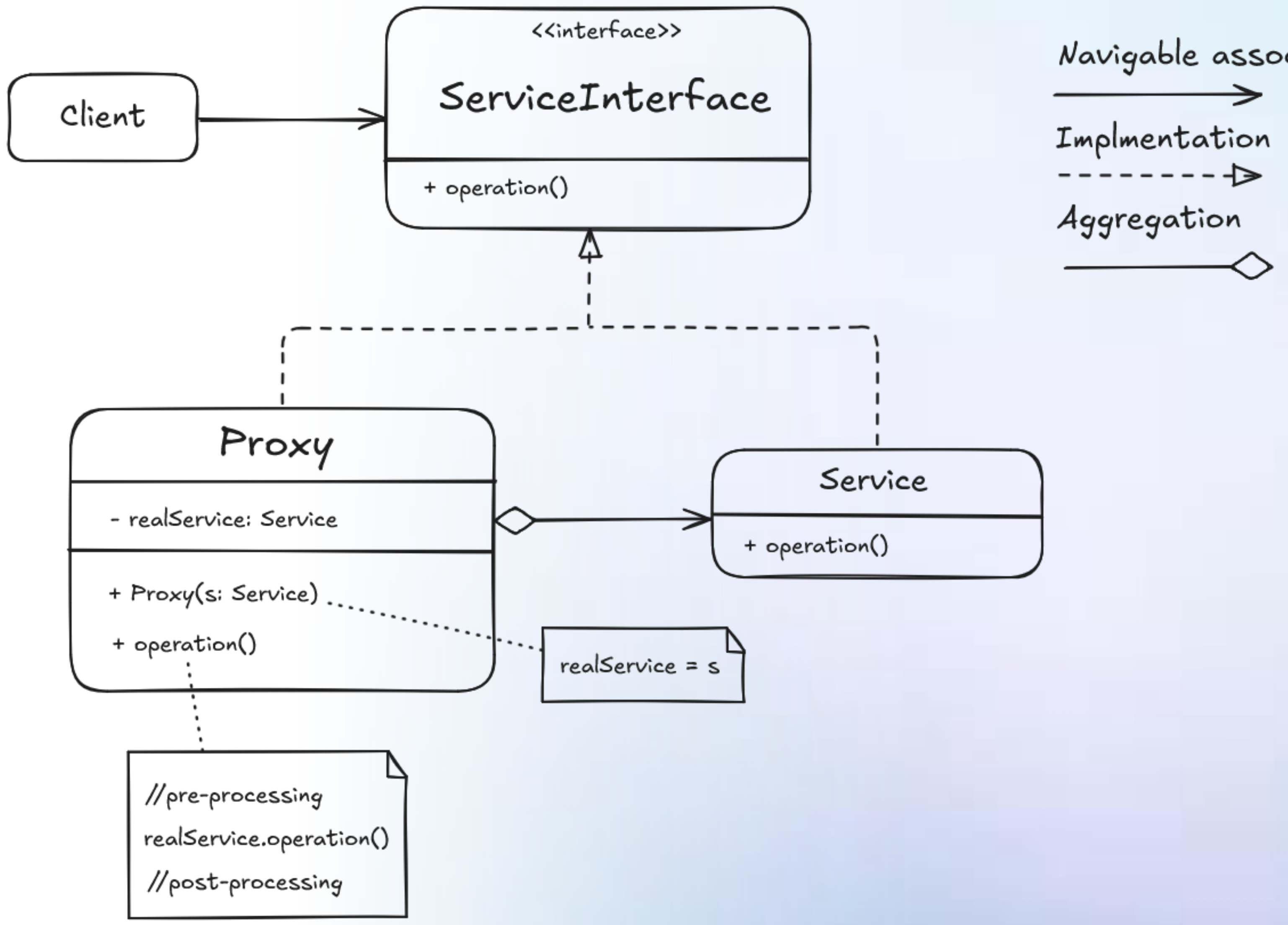
Leaf Transaction

CODEUS_

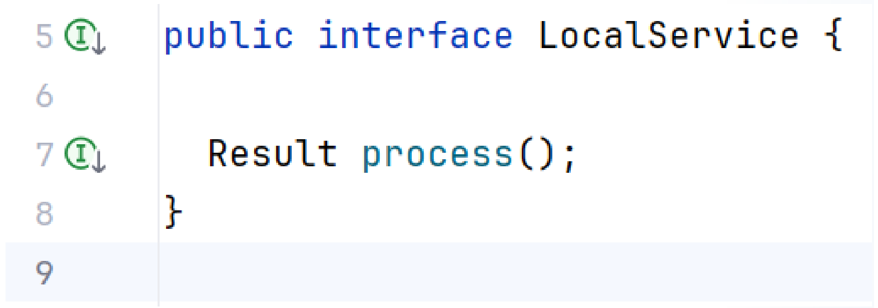
