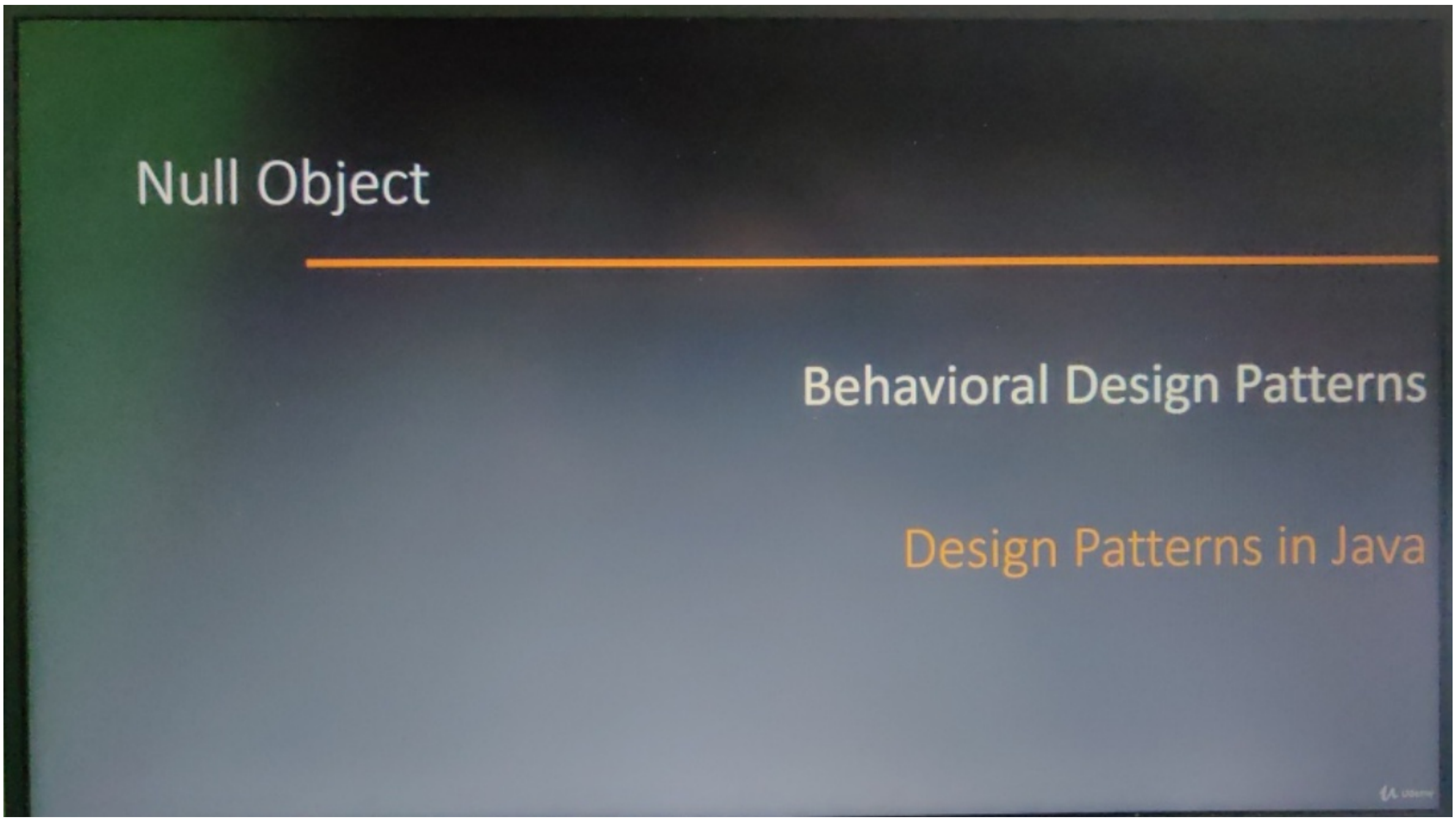
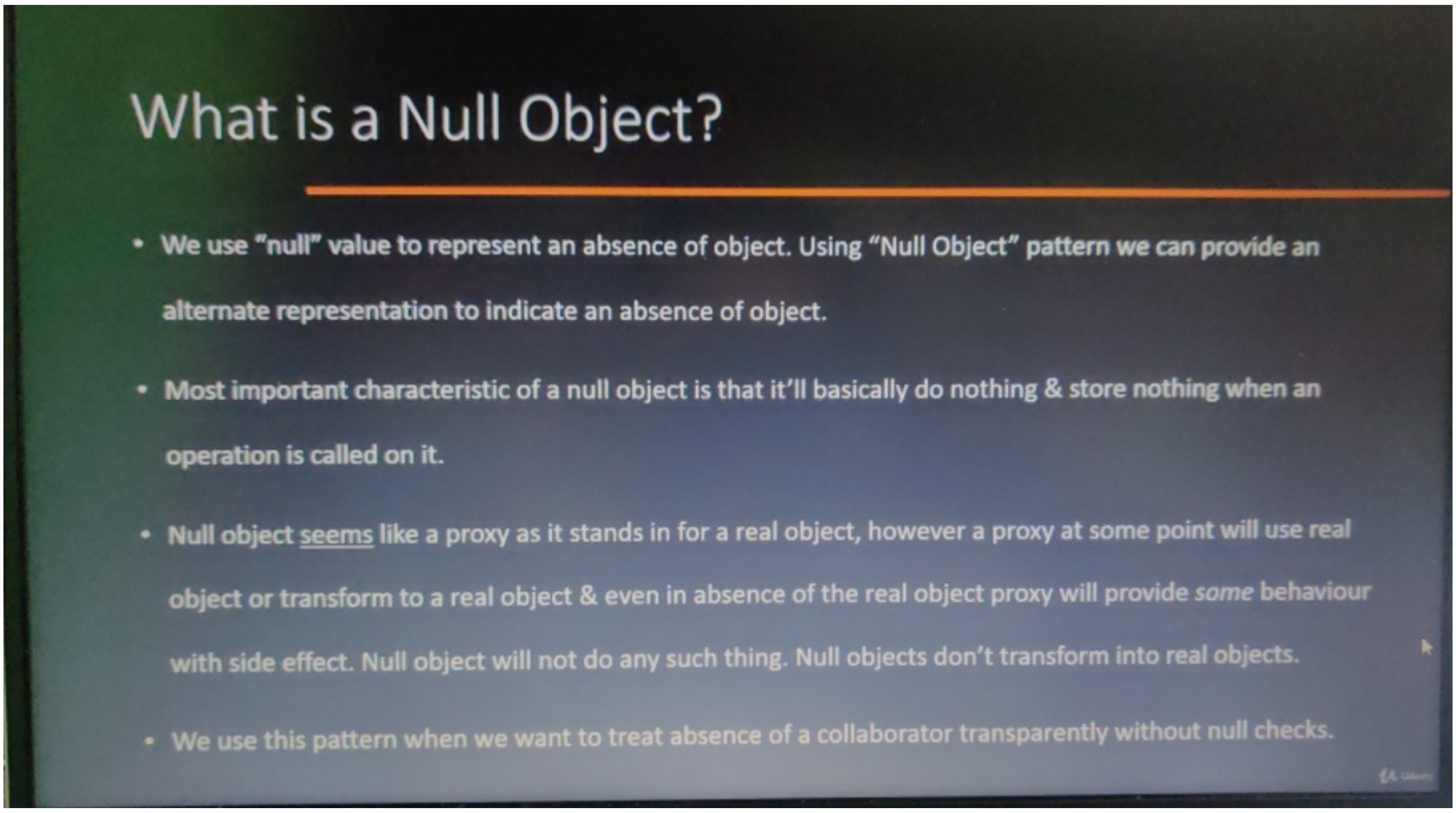


