

Exposing and Consuming Services



Sander Mak

@sander_mak www.branchandbound.net





The limits of encapsulation



The limits of encapsulation

Introducing services



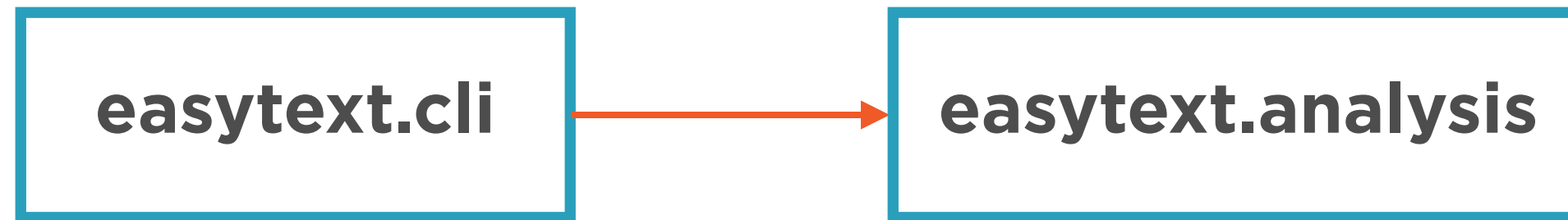
The limits of encapsulation

Introducing services

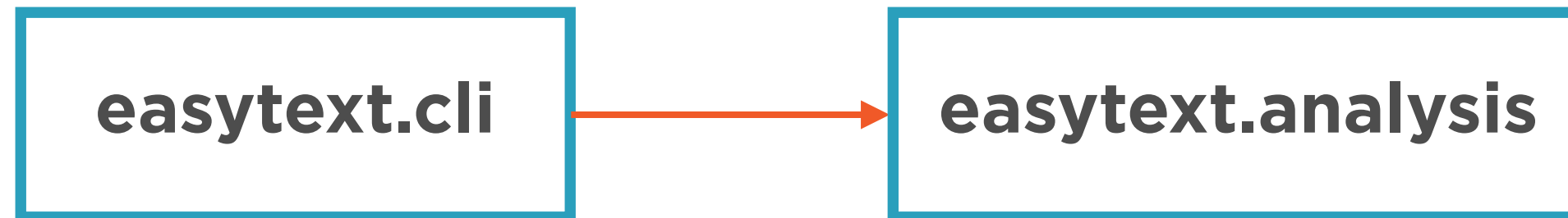
Adding services to EasyText

Remember: Encapsulation

Remember: Encapsulation



Remember: Encapsulation



**Do you read
the module?**

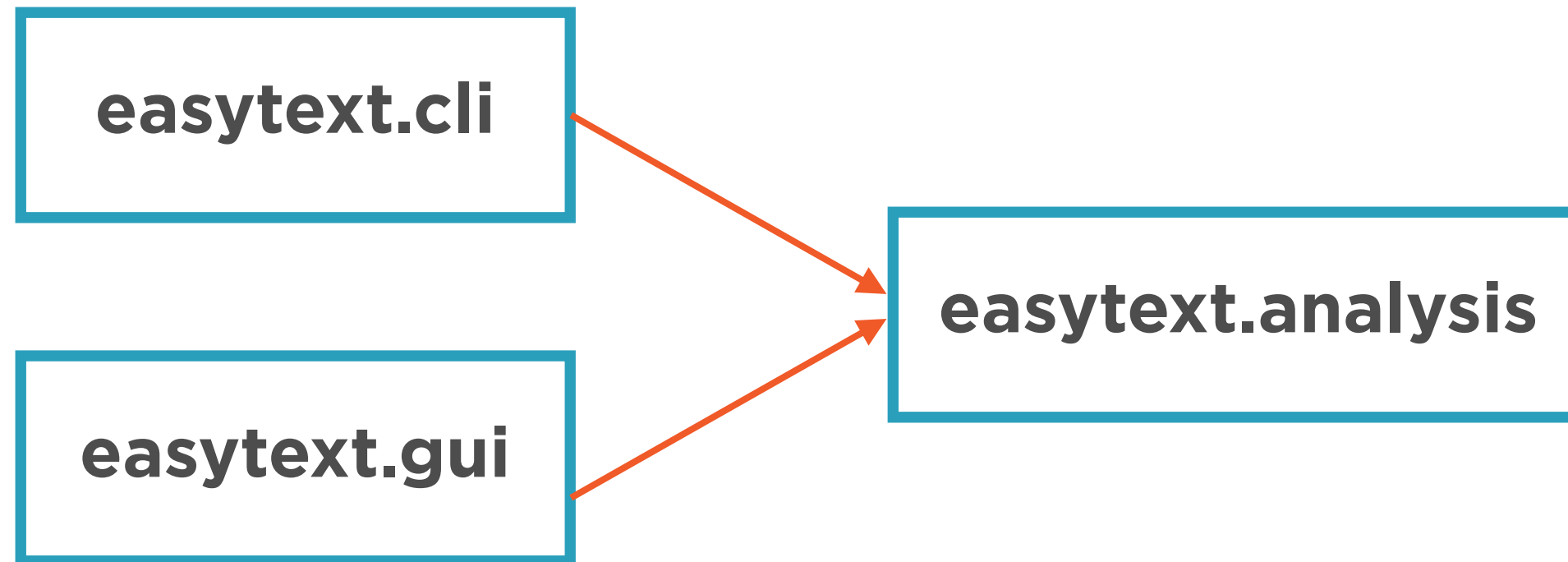


**Is the package
exported?**

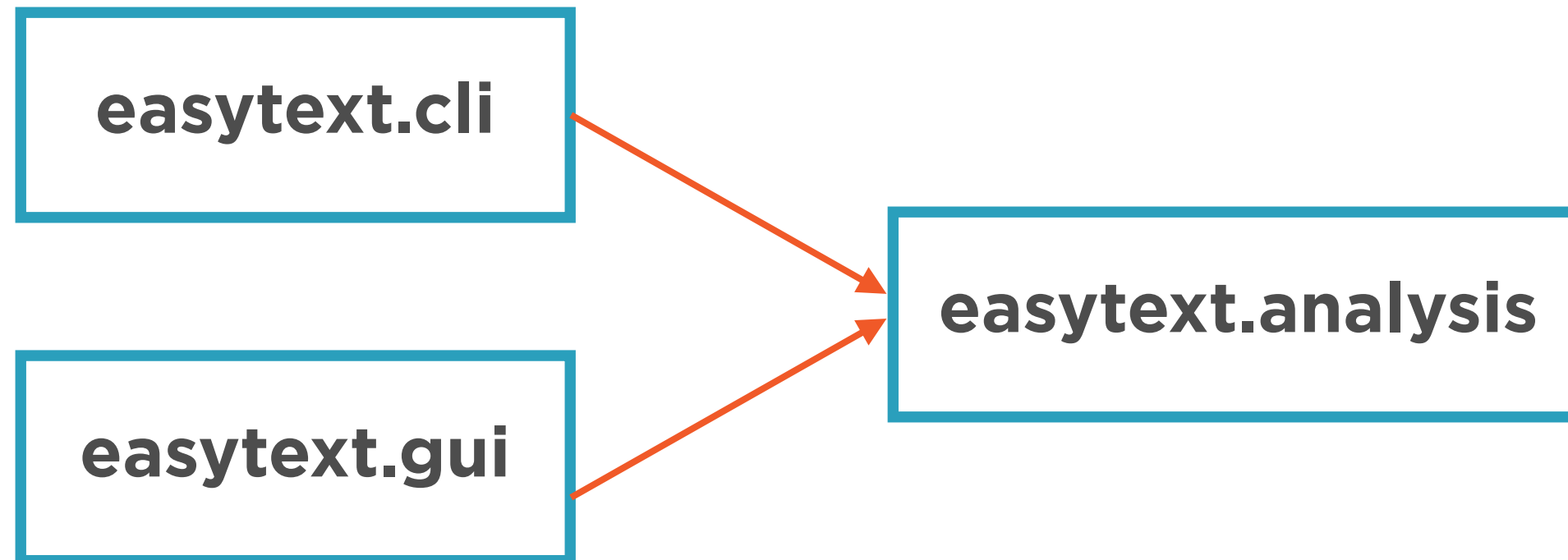


**Is the type
public?**

Modular and Extensible



Modular and Extensible



What if we want to add more analyses?

