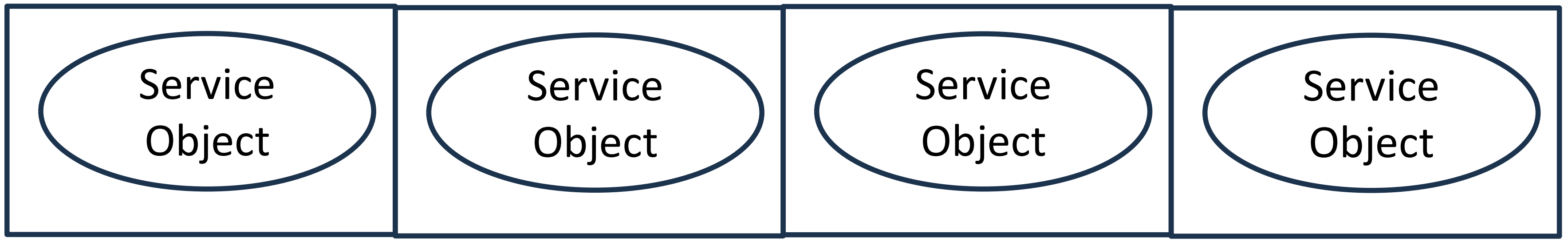
ASD PROJECT: WRITE YOU OWN SPRING BOOT FRAMEWORK

Find and instantiate @Service classes

1. Find all classes with @Service
2. Instantiate them
3. Add the objects to the service list



Field injection

- By Type
  1. Loop over list of all service classes
  2. Find field with @Autowired annotation
  3. Get the type of the field
  4. Find object with this type in the service list
  5. Set the field (field.set())
- By Name
  1. Loop over list of all service classes
  2. Find field with @Autowired + @Qualifier annotation
  3. Get the name from the @Qualifier annotation
  4. Find object with this name in the service list (use map)
  5. Set the field (field.set())

Value injection

1. Loop over list of all service classes
2. Find field with @Value(name=…) annotation
3. Get the name of the attribute
4. Get the property value of that name from the *application.properties* file
5. Set the field (field.set())

Setter injection

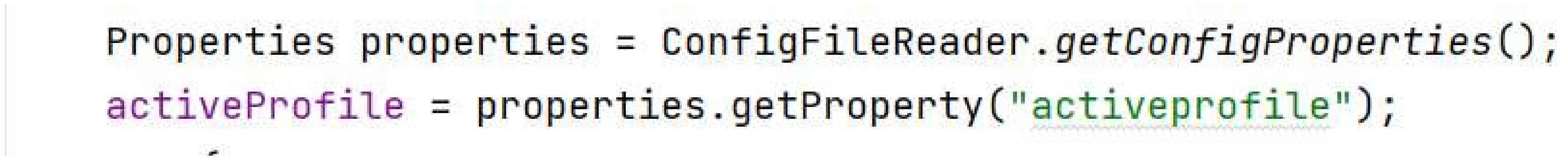
1. Loop over list of all service classes
2. Find method with @Autowired annotation
3. Get the type of the parameter
4. Find object with this type in the service list
5. Call the setter method (method.invoke())

Constructor injection

1. Loop over list of all service classes
2. Find constructor with @Autowired annotation
3. Get the type of the parameter
4. Find object-to-inject with this type in the service list
5. Instantiate the object with the object-to-inject as parameter
6. Replace existing service object with the just created service object in the service list

