# OOPS ABAP

## Why OOPS?

1. **Encapsulation** and hence improved Security
2. **Abstraction**

3. **Inheritance**

4. Polymorphism

Code Redundancy is going to be optimized

5. Modularization → Readability and Maintainability

6. Design Pattern - Factory Design Pattern, Façade, Single Design

## Drawbacks of the procedural programming

1. Security Issue

2. Handling Large Project – To maintain the large number of source code is very-very difficult
Subroutine
Function Module

3. All the main features of OOPS are not available listed in left

What is the Benefits of Modularization?

1. Large set of code is going to be converted into small set of code
2. Less work force and less time
3. Hence Cost is less
4. Better Maintainability

## What is the important properties of OOPS ABAP ?

1. Encapsulation – Grouping of attributes and methods together
   To achieve this we have the concept of Class

2. Data Hiding
In OOPS we can easily achieve the DATA hiding concept
How?
Visibility set on the data:
Public
Protected
Private

3. Abstraction:
Hiding the method Implementation

4. Inheritance

a) Multiple
b) Multilevel Inheritance

5. Polymorphism

1. Overriding
2. Overloading

## What is Class in OOPS ABAP?

VVI

Class is a **User Defined Data type** having component in form of attributes and Methods, events, interfaces and they are bundled together.

**<u>In C++</u>**

Class is combination of data members + Member functions

**<u>In Java</u>**

Class -> Instance Variables
+ Methods

**<u>ABAP</u>**

Class -> Attributes
+ Methods
+ Interfaces
+ Events
+ Aliases

## How many different types of Class in OOPS ABAP?

Class

**Global Class** –
It can be accessed in any of the Reports/FM/Methods if its visibility sets to Public

T-Code: **SE24**( Class Builder )
Global class is accessible in any program

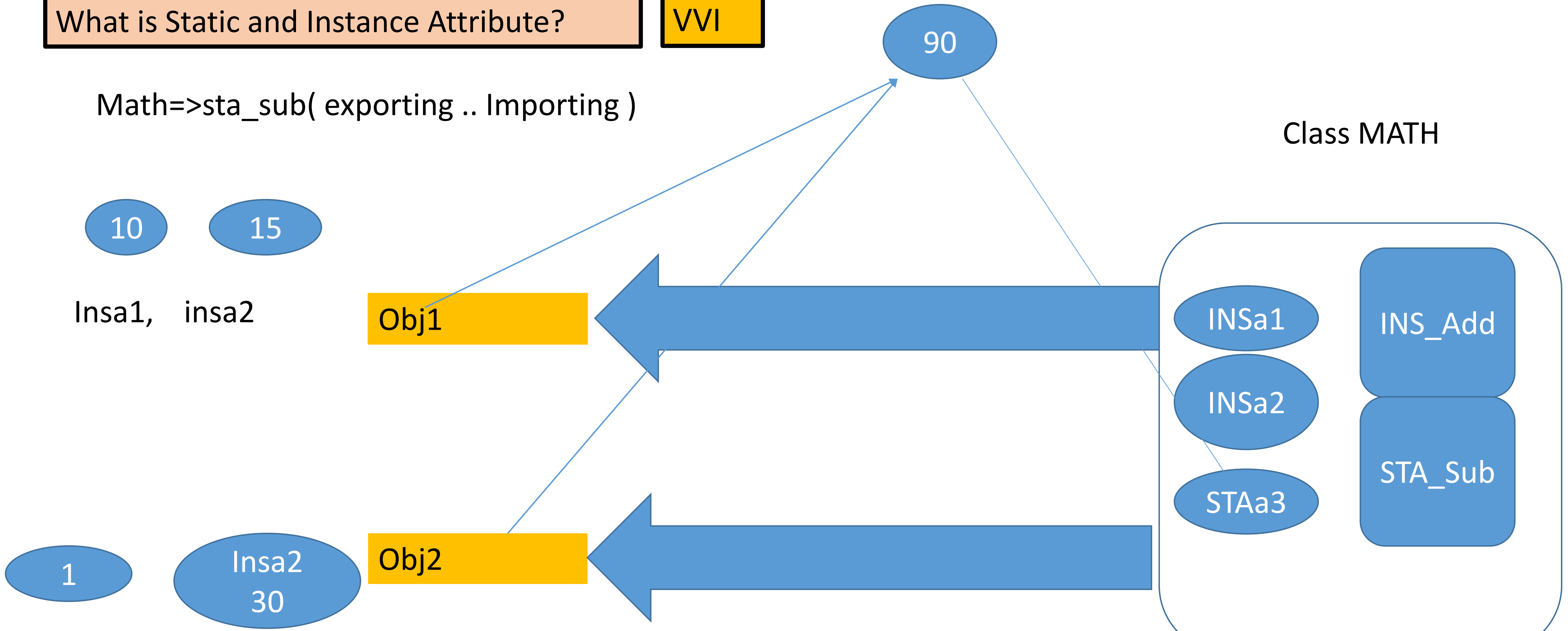
**Local Class–**
It is created inside a report and accessed in that report only

Its scope is specific to the program where it has been written.

## What is Static and Instance Attribute?

VVI

Math=>sta_sub( exporting .. Importing )



Obj1->staa3 = 20
Obj1->staa3 = 40
Obj2->staa3 = 90

Obj1->insa1 = 10
Ojb1->insa2 = 15

Obj2->insa1 = 20
Obj2->insa2 = 30

static Attribute:

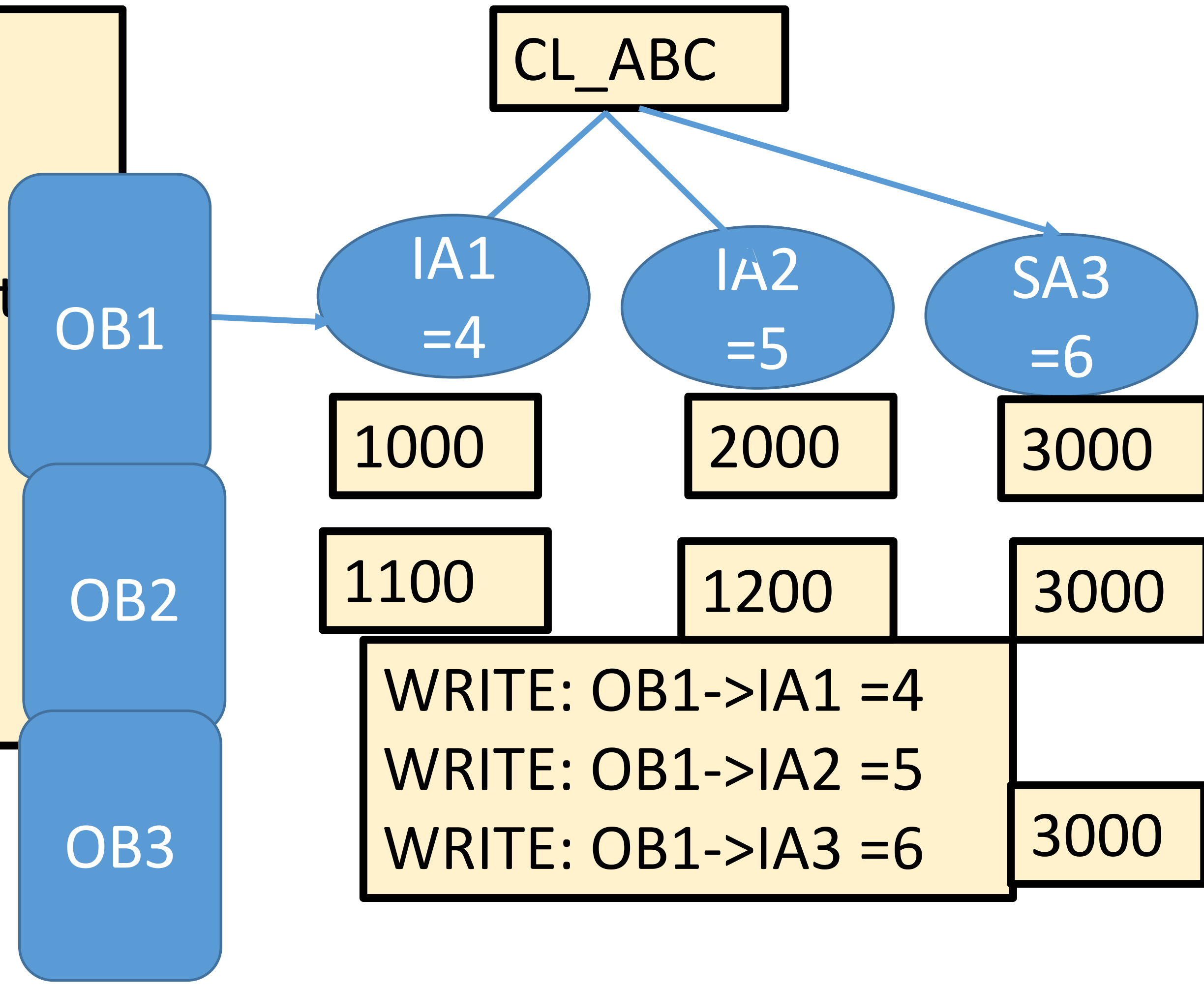
Accessed - Through Class name and Object instance

