

# Encapsulation

- Encapsulation
  - A Class/Bounded Context/Component/Library should be as shy as possible



30-Oct-15 8:18 PM

By Vijay – caretrainings.co.in

1

It is about managing complexity

Put state information in the class that works on it. Collaborating classes are unaware of the internal state of this object.

Keep a variable/function private, if possible. Use minimum visibility.

Every module should have a secret. If it does not have a secret, why does it exist?

## Compare

```
List<Employee> employees = getEmployees();  
if (employees != null) {  
    for (Employee e : employees)  
        totalPay += e.getPay();  
}
```



```
List<Employee> employees = getEmployees();  
for( Employee e : employees)  
    totalPay += e.getPay();
```

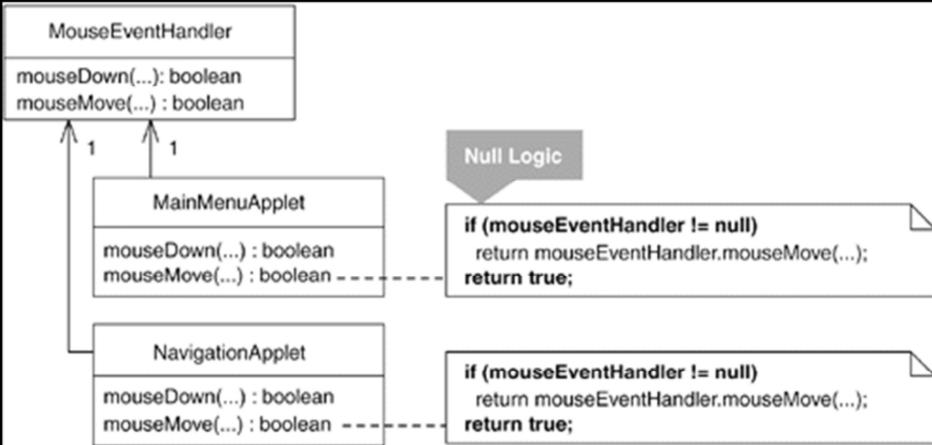
30-Oct-15 8:18 PM

By Vijay – caretainings.co.in

2

Java has Collections.emptyList() for this. It is immutable.

# Null Pattern



30-Oct-15 8:18 PM

By Vijay – caretrainings.co.in

3

## Null pattern

Avoid NullPointerException in code

Return "" instead of null for String class

Return an array with zero elements instead of null.

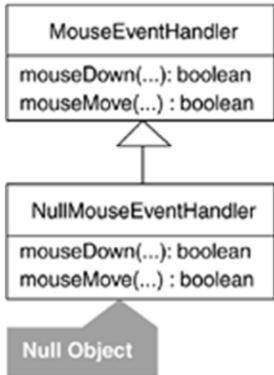
=====

## Benefits and Liabilities

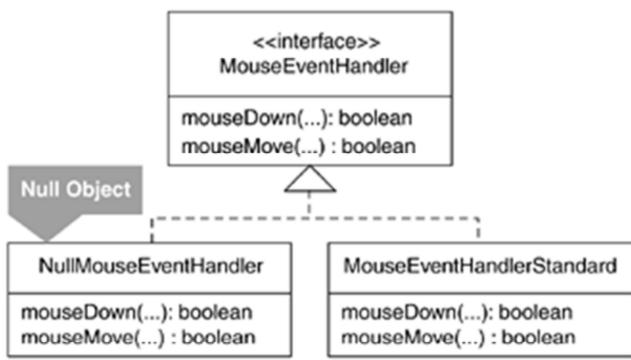
- + Prevents null errors without duplicating null logic.
- + Simplifies code by minimizing null tests.
- Complicates a design when a system needs few null tests.
- Can yield redundant null tests if programmers are unaware of a Null Object implementation.
- Complicates maintenance. Null Objects that have a superclass must override all newly inherited public methods.

# Null Object by Interface

subclassing approach



interface approach



30-Oct-15 8:18 PM

By Vijay – caretrainings.co.in

4

## Guideline

- Unless a method declares in its documentation that null is accepted as a parameter or can be returned from a method as its result, then the method won't accept it or it will never return it.
- Return/Accept Optional object instead of null.



30-Oct-15 8:18 PM

By Vijay – caretrainings.co.in

5

Q28Artists

Optional class is present in Java 8 and Guava library. It has been coded for C# and stored in same directory other examples.

## Guideline

- Don't return String that the client has to parse
- Method overloading ???
  - Bad: TreeSet is sorted in the 2<sup>nd</sup> case

```
public TreeSet (Collection c);  
public TreeSet (SortedSet s);
```

SortedSet extends Collection!

30-Oct-15 8:18 PM

By Vijay – caretrainings.co.in

6

Return of sting with multiple values is bad because

Future modifications will become difficult

Bad: In Java, printStackTrace of Throwable class

=====

Overloading - Avoid same name for multiple methods with same number of arguments

In C# and C++, operator overloading should be used only when it is clear. It should not be very frequent.