Profiles

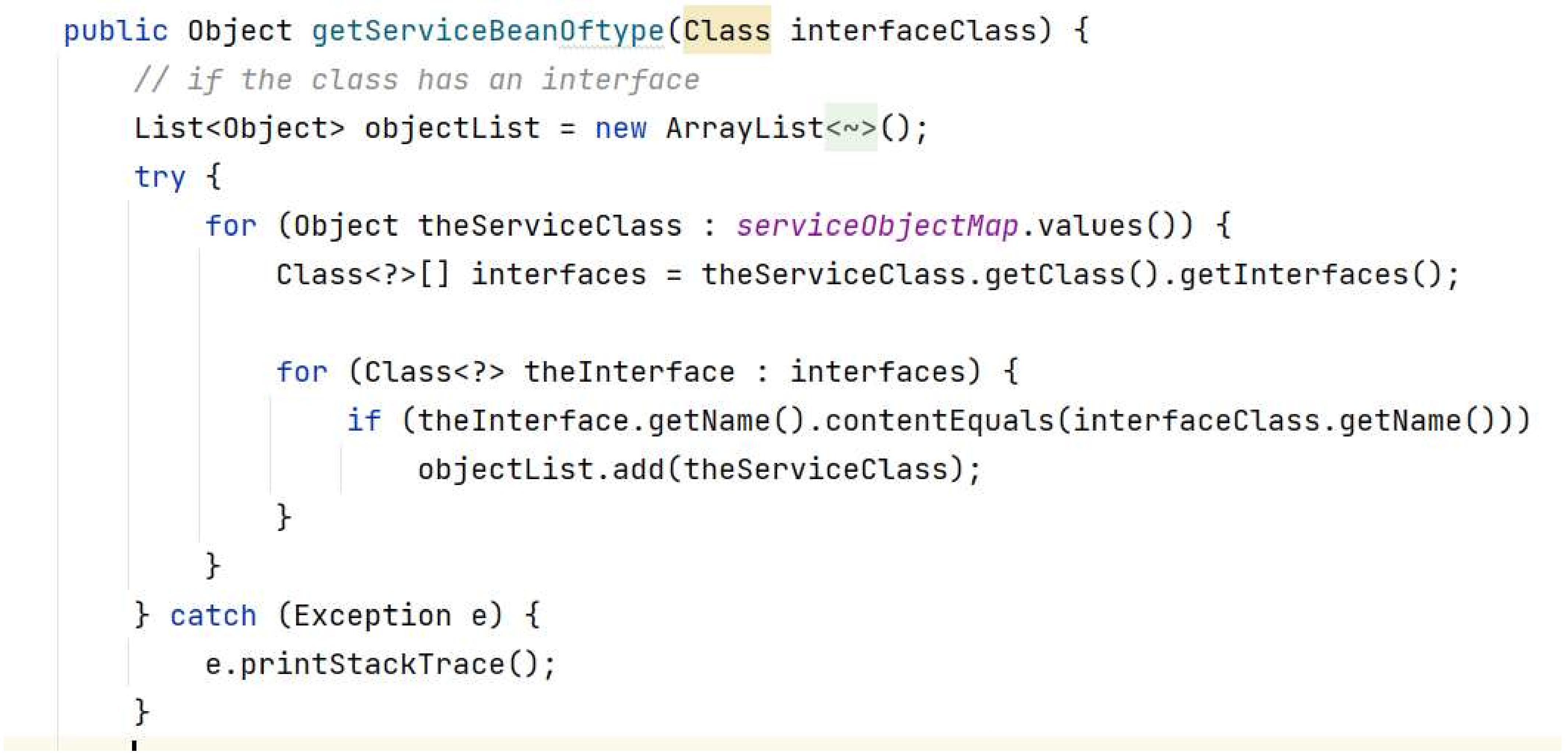
1. Get the active profile from *application.properties*



```
Properties properties = ConfigFileReader.getConfigProperties();
activeProfile = properties.getProperty("activeprofile");
```