Null Object

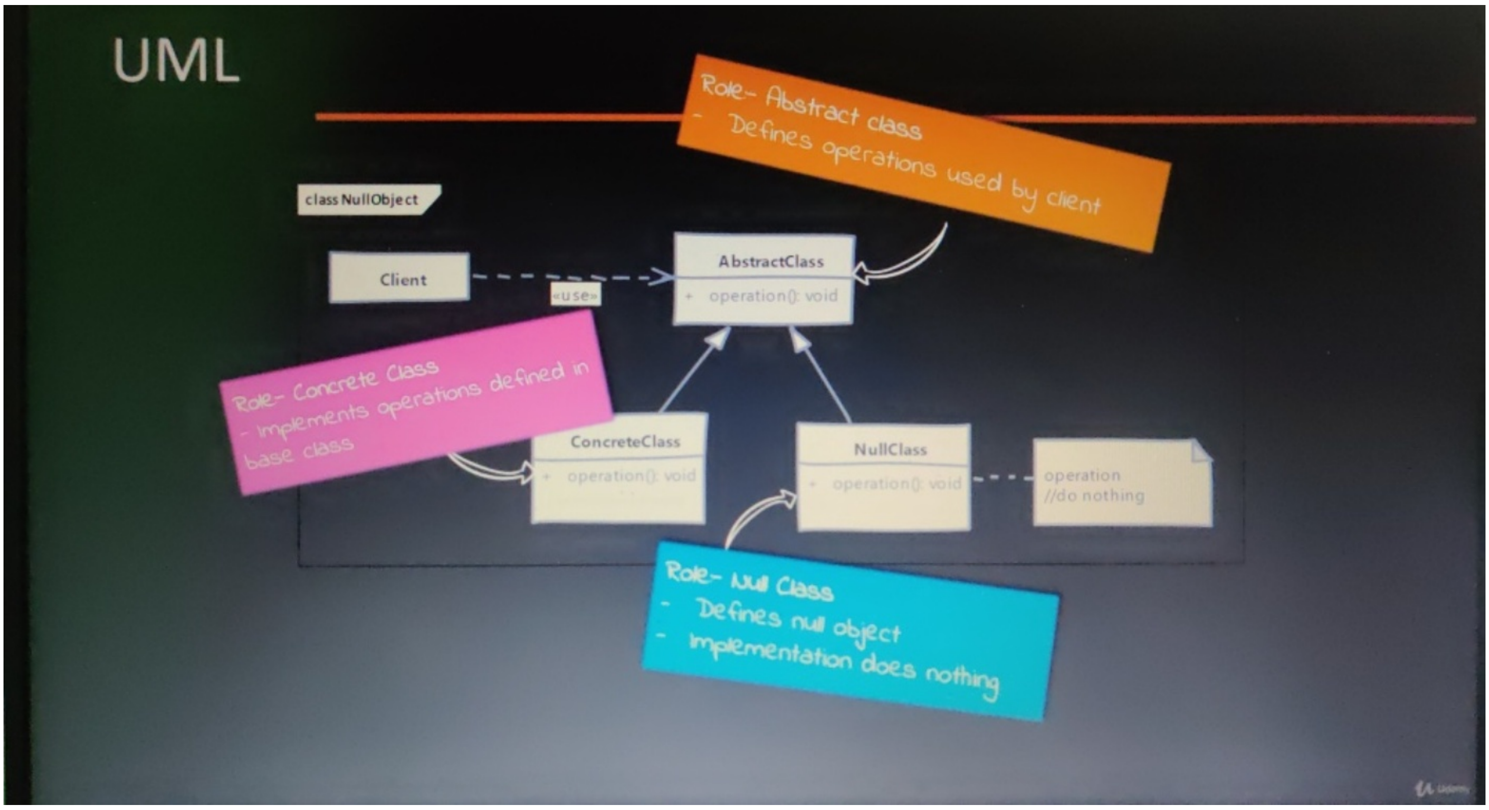
Behavioral Design Patterns

Design Patterns in Java



What is a Null Object?