# Compare

## JUnit3

```
public static Test
suite() {
    Test r = new
        TestSuite();
    //...
    return r;
}
```

## JUnit 4

```
@RunWith(Suite.class)
@TestClasses({
    //List of classes
});
class AllTests {
}
```

30-Oct-15 8:18 PM

By Vijay – caretainings.co.in

7

The code on the right

- Is flexible because any runner can be used
- Less prone to error, because the caller does not have to worry about exceptions

## Improve

```
Collection<String> keys =  
    new ArrayList<String>();  
keys.add(key1);  
keys.add(key2);  
object.index(keys);
```

- Use varargs

```
object.index(key1, key2);
```

30-Oct-15 8:18 PM

By Vijay – caretrainings.co.in

8

## Compare

```
static int min(  
    int ... args) {  
    if (args.length  
        <= 0) {  
        //Throw  
        //exception  
    }  
    //Compute Minimum  
}
```

```
static int min(  
    int firstarg,  
    int ... args) {  
    //Compute Minimum  
}
```

30-Oct-15 8:18 PM

By Vijay – caretainings.co.in

9

The code on the right

- Is flexible because any runner can be used
- Less prone to error, because the caller does not have to worry about exceptions

In C#, the syntax is

```
static int min(int firstarg, params int[] args)
```

# Java Date and Time

```
Date date =
    new Date(2007,12,13,16,40);
TimeZone zone = TimeZone.getTimeZone
("Europe/Bruxelles");
Calendar cal = new
    GregorianCalendar (zone);
cal.setTime(date);
DateFormat fm = new SimpleDateFormat
("HH:mm Z");
String str = fm.format(cal);
```

Identify the bugs

30-Oct-15 8:18 PM

By Vijay – caretrainings.co.in

10

Bug 1: Year should be 107, since base is 1900.

Bug 2: Month should be 11 instead of 12. Since Jan is 0.

Bug 3: “Europe/Bruxelles”. Bruxelles is capital of Brussels. It is capital of Belgium. Different people pronounce it different ways. Java returns GMT.

Bug 4: We are creating cal object with invalid or wrong value of date.

Not sure - Bug 5: fm.format gives runtime exception because it cannot format calendar. It can format only dates. So we need to call “cal.getTime()” and pass the returned date to “fm.format”

Not sure - Bug 6: We have not set the timezone in DateFormat. It needs to be set before calling format.

## Problems in Java API

- `java.util.Date`, `java.util.Calendar`, `java.util.DateFormat` are mutable
- Jan is 0, Dec is 11
- Date is not a date
- Calendar cannot be formatted
- DateFormat is not threadsafe
- `java.util.Date` is base for `java.sql.Date` and `java.sql.Time`



30-Oct-15 8:18 PM

By Vijay – caretrainings.co.in

11

Q53 – Piano

Q73 – Student, Teacher

---

Date is not a date because: It has time & It uses from 1900

`java.util.Date` should not base class for `java.sql.Date` and `Time` because `getYear` on `java.sql.Time` throws an illegal argument exception.

## Principle of Least Astonishment

- User of API should not be surprised by behavior
  - interrupted method in Thread class clears interrupted flag!



30-Oct-15 8:18 PM

By Vijay – caretrainings.co.in

12

It should have been called clearInterruptedFlag

## Find the flaw

```
public boolean checkPassword(
    String userName, String password) {
    User user = UserGateway.findByName(userName);
    if (user != User.NULL) {
        String codedPhrase =
            user.getPhraseEncodedByPassword();
        String phrase = cryptographer.decrypt(
            codedPhrase, password);
        if ("Valid Password".equals(phrase)) {
            Session.initialize();
            return true;
        }
    }
    return false;}}
```

30-Oct-15 8:18 PM

By Vijay – caretrainings.co.in

13

This function uses a standard algorithm to match a `userName` to a password. It returns true if they match and false if anything goes wrong. But it also has a side effect. Can you spot it?

The side effect is the call to `Session.initialize()`, of course. The `checkPassword` function, by its name, says that it checks the password. The name does not imply that it initializes the session. So a caller who believes what the name of the function says runs the risk of erasing the existing session data when he or she decides to check the validity of the user.

This side effect creates a temporal coupling. That is, `checkPassword` can only be called at certain times (in other words, when it is safe to initialize the session). If it is called out of order, session data may be inadvertently lost. Temporal couplings are confusing, especially when hidden as a side effect. If you must have a temporal coupling, you should make it clear in the name of the function. In this case we might rename the function `checkPasswordAndInitializeSession`, though that certainly violates "Do one thing."

## Guidelines

- Keep the command and queries segregated
  - Accessors, mutators, and predicates should be named for their value and prefixed with get, set, and is according to the standard.

```
String name =  
    employee.getName();  
customer.setName("mike");  
if (paycheck.isPosted())...
```

30-Oct-15 8:18 PM

By Vijay – caretrainings.co.in

14

Exception: DSL.

Commands : return void.

## Law of Demeter violated

```
// DOM code to write an XML document to a specified
// output stream
static final void writeDoc(Document doc,
    OutputStream out) throws IOException {
try {
    Transformer t = newTransformerFactory.
        newInstance().newTransformer();
    t.setOutputProperty(OutputKeys.DOCTYPE_SYSTEM,
        doc.getDoctype(), getSystemId());
    t.transform(new DOMSource(doc), new
        StreamResult(out));
} catch (TransformerException e) {
    throw new AssertionError(e); //can't happen
}
}
```

30-Oct-15 8:18 PM

By Vijay – caretrainings.co.in

15

Principle of least knowledge or Law of Demeter:

Each unit should only talk to its friends; Don't talk to strangers.

Don't make the client do anything, that the Module could do

Reduce the need for boilerplate code

Generally done via cut-and-paste

Ugly, Annoying and error-prone

## Reduce Coupling

```
public float getTemp() {  
    return  
    station.getThermometer().getTemp();  
}
```

Vs.

```
public float getTemp() {  
    return station.getTemp();  
}
```

30-Oct-15 8:18 PM

By Vijay – caretrainings.co.in

16