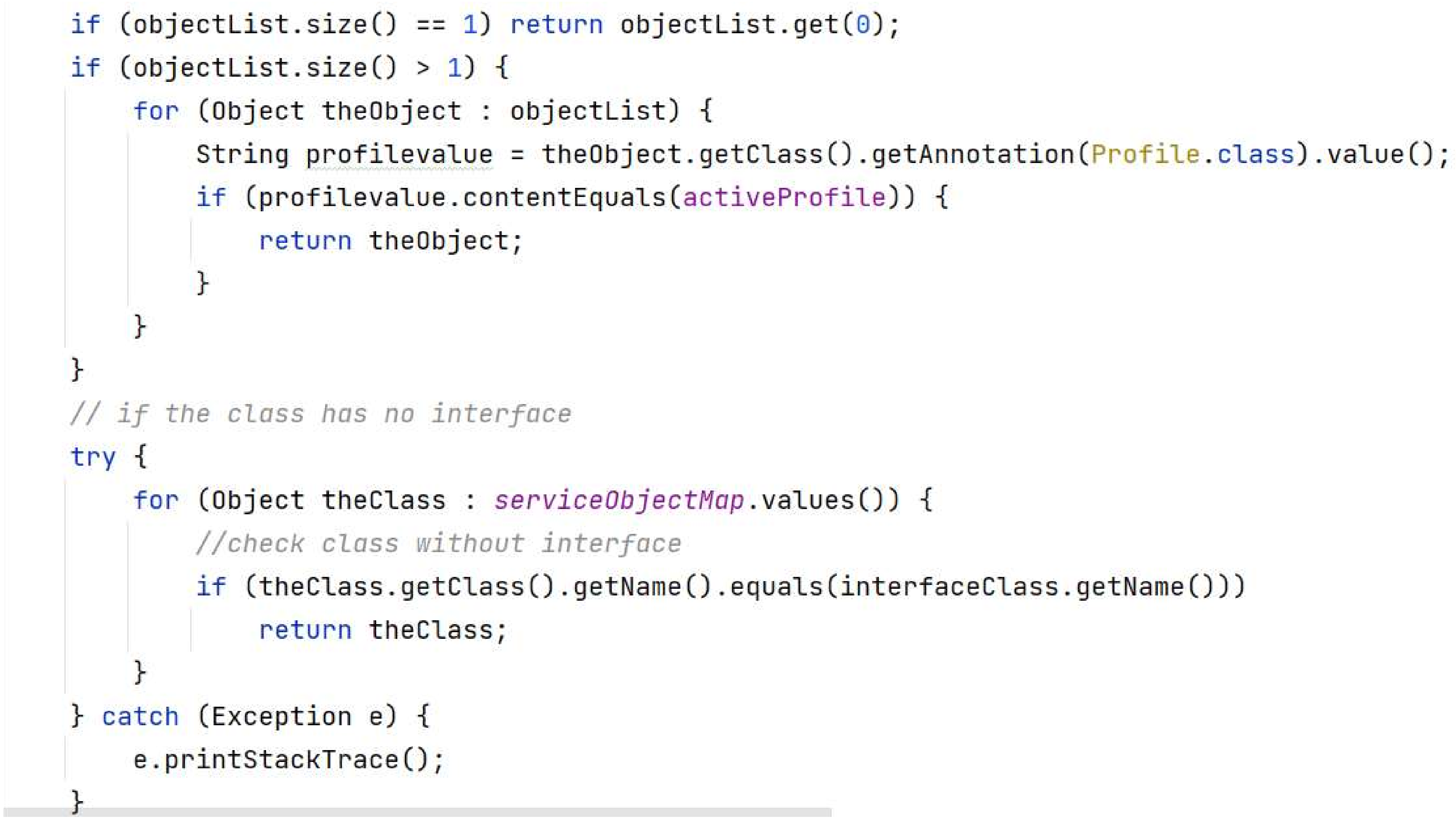
2. Method *getServiceBeanOfType()*
   1. Find all objects with the provided interface type
   2. If we found 1 object, return this object
   3. If we found multiple objects, return the object with the active profile
   4. If we did not found an object with the provided interface type
      a) Find and return the object of the provided class type

Profiles



```java
public Object getServiceBeanOftype(Class interfaceClass) {
    // if the class has an interface
    List<Object> objectList = new ArrayList<~>();
    try {
        for (Object theServiceClass : serviceObjectMap.values()) {
            Class<?>[] interfaces = theServiceClass.getClass().getInterfaces();

            for (Class<?> theInterface : interfaces) {
                if (theInterface.getName().contentEquals(interfaceClass.getName()))
                    objectList.add(theServiceClass);
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
```