Content

Proxy Pattern

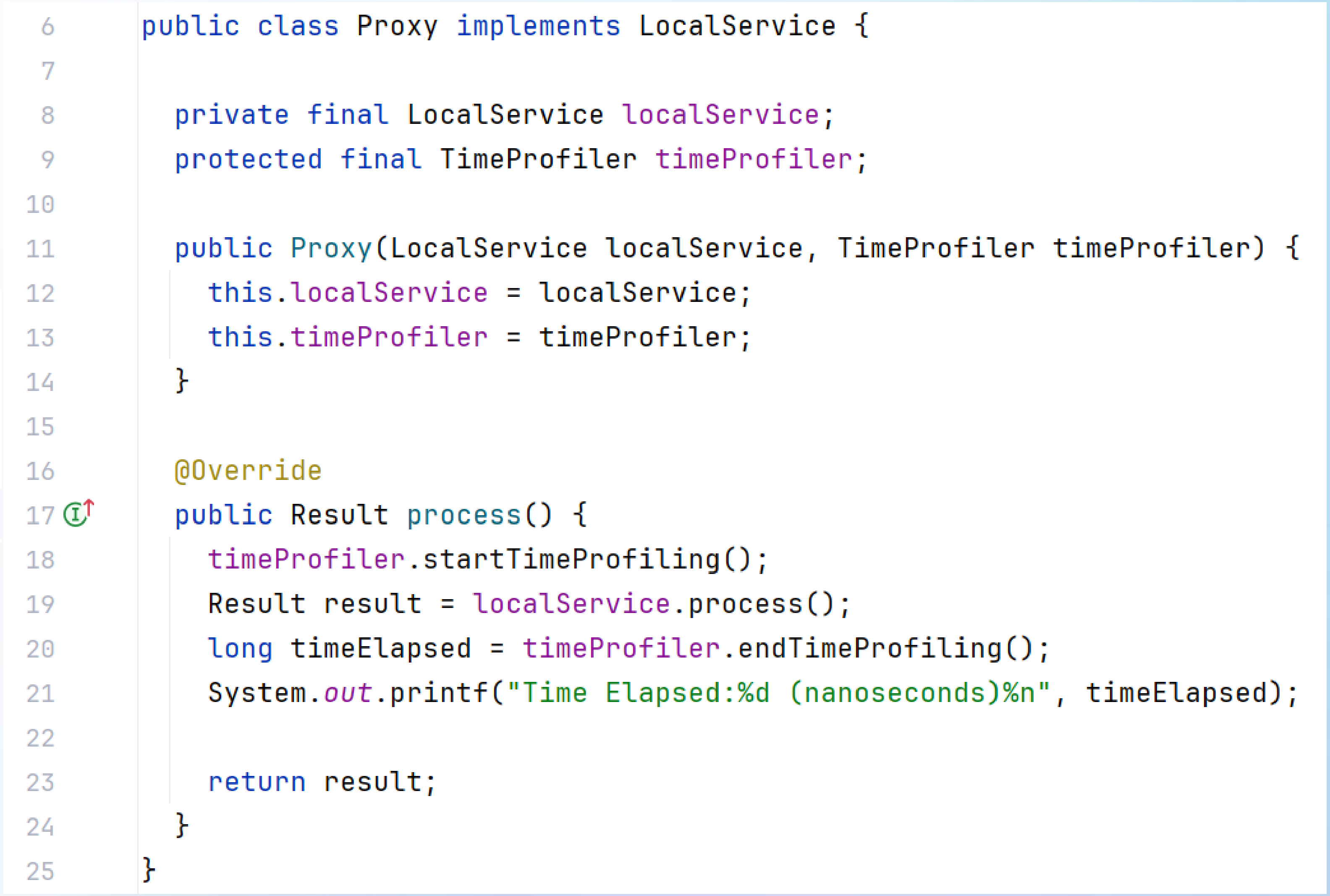


CODEUS_

Proxy Pattern: Example