Memory – It is specific to class and always points to same Memory location

Instance Attribute:

Accessed - Only through the object instance

Memory -



WRITE: OB1->IA1 =4
WRITE: OB1->IA2 =5
WRITE: OB1->IA3 =6

WRITE: OB2->IA1 =4
WRITE: OB2->IA2 =5
WRITE: OB2->IA3 =6

Ob1->ia1 = 4.
Ob1->ia2 = 5.
ob1->**sa3** = 10
**ob2->ia1 = 1000.**
Ob2->ia2 = 2000.
**Ob2->ia1 = 30000.**
Ob2->**sa3** = 9999.

Write: ob1->ia1, ob2->ia2, ob1->sa3, ob2->ia1

## What is Object?

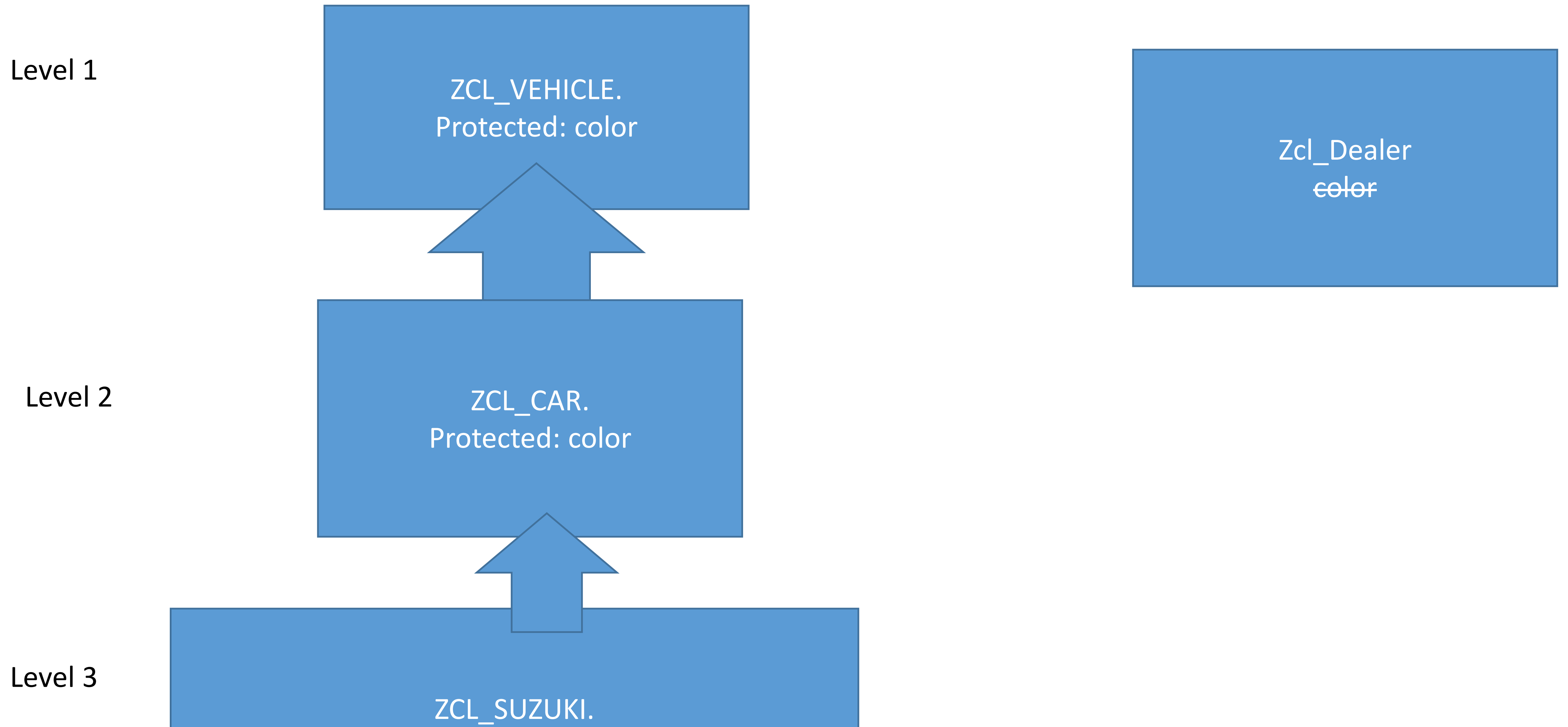
Object : Instantiation of the class.

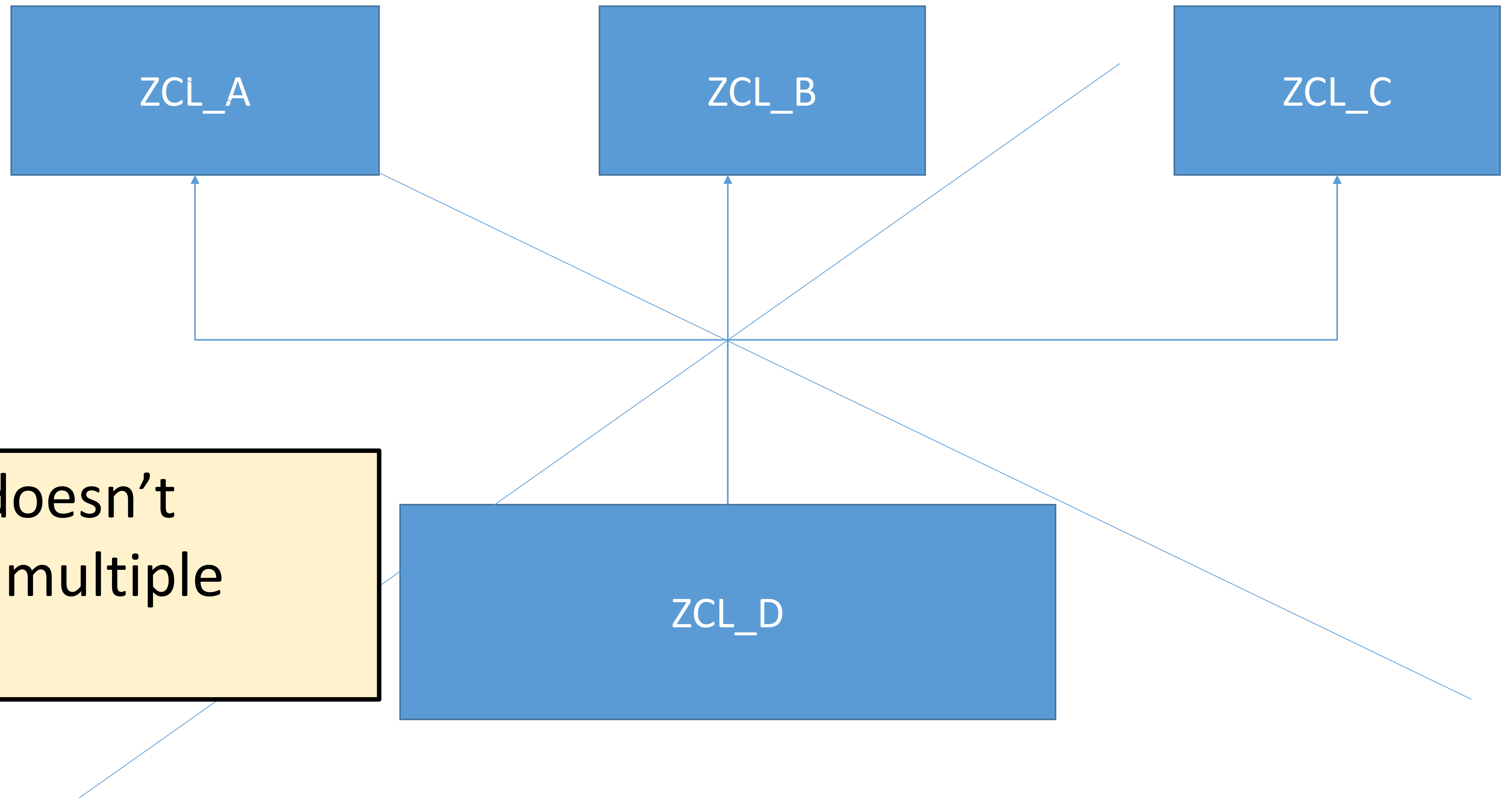
To Create Object –( Instantiation of the class )