Profiles



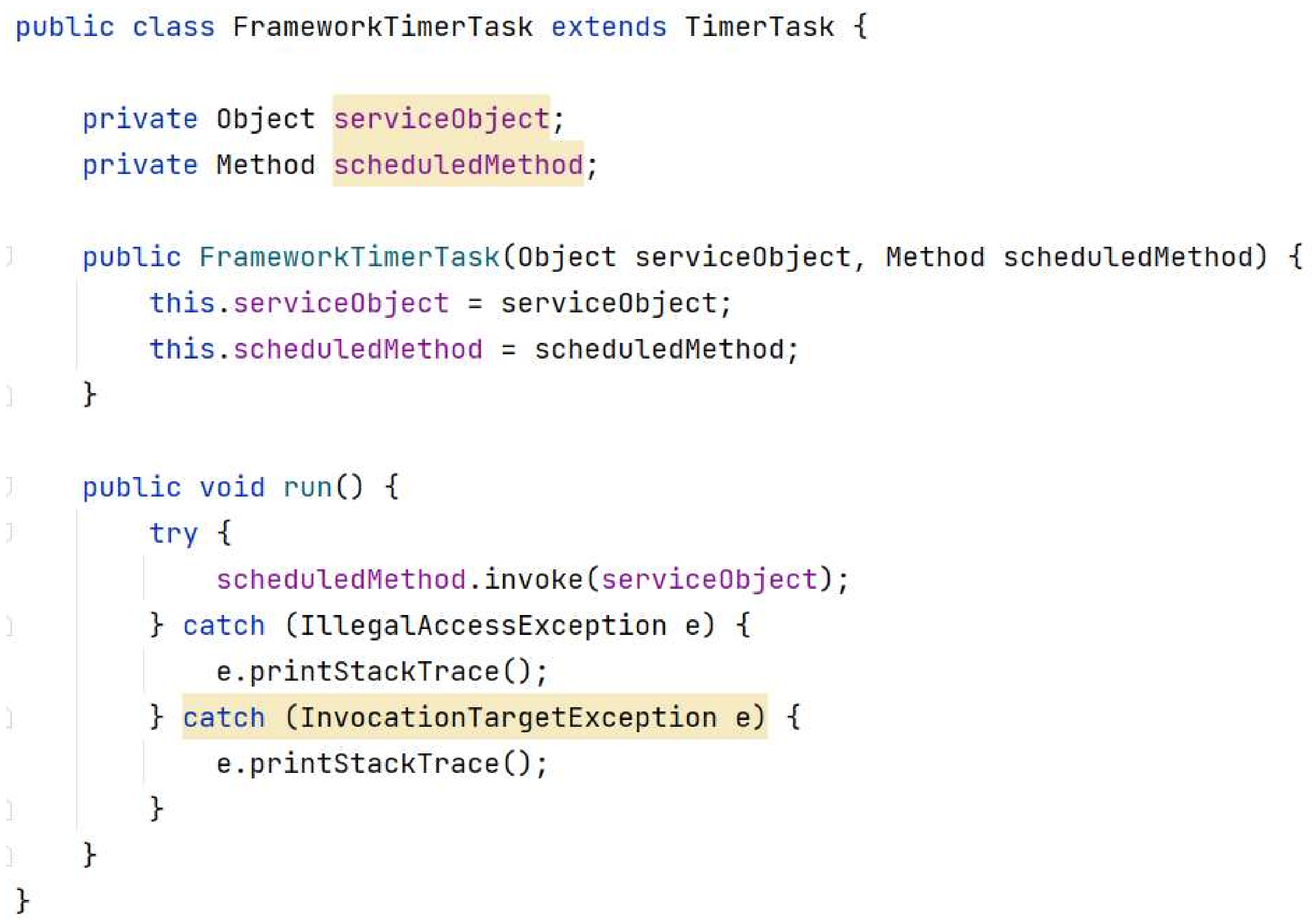
```
if (objectList.size() == 1) return objectList.get(0);
if (objectList.size() > 1) {
    for (Object theObject : objectList) {
        String profilevalue = theObject.getClass().getAnnotation(Profile.class).value();
        if (profilevalue.contentEquals(activeProfile)) {
            return theObject;
        }
    }
}
// if the class has no interface
try {
    for (Object theClass : serviceObjectMap.values()) {
        //check class without interface
        if (theClass.getClass().getName().equals(interfaceClass.getName()))
            return theClass;
    }
} catch (Exception e) {
    e.printStackTrace();
}
```

Scheduling

1. Create a generic TimerTask(Object, method)
2. Find methods with @Scheduled annotation
   - With fixedrate attribute
     1. Get fixedrate attribute value
     2. Create timer object
     3. Start timer with the object and scheduled method
   - With cron attribute
     1. Get cron attribute value
     2. Create timer object
     3. Start timer with the object and scheduled method