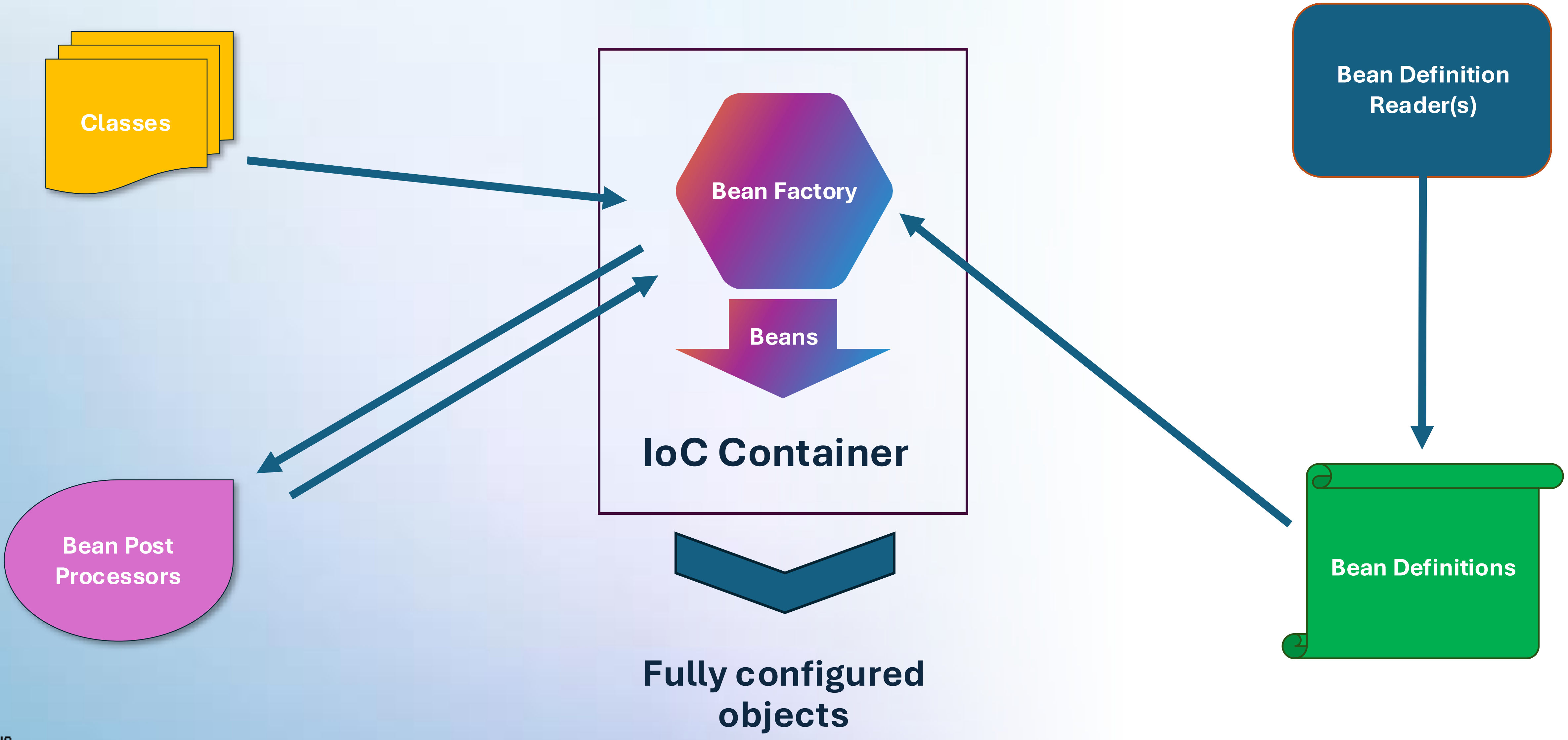
```java
public interface LocalService {

    Result process();
}
```