- We use "null" value to represent an absence of object. Using "Null Object" pattern we can provide an alternate representation to indicate an absence of object.
- Most important characteristic of a null object is that it'll basically do nothing & store nothing when an operation is called on it.
- Null object <u>seems</u> like a proxy as it stands in for a real object, however a proxy at some point will use real object or transform to a real object & even in absence of the real object proxy will provide *some* behaviour with side effect. Null object will not do any such thing. Null objects don't transform into real objects.
- We use this pattern when we want to treat absence of a collaborator transparently without null checks.

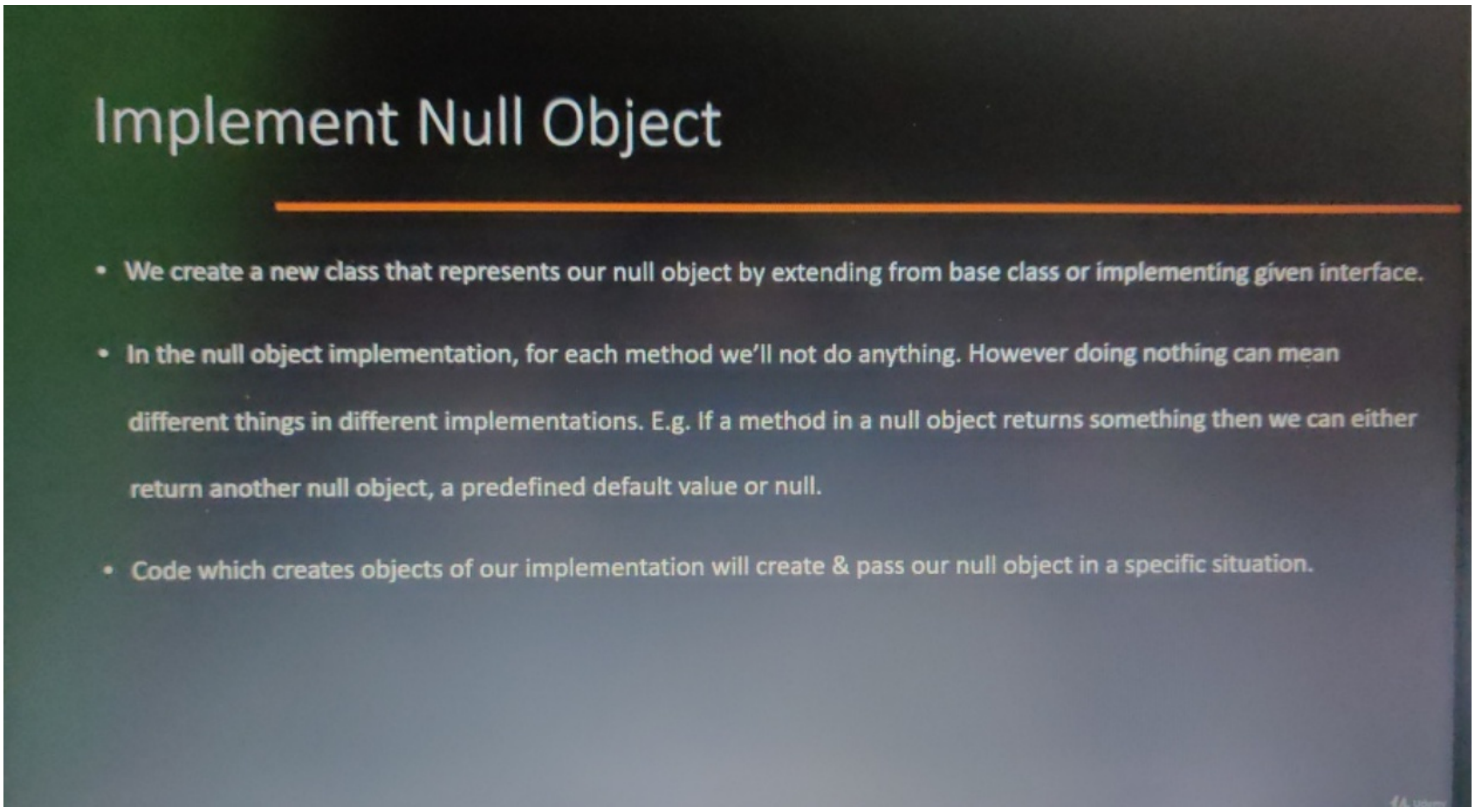


UML

Role- Abstract class
- Defines operations used by client

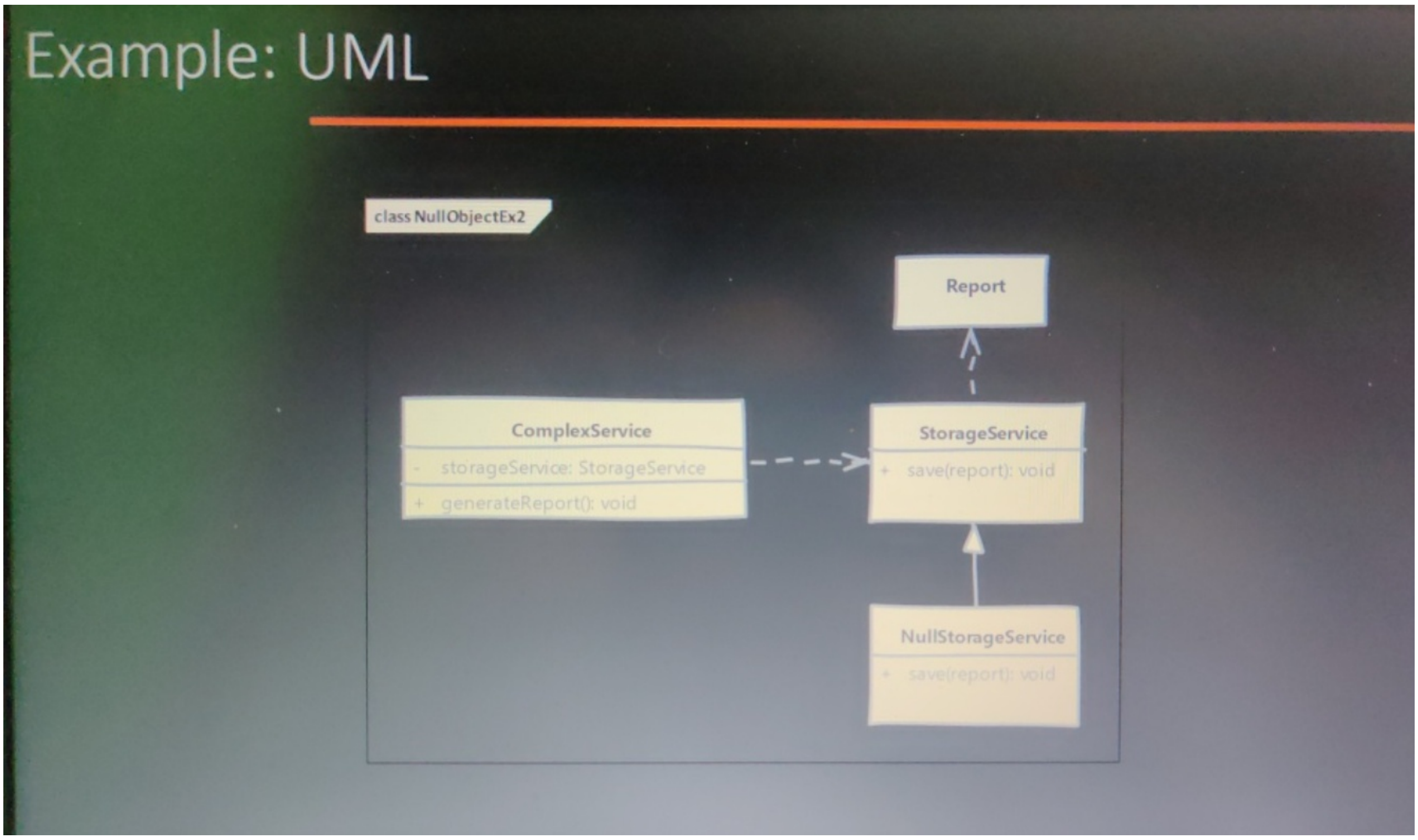
Role- Concrete Class
- Implements operations defined in base class