Scheduling

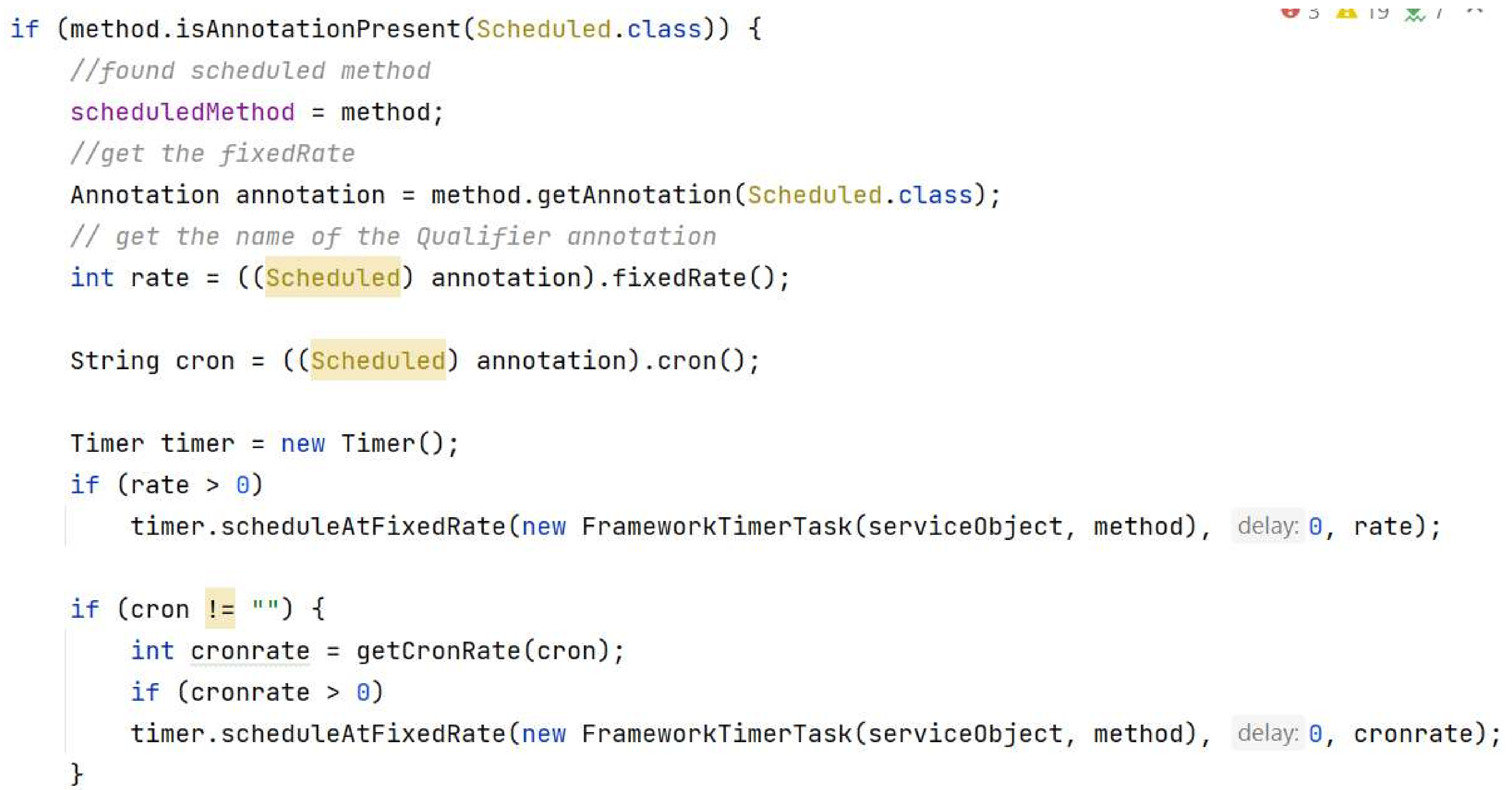
1. Create a generic TimerTask(Object, method)



```java
public class FrameworkTimerTask extends TimerTask {

    private Object serviceObject;
    private Method scheduledMethod;

    public FrameworkTimerTask(Object serviceObject, Method scheduledMethod) {
        this.serviceObject = serviceObject;
        this.scheduledMethod = scheduledMethod;
    }

    public void run() {
        try {
            scheduledMethod.invoke(serviceObject);
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        } catch (InvocationTargetException e) {
            e.printStackTrace();
        }
    }
}
```