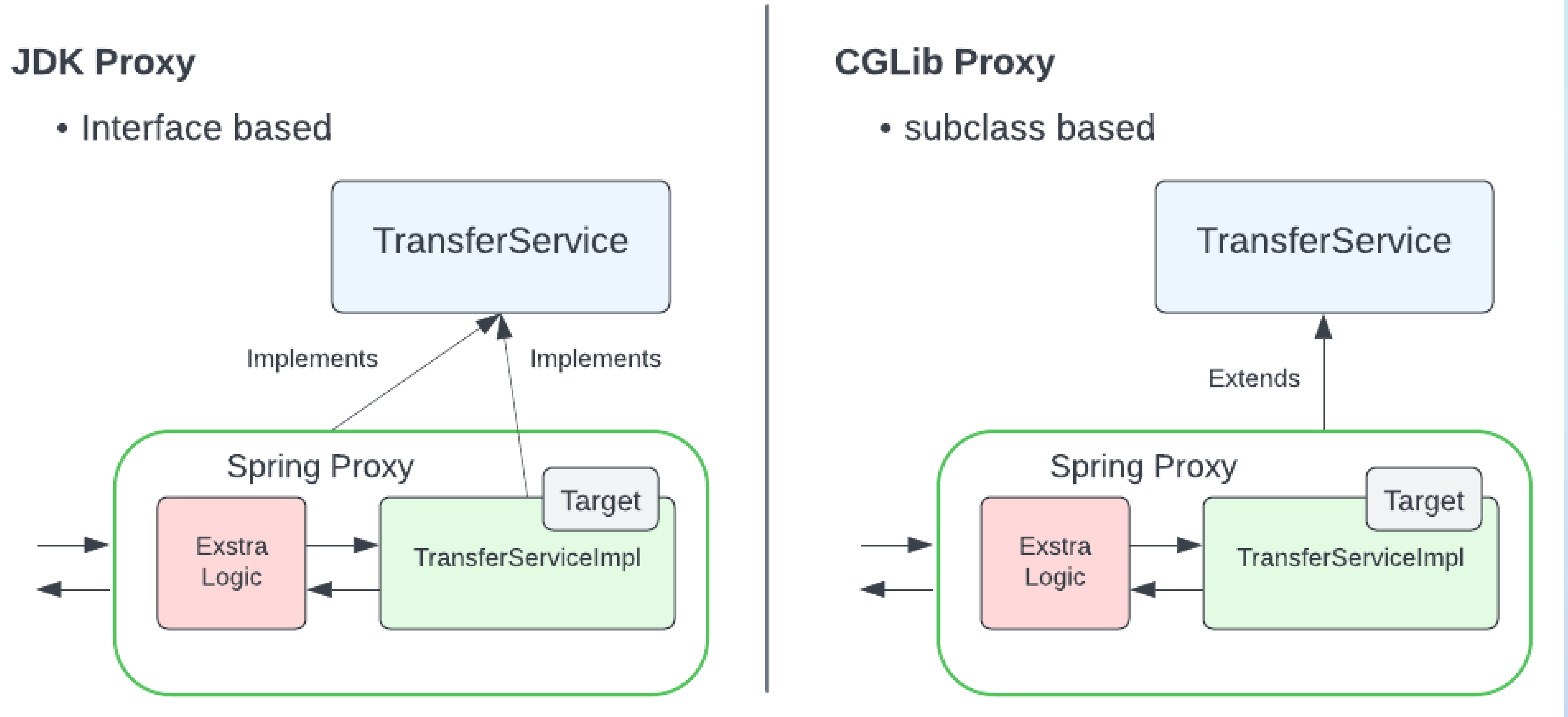
```java
public class Proxy implements LocalService {

    private final LocalService localService;
    protected final TimeProfiler timeProfiler;

    public Proxy(LocalService localService, TimeProfiler timeProfiler) {
        this.localService = localService;
        this.timeProfiler = timeProfiler;
    }

    @Override
    public Result process() {
        timeProfiler.startTimeProfiling();
        Result result = localService.process();
        long timeElapsed = timeProfiler.endTimeProfiling();
        System.out.printf("Time Elapsed:%d (nanoseconds)%n", timeElapsed);

        return result;
    }
}
```

CODEUS_

How does Spring work?

