La excepción está causada por el [Sistema de Módulos de la Plataforma](http://openjdk.java.net/projects/jigsaw/spec/sotms) Java que se introdujo en Java 9, en particular su implementación de encapsulación fuerte. Sólo permite el [acceso](http://openjdk.java.net/projects/jigsaw/spec/sotms/#accessibility) bajo ciertas condiciones, las más destacadas son:

* el tipo debe ser público
* el paquete propietario debe ser exportado

Las mismas limitaciones se aplican a la reflexión, que el código causante de la excepción intentó utilizar. Más concretamente, la excepción está causada por una llamada a [setAccessible](https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/AccessibleObject.html#setAccessible-boolean-). Esto se puede ver en el stack trace anterior, donde las líneas correspondientes en javassist.util.proxy.SecurityActions tienen el siguiente aspecto:

static void setAccessible**(**final AccessibleObject ao**,**

final boolean accessible**)** **{**

**if** **(**System**.**getSecurityManager**()** **==** **null)**

ao**.**setAccessible**(**accessible**);** // <~ Dragons

**else** **{**

AccessController**.**doPrivileged**(new** PrivilegedAction**()** **{**

public Object run**()** **{**

ao**.**setAccessible**(**accessible**);** // <~ moar Dragons

**return** **null;**

**}**

**});**

**}**

**}**

Para que el programa se ejecute correctamente hay que convencer al sistema de módulos para que permita el acceso al elemento sobre el que se ha llamado a setAccessible. Toda la información necesaria para ello está contenida en el mensaje de excepción, pero existen [varios mecanismos](https://www.sitepoint.com/reflection-vs-encapsulation-in-the-java-module-system/) para conseguirlo. Cuál es el mejor depende del escenario exacto que la causó.

No se puede hacer accesible {member}: el módulo {A} no 'abre {package}' a {B}

Con mucho, los escenarios más prominentes son los dos siguientes:

1. Una librería o framework utiliza reflection para llamar a un módulo JDK. En este caso
   * {A} es un módulo Java (prefijado con java. o jdk.)
   * {member} y {package} son partes de la API de Java
   * {B} es una biblioteca, framework o módulo de aplicación; a menudo un módulo sin nombre @...
2. Una biblioteca/framework basado en reflexión como Spring, Hibernate, JAXB, ... refleja sobre el código de la aplicación para acceder a beans, entidades,... En este escenario
   * {A} es un módulo de aplicación
   * {member} y {package} forman parte del código de la aplicación
   * {B} es un módulo del framework o un módulo sin nombre @...

Ten en cuenta que algunas librerías (JAXB, por ejemplo) pueden fallar en ambos casos, ¡así que fíjate bien en qué escenario te encuentras! El de la pregunta es el caso 1.

**1. Llamada reflexiva al JDK**

Los módulos del JDK son inmutables para los desarrolladores de aplicaciones, por lo que no podemos cambiar sus propiedades. Esto sólo deja una solución posible: [las banderas de línea de comandos](https://www.sitepoint.com/reflection-vs-encapsulation-in-the-java-module-system/#commandlineescapehatches). Con ellos es posible abrir paquetes específicos para la reflexión.

Así que en un caso como el anterior (abreviado)...

Unable to make field private java.beans.PropertyChangeSupport$PropertyChangeListenerMap java.beans.PropertyChangeSupport.map accessible: module java.desktop does not "opens java.beans" to module neodatis.odb

... la solución correcta es lanzar la JVM de la siguiente manera:

# --add-opens tiene la siguiente sintaxis: {A}/{paquete}={B} java --add-opens java.base/java.lang=ALL-UNNAMED

Si el código reflectante se encuentra en un módulo con nombre, ALL-UNNAMED puede sustituirse por su nombre.