Scheduling

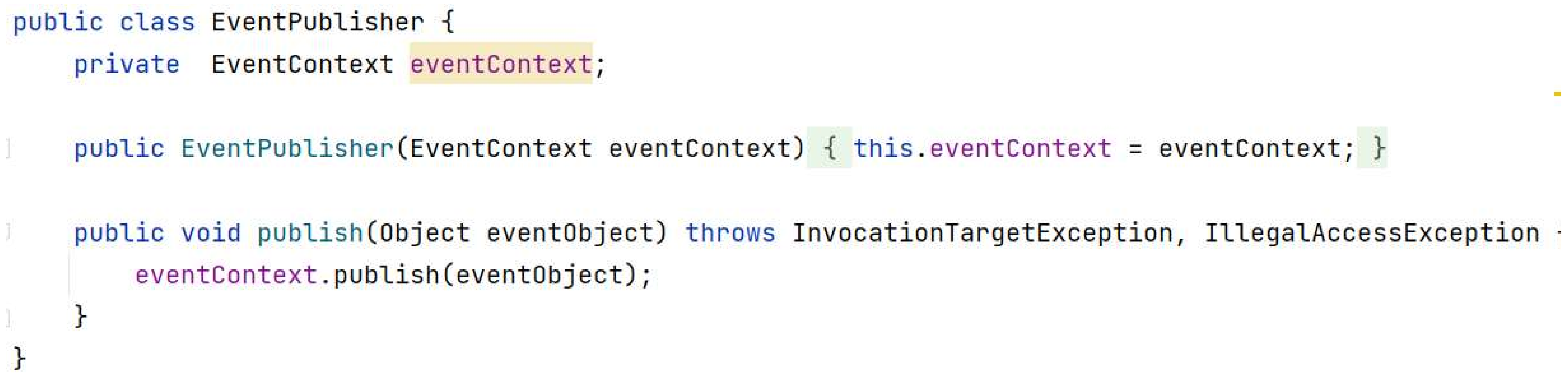
1. Get cron attribute value
2. Create timer object
3. Start timer with the object and scheduled method



```java
if (method.isAnnotationPresent(Scheduled.class)) {
    //found scheduled method
    scheduledMethod = method;
    //get the fixedRate
    Annotation annotation = method.getAnnotation(Scheduled.class);
    // get the name of the Qualifier annotation
    int rate = ((Scheduled) annotation).fixedRate();

    String cron = ((Scheduled) annotation).cron();

    Timer timer = new Timer();
    if (rate > 0)
        timer.scheduleAtFixedRate(new FrameworkTimerTask(serviceObject, method),  delay: 0, rate);

    if (cron != "") {
        int cronrate = getCronRate(cron);
        if (cronrate > 0)
        timer.scheduleAtFixedRate(new FrameworkTimerTask(serviceObject, method),  delay: 0, cronrate);
    }
```

Events

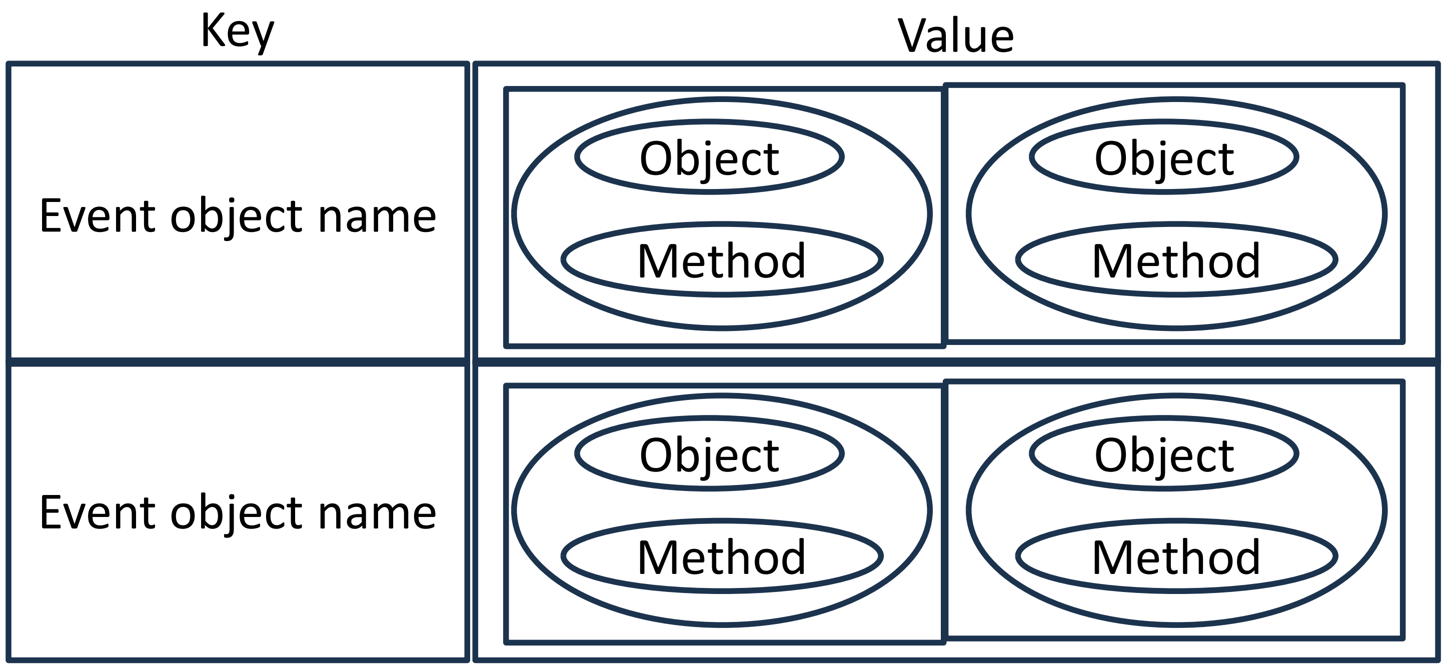
1. Create an EventPublisher that can publish() any Object



```java
public class EventPublisher {
    private  EventContext eventContext;

    public EventPublisher(EventContext eventContext) { this.eventContext = eventContext; }

    public void publish(Object eventObject) throws InvocationTargetException, IllegalAccessException
        eventContext.publish(eventObject);
    }
}
```

