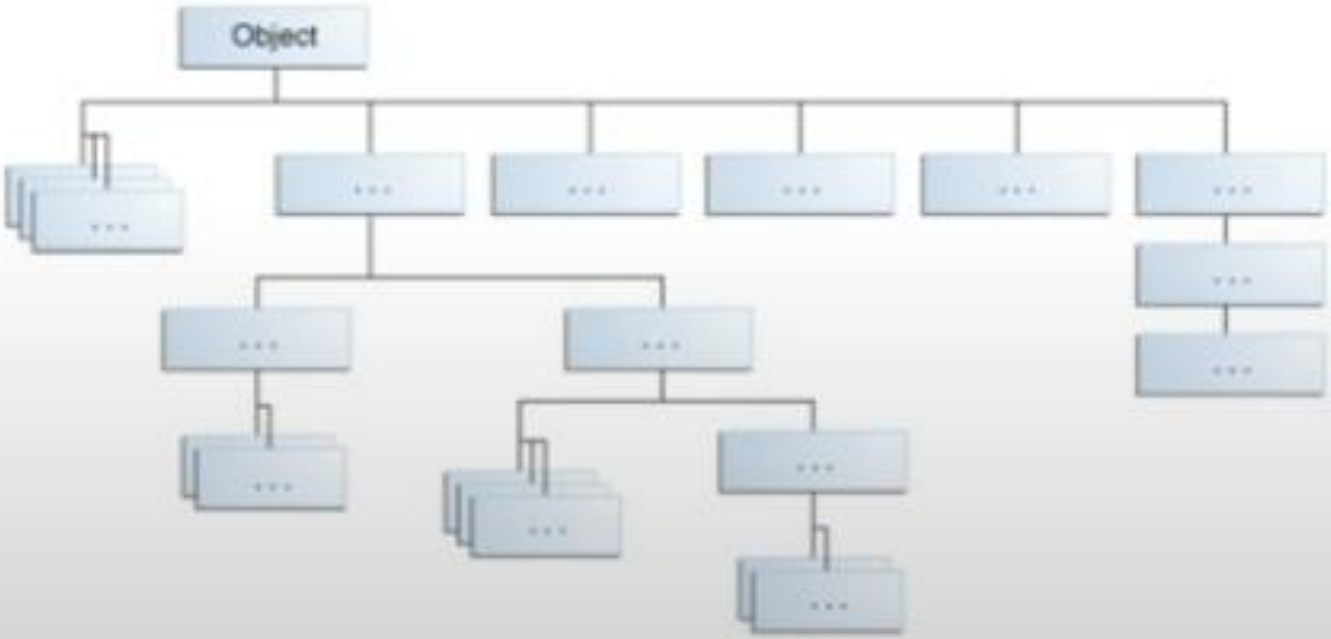
- Leer datos de la línea de comandos
- Buscar cadenas
- Analizar cadenas
- Crear cadenas mediante StringBuilder
- Buscar cadenas mediante el uso de expresiones regulares
- Analizar cadenas mediante el uso de expresiones regulares
- Sustituir cadenas mediante el uso de expresiones regulares

Modulos con JAVA

Edwin Maravi

Desde Java 9

Los programas son clases



CJAVA
siempre para apoyarte

El infierno del ClassPath

```
common/hadoop-common-3.0.0-SNAPSHOT.jar:common/hadoop-nfs-3.0.0-SNAPSHOT.jar:common/lib/activation-1.1.jar:common/lib/apacheds-i\
18n-2.0.0-M15.jar:common/lib/apacheds-kerberos-codec-2.0.0-M15.jar:common/lib/api-asn1-api-1.0.0-M20.jar:common/lib/api-util-1.0\
.0-M20.jar:common/lib/asm-3.2.jar:common/lib/avro-1.7.4.jar:common/lib/commons-beanutils-1.7.0.jar:common/lib/commons-beanutils-\
core-1.8.0.jar:common/lib/commons-cli-1.2.jar:common/lib/commons-codec-1.4.jar:common/lib/commons-collections-3.2.1.jar:common/l\
ib/commons-compress-1.4.1.jar:common/lib/commons-configuration-1.6.jar:common/lib/commons-digester-1.8.jar:common/lib/commons-ht\
tpclient-3.1.jar:common/lib/commons-io-2.4.jar:common/lib/commons-lang-2.6.jar:common/lib/commons-logging-1.1.3.jar:common/lib/c\
ommons-math3-3.1.1.jar:common/lib/commons-net-3.1.jar:common/lib/curator-client-2.7.1.jar:common/lib/curator-framework-2.7.1.jar\
:common/lib/curator-recipes-2.7.1.jar:common/lib/gson-2.2.4.jar:common/lib/guava-11.0.2.jar:common/lib/hadoop-annotations-3.0.0-\
SNAPSHOT.jar:common/lib/hadoop-auth-3.0.0-SNAPSHOT.jar:common/lib/hamcrest-core-1.3.jar:common/lib/htrace-core4-4.0.1-incubating\
.jar:common/lib/httpclient-4.2.5.jar:common/lib/httpcore-4.2.5.jar:common/lib/jackson-core-asl-1.9.13.jar:common/lib/jackson-jax\
rs-1.9.13.jar:common/lib/jackson-mapper-asl-1.9.13.jar:common/lib/jackson-xc-1.9.13.jar:common/lib/java-xmlbuilder-0.4.jar:commo\
n/lib/jaxb-api-2.2.2.jar:common/lib/jaxb-impl-2.2.3-1.jar:common/lib/jcip-annotations-1.0.jar:common/lib/jersey-core-1.9.jar:com\
mon/lib/jersey-json-1.9.jar:common/lib/jersey-server-1.9.jar:common/lib/jets3t-0.9.0.jar:common/lib/jettison-1.1.jar:common/lib/\
jetty-6.1.26.jar:common/lib/jetty-util-6.1.26.jar:common/lib/jsch-0.1.51.jar:common/lib/json-smart-1.1.1.jar:common/lib/jsp-api-\
2.1.jar:common/lib/jsr305-3.0.0.jar:common/lib/junit-4.11.jar:common/lib/log4j-1.2.17.jar:common/lib/mockito-all-1.8.5.jar:commo\
n/lib/netty-3.6.2.Final.jar:common/lib/nimbus-jose-jwt-3.9.jar:common/lib/paranamer-2.3.jar:common/lib/protobuf-java-2.5.0.jar:c\
ommon/lib/servlet-api-2.5.jar:common/lib/slf4j-api-1.7.10.jar:common/lib/slf4j-log4j12-1.7.10.jar:common/lib/snappy-java-1.0.4.1\
.jar:common/lib/stax-api-1.0-2.jar:common/lib/xmlenc-0.52.jar:common/lib/xz-1.0.jar:common/lib/zookeeper-3.4.6.jar:hdfs/hadoop-h\
dfs-3.0.0-SNAPSHOT.jar:hdfs/hadoop-hdfs-nfs-3.0.0-SNAPSHOT.jar:hdfs/lib/commons-daemon-1.0.13.jar:hdfs/lib/hadoop-hdfs-client-3.\
0.0-SNAPSHOT.jar:hdfs/lib/hpack-0.11.0.jar:hdfs/lib/leveldbjni-all-1.8.jar:hdfs/lib/netty-all-4.1.0.Beta5.jar:hdfs/lib/okhttp-2.\
4.0.jar:hdfs/lib/okio-1.4.0.jar:hdfs/lib/xercesImpl-2.9.1.jar:mapreduce/hadoop-mapreduce-client-app-3.0.0-SNAPSHOT.jar:mapreduce\
/hadoop-mapreduce-client-common-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-client-core-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-ma\
preduce-client-hs-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-client-hs-plugins-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-\
client-jobclient-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-client-nativetask-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-c\
lient-shuffle-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-examples-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-api-3.0.0-SNAPSHOT.jar:\
yarn/hadoop-yarn-applications-distributedshell-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-applications-unmanaged-am-launcher-3.0.0-SNAP\
SHOT.jar:yarn/hadoop-yarn-client-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-common-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-registry-3.0.0-S\
NAPSHOT.jar:yarn/hadoop-yarn-server-applicationhistoryservice-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-server-common-3.0.0-SNAPSHOT.j\
ar:yarn/hadoop-yarn-server-nodemanager-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-server-resourcemanager-3.0.0-SNAPSHOT.jar:yarn/hadoop\
-yarn-server-sharedcachemanager-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-server-web-proxy-3.0.0-SNAPSHOT.jar:yarn/lib/aopalliance-1.0\
.jar:yarn/lib/commons-math-2.2.jar:yarn/lib/curator-test-2.7.1.jar:yarn/lib/fst-2.24.jar:yarn/lib/guice-3.0.jar:yarn/lib/guice-s\
ervlet-3.0.jar:yarn/lib/javassist-3.18.1-GA.jar:yarn/lib/javax.inject-1.jar:yarn/lib/jersey-client-1.9.jar:yarn/lib/jersey-guice\
```