Modular and Extensible

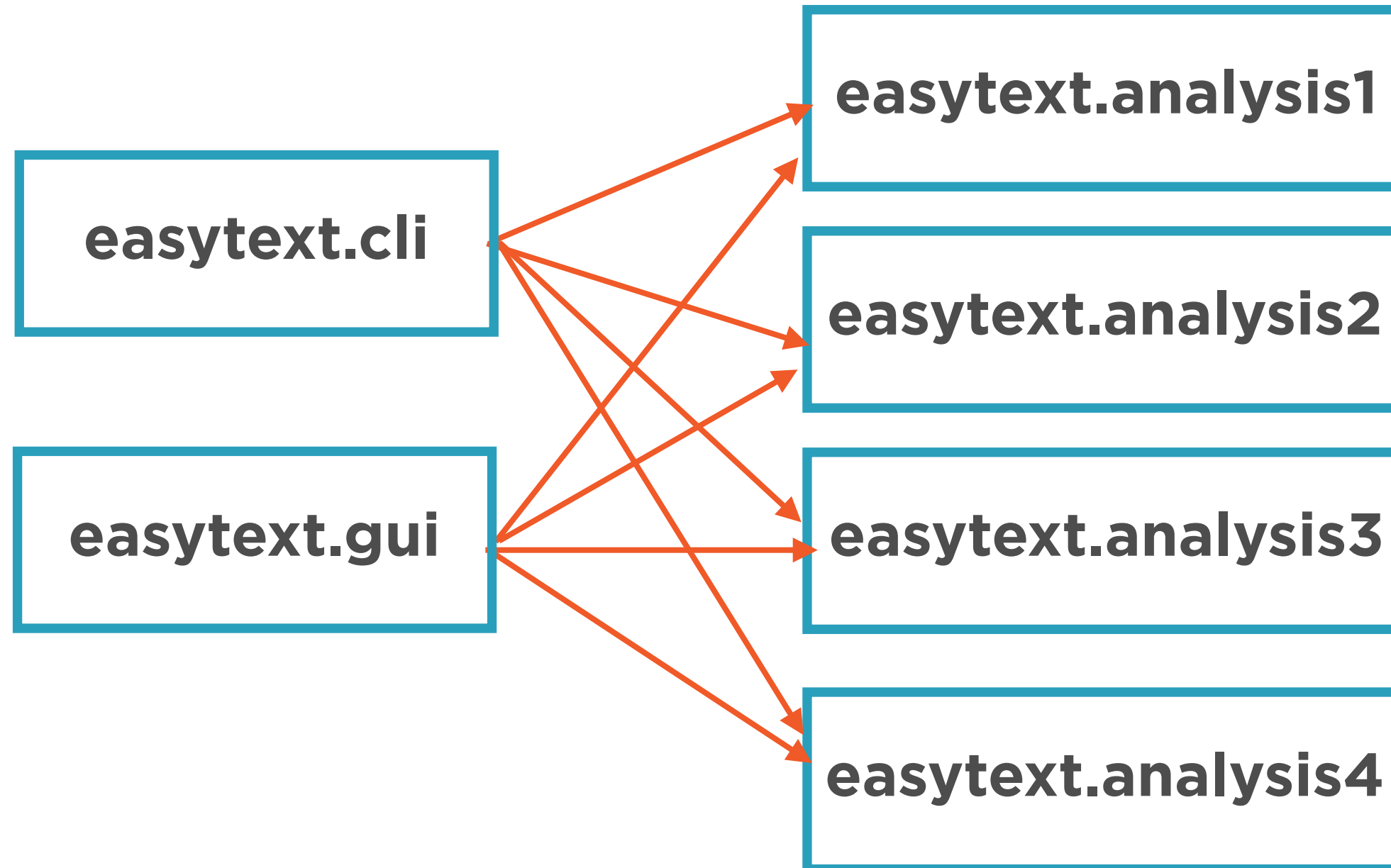
easytext.analysis1

easytext.analysis2

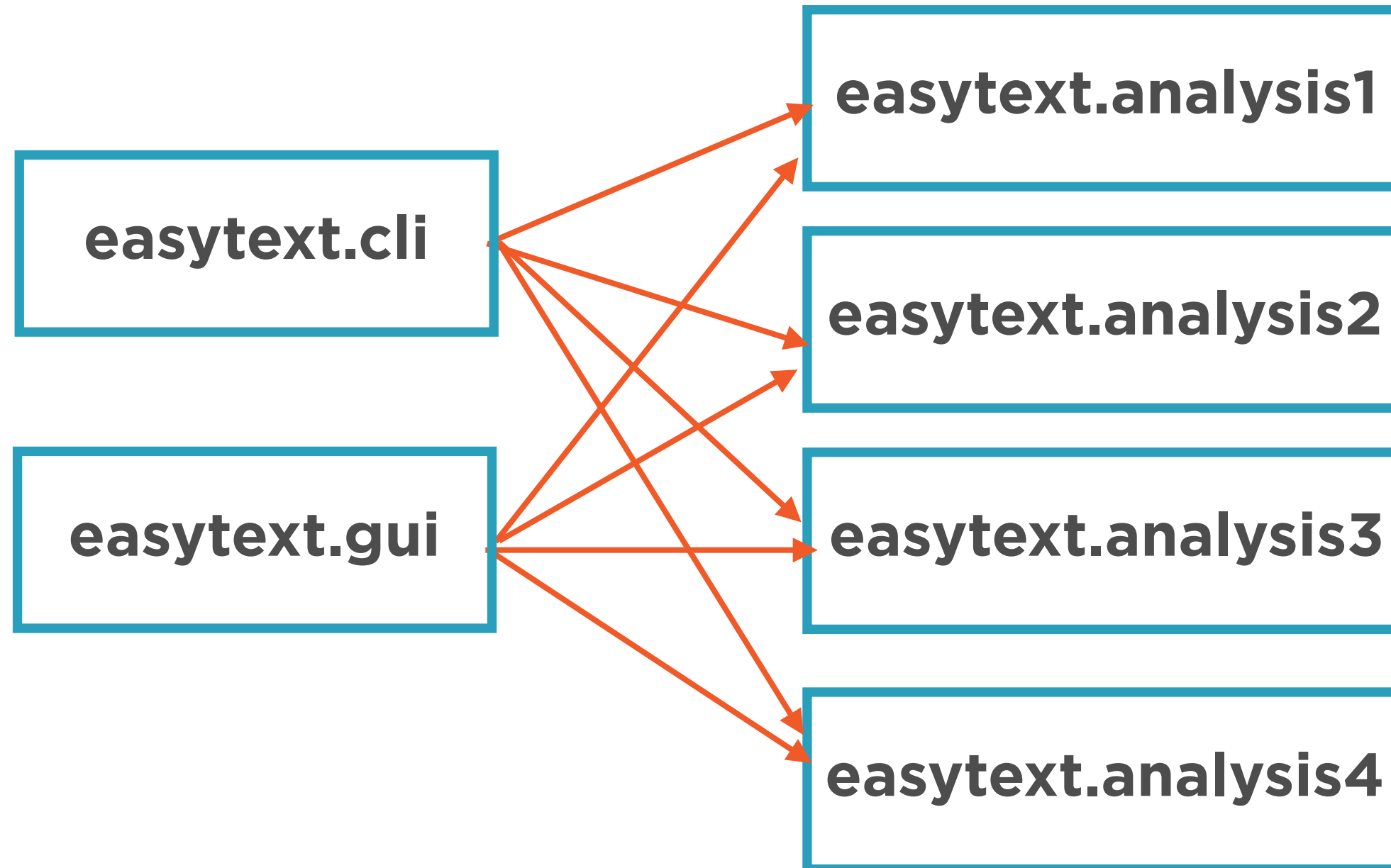
easytext.analysis3

easytext.analysis4

Modular and Extensible



Modular and Extensible



Every front-end needs to know every analysis *implementation*.

What's Wrong With This Code?

```
new FleschKincaid().analyze(sentences);
```

What's Wrong With This Code?

```
new FleschKincaid().analyze(sentences);
```

Depends on implementation, not an interface

What's Wrong With This Code?

```
new FleschKincaid().analyze(sentences);
```

Depends on implementation, not an interface

Implementation class needs to be exported

What's Wrong With This Code?

```
new FleschKincaid().analyze(sentences);
```

Depends on implementation, not an interface

Implementation class needs to be exported

Tight coupling from front-ends to analysis module

What's Wrong With This Code?

```
new FleschKincaid().analyze(sentences);
```

Depends on implementation, not an interface

Implementation class needs to be exported

Tight coupling from front-ends to analysis module

Not extensible

The Limits of Encapsulation

easytext.analysis

j.e.a.KincaidAnalyzer



j.e.a.internal.SyllableCounter

The Limits of Encapsulation

easytext.analysis

j.e.a.KincaidAnalyzer