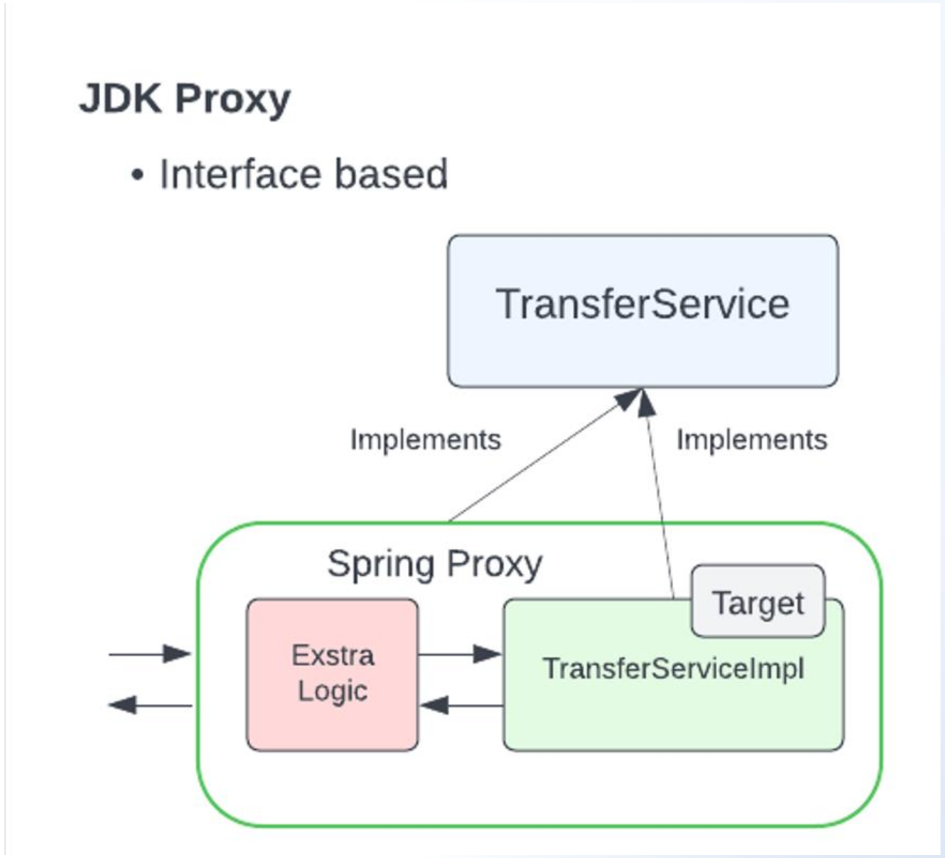


How does Spring work?