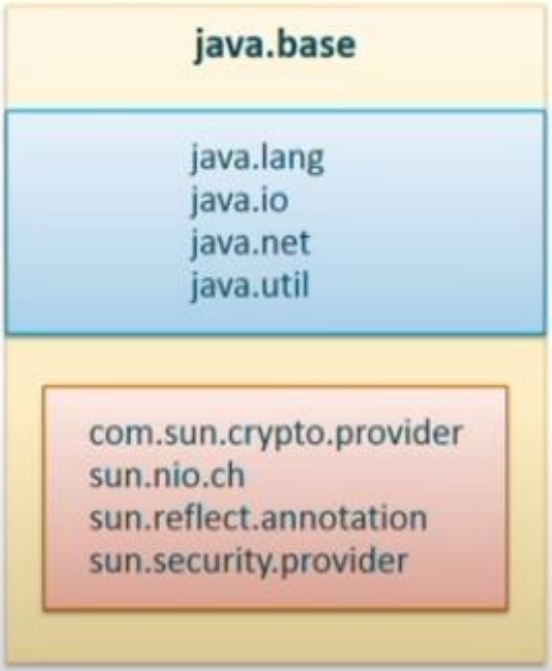
CJAVA
siempre para apoyarte

Los programas son ~~clases~~ paquetes

Un Módulo es un conjunto de paquetes diseñados para reutilización.

CJAVA
siempre para apoyarte

Java.base




```
// module-info.java
module java.base {
  exports java.lang;
  exports java.io;
  exports java.net;
  exports java.util;
}
```

CJAVA
siempre para apoyarte

Accesibilidad

Accessibility (JDK 1 – JDK 8)

- public
- protected
- package
- private

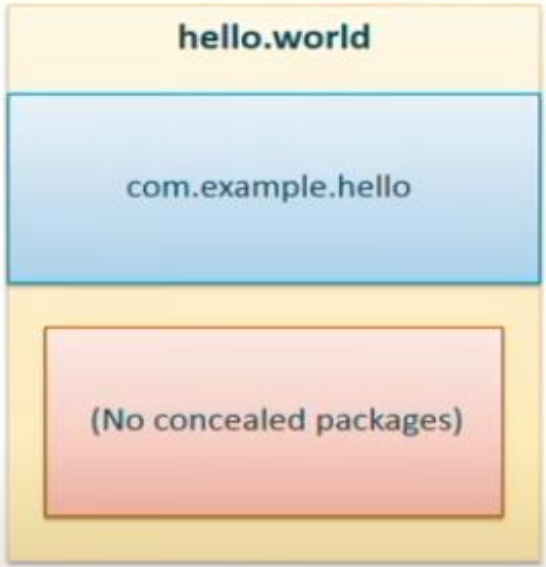
Accessibility (JDK 9)

- *public to everyone*
- *public but only to friend modules*
- *public only within a module*
- protected
- package
- private

Ejemplo

"Hello world" as a Module

```
// src/com/example/hello/SayHello.java
package com.example.hello;
import java.lang.*;
public class SayHello {
  public static void main(String[] args) {
    System.out.println("Hello world");
  }
}

// src/module-info.java
module hello.world {
  exports com.example.hello;
}
```

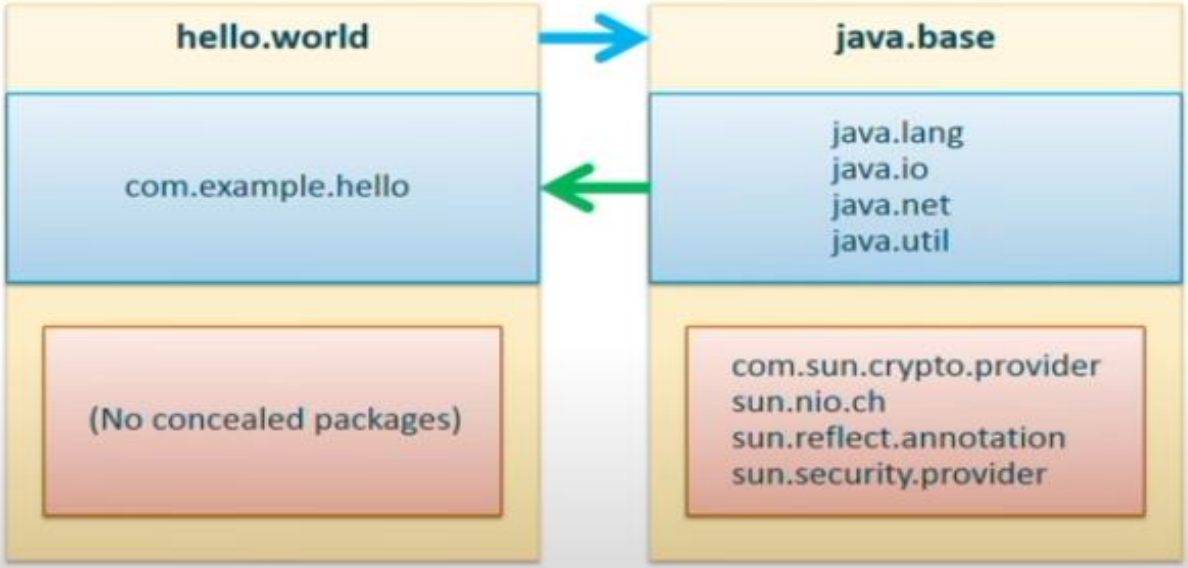



```
javac -d classes -sourcepath src
      module-info.java
      com/example/hello/SayHello.java
```

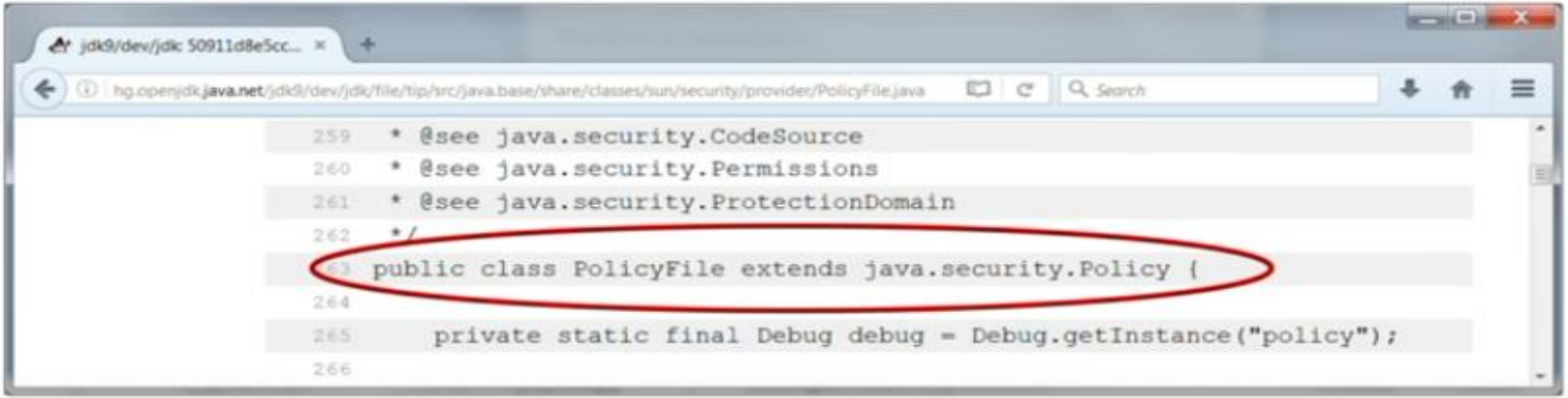
Reusando un módulo

Reusing a Module

```
// src/module-info.java
module hello.world {
  exports  com.example.hello;
  requires java.base;
}
```



Errores en el camino de la ejecución.