1. **Reference of the class** – Naming the address in memory where your object will sit :
   data: go_obj1 TYPE REF TO CL_ABC.
1. **Create the object ( Instantiation )-**
   **CREATE OBJECT** go_obj1

New Syntax(NW7.40)
**DATA** (ob1) = **NEW** cl_sample()

DATA(go_obj1) = NEW CL_ABC( ).





Oops abap doesn’t
upport the multiple
nheritence

What kind of inheritance is being supported by SAP ABAP?

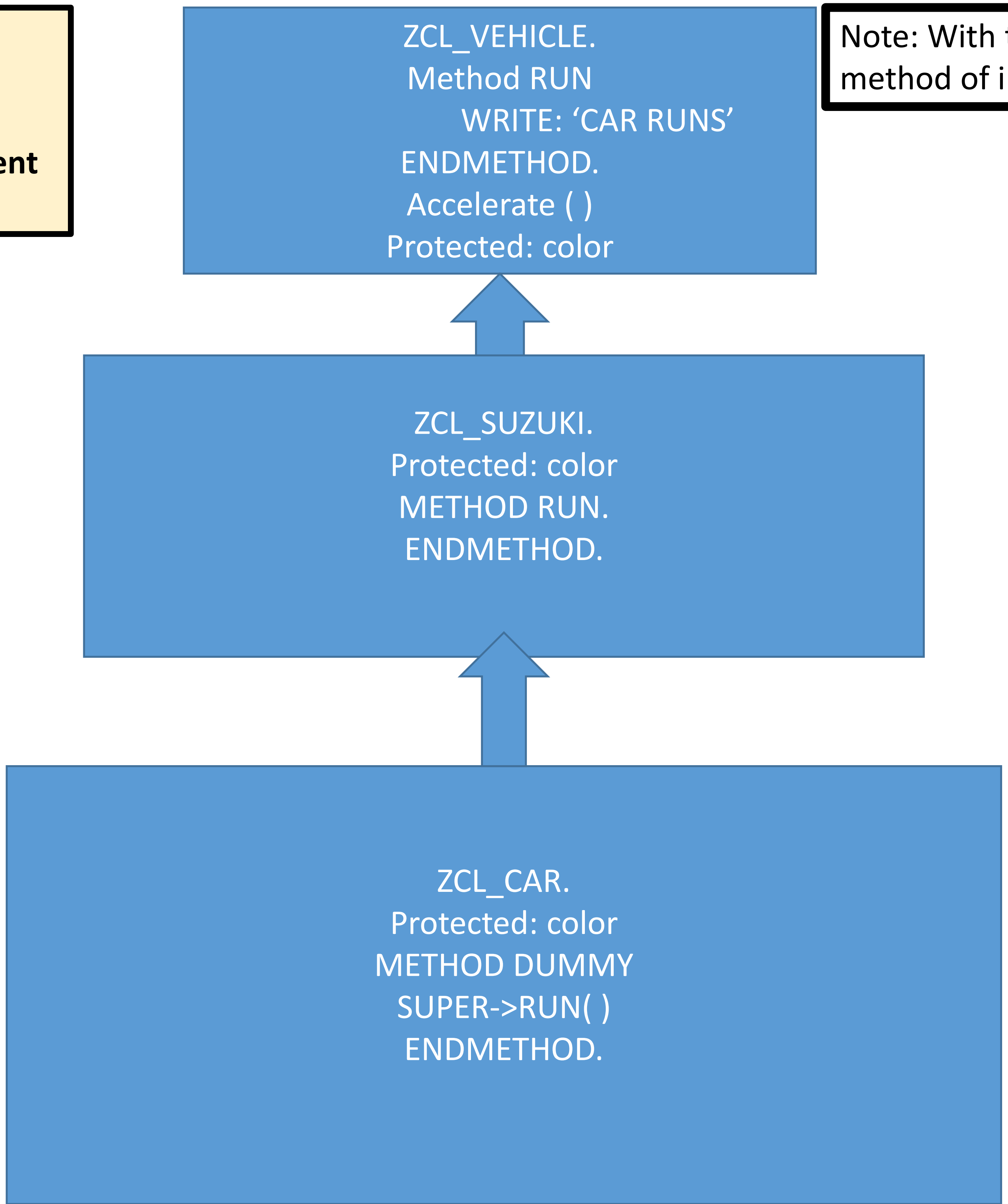
Ans.  Only **Multilevel** Inheritance is being supported by SAP ABAP

Q. If Multiple Inheritance is not supported, is there any way that it can be achieved?

Ans. Using the Interface we can achieve the **Multiple Inheritance.**

What is **SUPER Key word?**

**Ans.** It is used to access the **Immediate Parent Class**



```
ZCL_VEHICLE.
Method RUN
        WRITE: ‘CAR RUNS’
ENDMETHOD.
Accelerate ( )
Protected: color
```

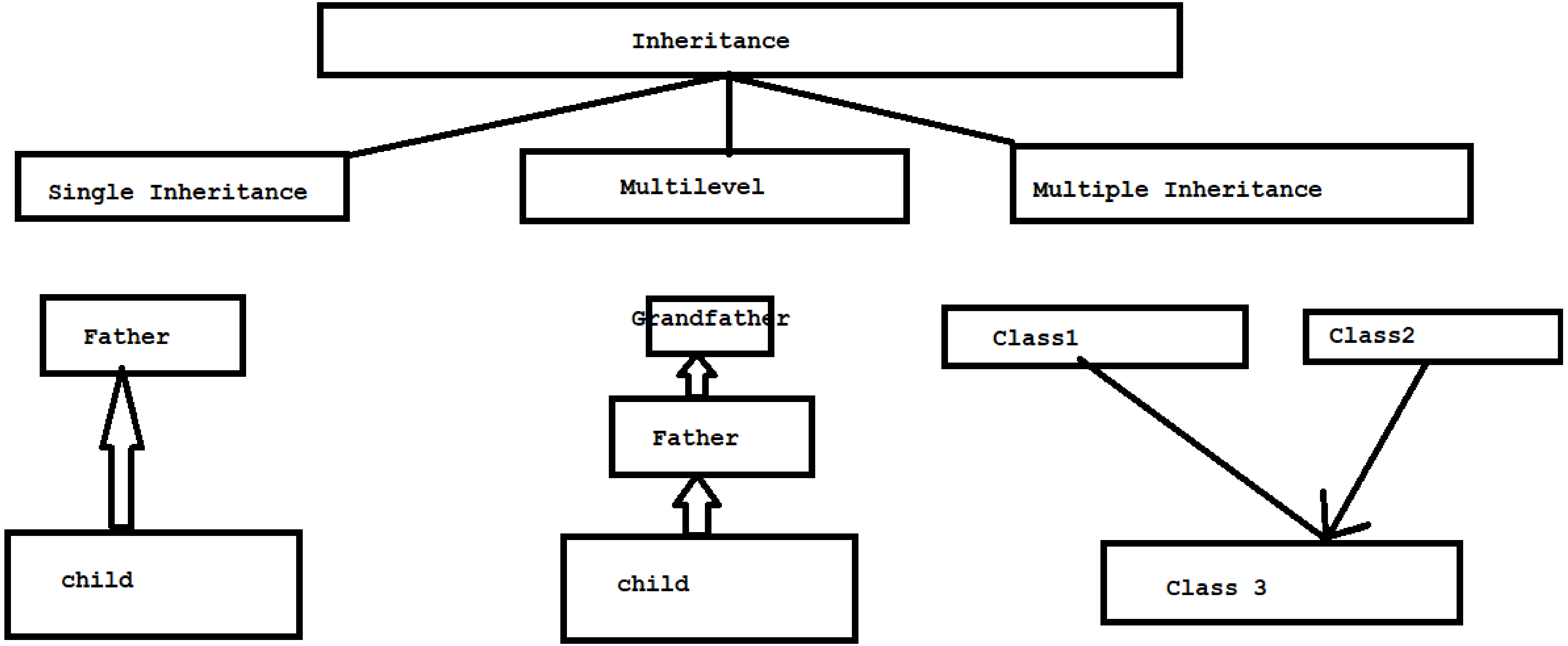
```
ZCL_SUZUKI.
Protected: color
METHOD RUN.
ENDMETHOD.
```

```
ZCL_CAR.
Protected: color
METHOD DUMMY
SUPER->RUN( )
ENDMETHOD.
```