j.e.a.internal.SyllableCounter

```
interface Analyzer {  
    String getName();  
  
    double analyze(List<List<String> sentences);  
  
}
```

The Limits of Encapsulation

easytext.analysis

j.e.a.KincaidAnalyzer



j.e.a.internal.SyllableCounter

easytext.analysis

j.e.a.Analyzer



j.e.a.KincaidAnalyzer
j.e.a.internal.SyllableCounter

The Limits of Encapsulation

easytext.analysis

j.e.a.KincaidAnalyzer



j.e.a.internal.SyllableCounter

easytext.analysis

j.e.a.Analyzer



j.e.a.KincaidAnalyzer
j.e.a.internal.SyllableCounter

How to instantiate KincaidAnalyzer?!

Services

Services

Service Catalog/Registry

Services

Service Catalog/Registry

MyServiceImpl
implements
MyService

Service Provider

provides

```
graph LR; SP[Service Provider] -- provides --> SCR[Service Catalog/Registry];
```

The diagram illustrates a service discovery architecture. On the left, a large blue-bordered box represents the 'Service Catalog/Registry'. Inside this box, the text 'MyServiceImpl implements MyService' is displayed. On the right, three overlapping light blue boxes represent 'Service Providers'. An orange curved arrow points from the bottom of the 'Service Provider' boxes to the bottom of the 'Service Catalog/Registry' box, with the word 'provides' written in orange text above the arrow.