CJAVA
siempre para apoyarte

Reusando un Modulo

```
// src/module-info.java
module hello.world {
  exports  com.example.hello;
  requires java.base;
}
```

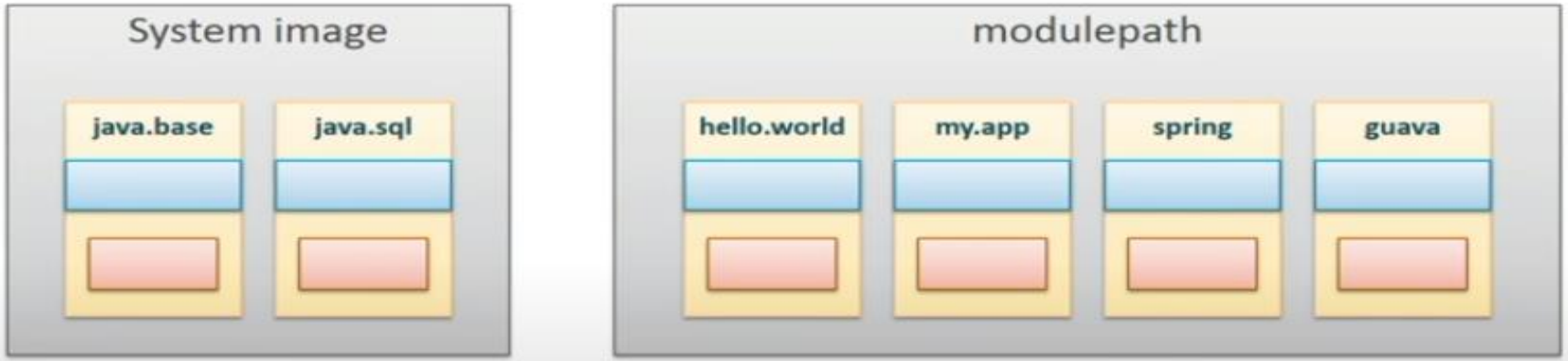
Comunicación entre módulos

> "The unit of reuse is the unit of release."

```
// src/module-info.java
module hello.world {
  exports  com.example.hello;
  requires java.base;
}
```

Reusando un Modulo

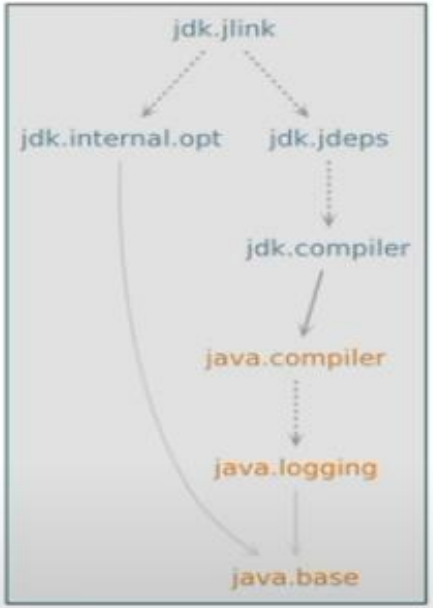
Running a Modular Application




```
java -p mods -m hello.world
```

Reusando un Modulo

Running a Modular Application

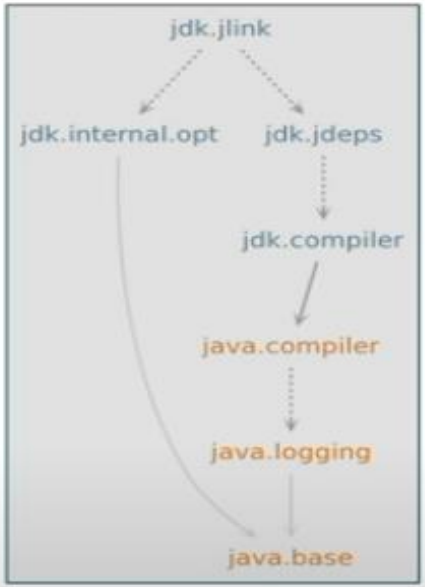



```
java -p mods -m hello.world
```

Reusando un Modulo

Running a Modular Application

- No missing dependencies
- No cyclic dependencies
- No split packages




```
java -p mods -m hello.world
```