Note: With the help of **SUPER** key word very first implementation of the method of immediate super class is called.

Output:-

## What are the different types of Inheritance?



ABSTRACT CLASS

ZCL_VEHICLE
ZCL_ANIMALS
ZCL_LIVING
ZCL_ORE

Notes: -

1. **Abstract Class can’t be instantiated.**
2. At least one method must be abstract.

INTERAFACE

1. This can also be not instantiated.

## What is the difference between Abstract Class and Interface? (VVI)

| **Abstract Class** | **Interface** |
|---|---|
| 1. Abstract class must have at least one empty method( Abstract Method ) | 1. All the methods of the interface must be empty. |
| 2. We can go for it if we know some of the behavior( method ) in advance | 2. We can go for it if we don’t know any of the behavior. |
| 3. You can’t achieve multiple inheritance with the help of Abstract Class | 3. Multiple inheritance with the help of Interface can be achieved. |

What is Constructor?
And what are the different types of Constructor?

CONSTRUCTOR

Definition:
Special method To set the state of an Object/
Special method To initialize the Variable of the object/Class

CONSTRUCTOR/INSTANCE CONSTRUCTOR/OBJECT CONSTRUCTOR

STATIC CONSTRUCTOR/CLASS CONSTRUCTOR

## What is the difference between Class Constructor and Instance Constructor?

Class Constructor(CLASS-CONSTRUCTOR)

1. It is called only one time whenever a static component of a class is accessed or object is created.

2. Name of the class constructor:
**CLASS_CONSTRUCTOR**

Instance Constructor(CONSTRUCTOR)

1. This is going to be called every time whenever you create the object of the class

2. Name: **CONSTRUCTOR**

## Can a Constructor have a Import and export parameter both?

Ans. No Not possible. It is only having the import parameter.

## When the constructor is going to be called – Before object creation or After object Creation?

Ans. Before Object Creation

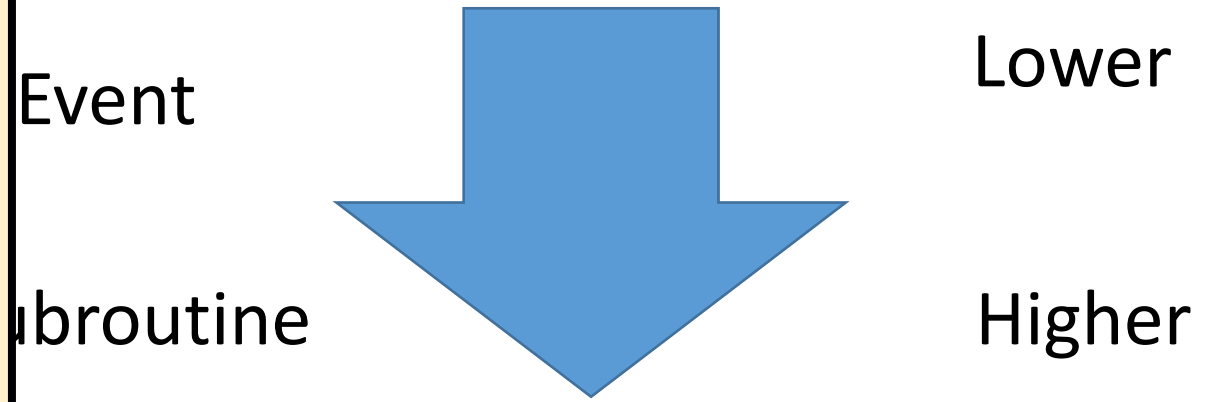
What is the rule for Static Constructor?

**Rule for Static Constructor:**

1. If any object is created then very first time class constructor will be called.
2. If in the program any static component of the class has been accessed then while loading the event in which that code has been written which is accessing the component.
3. Same thing is true for Subroutine and Module.

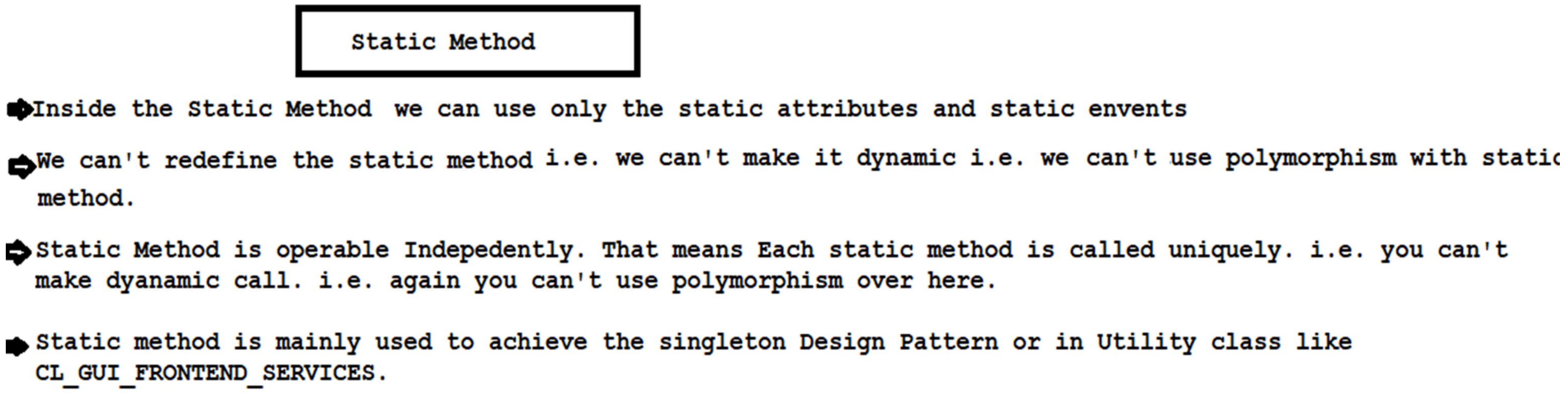
CL_ABC
SC1 – STATIC ATTRIBUTE
SC2 - STATIC METHOD

```
PROGRAM ZABC.

*CL_ABC=>SC1

DATA: OB1 TYPE REF TO ZABC.
CREATE OBJECT OB1.-2
DATA: OB2 TYPE REF TO ZABC
CREATE OBJECT OB2.
```



| Normal Method | Special Method | |
|---|---|---|
| | Constructor | Class Constructor |
| 1. It can Public, Protected or Private | This depends on the Class Visibility. It is same as class.If class is public then it must be under public section. if class is protected then it must be under protected and if it is privated then it must be under private | It is always Public |
| 2. should be called explicitly | 2. called implicitly | called implicitly |
| 3. Can be called many number of times | 3. Can be called only once when first time object is created | 3. Will be called only once in a session |
| 4.Can contain any types of parameters | 4. Can contain only import and exception parameter | No parameter at all |
| 5.Its name can be anything | 5. Its name must be CONSTRUCTOR | 5. Its name must be CLASS_CONSTRUCTOR |

## What is Static Method and it’s prosperities



Static Method

- Inside the Static Method we can use only the static attributes and static envents
- We can't redefine the static method i.e. we can't make it dynamic i.e. we can't use polymorphism with static method.
- Static Method is operable Indepedently. That means Each static method is called uniquely. i.e. you can't make dyanamic call. i.e. again you can't use polymorphism over here.
- Static method is mainly used to achieve the singleton Design Pattern or in Utility class like CL_GUI_FRONTEND_SERVICES.