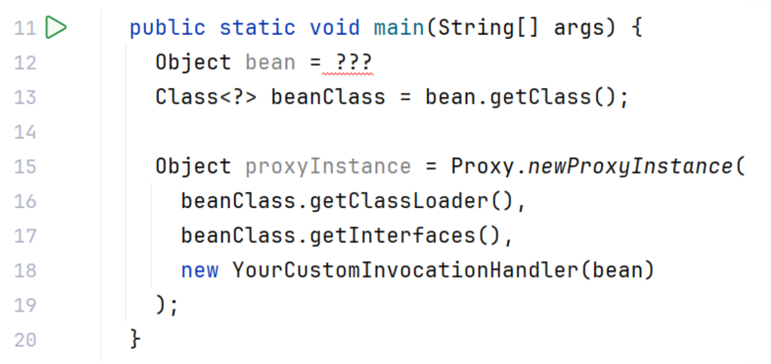
CODEUS_

Dynamic-Proxy

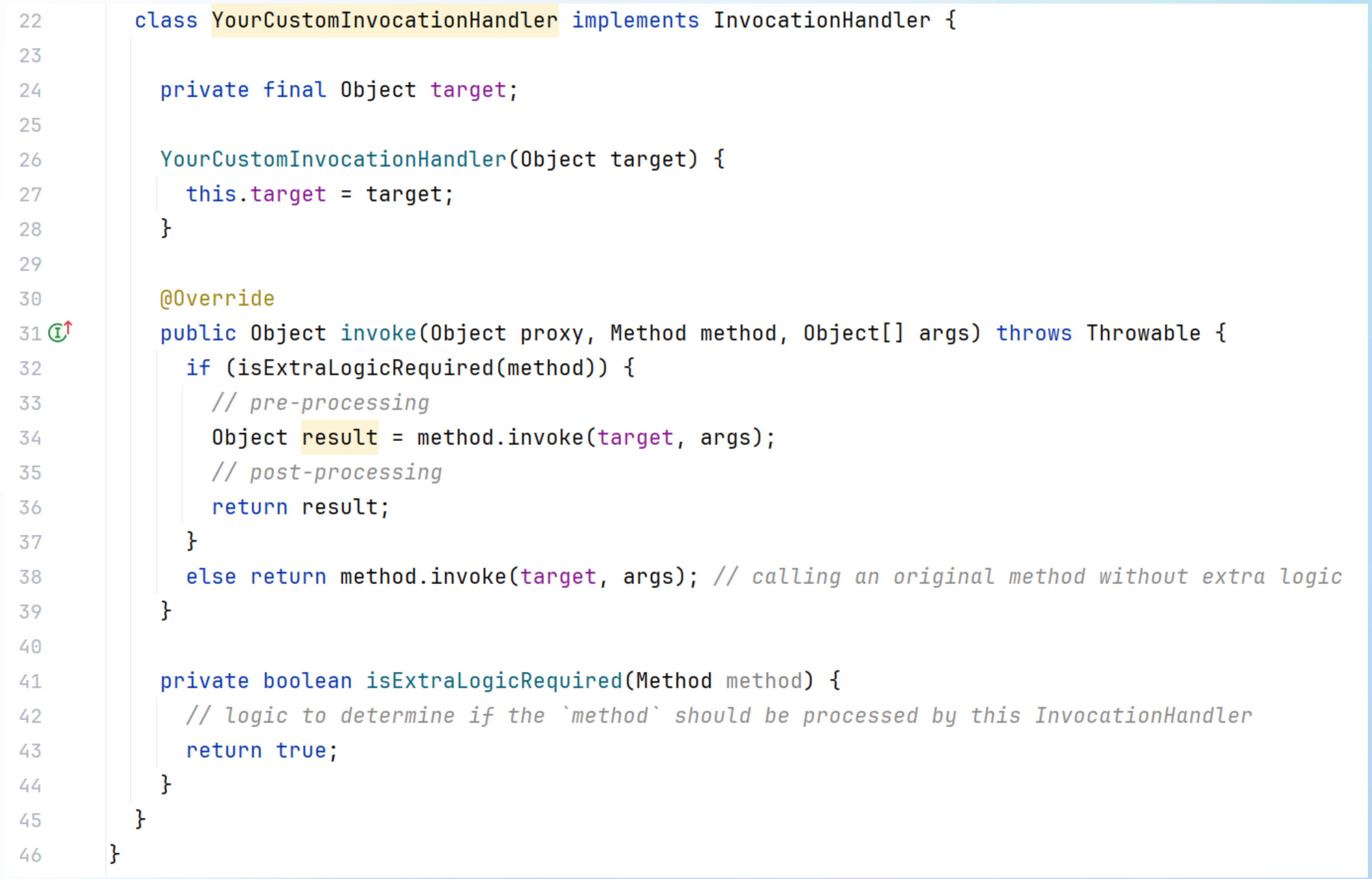


- JDK Proxy follows the same design as well known Proxy Design Pattern

Dynamic-Proxy



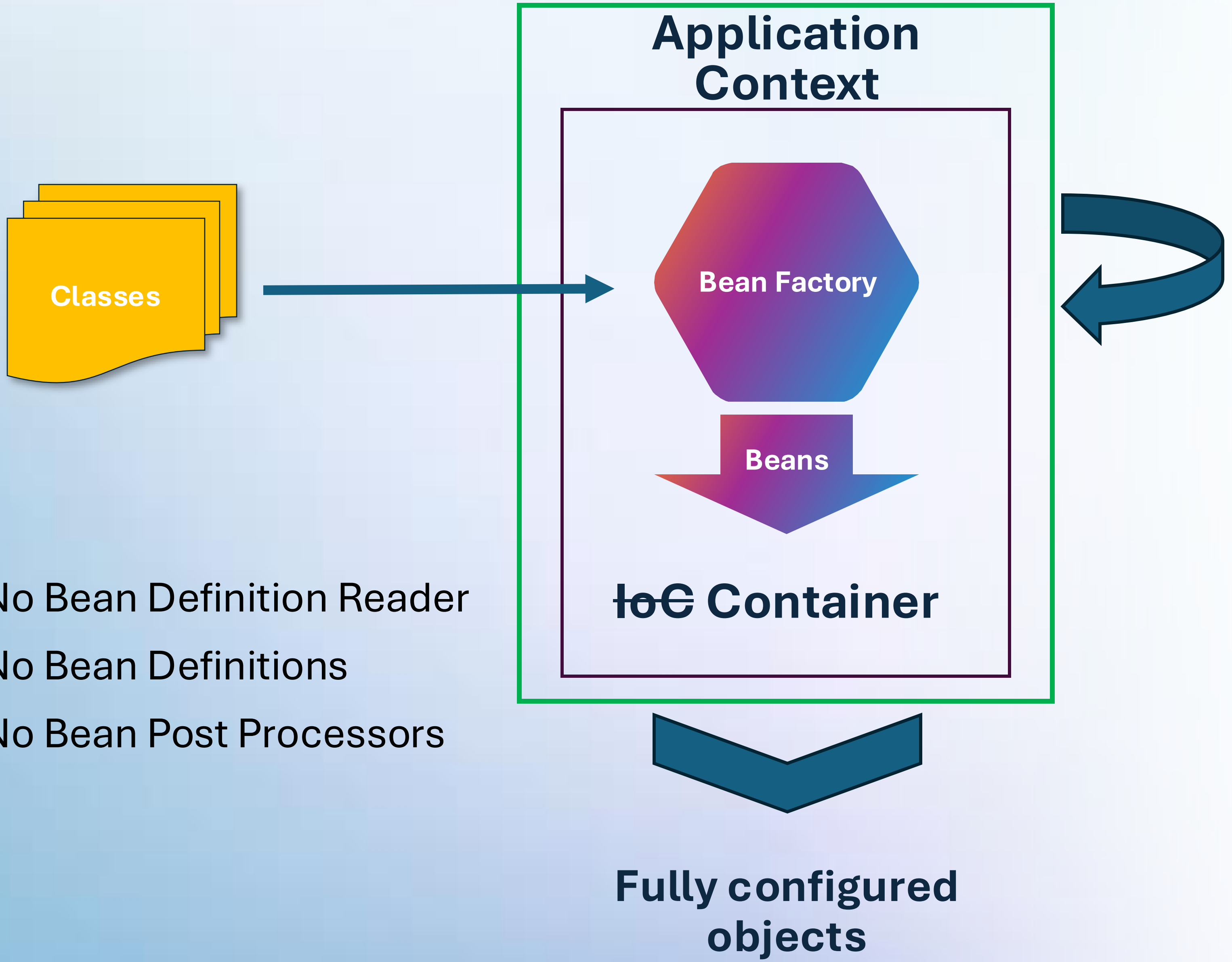
```java
public static void main(String[] args) {
  Object bean = ???
  Class<?> beanClass = bean.getClass();

  Object proxyInstance = Proxy.newProxyInstance(
    beanClass.getClassLoader(),
    beanClass.getInterfaces(),
    new YourCustomInvocationHandler(bean)
  );
}
```



```java
  class YourCustomInvocationHandler implements InvocationHandler {

    private final Object target;

    YourCustomInvocationHandler(Object target) {
      this.target = target;
    }

    @Override
    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
      if (isExtraLogicRequired(method)) {
        // pre-processing
        Object result = method.invoke(target, args);
        // post-processing
        return result;
      }
      else return method.invoke(target, args); // calling an original method without extra logic
    }

    private boolean isExtraLogicRequired(Method method) {
      // logic to determine if the `method` should be processed by this InvocationHandler
      return true;
    }
  }
}
```

CODEUS_

Task Setup: Code base



- Full custom code
- Creates a limited set of preconfigured beans
- Doesn't use Spring's interfaces
- Represents only part of the Spring functionality

- !No Bean Definition Reader
- !No Bean Definitions
- !No Bean Post Processors

CODEUS_

Task Setup: Goal

Replace manual transaction handling in the task code with a Proxy.

There are 3 main steps:

- Create @Transactional annotation.
- Create Proxy wrapper
- Implement Proxy wrapping logic for objects that have methods annotated with @Transactional

There is a set of test cases to verify your solution.

Note: you can start with basic proxy and later proceed with dynamic proxy.

Thank you

- Author: Serhii Kravchuk
- My LinkedIn
- Date: 13 December 2024
- Join Codeus community in Discord
- Join Codeus community in LinkedIn