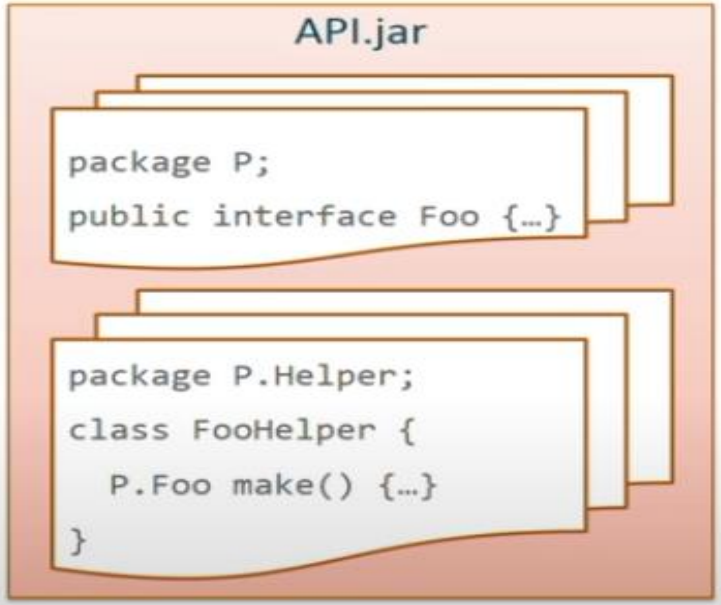
CJAVA
siempre para apoyarte

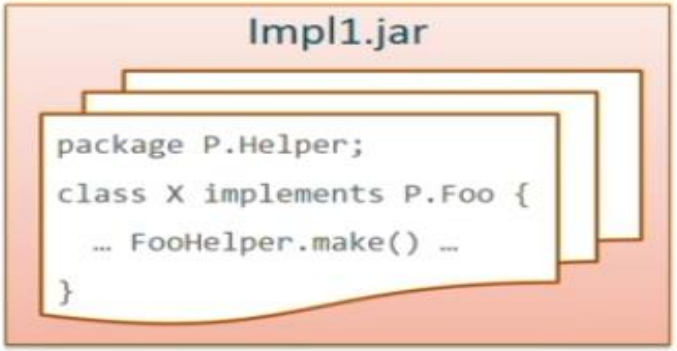
Reusando un Modulo

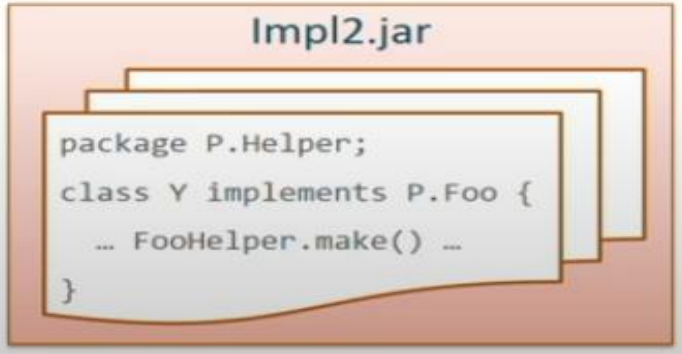
```
common/hadoop-common-3.0.0-SNAPSHOT.jar:common/hadoop-nfs-3.0.0-SNAPSHOT.jar:common/lib/activation-1.1.jar:common/lib/apacheds-i\
18n-2.0.0-M15.jar:common/lib/apacheds-kerberos-codec-2.0.0-M15.jar:common/lib/api-asn1-api-1.0.0-M20.jar:common/lib/api-util-1.0\
.0-M20.jar:common/lib/asm-3.2.jar:common/lib/avro-1.7.4.jar:common/lib/commons-beanutils-1.7.0.jar:common/lib/commons-beanutils-\
core-1.8.0.jar:common/lib/commons-cli-1.2.jar:common/lib/commons-codec-1.4.jar:common/lib/commons-collections-3.2.1.jar:common/l\
ib/commons-compress-1.4.1.jar:common/lib/commons-configuration-1.6.jar:common/lib/commons-digester-1.8.jar:common/lib/commons-ht\
tpclient-3.1.jar:common/lib/commons-io-2.4.jar:common/lib/commons-lang-2.6.jar:common/lib/commons-logging-1.1.3.jar:common/lib/c\
ommons-math3-3.1.1.jar:common/lib/commons-net-3.1.jar:common/lib/curator-client-2.7.1.jar:common/lib/curator-framework-2.7.1.jar\
:common/lib/curator-recipes-2.7.1.jar:common/lib/gson-2.2.4.jar:common/lib/guava-11.0.2.jar:common/lib/hadoop-annotations-3.0.0-\
SNAPSHOT.jar:common/lib/hadoop-auth-3.0.0-SNAPSHOT.jar:common/lib/hamcrest-core-1.3.jar:common/lib/htrace-core4-4.0.1-incubating\
.jar:common/lib/httpclient-4.2.5.jar:common/lib/httpcore-4.2.5.jar:common/lib/jackson-core-asl-1.9.13.jar:common/lib/jackson-jax\
rs-1.9.13.jar:common/lib/jackson-mapper-asl-1.9.13.jar:common/lib/jackson-xc-1.9.13.jar:common/lib/java-xmlbuilder-0.4.jar:commo\
n/lib/jaxb-api-2.2.2.jar:common/lib/jaxb-impl-2.2.3-1.jar:common/lib/jcip-annotations-1.0.jar:common/lib/jersey-core-1.9.jar:com\
mon/lib/jersey-json-1.9.jar:common/lib/jersey-server-1.9.jar:common/lib/jets3t-0.9.0.jar:common/lib/jettison-1.1.jar:common/lib/\
jetty-6.1.26.jar:common/lib/jetty-util-6.1.26.jar:common/lib/jsch-0.1.51.jar:common/lib/json-smart-1.1.1.jar:common/lib/jsp-api-\
2.1.jar:common/lib/jsr305-3.0.0.jar:common/lib/junit-4.11.jar:common/lib/log4j-1.2.17.jar:common/lib/mockito-all-1.8.5.jar:commo\
n/lib/netty-3.6.2.Final.jar:common/lib/nimbus-jose-jwt-3.9.jar:common/lib/paranamer-2.3.jar:common/lib/protobuf-java-2.5.0.jar:c\
ommon/lib/servlet-api-2.5.jar:common/lib/slf4j-api-1.7.10.jar:common/lib/slf4j-log4j12-1.7.10.jar:common/lib/snappy-java-1.0.4.1\
.jar:common/lib/stax-api-1.0-2.jar:common/lib/xmlenc-0.52.jar:common/lib/xz-1.0.jar:common/lib/zookeeper-3.4.6.jar:hdfs/hadoop-h\
dfs-3.0.0-SNAPSHOT.jar:hdfs/hadoop-hdfs-nfs-3.0.0-SNAPSHOT.jar:hdfs/lib/commons-daemon-1.0.13.jar:hdfs/lib/hadoop-hdfs-client-3.\
0.0-SNAPSHOT.jar:hdfs/lib/hpack-0.11.0.jar:hdfs/lib/leveldbjni-all-1.8.jar:hdfs/lib/netty-all-4.1.0.Beta5.jar:hdfs/lib/okhttp-2.\
4.0.jar:hdfs/lib/okio-1.4.0.jar:hdfs/lib/xercesImpl-2.9.1.jar:mapreduce/hadoop-mapreduce-client-app-3.0.0-SNAPSHOT.jar:mapreduce\
/hadoop-mapreduce-client-common-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-client-core-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-ma\
preduce-client-hs-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-client-hs-plugins-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-\
client-jobclient-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-client-nativetask-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-c\
lient-shuffle-3.0.0-SNAPSHOT.jar:mapreduce/hadoop-mapreduce-examples-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-api-3.0.0-SNAPSHOT.jar:\
yarn/hadoop-yarn-applications-distributedshell-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-applications-unmanaged-am-launcher-3.0.0-SNAP\
SHOT.jar:yarn/hadoop-yarn-client-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-common-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-registry-3.0.0-S\
NAPSHOT.jar:yarn/hadoop-yarn-server-applicationhistoryservice-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-server-common-3.0.0-SNAPSHOT.j\
ar:yarn/hadoop-yarn-server-nodemanager-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-server-resourcemanager-3.0.0-SNAPSHOT.jar:yarn/hadoop\
-yarn-server-sharedcachemanager-3.0.0-SNAPSHOT.jar:yarn/hadoop-yarn-server-web-proxy-3.0.0-SNAPSHOT.jar:yarn/lib/aopalliance-1.0\
.jar:yarn/lib/commons-math-2.2.jar:yarn/lib/curator-test-2.7.1.jar:yarn/lib/fst-2.24.jar:yarn/lib/guice-3.0.jar:yarn/lib/guice-s\
ervlet-3.0.jar:yarn/lib/javassist-3.18.1-GA.jar:yarn/lib/javax.inject-1.jar:yarn/lib/jersey-client-1.9.jar:yarn/lib/jersey-guice\
-1.9.jar:yarn/lib/objenesis-2.1.jar
```