**What is the syntax to access the static method in the program?**

Ans.->
This is first way using the class name -> CLASS NAME **=>** Static Method Name
Data: ob1 type ref to CL_ABC.
CREATE OB1
**Ob1->Static Method**

**Inside a static method can you use the instance component of a class?**
**No – Not possible**

**Is it possible to override(redefine) the static method in the subclass?**
**Ans -> No not possible**

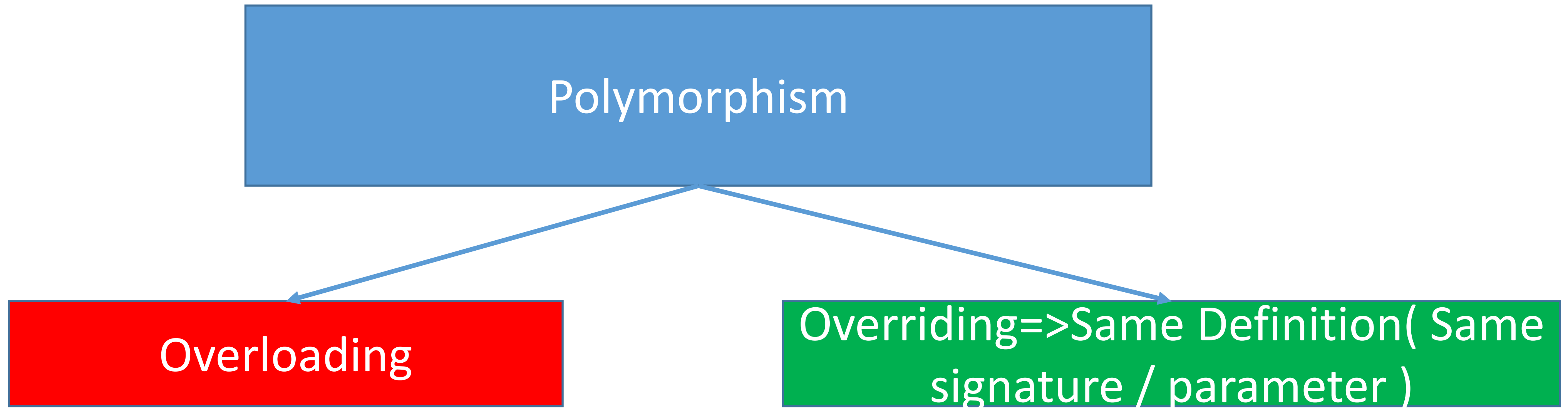
**Why We are supposed to create or use the instance component( method, attributes, events) over static component?**
**Ans. – Because we can’t use the polymorphism which is one of the**

Then, when to use Static Method?

Every time we don't require polymorphism for example in case of Factory method.

When we have to create the test program we can go for it because accessing the method is very easier here

Class Name => Method.



$$\begin{tikzcd}
 & \text{Polymorphism( Different Form)} \arrow[dl] \arrow[dr] & \\
\xcancel{\text{Overloading}} & & \text{Overriding} \checkmark
\end{tikzcd}$$

Class:-

Mehtod : run ->1

Method : run **importing x type i. ->2**

Method : run **<u>importing y type char5</u>**. ->3

Method:  run  importing x type I  z type I

exporting y time char10. ->4

Obj->run( 5 )

Obj->run(  ‘5’ )

Obj->run(5,6)

Polymorphism and Inheritance

->Class Constructor can never be inherited.

->Constructor can be inherited and can not be redefined but a subclass can have its own constructor but must use the SUPER-CONSTRUCTOR.

Q. Can a Class Constructor will inherited?

Ans. **No**

Q. Can a Constructor will be inherited

**Ans. Yes .**

Q. Can a constructor will be redefined?

**Ans. No.**

Q. Can a subclass have its own constructor defined?

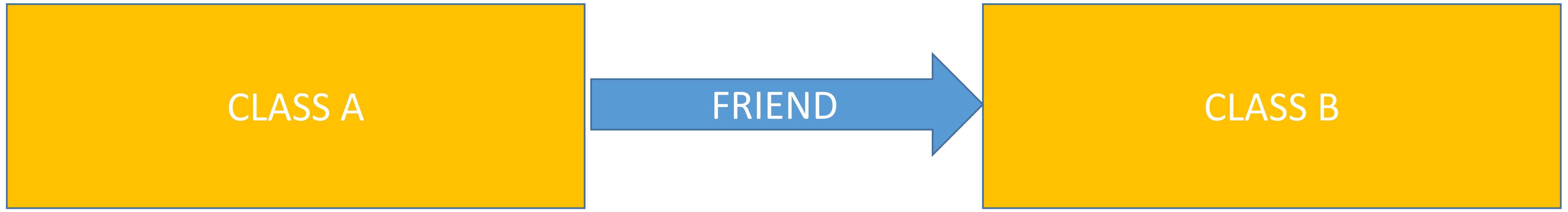
**Ans. Yes.**

If a subclass have a constructor redefined then is it necessary to call the super->constructor in the redefined constructor?

Ans.-> We must have to use the SUPER Key word .

## Friend Key word / Friend Class

If a class A is friend of Class B, then Class A can access the Protected or Private component ( attributes, Methods and Events)
Of Class B.



## Local Class SE38

1. Local class is created inside the Program.
2. The Scope of the local class is just inside that program
3. You can create the Global class from Local Class

What is the major / Important / basic / fundamental difference between Friend Class and Subclass.

**Friend Class** -> public, protected , private , no polymorphism.
**Subclass**->Public and protected. Polymorphism.

Agenda:

1. Load Key Word( **Obsolete** Key Word )

```
Report zdummyloaddemo.
class zcl_eb22_vehicle DEFINITION load.
" Whenever you are going to access the static component of the class
zcl_eb22_vehicle=>test_static( ).
```

1. **Deferred** Key Word

What is Deferred Key word?

**Ans.** Just to let the compiler know that the used class is somewhere declared later in the program.

Q. What is Interface?

Ans. A user defined data type which has below property -

- It doesn’t have the **implementation** of any method.
- It have only the method definition.
- It can’t be instantiated.

What is Abstract Class and Interface? What is the Similarity and difference between Both? (VVI)
Ans –

## Abstract Class

**Similarity** – 1. As its name suggests, it is an abstract, not concrete, so ***This class can't be instantiated***, If you try to instantiate in ***ABAP editor it'll throw you an error***. This is similarity. It's also used to achieve ***polymorphism through Overriding/Redefinition.***

***2. Power of polymorphism can be realized with the help of narrow casting.***

Note – ***Exception is Static method***. This can't be overridden/redefined, because its all the attributes are static which share the same

## Interface

1. **Similarity** - ***This also can't be instantiated***. ***This is also used to achieve polymorphism by Overriding/redefinition***.