Role- Null Class
- Defines null object
- Implementation does nothing



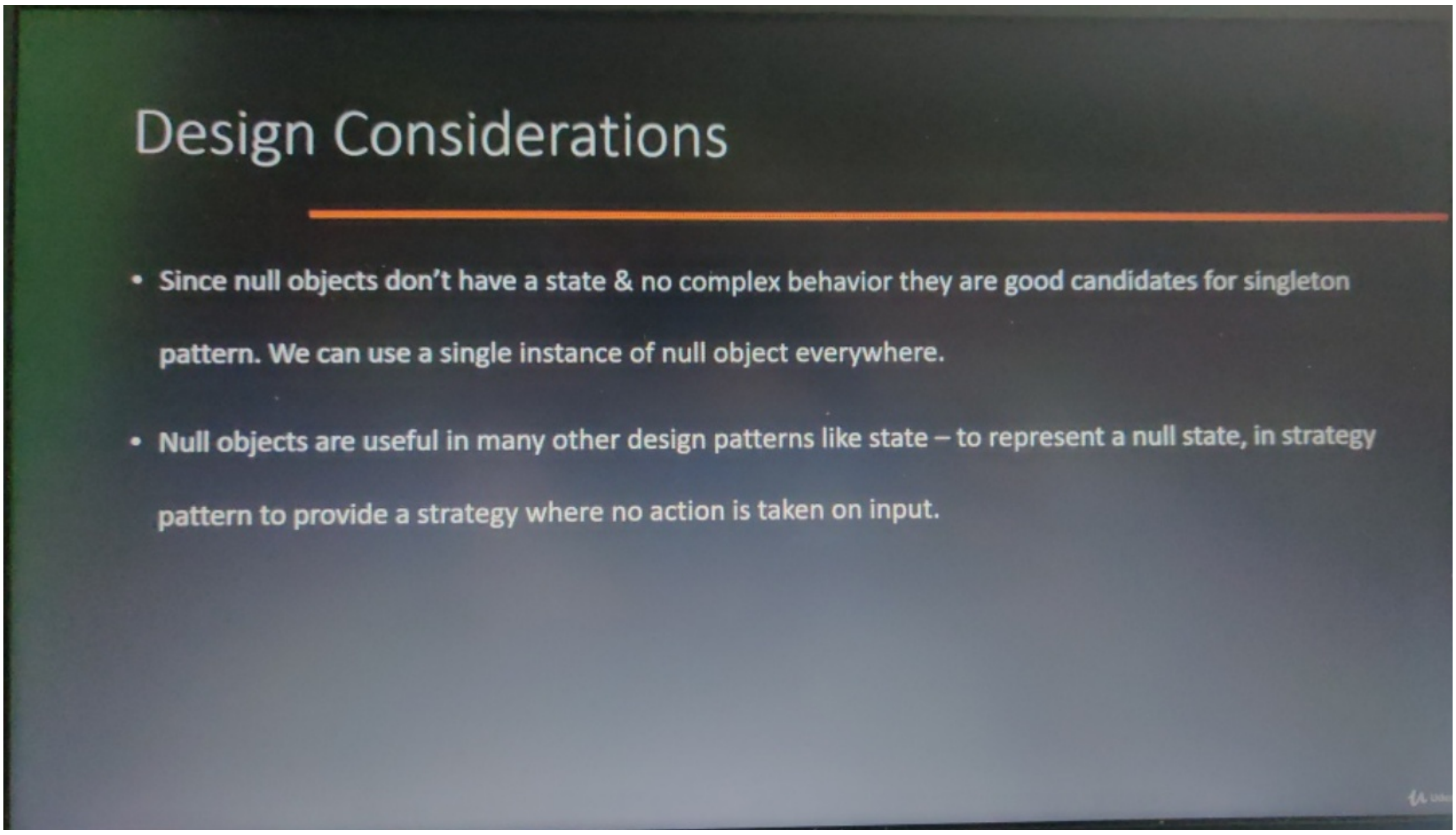
Implement Null Object

- We create a new class that represents our null object by extending from base class or implementing given interface.
- In the null object implementation, for each method we'll not do anything. However doing nothing can mean different things in different implementations. E.g. If a method in a null object returns something then we can either return another null object, a predefined default value or null.
- Code which creates objects of our implementation will create & pass our null object in a specific situation.



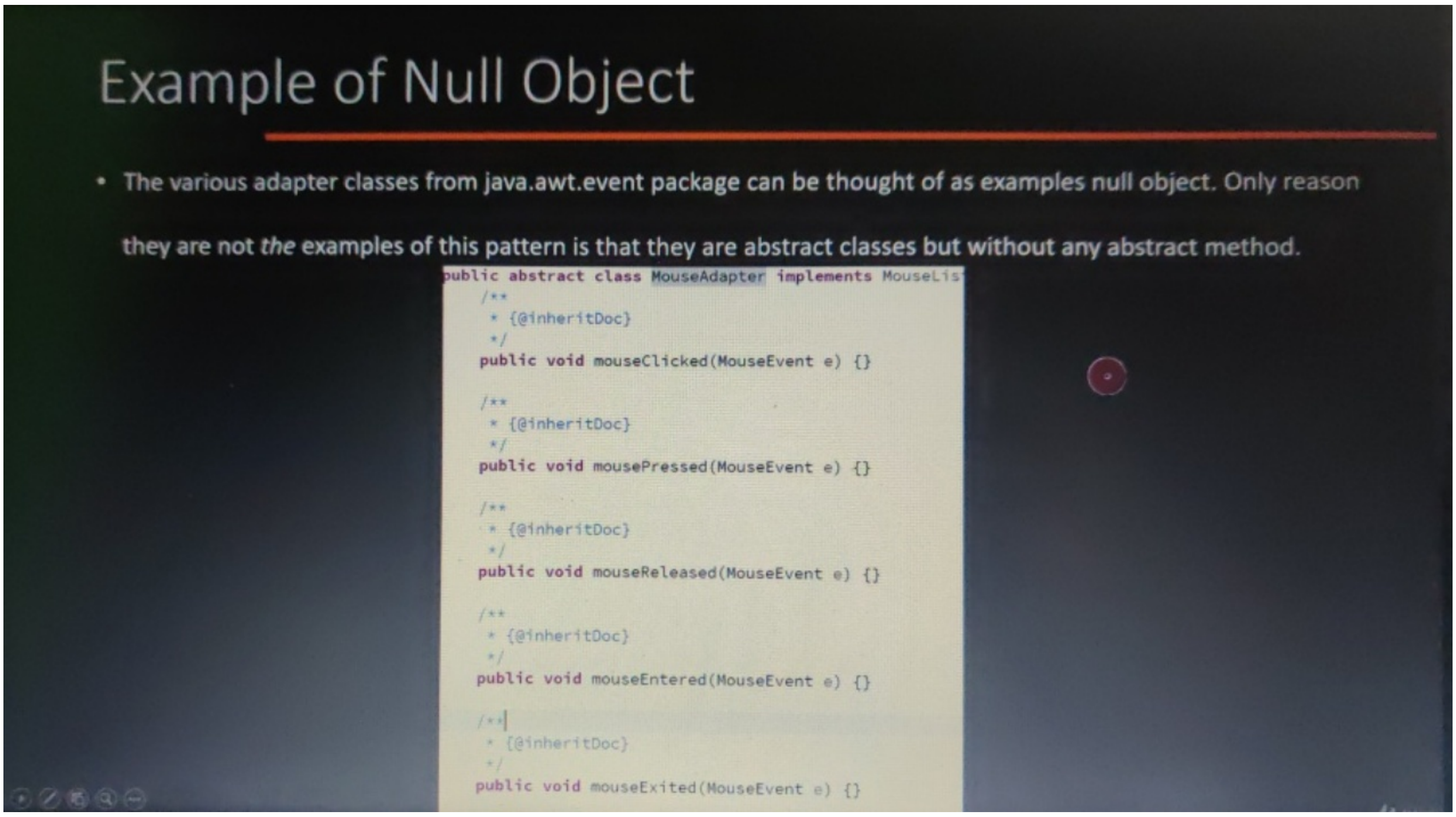
Example: UML

class NullObjectEx2
Report
ComplexService
- storageService: StorageService
+ generateReport(): void
StorageService
+ save(report): void
NullStorageService
+ save(report): void



