Java 21 Module System Example — Service-Based Version

This example demonstrates how to design modular Java 21 applications using ServiceLoader, where one module defines an interface, another provides an implementation, and a third uses it without compile-time coupling.

Module 1 — API module

module-info.java

```
module com.example.api { exports com.example.api; }
```

com/example/api/Greeter.java

```
package com.example.api; public interface Greeter { String greet(String name);
}
```

Module 2 — Implementation module

module-info.java

```
module com.example.impl { requires com.example.api; provides
com.example.api.Greeter with com.example.impl.SimpleGreeter; }
```

com/example/impl/SimpleGreeter.java

```
package com.example.impl; import com.example.api.Greeter; public class
SimpleGreeter implements Greeter { @Override public String greet(String name) {
return "Hello, " + name + " from ServiceLoader!"; } }
```

Module 3 — Application module

module-info.java

```
module com.example.app { requires com.example.api; uses
com.example.api.Greeter; }
```

com/example/app/Main.java

```
package com.example.app; import com.example.api.Greeter; import
java.util.ServiceLoader; public class Main { public static void main(String[]
args) { ServiceLoader loader = ServiceLoader.load(Greeter.class); for (Greeter
greeter : loader) { System.out.println(greeter.greet("Thierry")); } } }
```

Build and Run Instructions

Compile:

```
javac -d mods/com.example.api com.example.api/module-info.java
com.example.api/src/com/example/api/*.java javac --module-path mods -d
mods/com.example.impl com.example.impl/module-info.java
com.example.impl/src/com/example/impl/*.java javac --module-path mods -d
mods/com.example.app com.example.app/module-info.java
com.example.app/src/com/example/app/*.java
```

Run:

```
java --module-path mods -m com.example.app/com.example.app.Main
```

Expected Output:

```
Hello, Thierry from ServiceLoader!
```

Why this version is better

Feature	Direct reference	ServiceLoader version
Compile-time dependency on implementation	Yes	No
Decoupled design	No	Yes
Extensible with new implementations	No	Yes (add new provider modules)
Typical in modular applications	Less common	Standard