***2. Power of polymorphism can be realized with the help of narrow casting.***

# Constructor Hierarchy Made Easy

Q. When you create the Object of the Subclass in below scenario in what sequence call will take place?

**SUPER CLASS**
CLASS_CONSTRUCTOR
INSTANCE CONSTRUCTOR

**SUB CLASS**
**CLASS_CONSTRUCTOR**
INSTANCE CONSTRUCTOR
SUPER->CONSTRUCTOR

**CREATE THE OBJECT OF THE SUBCLASS:**

And below is the Sequence -

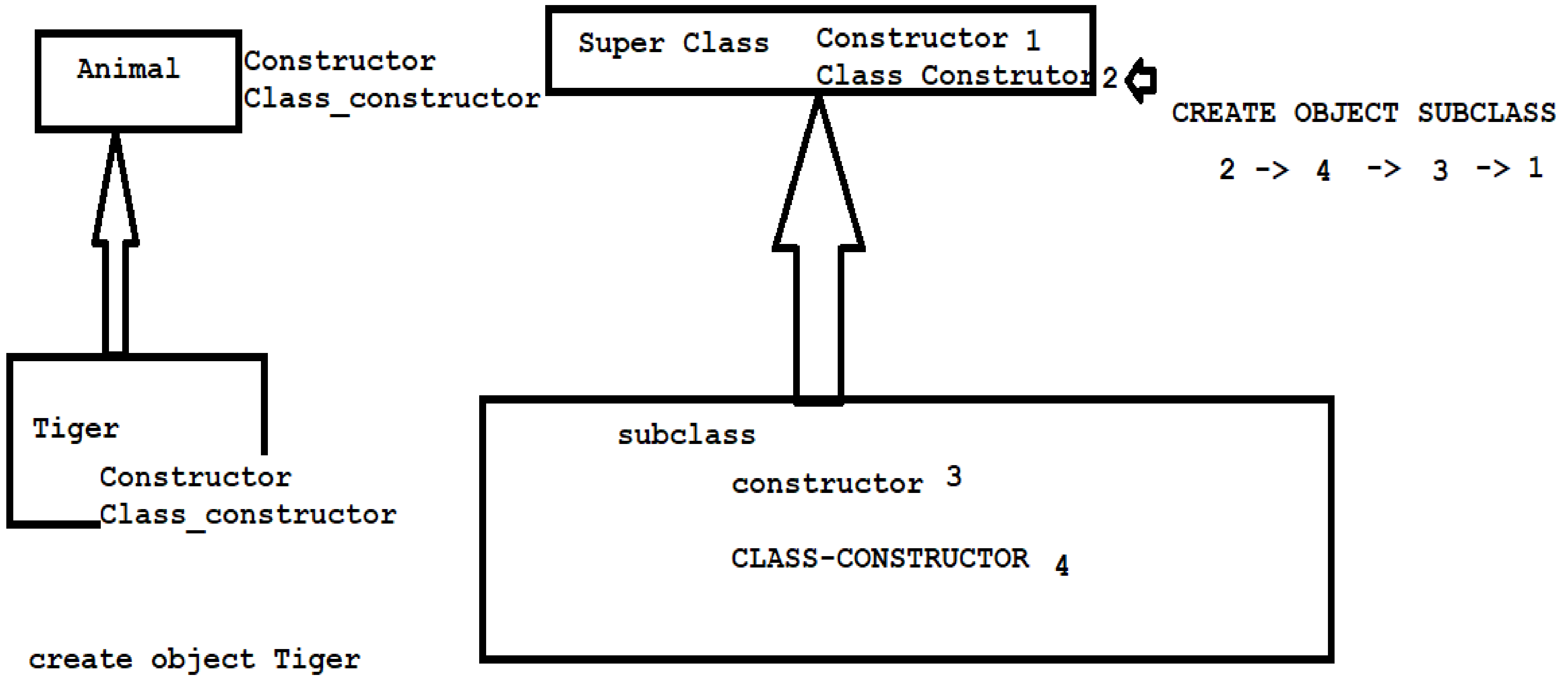
**SUPERCLASS->** CLASS CONSTRUCTOR

**SUBCLASS** -> CLASS CONSTRUCTOR

**SUBCLASS->** Instance Constructor

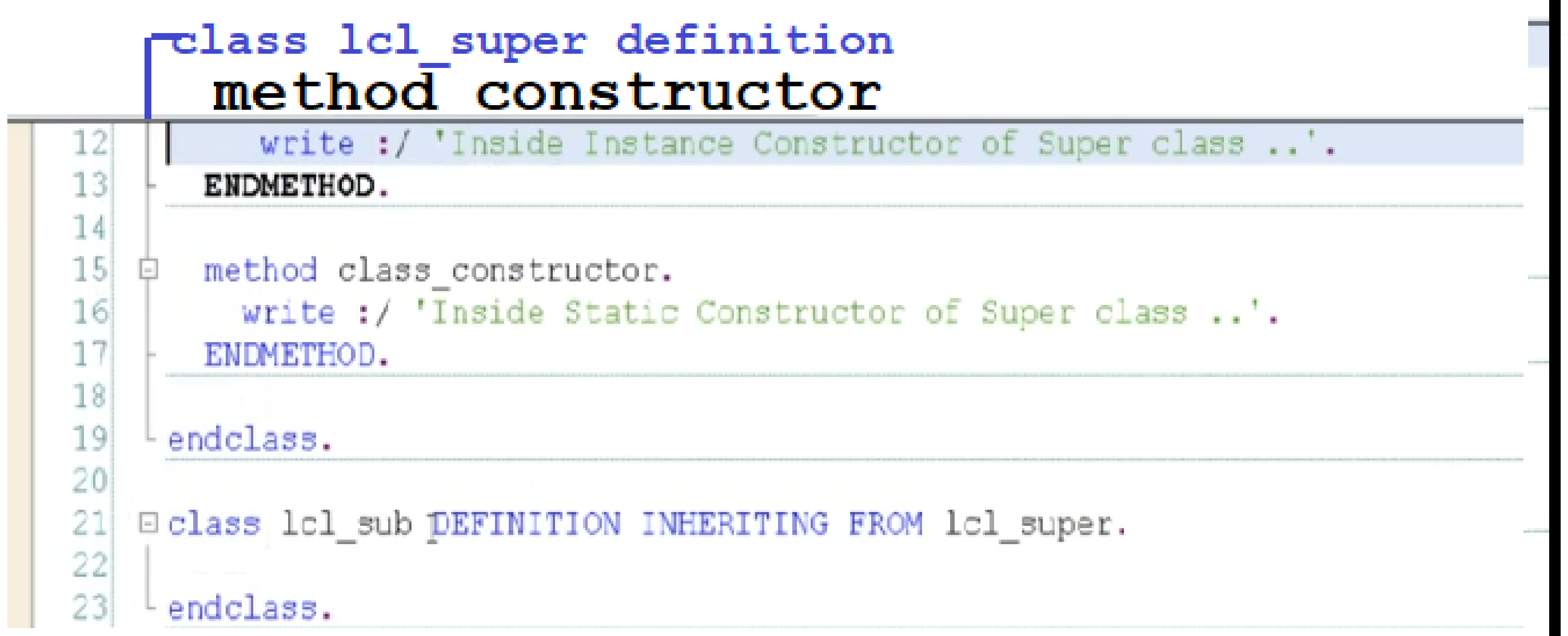