Design Considerations

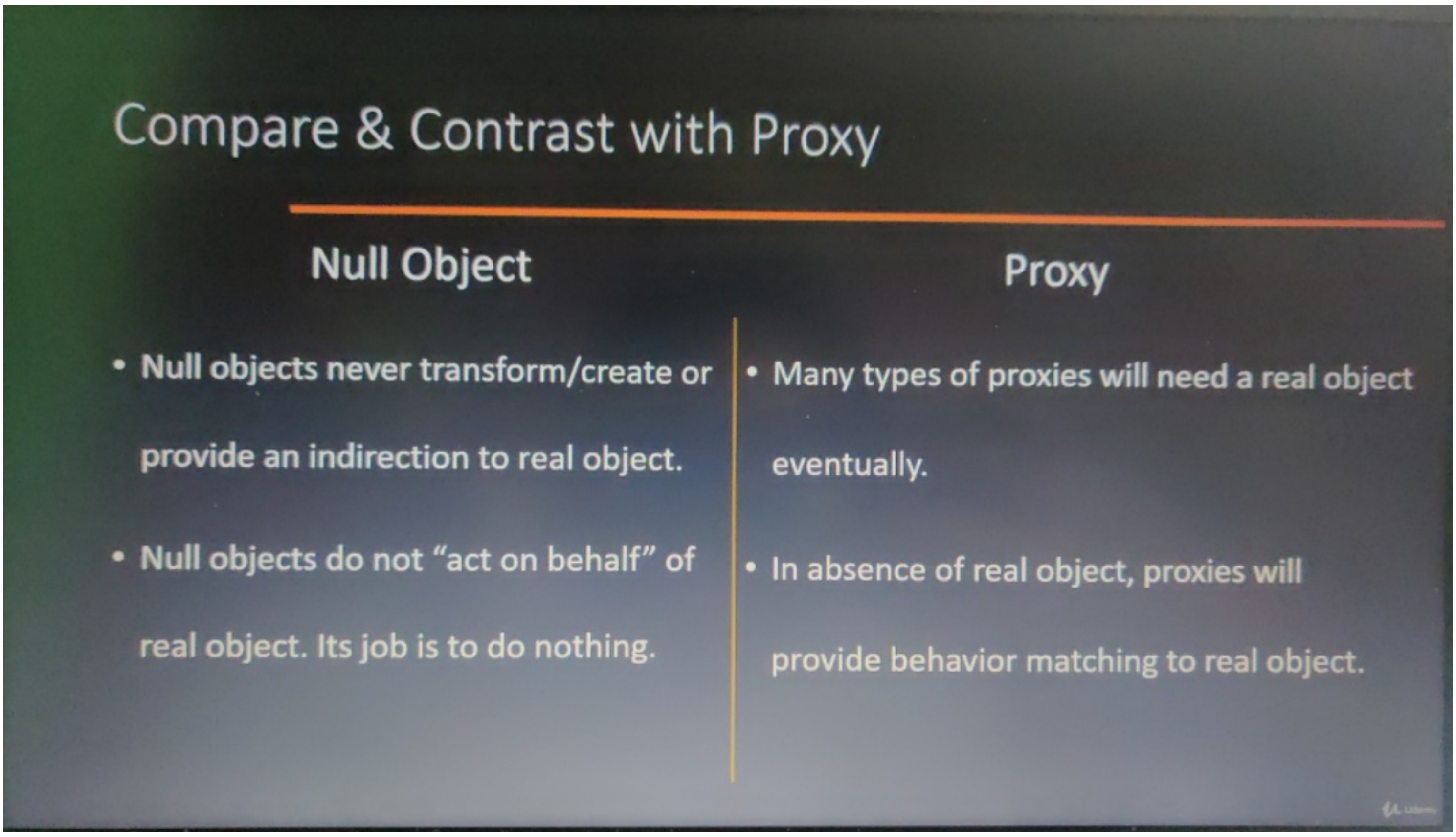
- Since null objects don't have a state & no complex behavior they are good candidates for singleton pattern. We can use a single instance of null object everywhere.
- Null objects are useful in many other design patterns like state – to represent a null state, in strategy pattern to provide a strategy where no action is taken on input.