Services

Service Catalog/Registry

MyService

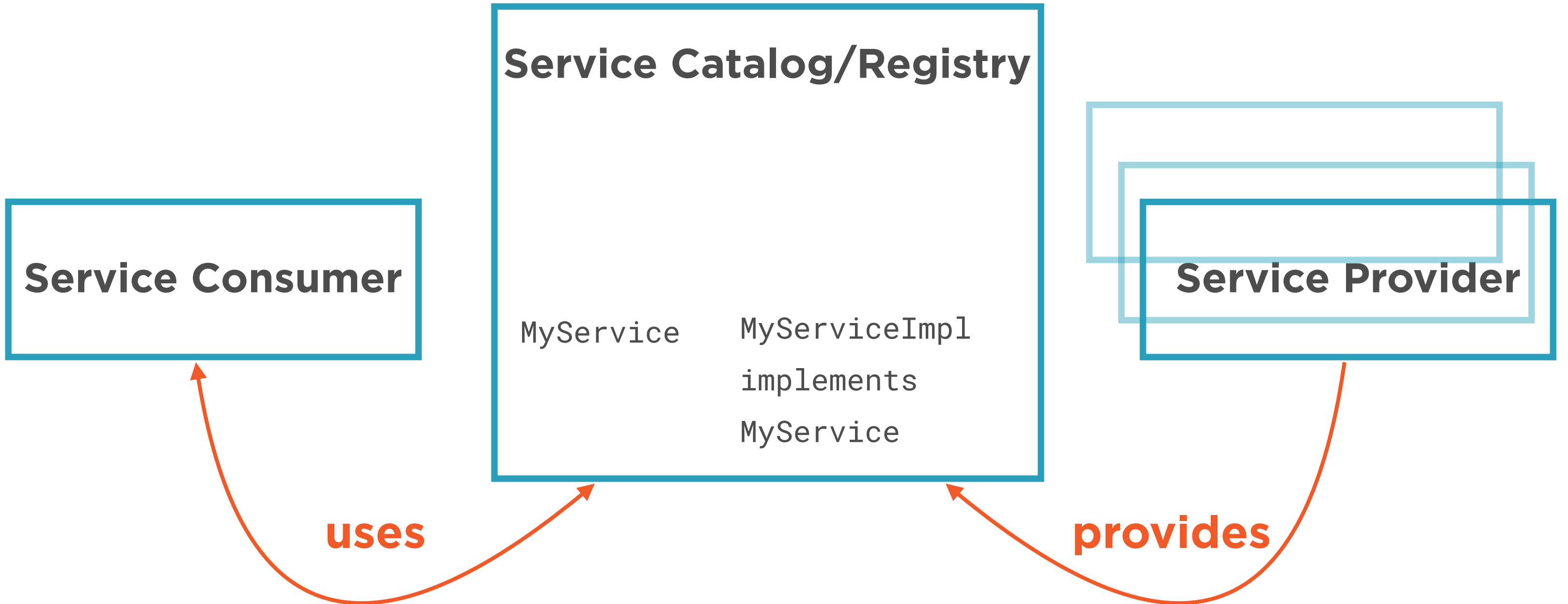
MyServiceImpl
implements
MyService

Service Consumer

Service Provider

uses

provides



Services

Service Catalog/Registry

MyService

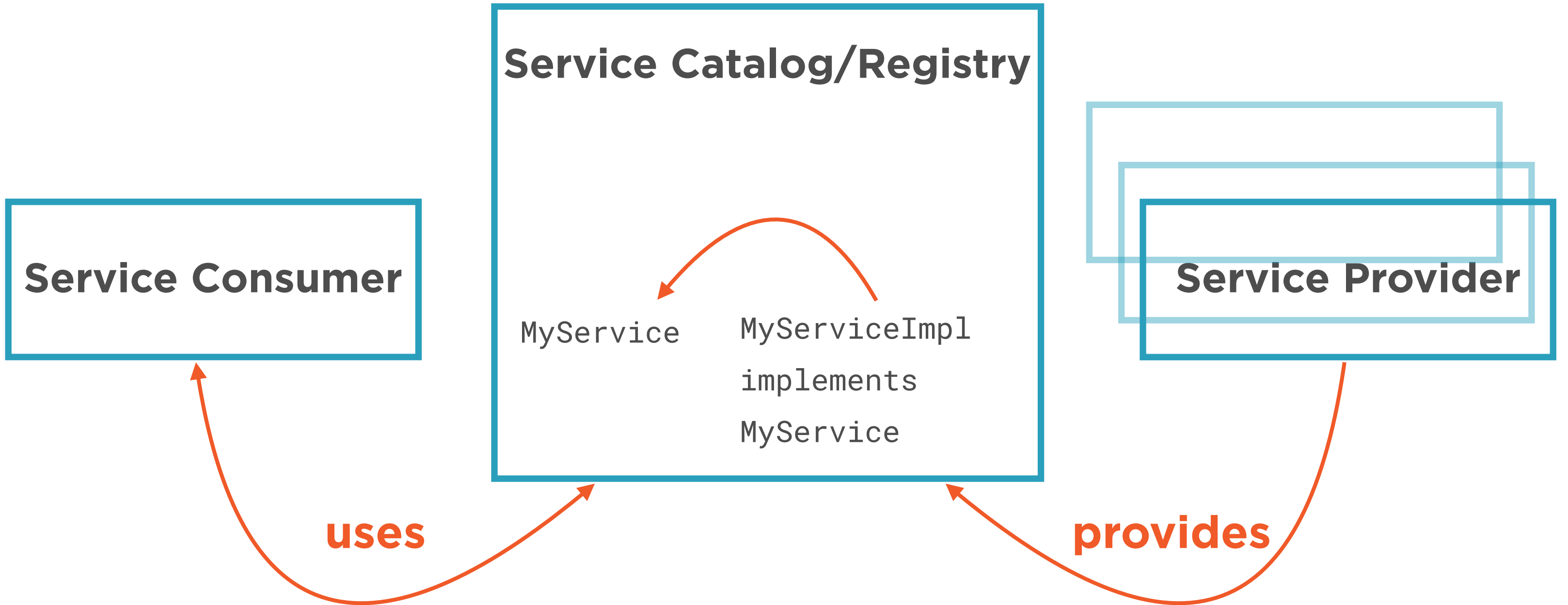
MyServiceImpl
implements
MyService

Service Consumer

Service Provider

uses

provides



Services and ServiceLoader

```
module myApi {  
    exports com.api;  
}  
  
interface MyService {  
    void doWork();  
}
```