Tenga en cuenta que a veces puede ser difícil encontrar una manera de aplicar esta bandera a la JVM que realmente ejecutará el código reflectante. Esto puede ser particularmente difícil si el código en cuestión es parte del proceso de construcción del proyecto y se ejecuta en una JVM que la herramienta de construcción generó.

Si hay demasiadas banderas que añadir, puedes considerar usar el [interruptor de encapsulación](http://mail.openjdk.java.net/pipermail/jigsaw-dev/2017-March/011763.html) --permit-illegal-access en su lugar. Permitirá que todo el código de la ruta de clases refleje los módulos con nombre en general. ¡Tenga en cuenta que esta bandera **sólo funcionará en Java 9**!

**2. Reflexión sobre el código de la aplicación**

En este escenario, es probable que pueda editar el módulo en el que se utiliza la reflexión para entrar. (Si no es así, está efectivamente en el caso 1.) Esto significa que las banderas de línea de comandos no son necesarias y en su lugar se puede utilizar el descriptor del módulo {A} para abrir su interior. Hay varias opciones:

* exportar el paquete con exports {paquete}, que lo hace disponible en tiempo de compilación y ejecución para todo el código
* exportar el paquete al módulo de acceso con exports {paquete} a {B}, lo que lo hace disponible en tiempo de compilación y ejecución pero sólo para {B}.
* abrir el paquete con opens {paquete}, que lo hace disponible en tiempo de ejecución (con o sin reflexión) para todo el código
* abrir el paquete al módulo de acceso con opens {paquete} a {B}, que lo hace disponible en tiempo de ejecución (con o sin reflexión) pero sólo a {B}
* abrir todo el módulo con open module {A} { ... }, que hace que todos sus paquetes estén disponibles en tiempo de ejecución (con o sin reflexión) para todo el código.

Véase [este](https://blog.codefx.org/java/reflection-vs-encapsulation/) post para una discusión más detallada y una comparación de estos enfoques.

**Te explico la causa del fallo y posibles soluciones:**

**Problema:**

La excepción java.lang.reflect.InaccessibleObjectException indica un conflicto de acceso por reflexión a un paquete interno del módulo java.desktop/java.beans.

El módulo neodatis.odb intenta utilizar la reflexión para acceder al campo map de la clase java.beans.PropertyChangeSupport, pero el módulo java.desktop no ha abierto ese paquete a la reflexión.

**Causas Posibles:**

* **Encapsulación de JDK Internos:** Java 9 y posteriores encapsulan algunos paquetes internos para mejorar la seguridad y modularidad. Esto puede causar problemas a librerías o frameworks que antes usaban reflexión a esos paquetes.
* **Configuración de Módulo Incorrecta:** El módulo neodatis.odb podría no tener los permisos adecuados para usar la reflexión en java.beans.

**Soluciones:**

1. **Añadir** opens java.beans to java.desktop**:**
   * Los módulos JDK son inmutables y no se pueden modificar. Si se tuviese control sobre java.desktop, se podría modificar su module-info.java para abrir java.beans a la reflexión.
   * Si no tienes control, podrías usar opciones del JRE/JDK para relajar la encapsulación.
2. **Modificar** neodatis.odb**:**
   * Si es posible, actualiza la librería o busca alternativas que no requieran reflexión a java.beans.
   * Refactoriza tu código para evitar la reflexión a ese paquete.
3. **Usar Opciones de JRE/JDK:**
   * Añade --add-opens java.desktop/java.beans=ALL-UNNAMED al comando de ejecución para abrir el paquete temporalmente.
4. **Inhabilitar Encapsulación (No Recomendado):**
   * Usa la opción --illegal-access=permit (Java 9-15) o --add-exports java.desktop/java.beans=ALL-UNNAMED (Java 16+) para permitir acceso sin restricciones. Esto podría tener implicaciones de seguridad.

**Elecciones Recomendadas:**

* Priorizar opciones 1 o 2 para mantener la encapsulación y modularidad.
* Usar opciones 3 o 4 solo como último recurso, considerando riesgos.