**SUPERCLASS->** Instance Constructor

Sequence in which objects are called



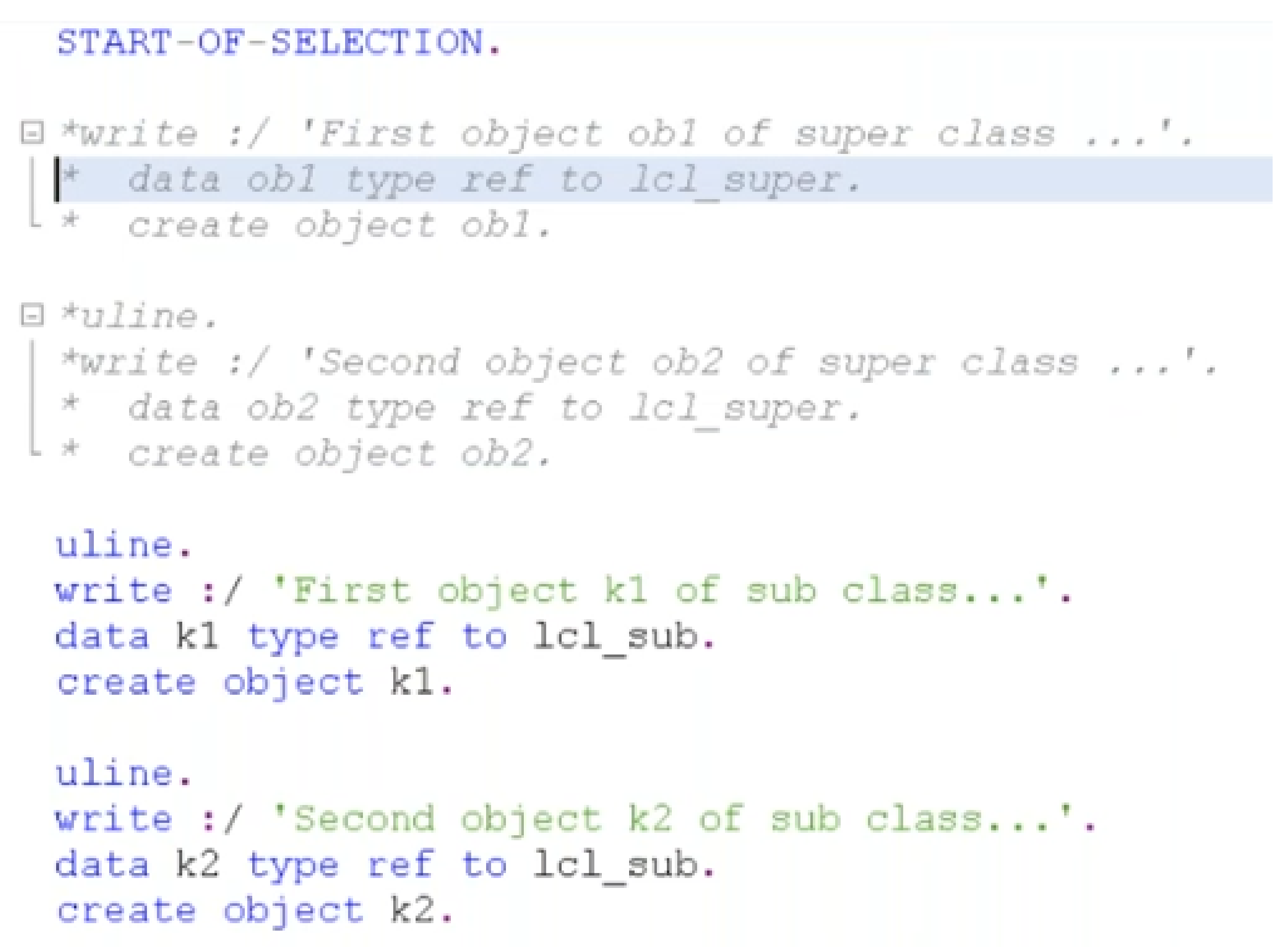
## Constructor Hierarchy

### Scenario 1: When Instance Constructor is not defined in Subclass



```
class lcl_super definition
 method constructor
    write :/ 'Inside Instance Constructor of Super class ..'.
  ENDMETHOD.

  method class_constructor.
    write :/ 'Inside Static Constructor of Super class ..'.
  ENDMETHOD.

endclass.

class lcl_sub DEFINITION INHERITING FROM lcl_super.

endclass.
```

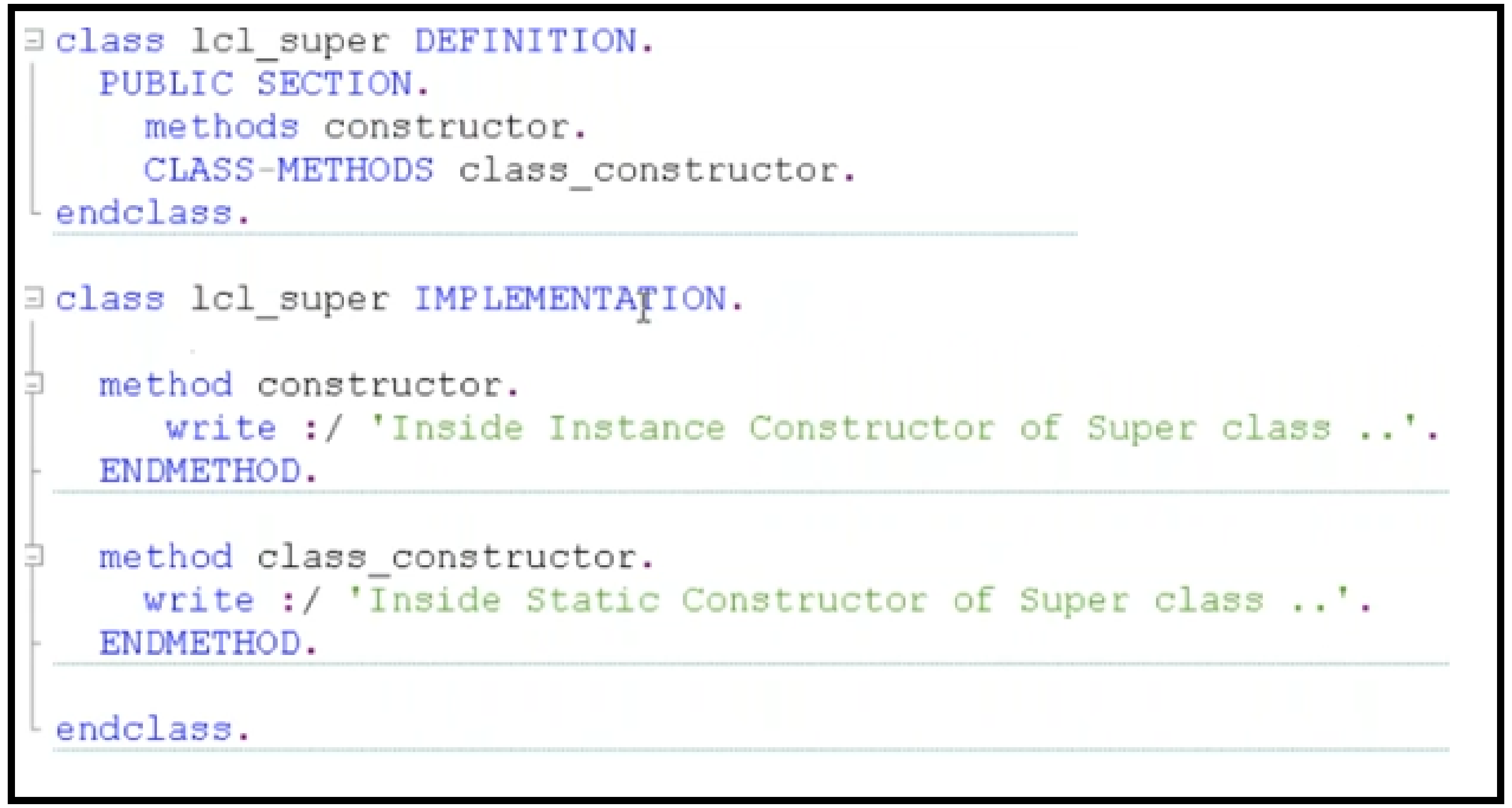
Note that here
no Subclass Implementation