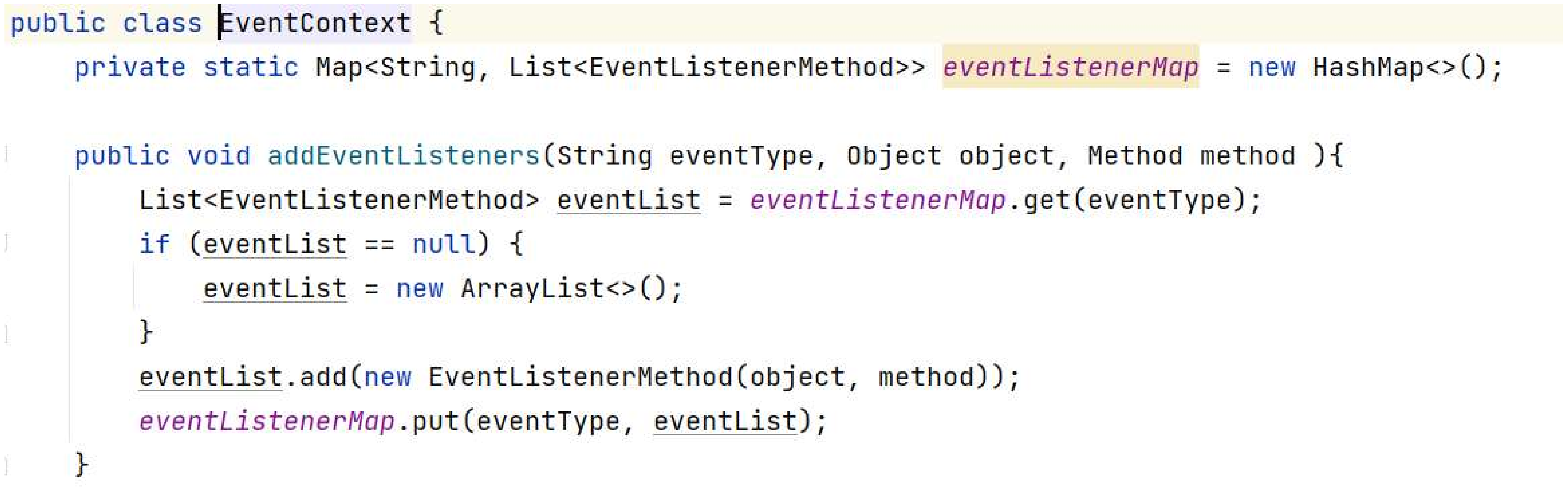
Events

2. Create an EventContext
    a) Add all @EventListener methods in a map
    b) Write a publish() method that invokes all @EventListener methods



