# Module Import Declarations (Second Preview)

***Changes to the Java® Virtual Machine Specification • Version 24-internal-adhoc.gbierman.20241024***

This document describes changes to the Java Virtual Machine Specification ⇗ to support *Module Import Declarations*, which is a preview feature of Java SE 24. See JEP 494 ⇗ for an overview of the feature.

A companion document describes the changes needed to the Java Language Specification ⇗ to support Module Import Declarations.

Changes are described with respect to existing sections of the JVMS. New text is indicated like this and deleted text is indicated ~~like this~~. Explanation and discussion, as needed, is set aside in grey boxes.

*Changelog:*

*2024-10: First draft.*

# Chapter 4: The `class` File Format

## 4.7 Attributes

### 4.7.25 The `Module` Attribute

The `Module` attribute is a variable-length attribute in the `attributes` table of a `ClassFile` structure (4.1 ⇗). The `Module` attribute indicates the modules required by a module; the packages exported and opened by a module; and the services used and provided by a module.

There may be at most one `Module` attribute in the `attributes` table of a `ClassFile` structure.

The `Module` attribute has the following format:

```
Module_attribute {
    u2 attribute_name_index;
    u4 attribute_length;

    u2 module_name_index;
    u2 module_flags;
```

```
        u2 module_version_index;

        u2 requires_count;
        {   u2 requires_index;
            u2 requires_flags;
            u2 requires_version_index;
        } requires[requires_count];

        u2 exports_count;
        {   u2 exports_index;
            u2 exports_flags;
            u2 exports_to_count;
            u2 exports_to_index[exports_to_count];
        } exports[exports_count];

        u2 opens_count;
        {   u2 opens_index;
            u2 opens_flags;
            u2 opens_to_count;
            u2 opens_to_index[opens_to_count];
        } opens[opens_count];

        u2 uses_count;
        u2 uses_index[uses_count];

        u2 provides_count;
        {   u2 provides_index;
            u2 provides_with_count;
            u2 provides_with_index[provides_with_count];
        } provides[provides_count];
    }
```

The items of the `Module_attribute` structure are as follows:

attribute_name_index
: The value of the `attribute_name_index` item must be a valid index into the `constant_pool` table. The `constant_pool` entry at that index must be a `CONSTANT_Utf8_info` structure (4.4.7 ⇗) representing the string "`Module`".

attribute_length
: The value of the `attribute_length` item indicates the length of the attribute, excluding the initial six bytes.

module_name_index
: The value of the `module_name_index` item must be a valid index into the `constant_pool` table. The `constant_pool` entry at that index must be a `CONSTANT_Module_info` structure (4.4.11 ⇗) denoting the current module.

module_flags
: The value of the `module_flags` item is as follows:

    0x0020 (`ACC_OPEN`)

Indicates that this module is open.

0x1000 (`ACC_SYNTHETIC`)
Indicates that this module was not explicitly or implicitly declared.

0x8000 (`ACC_MANDATED`)
Indicates that this module was implicitly declared.

module_version_index
The value of the `module_version_index` item must be either zero or a valid index into the `constant_pool` table. If the value of the item is zero, then no version information about the current module is present. If the value of the item is nonzero, then the `constant_pool` entry at that index must be a `CONSTANT_Utf8_info` structure representing the version of the current module.

requires_count
The value of the `requires_count` item indicates the number of entries in the `requires` table.

If the current module is `java.base`, then `requires_count` must be zero.

If the current module is not `java.base`, then `requires_count` must be at least one.

requires[]
Each entry in the `requires` table specifies a dependence of the current module. The items in each entry are as follows:

requires_index
The value of the `requires_index` item must be a valid index into the `constant_pool` table. The `constant_pool` entry at that index must be a `CONSTANT_Module_info` structure denoting a module on which the current module depends.

At most one entry in the `requires` table may specify a module of a given name with its `requires_index` item.

requires_flags
The value of the `requires_flags` item is as follows:

0x0020 (`ACC_TRANSITIVE`)
Indicates that any module which depends on the current module, implicitly declares a dependence on the module indicated by this entry.

0x0040 (`ACC_STATIC_PHASE`)
Indicates that this dependence is mandatory in the static phase, i.e., at compile time, but is optional in the dynamic phase, i.e., at run time.

0x1000 (`ACC_SYNTHETIC`)
Indicates that this dependence was not explicitly or implicitly declared in the source of the module declaration.

0x8000 (`ACC_MANDATED`)
Indicates that this dependence was implicitly declared in the source of the module declaration.

requires_version_index

The value of the `requires_version_index` item must be either zero or a valid index into the `constant_pool` table. If the value of the item is zero, then no version information about the dependence is present. If the value of the item is nonzero, then the `constant_pool` entry at that index must be a `CONSTANT_Utf8_info` structure representing the version of the module specified by `requires_index`.