Services and ServiceLoader

```
module myApi {  
    exports com.api;  
}  
  
interface MyService {  
    void doWork();  
}
```

```
module myConsumer {  
    requires myApi;  
    uses com.api.MyService;  
}
```

Services and ServiceLoader

```
module myApi {  
    exports com.api;  
}  
  
interface MyService {  
    void doWork();  
}
```

```
module myConsumer {  
    requires myApi;  
    uses com.api.MyService;  
}
```

```
module myProvider {  
    requires myApi;  
    provides com.api.MyService  
        with myProvider.MyServiceImpl;  
}
```

Services and ServiceLoader

```
module myApi {  
    exports com.api;  
}  
  
interface MyService {  
    void doWork();  
}
```

```
module myConsumer {  
    requires myApi;  
    uses com.api.MyService;  
}
```

```
Iterable<MyService> services =  
    ServiceLoader.load(MyService.class);  
  
for (MyService svc: services) {  
    svc.doWork();  
}
```

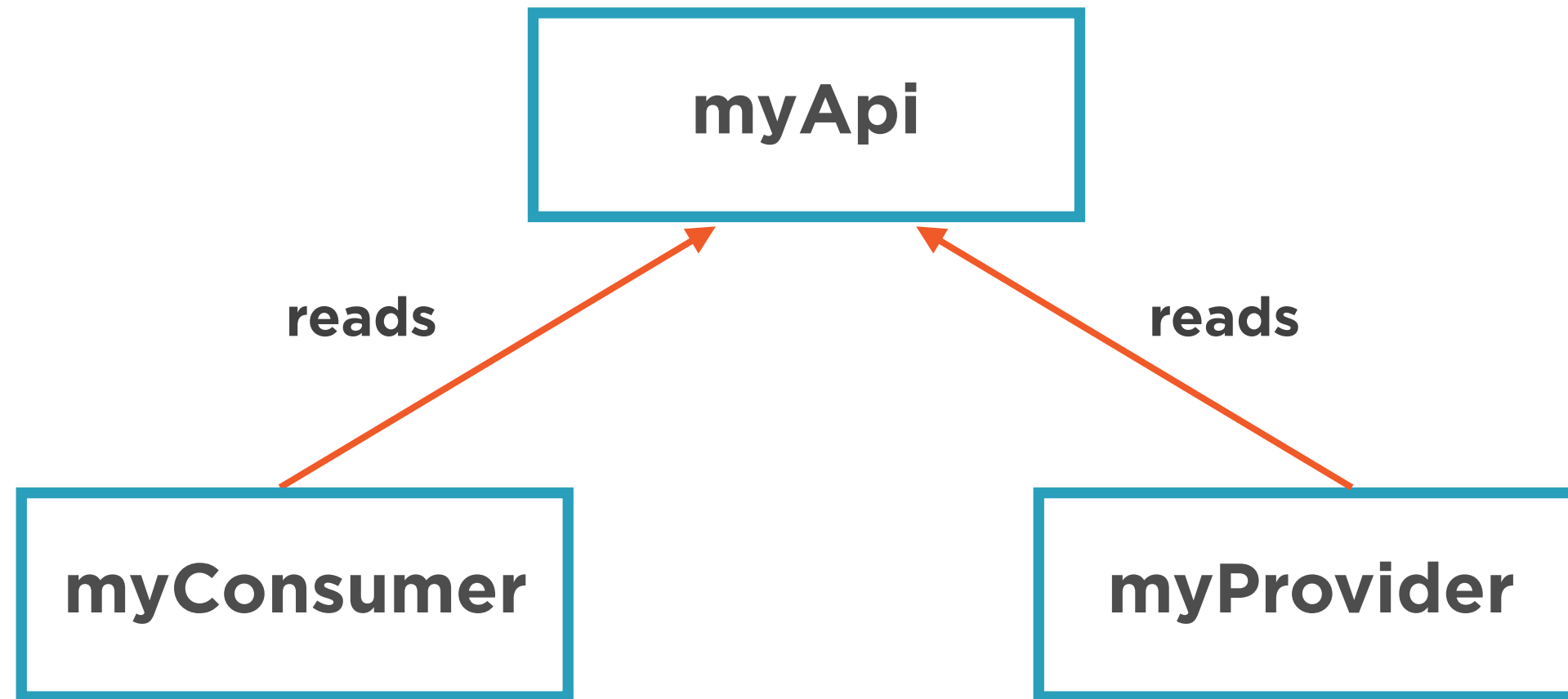
```
module myProvider {  
    requires myApi;  
    provides com.api.MyService  
        with myProvider.MyServiceImpl;  
}
```