Events

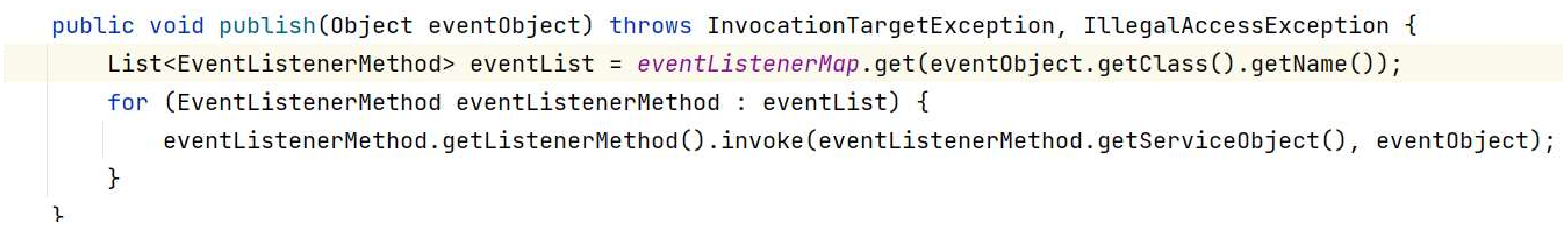
2. Create an EventContext
    a) Add all @EventListener methods in a map



```
public class EventContext {
    private static Map<String, List<EventListenerMethod>> eventListenerMap = new HashMap<>();

    public void addEventListeners(String eventType, Object object, Method method ){
        List<EventListenerMethod> eventList = eventListenerMap.get(eventType);
        if (eventList == null) {
            eventList = new ArrayList<>();
        }
        eventList.add(new EventListenerMethod(object, method));
        eventListenerMap.put(eventType, eventList);
    }
```

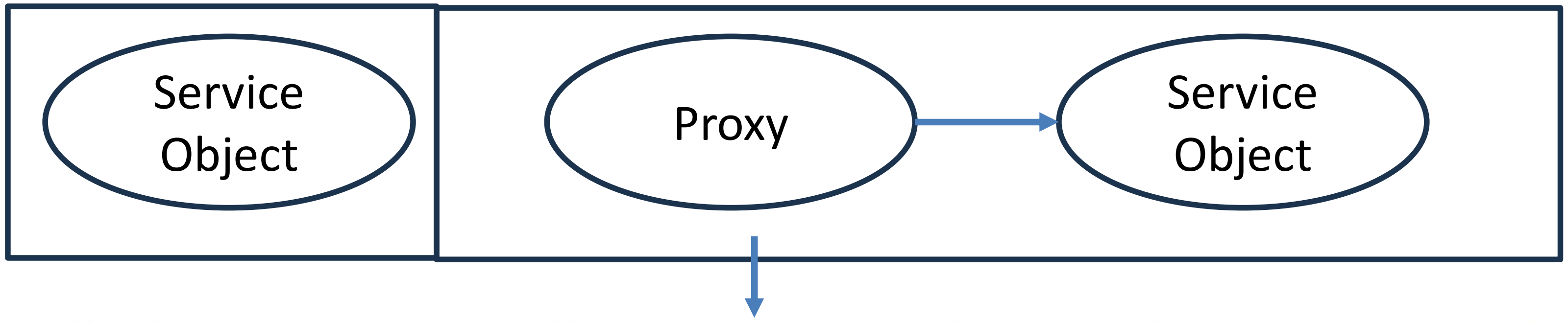
Events

2. Create an EventContext
   a) Add all @EventListener methods in a map
   b) Write a publish() method that invokes all @EventListener methods



```java
public void publish(Object eventObject) throws InvocationTargetException, IllegalAccessException {
    List<EventListenerMethod> eventList = eventListenerMap.get(eventObject.getClass().getName());
    for (EventListenerMethod eventListenerMethod : eventList) {
        eventListenerMethod.getListenerMethod().invoke(eventListenerMethod.getServiceObject(), eventObject);
    }
}
```

@Async



```
if (targetMethod != null && targetMethod.isAnnotationPresent(Async.class)) {
    System.out.println("Executing method asynchronously: " + method.getName());
    CompletableFuture.runAsync(() -> {
        try {
            method.invoke(targetObject, args);
        } catch (Throwable e) {
            e.printStackTrace();
        }
    }, Executors.newCachedThreadPool());
    return null;
} else {
    System.out.println("Executing method synchronously: " + method.getName());
    return method.invoke(targetObject, args);
}
```