Example of Null Object

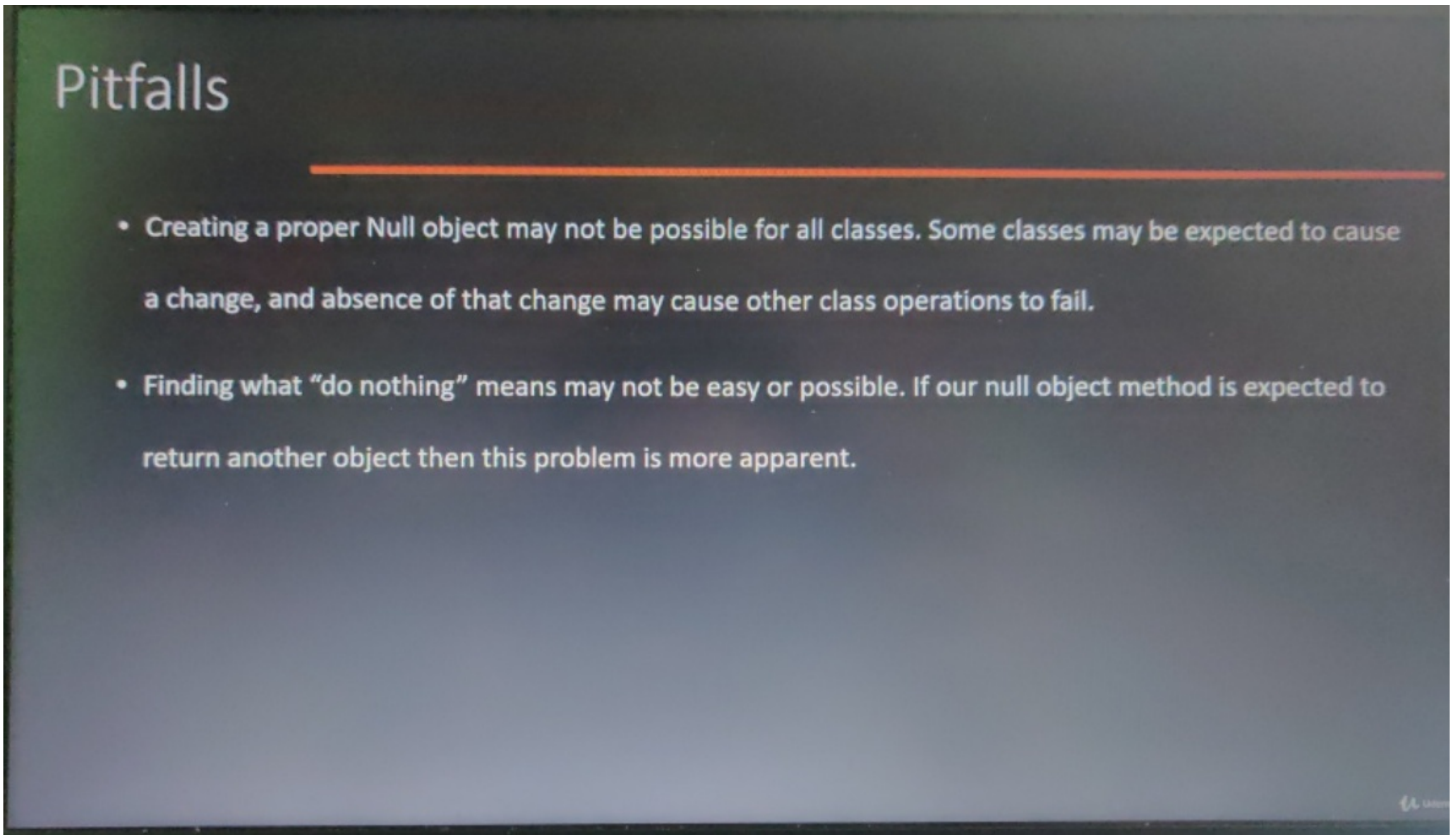
- The various adapter classes from java.awt.event package can be thought of as examples null object. Only reason they are not *the* examples of this pattern is that they are abstract classes but without any abstract method.

```
public abstract class MouseAdapter implements MouseLis
    /**
     * {@inheritDoc}
     */
    public void mouseClicked(MouseEvent e) {}

    /**
     * {@inheritDoc}
     */
    public void mousePressed(MouseEvent e) {}

    /**
     * {@inheritDoc}
     */
    public void mouseReleased(MouseEvent e) {}

    /**
     * {@inheritDoc}
     */
    public void mouseEntered(MouseEvent e) {}

    /**
     * {@inheritDoc}
     */
    public void mouseExited(MouseEvent e) {}
```