Services and ServiceLoader

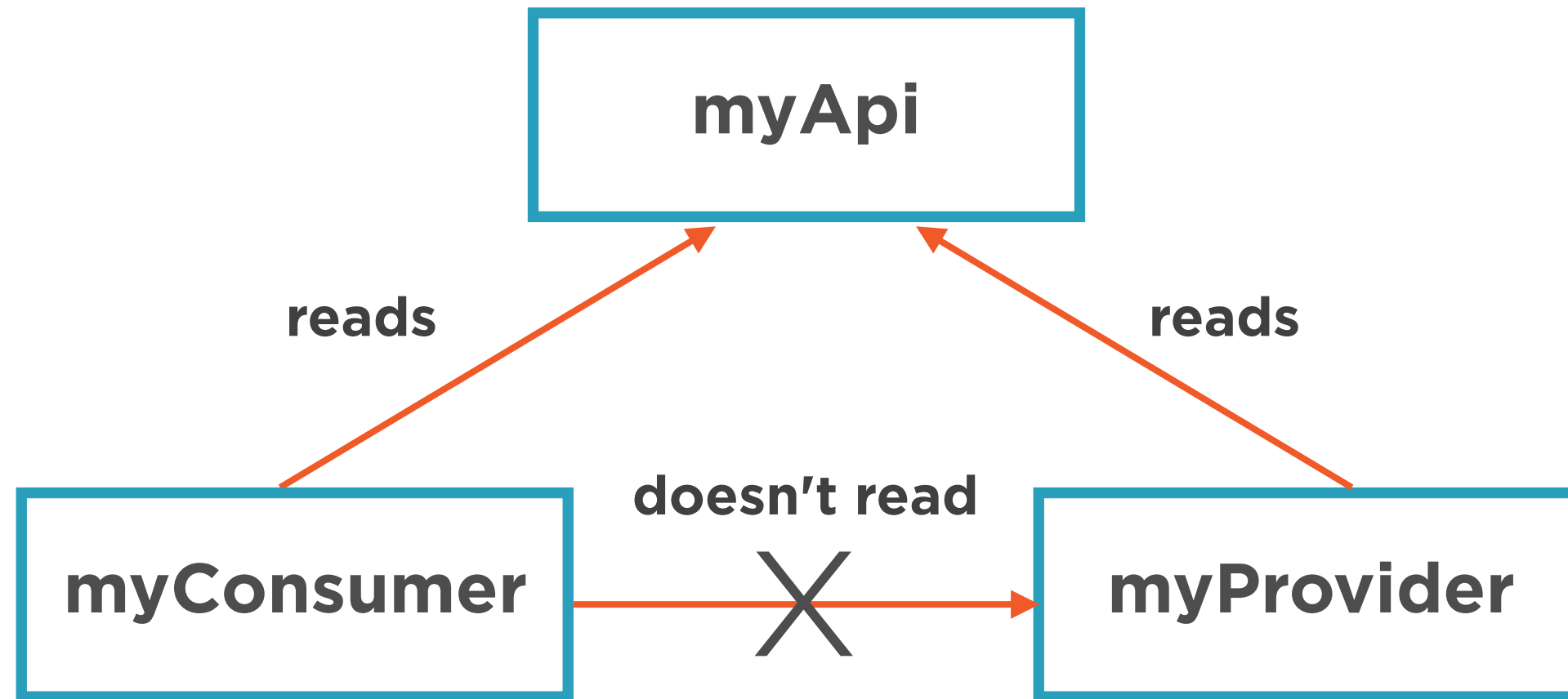


myApi

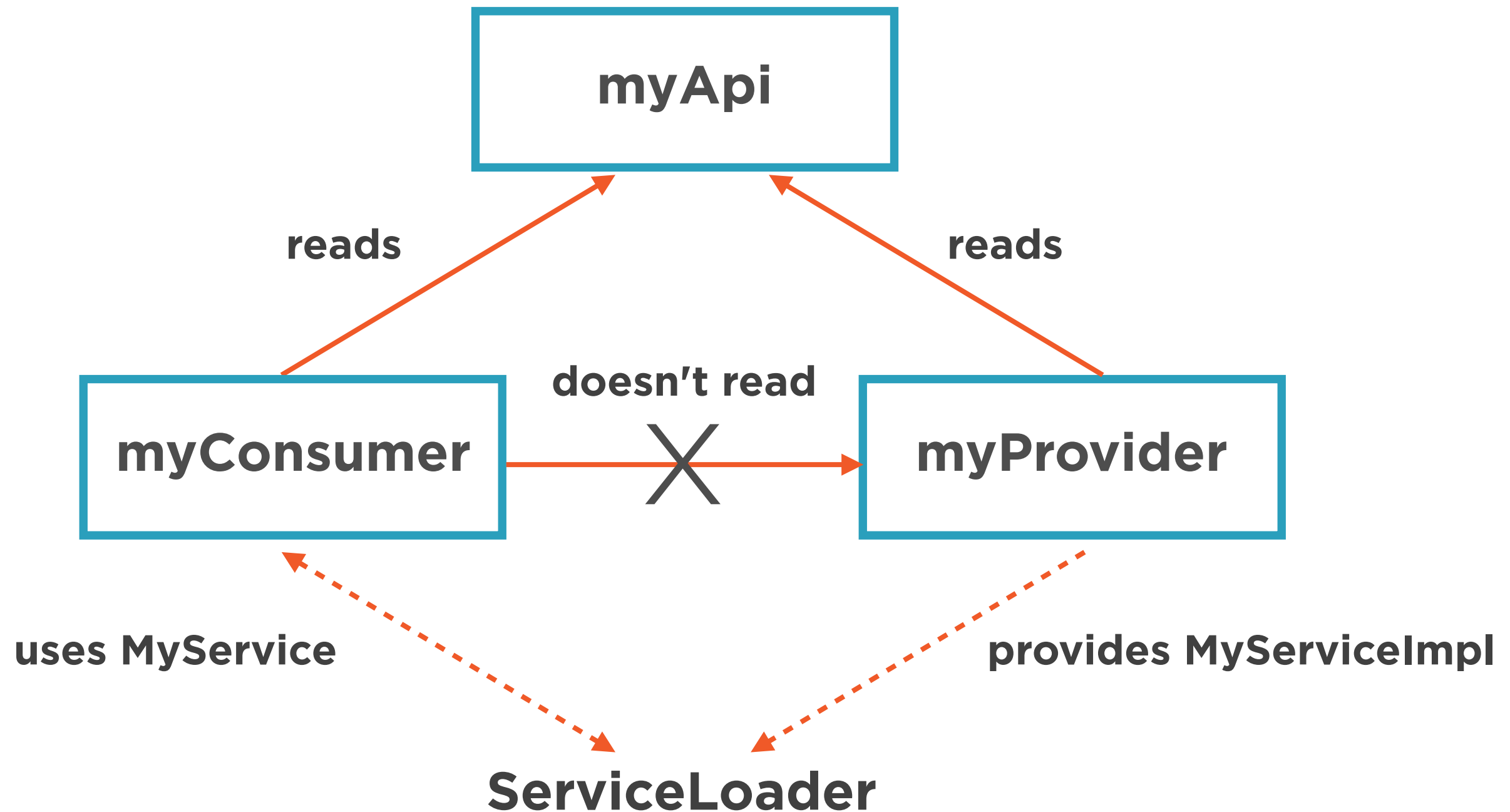
Services and ServiceLoader



Services and ServiceLoader

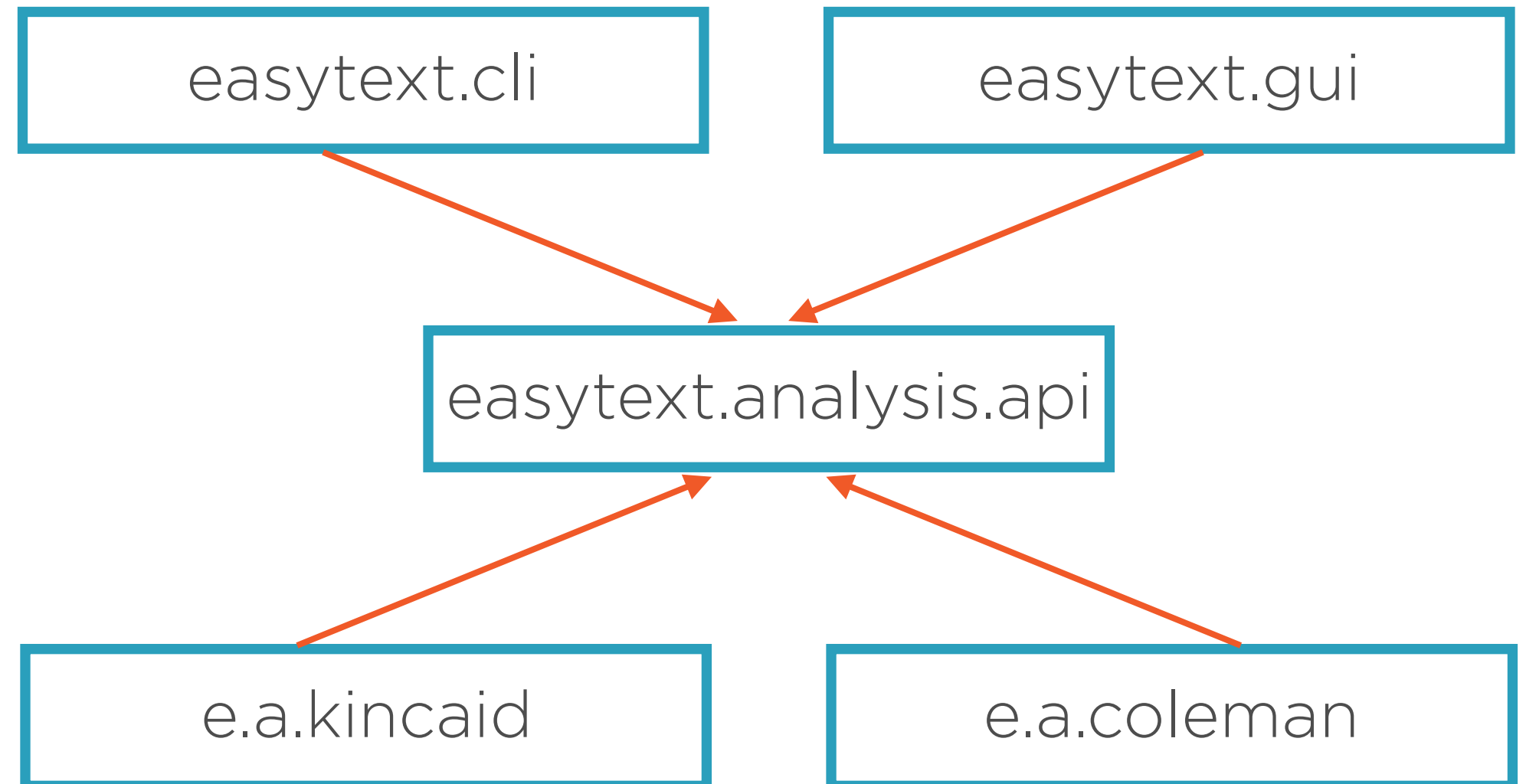