Paquetes separados en el mundo Real

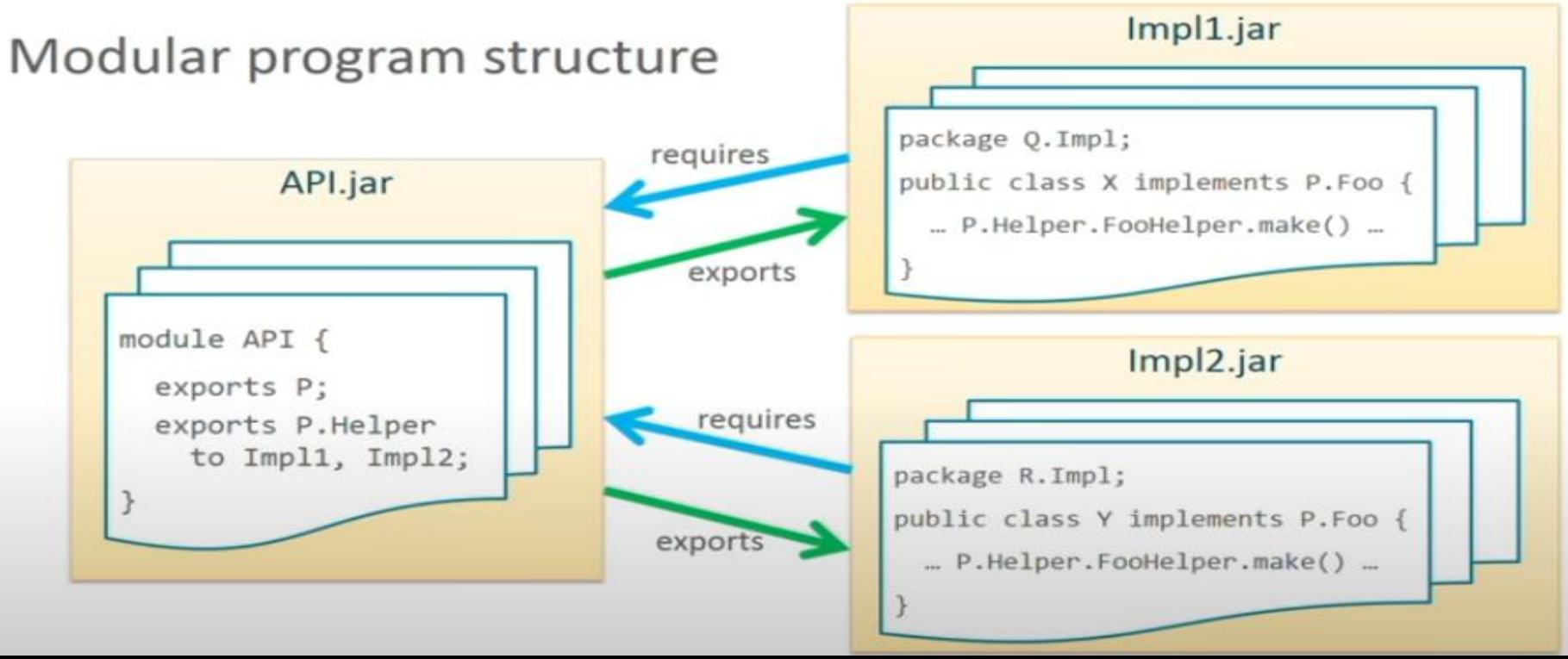
Manejando Paquetes separados







Así funciona



Que tenemos ?

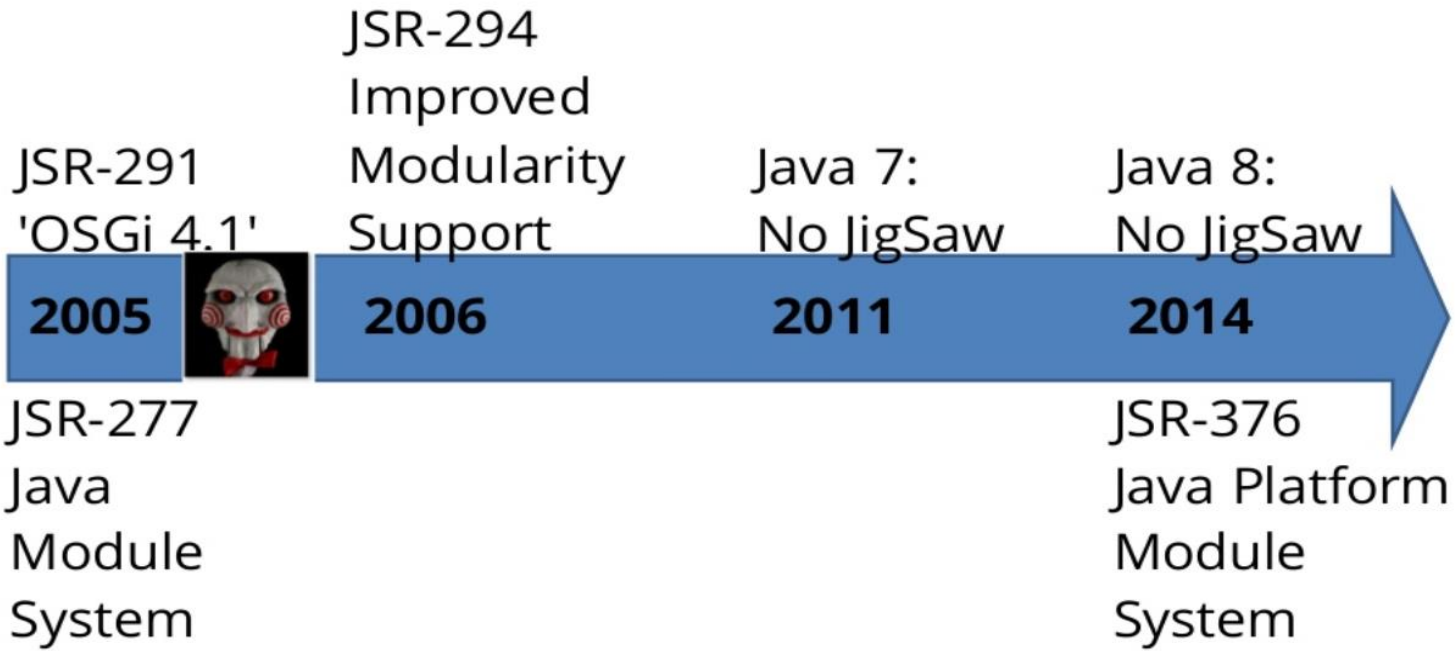
OSGi Java EE™

Runtime

Compile time

ivy Maven™ Gradle

Proyecto JigSaw



CJAVA
siempre para apoyarte

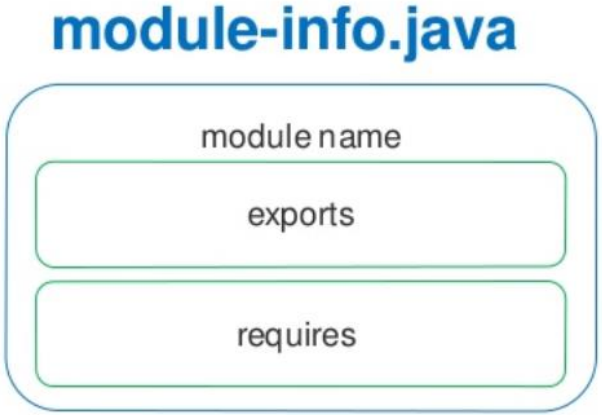
Por que Módulos ??

- Reliable **configuration** -Compile + Runtime
  +
- Strong **encapsulation** -Cleaner, more logical designs

- Leads to:
  - Scalable Java platform -95 Modules
  - Greater platform integrity -sun.misc.Unsafe
  - Improved performance -Lazy loading

Definición de un módulo

- Its **name**
- The **packages** it makes available **publicly**
- The modules it **depends** on



CJAVA
siempre para apoyarte

Veamos en código con IntelliJ

CJAVA
siempre para apoyarte

Según modelo de clases:

```
1  public class Arabica implements Coffee {
2
3      static Logger logger;
4
5      static {
6          System.setProperty("java.util.logging.SimpleFormatter.format",
7                  "[%1$tF %1$tT] [%4$-7s] %5$s %n");
8          logger = Logger.getLogger(Robusta.class.getName());
9      }
10     @Override
11     public void drink() {
12         logger.log(Level.INFO,"Arabica");
13     }
14
15 }
```

El árbol:

Example – com.coffee

```
com.coffee
|
\---com
    \---coffee
        |   Coffee.class
        |   CoffeeMaker.class
        |
        \---types
                Arabica.class
                Robusta.class
```

Example – com.coffee

```
com.coffee
|   module-info.class ----------->
|
\---com
    \---coffee
        |   Coffee.class
        |   CoffeeMaker.class
        |
        \---types
                Arabica.class
                Robusta.class
```

```
module com.coffee {
    requires java.logging;
    exports com.coffee;
}
```