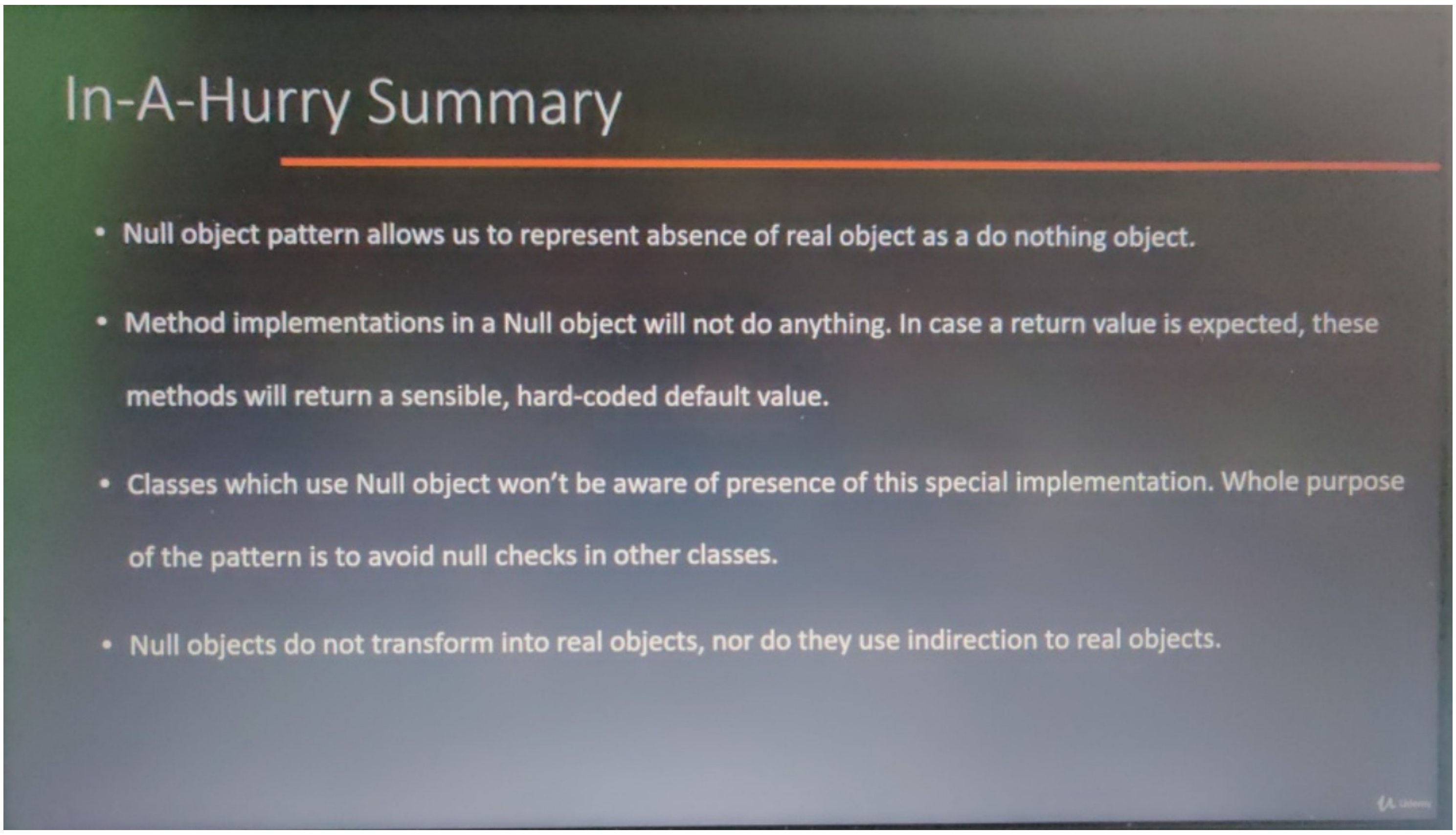
Compare & Contrast with Proxy

Null Object	Proxy
• Null objects never transform/create or provide an indirection to real object.	• Many types of proxies will need a real object eventually.
• Null objects do not "act on behalf" of real object. Its job is to do nothing.	• In absence of real object, proxies will provide behavior matching to real object.



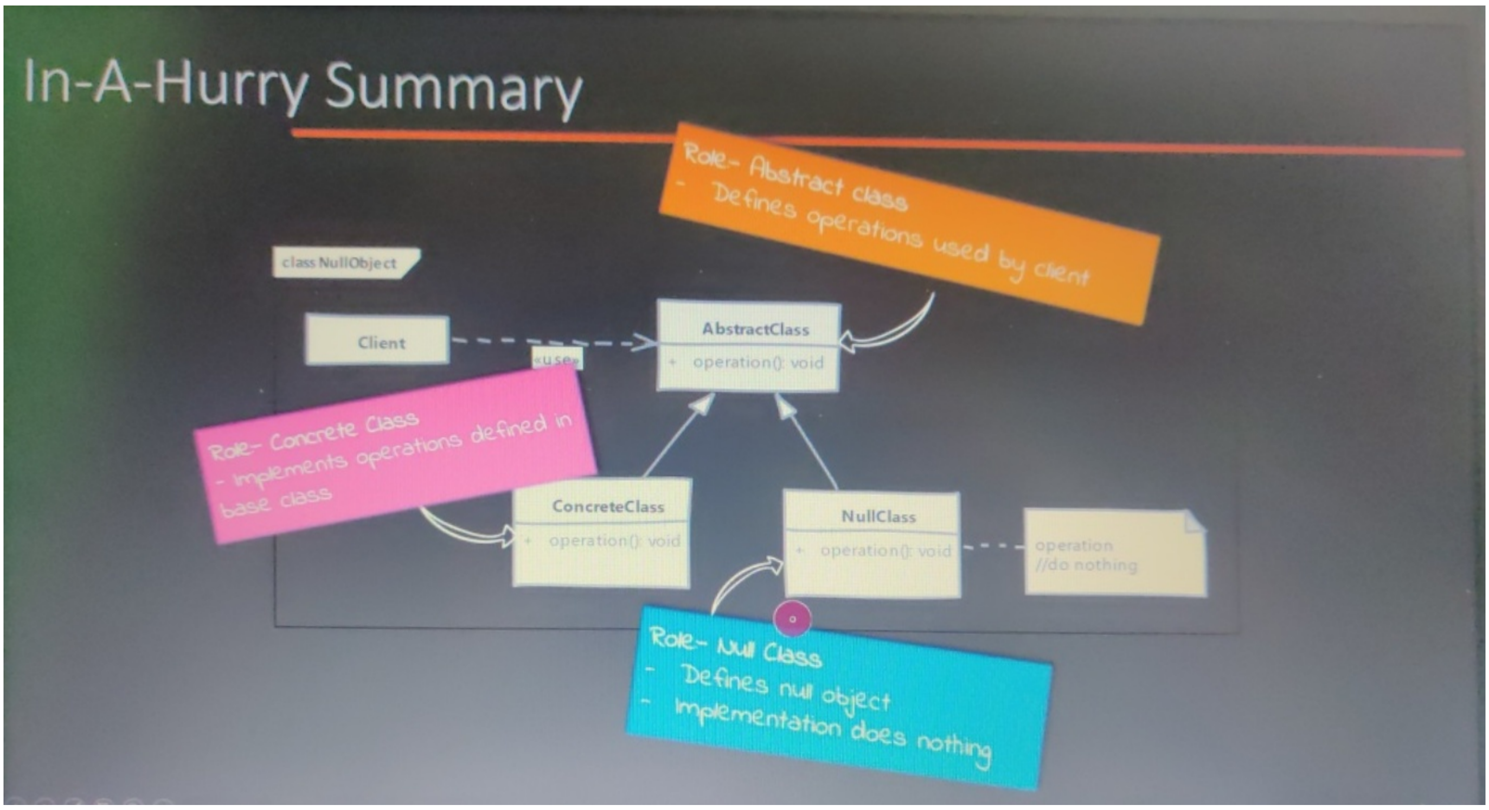
Pitfalls

- Creating a proper Null object may not be possible for all classes. Some classes may be expected to cause a change, and absence of that change may cause other class operations to fail.
- Finding what "do nothing" means may not be easy or possible. If our null object method is expected to return another object then this problem is more apparent.



In-A-Hurry Summary

- Null object pattern allows us to represent absence of real object as a do nothing object.
- Method implementations in a Null object will not do anything. In case a return value is expected, these methods will return a sensible, hard-coded default value.
- Classes which use Null object won't be aware of presence of this special implementation. Whole purpose of the pattern is to avoid null checks in other classes.
- Null objects do not transform into real objects, nor do they use indirection to real objects.



In-A-Hurry Summary