Services and ServiceLoader



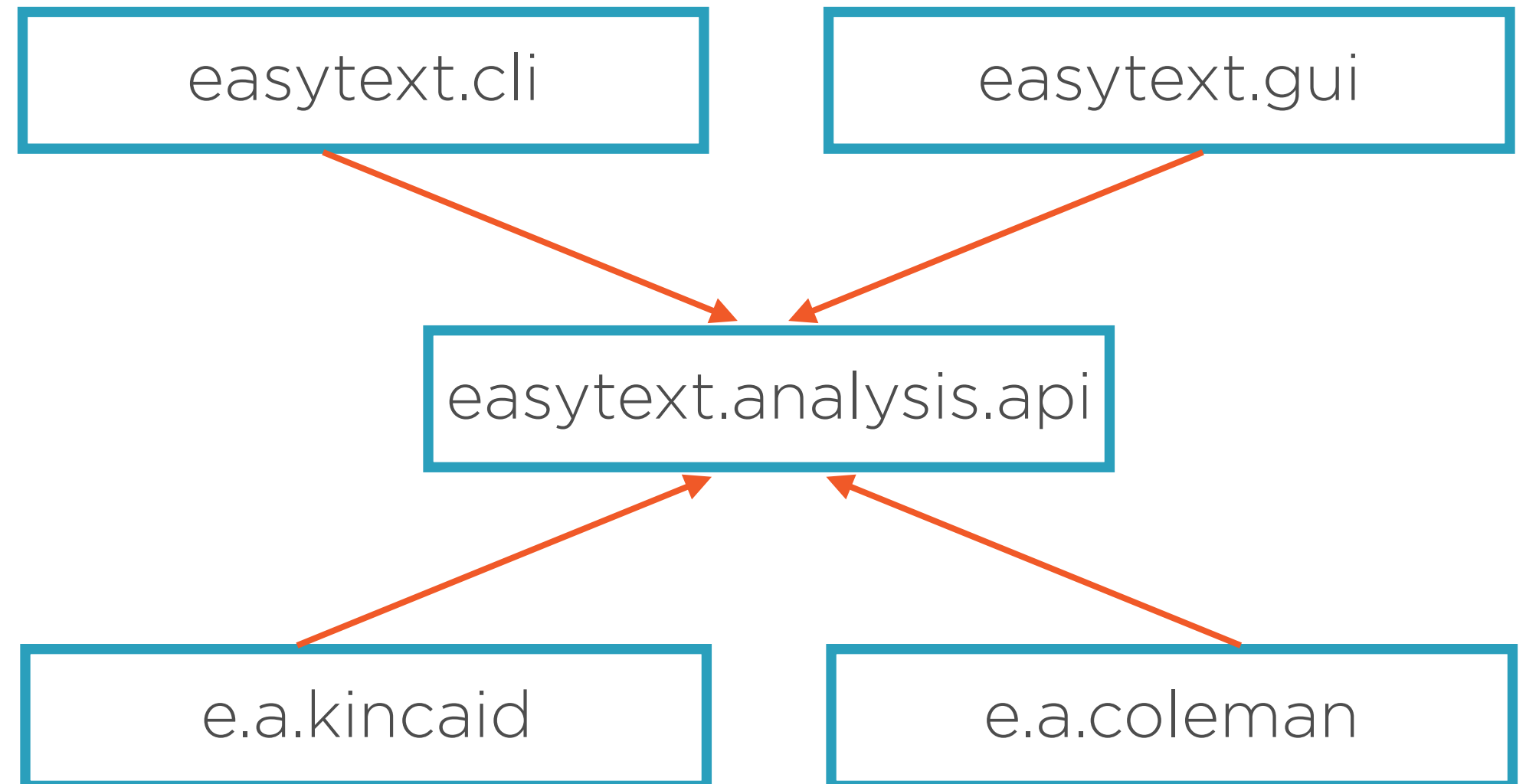
Demo

Adding services to EasyText



Demo

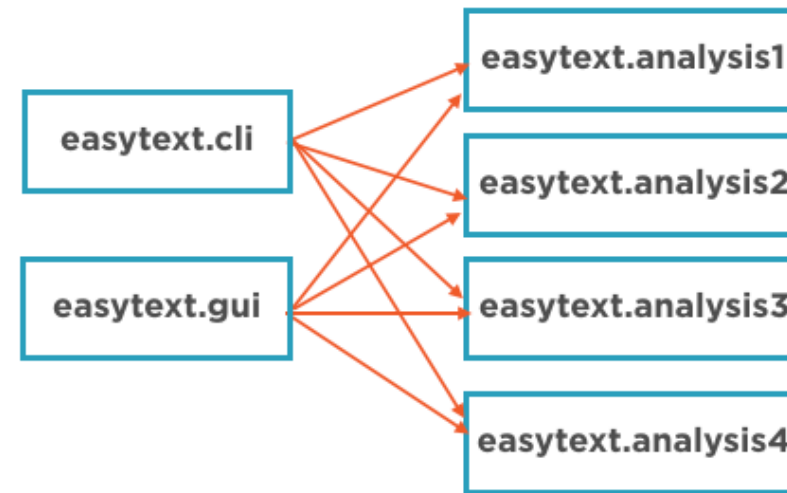
Adding services to EasyText



Summary