CJAVA
siempre para apoyarte

La Aplicación

```
1  public class CoffeeApp {
2
3      static Logger logger;
4
5      static {
6          System.setProperty("java.util.logging.SimpleFormatter.format",
7                  "[%1$tF %1$tT] [%4$-7s] %5$s %n");
8          logger = Logger.getLogger(CoffeeApp.class.getName());
9      }
10
11     public static void main(String[] args) {
12         CoffeeMaker coffeeMaker = new CoffeeMaker();
13         final Coffee wakeUpCoffee = coffeeMaker.brewRobusta();
14         final Coffee eveningCoffee = coffeeMaker.brewArabica();
15
16
17         wakeUpCoffee.drink();
18         eveningCoffee.drink();
19
20         if(!DriverManager.getDrivers().hasMoreElements()){
21             logger.log(Level.SEVERE,"To Use a DB - Please setup your JDBC drivers");
22         };
23
24     }
```

Lo que se ve

Example – com.app

```
com.app
|
\---com
    \---app
            CoffeeApp.class
            TeaApp.class
```

Definición de módulos

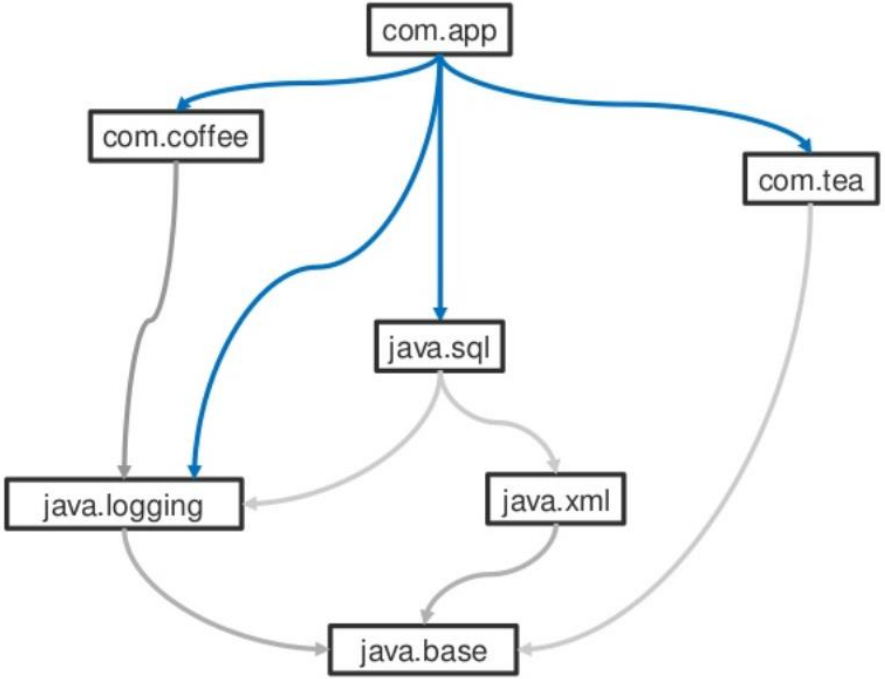
Example – com.app

```
com.app
|   module-info.class
|
\---com
    \---app
            CoffeeApp.class
            TeaApp.class
```

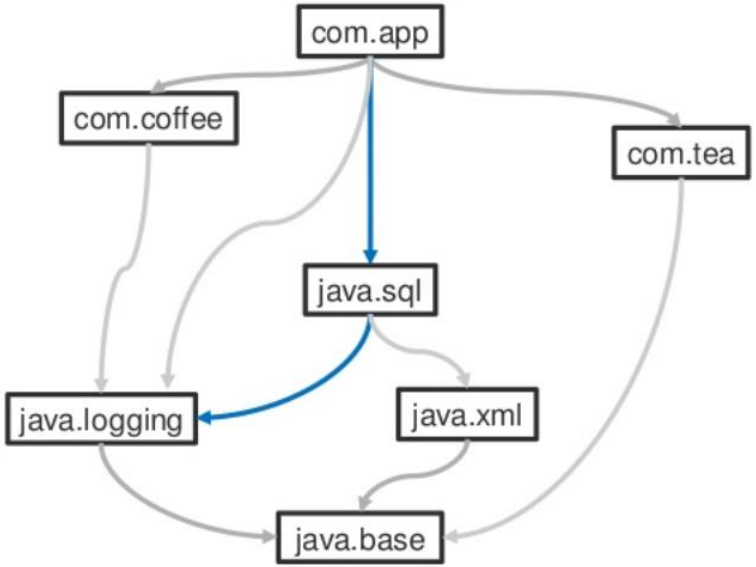
```
module com.app {
    requires com.coffee;
    requires com.tea;

    requires java.sql;
    requires java.logging;
}
```

CJAVA
siempre para apoyarte

Lectura y lectura directa



Lectura y lectura directa




```
module java.sql {
    requires transitive java.logging;
    requires transitive java.xml;
    exports java.sql;
    exports javax.sql;
    exports javax.transaction.xa;
}
```

```
String url = ...;
Properties props = ...;
Driver d = DriverManager.getDriver(url);
Connection c = d.connect(url, props);
d.getParentLogger().info("Connection acquired");
```

Accesibilidad

Accessibility

- Other modules can only **see** and **use** code if:
  1. **public**
  2. package is **exported** by target module
  3. package is **readable** from this module

- Combination referred to as "**accessibility**"

CJAVA
siempre para apoyarte

Accesibilidad

```
package com.app;

import com.coffee.Coffee;
import com.coffee.CoffeeMaker;
import com.coffee.types.Arabica;
import …

public class CoffeeApp {…
```

The type com.coffee.types.Arabica is **not accessible**

```
module com.app {
    requires com.coffee;
    requires com.tea;
    requires com.coffee.types;
}
```

com.coffee.types **cannot** be resolved to a module

CJAVA
siempre para apoyarte

Módulos sin nombre

- Used for
  - Java 8 and older versions
  - Small or temporary applications

- Added via the **class path**
- The unnamed module **reads** code in **any** other module
- The unnamed module **exports all** of its packages.

>jdeps minecraft1.12.jar

Warning: split package: javax.annotation jrt:/java.xml.ws.annotation server.jar
com.google.gson -> java.sql
<unnamed> -> com.google.common.base
....

CJAVA
siempre para apoyarte

Módulos automáticos

- **JAR** from the **class path** that has been put in the **module path.**
- **Don't wait for third party libraries** to become modularized.
- Becomes a named module with
  - **Automatic-Module-Name** in Manifest.mf
  - Otherwise, a name is derived from the JAR filename

- **Reads** every other **named** module.
  - **Exports all** of its packages.
  - It guarantees **direct** and **implied** readability to all other modules.

CJAVA
siempre para apoyarte

Resumen.