Unless the current module is `java.base`, exactly one entry in the `requires` table must have all of the following:

- A `requires_index` item that indicates `java.base`.
- A `requires_flags` item that has the `ACC_SYNTHETIC` flag not set. (The `ACC_MANDATED` flag may be set.)
- If the `class` file version number is 54.0 or above, a `requires_flags` item that has ~~both~~ the ~~`ACC_TRANSITIVE` and~~ `ACC_STATIC_PHASE` ~~flags~~ flag not set.

exports_count

The value of the `exports_count` item indicates the number of entries in the `exports` table.

exports[]

Each entry in the `exports` table specifies a package exported by the current module, such that `public` and `protected` types in the package, and their `public` and `protected` members, may be accessed from outside the current module, possibly from a limited set of "friend" modules.

The items in each entry are as follows:

exports_index

The value of the `exports_index` item must be a valid index into the `constant_pool` table. The `constant_pool` entry at that index must be a `CONSTANT_Package_info` structure (4.4.12 ↗) representing a package exported by the current module.

At most one entry in the `exports` table may specify a package of a given name with its `exports_index` item.

exports_flags

The value of the `exports_flags` item is as follows:

0x1000 (`ACC_SYNTHETIC`)

Indicates that this export was not explicitly or implicitly declared in the source of the module declaration.

0x8000 (`ACC_MANDATED`)

Indicates that this export was implicitly declared in the source of the module declaration.

exports_to_count

The value of the `exports_to_count` indicates the number of entries in the `exports_to_index` table.

If `exports_to_count` is zero, then this package is exported by the current module in an *unqualified* fashion; code in any other module may access the types and members in the package.

If `exports_to_count` is nonzero, then this package is exported by the current module in a *qualified* fashion; only code in the modules listed in the `exports_to_index` table may access the types and members in the package.

exports_to_index[]

The value of each entry in the `exports_to_index` table must be a valid index into the `constant_pool` table. The `constant_pool` entry at that index must be a `CONSTANT_Module_info` structure denoting a module whose code can access the types and members in this exported package.

For each entry in the `exports` table, at most one entry in its `exports_to_index` table may specify a module of a given name.

opens_count

The value of the `opens_count` item indicates the number of entries in the `opens` table.

`opens_count` must be zero if the current module is open.

opens[]

Each entry in the `opens` table specifies a package opened by the current module, such that all types in the package, and all their members, may be accessed from outside the current module via the reflection libraries of the Java SE Platform, possibly from a limited set of "friend" modules.

The items in each entry are as follows:

opens_index

The value of the `opens_index` item must be a valid index into the `constant_pool` table. The `constant_pool` entry at that index must be a `CONSTANT_Package_info` structure representing a package opened by the current module.

At most one entry in the `opens` table may specify a package of a given name with its `opens_index` item.

opens_flags

The value of the `opens_flags` item is as follows:

0x1000 (`ACC_SYNTHETIC`)

Indicates that this opening was not explicitly or implicitly declared in the source of the module declaration.

0x8000 (`ACC_MANDATED`)

Indicates that this opening was implicitly declared in the source of the module declaration.

opens_to_count

The value of the `opens_to_count` indicates the number of entries in the `opens_to_index` table.

If `opens_to_count` is zero, then this package is opened by the current module in an *unqualified* fashion; code in any other module may reflectively access the types and members in the package.

If `opens_to_count` is nonzero, then this package is opened by the current module in a *qualified* fashion; only code in the modules listed in the `opens_to_index` table may reflectively access the types and members in the package.

opens_to_index[]

The value of each entry in the `opens_to_index` table must be a valid index into the `constant_pool` table. The `constant_pool` entry at that index must be a `CONSTANT_Module_info` structure denoting a module whose code can access the types and members in this opened package.

For each entry in the `opens` table, at most one entry in its `opens_to_index` table may specify a module of a given name.

uses_count

The value of the `uses_count` item indicates the number of entries in the `uses_index` table.

uses_index[]

The value of each entry in the `uses_index` table must be a valid index into the `constant_pool` table. The `constant_pool` entry at that index must be a `CONSTANT_Class_info` structure (4.4.1 ⇗) representing a service interface which the current module may discover via `java.util.ServiceLoader`.

At most one entry in the `uses_index` table may specify a service interface of a given name.

provides_count

The value of the `provides_count` item indicates the number of entries in the `provides` table.

provides[]

Each entry in the `provides` table represents a service implementation for a given service interface.

The items in each entry are as follows:

provides_index

The value of the `provides_index` item must be a valid index into the `constant_pool` table. The `constant_pool` entry at that index must be a `CONSTANT_Class_info` structure representing a service interface for which the current module provides a service implementation.

At most one entry in the `provides` table may specify a service interface of a given name with its `provides_index` item.

provides_with_count

The value of the `provides_with_count` indicates the number of entries in the `provides_with_index` table.

`provides_with_count` must be nonzero.

provides_with_index[]

The value of each entry in the `provides_with_index` table must be a valid index into the `constant_pool` table. The `constant_pool` entry at that index must be a `CONSTANT_Class_info` structure representing a service implementation for the service interface specified by `provides_index`.

For each entry in the `provides` table, at most one entry in its `provides_with_index` table may specify a service implementation of a given name.