2. Core Concepts: DI & IoC

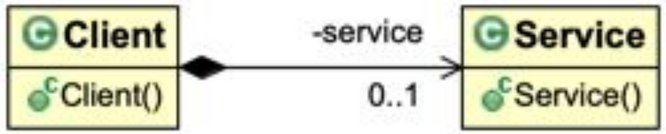
1. What Is Dependency?
2. Managing Dependency
3. What Is Dependency Injection?
4. What Is Inversion of Control?
5. DI and IoC
6. Lab Section: *Step by Step Into Spring DI*

What Is Dependency?

Dependency

- defines a relationship between service - client or consumer - provider
- service : *provides*
- client: *consumes*

Dependency: *Example*




```java
public class Client {

   private Service service;
}
```

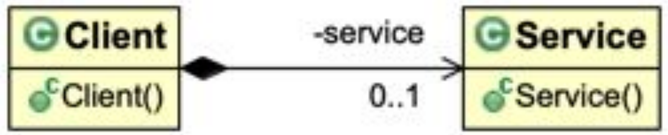
```java
public class Service {

}
```

Client is dependent to service

2. Core Concepts: DI & IoC

1. What Is Dependency?
2. Managing Dependency
3. What Is Dependency Injection?
4. What Is Inversion of Control?
5. DI and IoC
6. Lab Section: *Step by Step Into Spring DI*

Managing Dependency



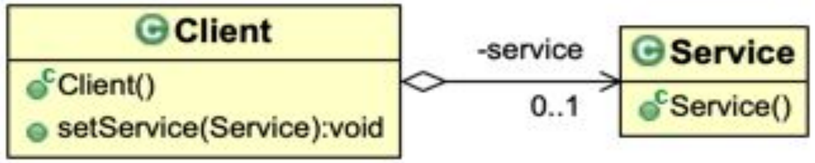

```java
public class Client {

   private Service service;

   public Client() {
       service = new Service();
   }
}
```

```java
public class Service {

}
```

Client object creates Service object.

High Coupling

Managing Dependency: A possible solution



1. constructor method

2. setter method

```java
public class Client {

   private Service service;

   public Client() {
       service = new Service();
   }

   public void setService(Service service) {
       this.service = service
   }
}
```

```java
public class Service {

}
```

Creating a Service object and passing it to the Client object

△ better solution for coupling and complexity of Client

2. Core Concepts: DI & IoC

1. What Is Dependency?
2. Managing Dependency
3. What Is Dependency Injection?
4. What Is Inversion of Control?
5. DI and IoC
6. Lab Section: *Step by Step Into Spring DI*

What Is Dependency Injection?

Dependency Injection

- is creating a dependent object and passing this object.
- aims simpler mechanism for component dependencies
- manages component's life cycles.

Client object does not have to be responsible for creating its parts or its Service objects.
Removing the responsibility of Service object creation allows the Client object to be **highly-cohesive** and **lowly-coupled**.

Now, if the client doesn't create the service, *who* does?
We will see the answer this question in the lab section.

2. Core Concepts: DI & IoC

1. What Is Dependency?
2. Managing Dependency
3. What Is Dependency Injection?
4. What Is Inversion of Control?
5. DI and IoC
6. Lab Section: *Step by Step Into Spring DI*

What Is Inversion of Control?

Inversion of Control

- is about inverting the owner of the control of the flow in an application.
- is a common principle applied by frameworks.
- shortly IoC

A framework when extended by an application takes over the control of the flow of that application and manages it by making calls on its objects and the objects of the application.

2. Core Concepts: DI & IoC

1. What Is Dependency?
2. Managing Dependency
3. What Is Dependency Injection?
4. What Is Inversion of Control?
5. DI and IoC
6. Lab Section: *Step by Step Into Spring DI*

DI and IOC

Dependency Injection (DI) is achieved by IoC. **IoC is used to enable DI.**

IoC is an underlying principle applied in the inner workings of frameworks and DI is one of many techniques where IoC is applied.

IoC is used in many places other than creating and wiring objects such as managing object's life cycle, events, AOP, etc.

2. Core Concepts: DI & IoC

1. What Is Dependency?
2. Managing Dependency
3. What Is Dependency Injection?
4. What Is Inversion of Control?
5. DI and IoC
6. Lab Section: *Step by Step Into Spring DI*

Lab Section


```
public static void main(String[] args) {
    System.out.println("Hello World!");
}
```

- What is wrong with this code?
- *Hint*: Think about dependencies.
- What if we want to say greeting in another language like “Selam”?
- Application must be changed for each different greeting.

Lab Section

Let’s do some refactor and learn what is Spring