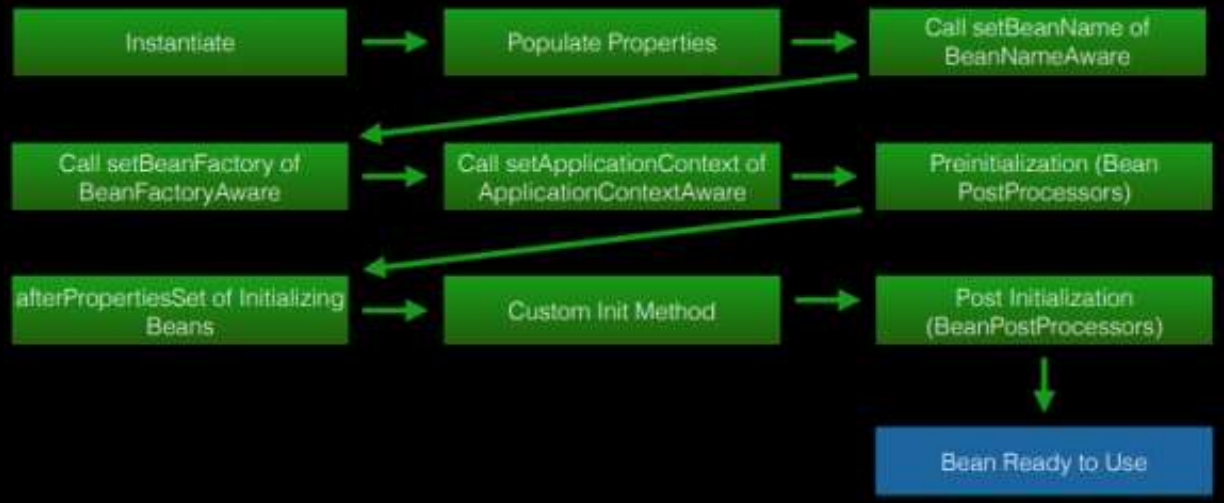
Spring Bean Lifecycle

Bean initialization stages



1. **Bean Definition**

Spring Bean will be defined using stereotype annotations or XML Bean configurations.

2. **Bean Creation and Instantiate**

As soon as bean created and It will be instantiated and loaded into ApplicationContext and JVM memory.

3. **Populating Bean properties**

Spring container will create a bean id, scope, default values based on the bean definition.

4. **Post-initialization**

Spring provides Aware interfaces to access application bean meta-data details and callback methods to hook into the bean life cycle to execute custom application-specific logic.

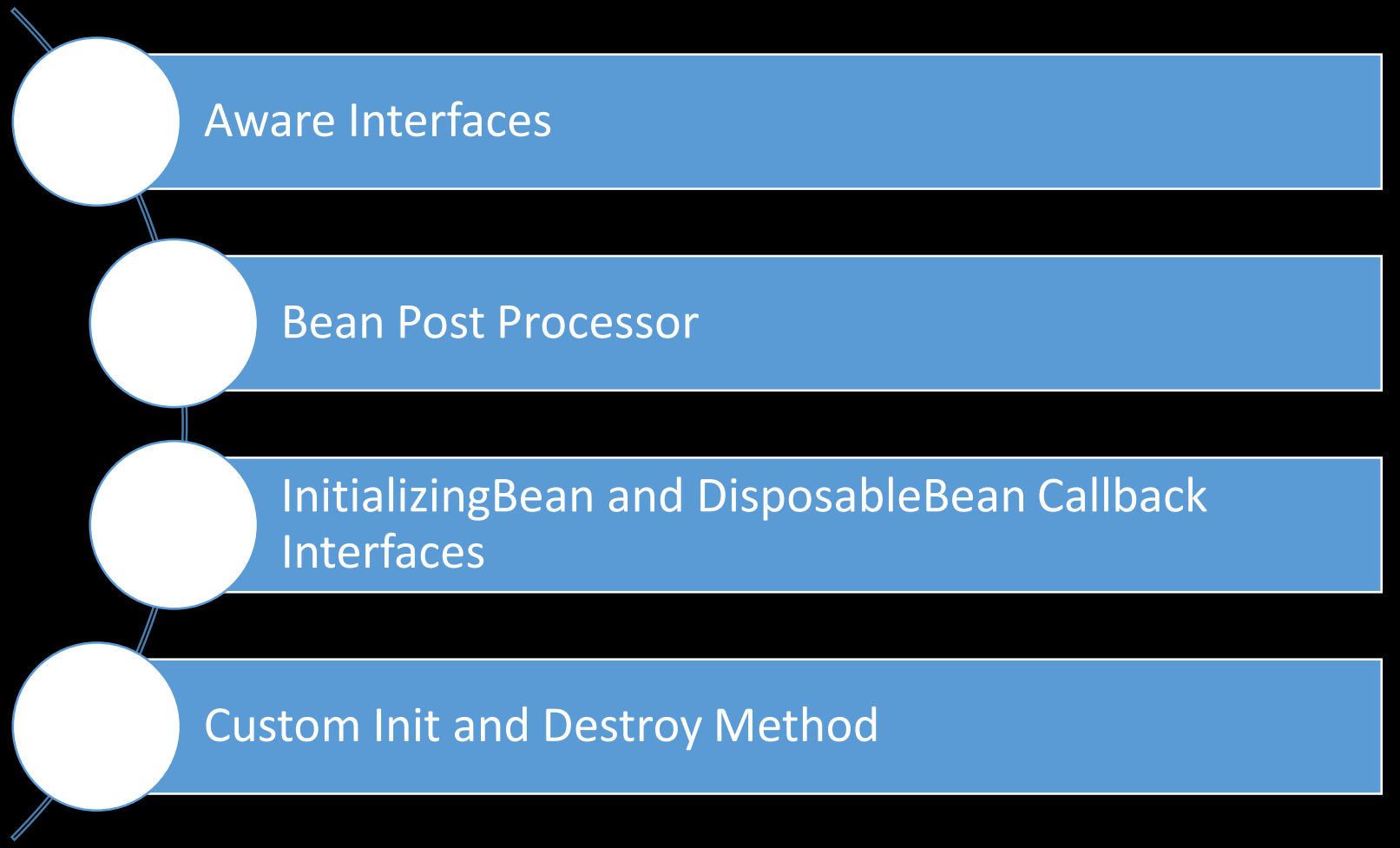
5. **Ready to Use**

Now, Bean is created and injected all the dependencies and should be executed all the Aware and callback methods implementation. Bean is ready to serve.

Bean destruction stages

1. **Pre-destroy**
Spring provides callback methods to execute custom application-specific logic and clean-ups before destroying a bean from ApplicationContext.
2. **Bean Destroyed**
Bean will be removed or destroyed from and JVM memory.

Ways to control bean lifecycle



Why Would I Need to Hook into the Bean Lifecycle?

Some of the use cases

- Assigning Default values for bean properties (ex: FILE PATH, MIN_VALUE, and MAX_VALUE) while creating bean.
- Opening and Closing the Files and Database connections as part of the bean creation and destruction.
- Loading application Metadata (ex: US state code values) information while creating a bean and clean up on bean destruction.
- Starting and Terminating a Process or Thread as part of the bean creation and destruction.
- To make sure application dependency (ex: Remote Database and External Services etc...) modules are up and running while creating bean.