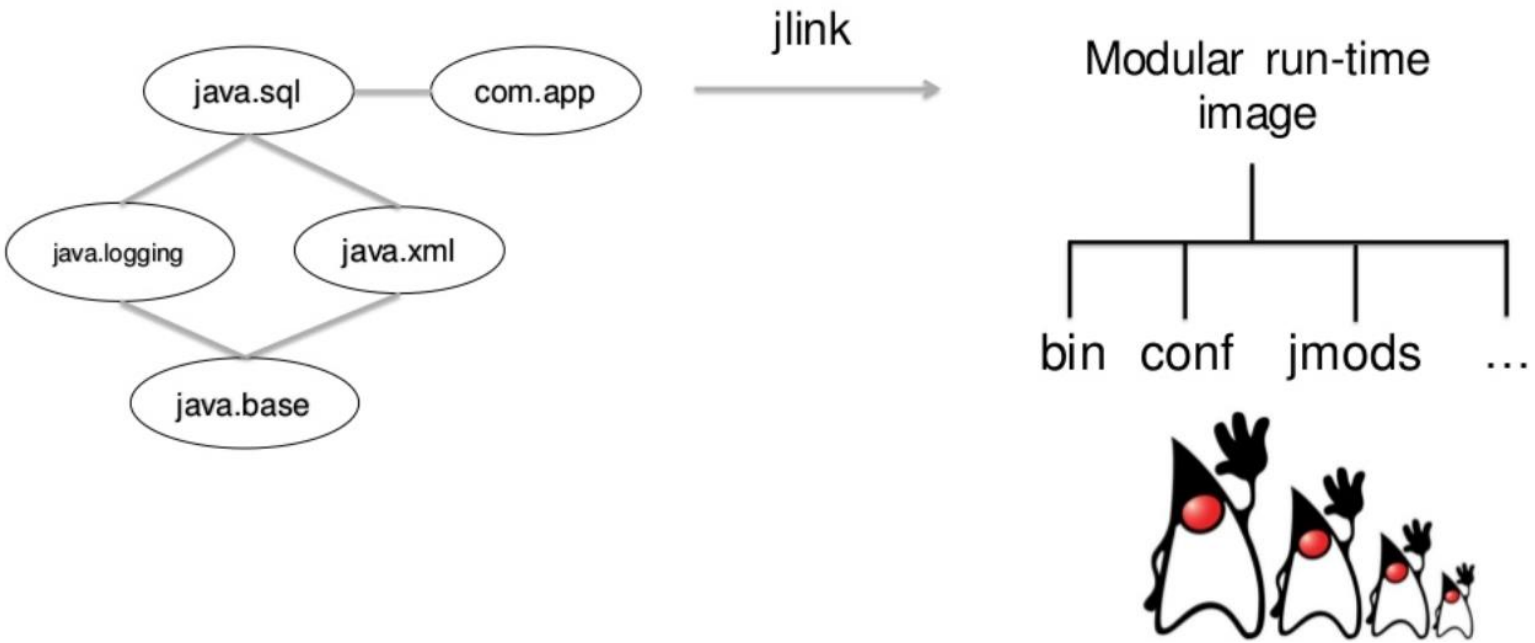
Module Type	Origin	Export Packages	Read Modules
Named Platform	Provided by **platform**	Explicitly	
Named Application	All JARS containing **module-info** on the module path	Explicitly	• Platform • Application • Automatic • ~~Unnamed~~
Automatic	All JARs **without module-info** but on module path	All	• Platform • Application • **Automatic** • Unnamed
Unnamed	**Classpath**	All	• Platform • Automatic • Application

JLink

Cloud-native is an approach to building and running applications that exploits the advantages of the cloud computing delivery model.

Source: https://pivotal.io/cloud-native

JLink


java.sql
com.app
java.logging
java.xml
java.base
jlink
Modular run-time image
bin
conf
jmods
...

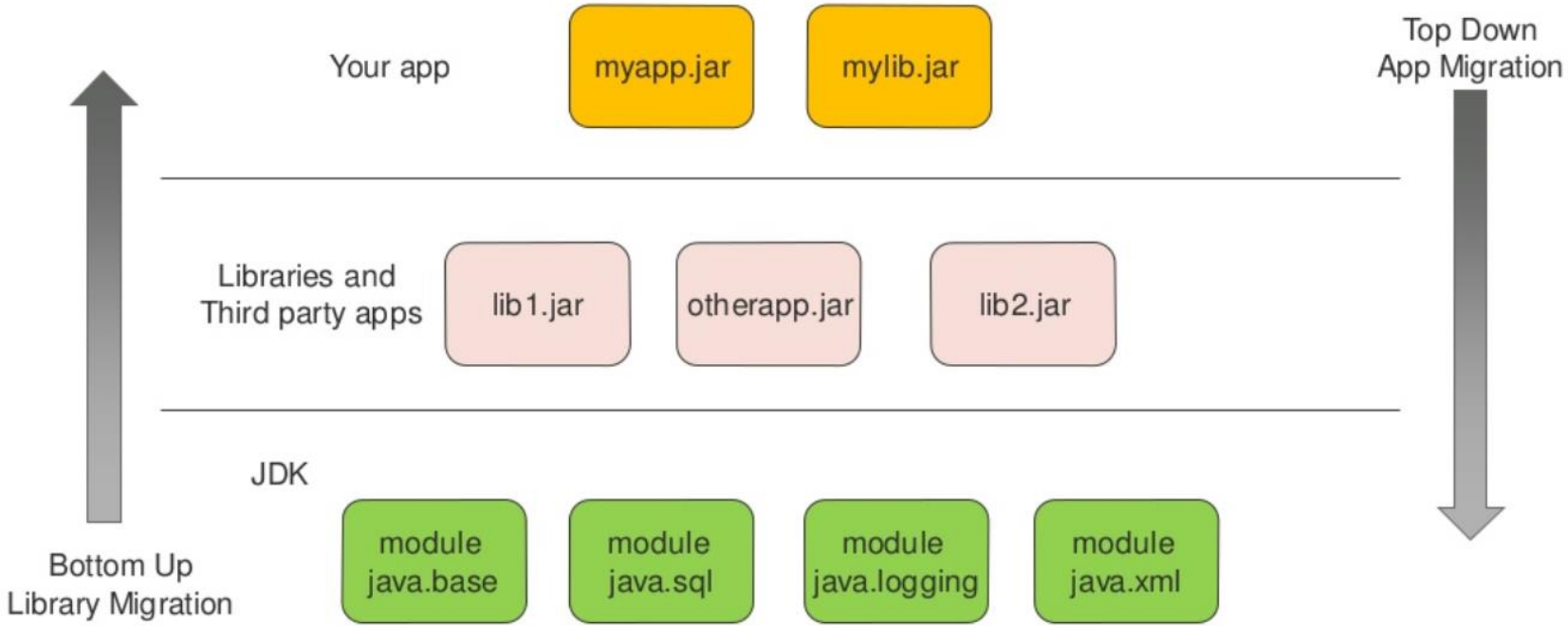
CJAVA
siempre para apoyarte

JLink

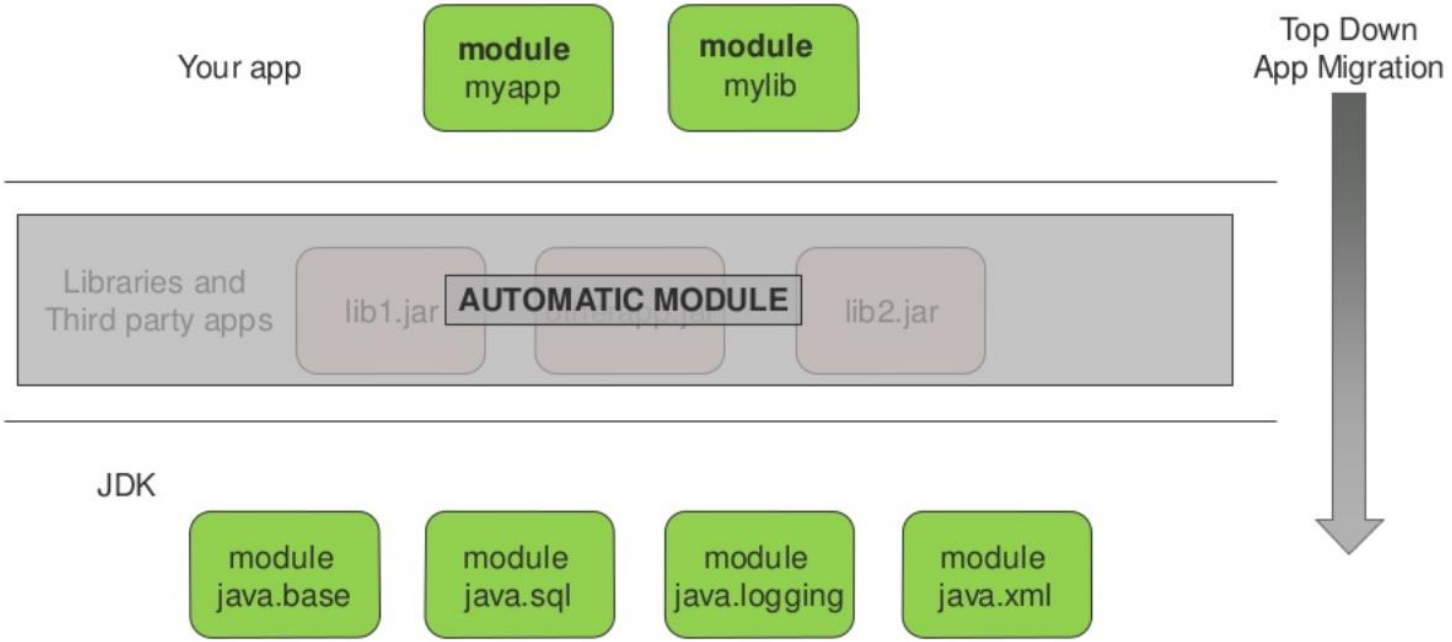
```
$ jlink --module-path $JDKMODS:$MYMODS \
    --add-modules com.app \
    --output myimage

$ myimage/bin/java --list-modules
com.app
com.coffee@1.0
com.tea@1.0
java.base@10
java.sql@10
java.logging@10
```