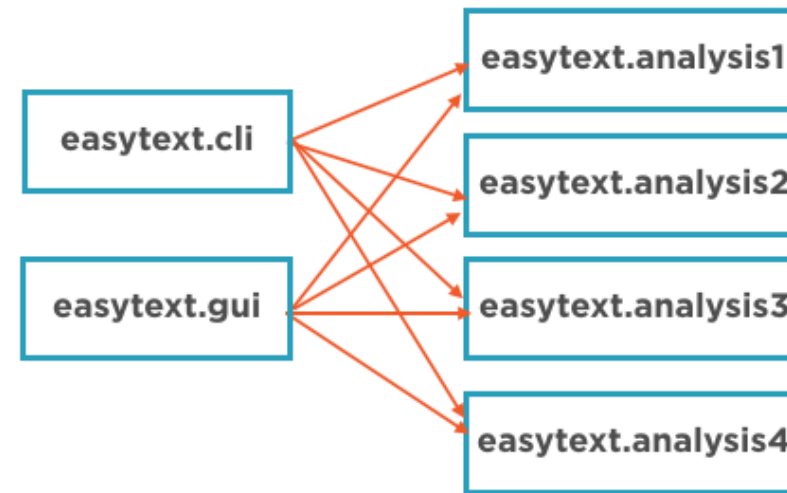
Summary

Summary

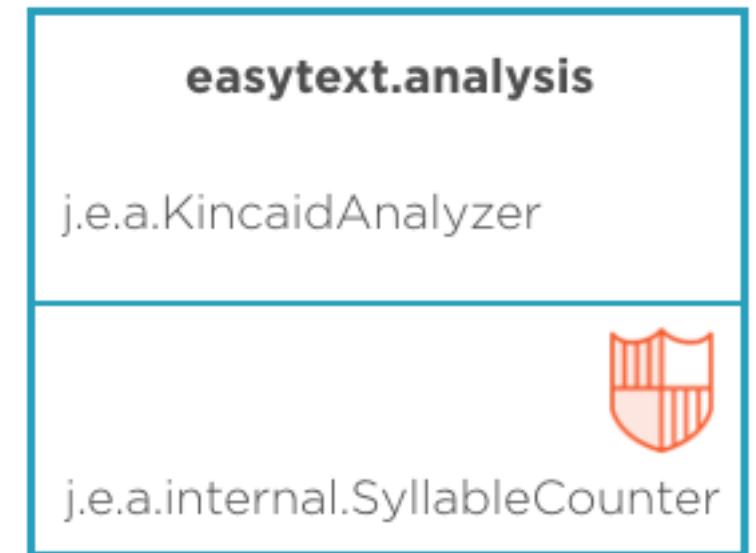


**Strong coupling at
compile-time**

Summary



**Strong coupling at
compile-time**



Weak encapsulation