Doubt Regarding Implementation

Yes, you can implement the `InitializingBean` interface in a class that itself is not a Spring bean. However, doing so would mean that the `afterPropertiesSet()` method defined by the `InitializingBean` interface will need to be manually invoked on an instance of that class.

Here's an example:

```java
public class MyInitializingBean implements InitializingBean {

    @Override
    public void afterPropertiesSet() throws Exception {
        // Initialization logic here
        System.out.println("MyInitializingBean initialized.");
    }

    public void doSomething() {
        // Some method in the class
        System.out.println("Doing something...");
    }
}
```

Now, to use the `afterPropertiesSet()` method, you would need to manually call it on an instance of `MyInitializingBean`:

```java
public class MyApp {
    public static void main(String[] args) {
        MyInitializingBean bean = new MyInitializingBean();
        bean.afterPropertiesSet(); // Manually invoking initialization
        bean.doSomething(); // Invoking other methods
    }
}
```

In this example, `MyInitializingBean` implements `InitializingBean`, but it's instantiated and used like any regular Java class. The `afterPropertiesSet()` method is manually invoked to trigger the initialization logic defined in the class.