27 MBs vs 226 MBs

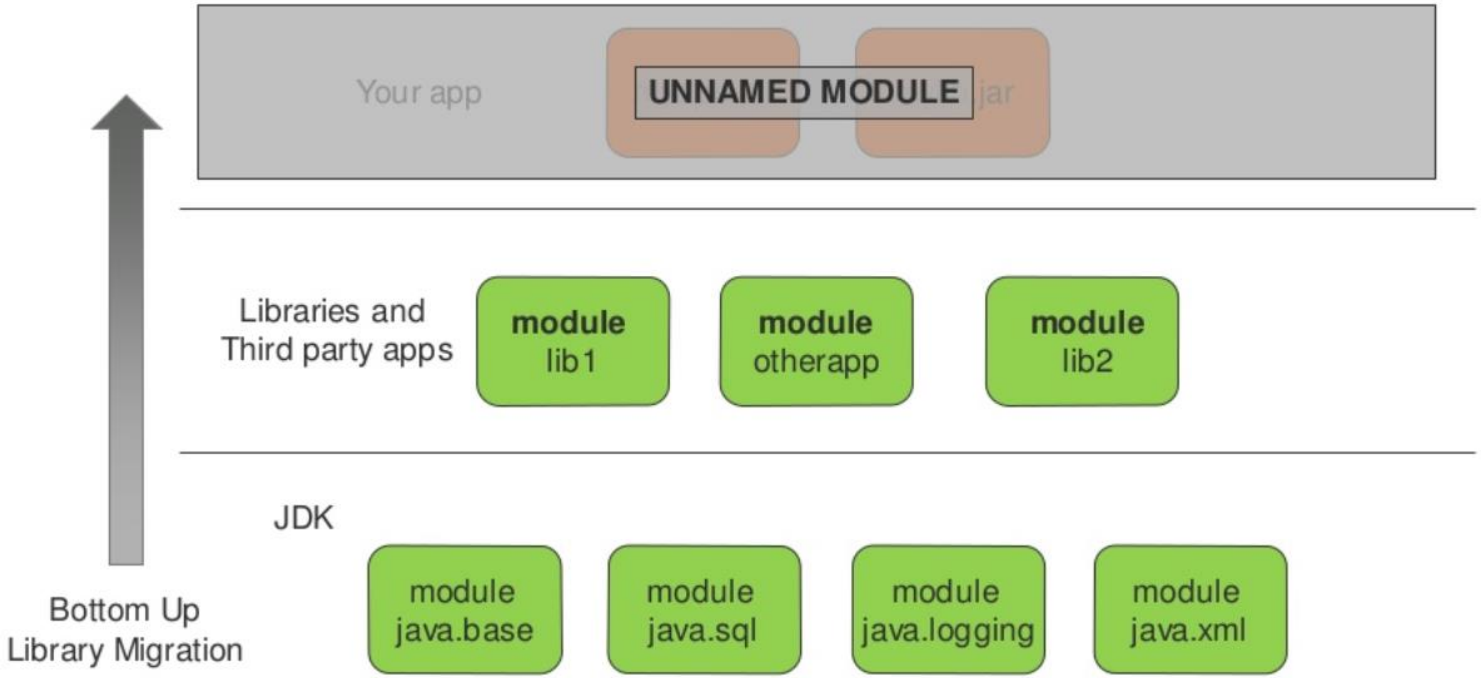
Migración de java 8 para atrás:



Migración de java 8 para atrás:



Migración de java 8 para atrás:



(Migrating the libraries first and using migrated modularized libraries while your app is still on the class path)

CJAVA
siempre para apoyarte

Como Modularizar

- Introduce a **module-info.java** file for each module
- A module-info.java file can be:
  - Created **manually** by the developers
  - Automatically **generated** using the **JDeps** tool with the option --generate-module-info

CJAVA
siempre para apoyarte

Características de la Modularización:

1) Encapsulated JDK internal APIs
2) Not resolved modules
3) Cyclic dependencies
4) Split packages

CJAVA
siempre para apoyarte

API interno de JDK encapsulado

- Most of the internal JDK APIs are inaccessible in Java 9
- Trying to access them causes a compilation error
- Internal APIs are in the **sun.*** package are part of the **jdk.unsupported** module . Not **java.base**
- The module **jdk.unsupported** provides:
  - sun.misc.Unsafe,
  - sun.reflect.Reflection,
  - ...

CJAVA
siempre para apoyarte

Apis del JDK internas encapsuladas:

- JDeps can report internal dependencies

```
$ jdeps --jdk-internals 'lib/batik/batik-codec.jar'
```

batik-codec.jar -> JDK removed internal API

JDK Internal API	Suggested Replacement
com.sun.image.codec.jpeg.JPEGCodec	Use javax.imageio @since 1.4
com.sun.image.codec.jpeg.JPEGDecodeParam	Use javax.imageio @since 1.4

CJAVA
siempre para apoyarte

Problemas con las APIs internas

- **Replace** each of your JDK internal APIs calls with **supported** APIs.

OR

- **Break** the encapsulation with **--add-exports**

--add-exports java.base/sun.net=ALL-UNNAMED

CJAVA
siempre para apoyarte

Problemas resueltos en Java 11

- From **jdk-9+118** running code on the **class path** , these types will not be visible by default:

- java.activation
- java.annotations.common
- java.corba
- java.transaction
- java.xml.bind
- java.xml.ws

Ya no van:

- Make that module available for compilation with **--add-modules**:

>Javac **--add-modules java.xml.bind**

- And again for execution:

>java **--add-modules java.xml.bind**

Cancelled by JEP 320

Dependencias cíclicas

```
module A {
   requires B;
}
```


```
module B {
   requires A;
}
```

- **Forbidden** at **compile-time** (between modules)
- JPMS does not allow cycles in the "**requires**" clauses
- JPMS allows cycles in the "**reads**" relation of run-time modules
- One solution is to **merge** the JARs into a **single module**

Paquetes separados

- Modules that contain packages having the same name must not interfere with each other

- Solution for **JAR** files:
  1. Create a single JAR file out of the two JAR files
  2. Rename one of the package
- Solution for **modules**:
  1. Create a single module out of two or more modules
  2. Create a third module
  3. Remove the package dependencies

CJAVA
siempre para apoyarte

El infierno de los módulos

Do not add a **module-info.java** module declaration until:

1. All of your **runtime dependencies** have been modularised
   - (either as a full module or with a MANIFEST.MF entry)
2. All those modularised dependencies have been **released** to Maven Central
3. Your library depends on the **updated versions**

If you can't meet these 3 criteria

- Add a **MANIFEST.MF** entry following the agreed module naming conventions

If everyone does this
we stand a reasonable chance of avoiding **Module Hell**

Contáctenos

Av. Arenales 395 oficina 405
Santa Beatriz - Lima 01 - Perú

Teléfono: 433-6948

RPC / WhatsApp: 932 656 459

Email: info@cjavaperu.com

Síguenos

/cjava.peru.1

/cjava_peru

/cjavaperu

CJAVA

siempre para apoyarte

#CJavaNoPara

Gracias