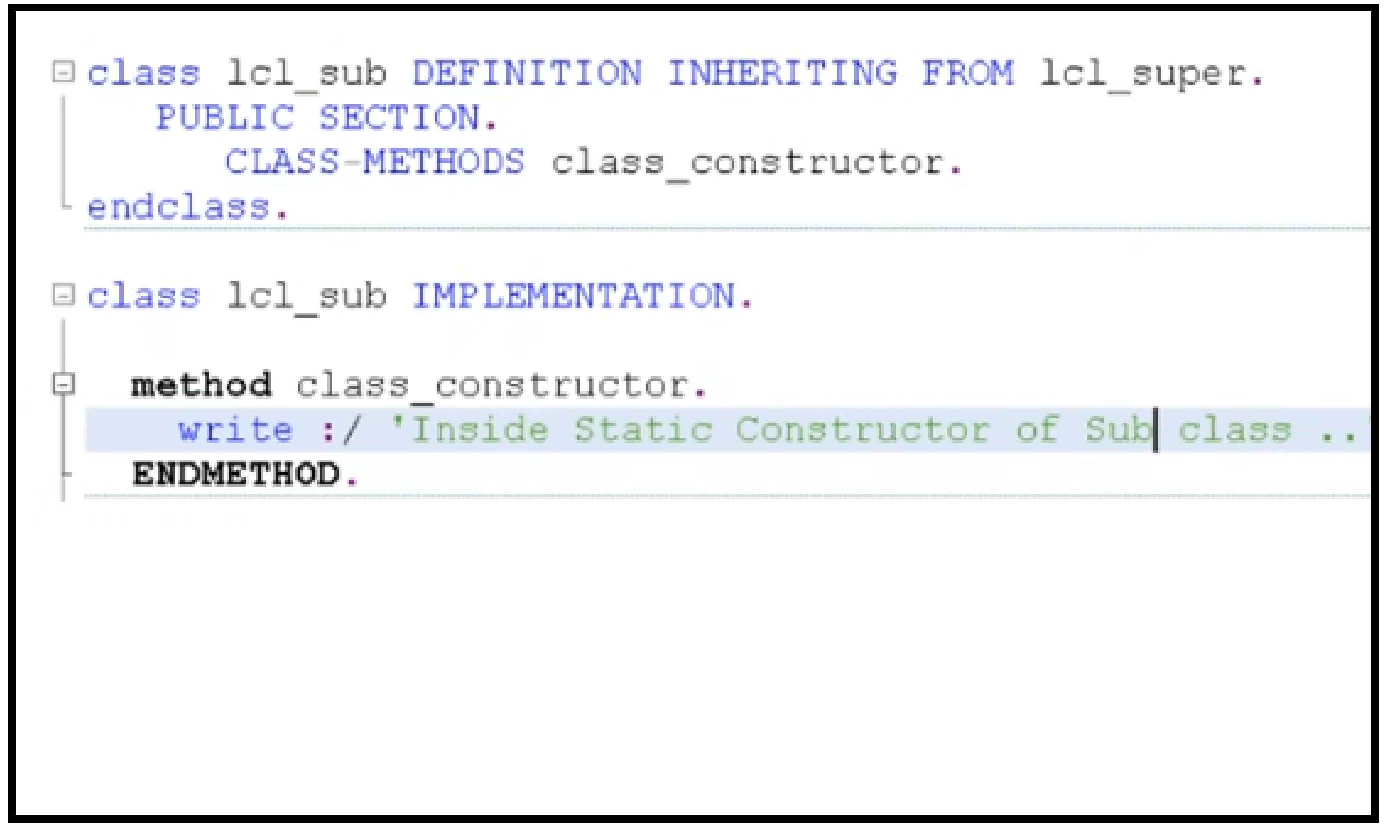
```
START-OF-SELECTION.

*write :/ 'First object ob1 of super class ...'.
*  data ob1 type ref to lcl_super.
*  create object ob1.

*uline.
*write :/ 'Second object ob2 of super class ...'.
*  data ob2 type ref to lcl_super.
*  create object ob2.

uline.
write :/ 'First object k1 of sub class...'.
data k1 type ref to lcl_sub.
create object k1.

uline.
write :/ 'Second object k2 of sub class...'.
data k2 type ref to lcl_sub.
create object k2.
```

# Constructor Hierarchy

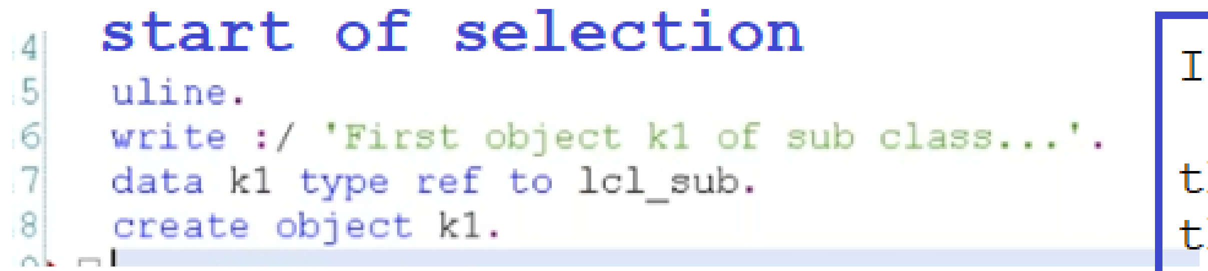
## Scenario 2 : When only Class Constructor is defined in Subclass



```
class lcl_super DEFINITION.
  PUBLIC SECTION.
    methods constructor.
    CLASS-METHODS class_constructor.
endclass.

class lcl_super IMPLEMENTATION.

  method constructor.
     write :/ 'Inside Instance Constructor of Super class ..'.
  ENDMETHOD.

  method class_constructor.
    write :/ 'Inside Static Constructor of Super class ..'.
  ENDMETHOD.

endclass.
```



```
class lcl_sub DEFINITION INHERITING FROM lcl_super.
   PUBLIC SECTION.
      CLASS-METHODS class_constructor.
endclass.

class lcl_sub IMPLEMENTATION.

 method class_constructor.
   write :/ 'Inside Static Constructor of Sub class ..
 ENDMETHOD.
```

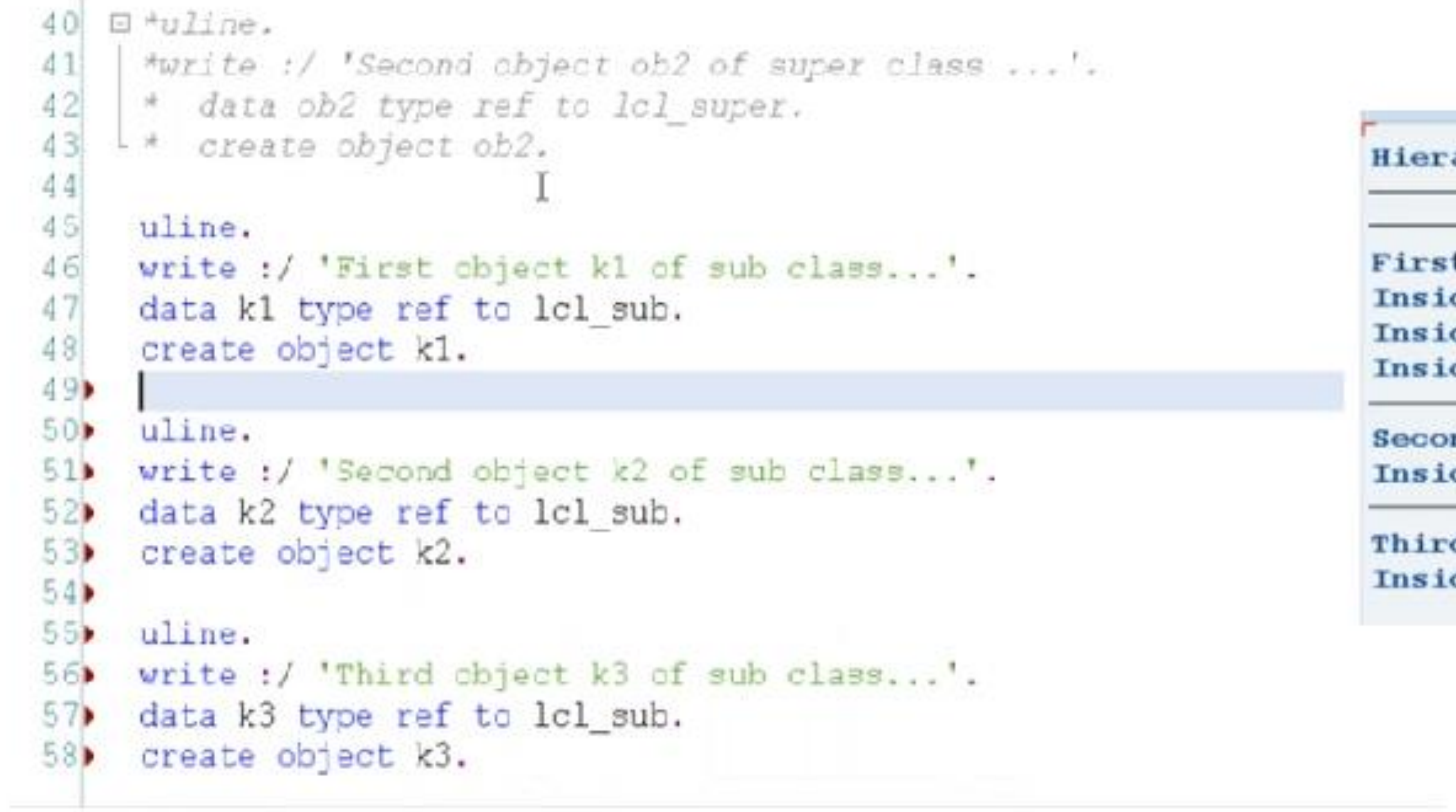


```
start of selection
 uline.
 write :/ 'First object k1 of sub class...'.
 data k1 type ref to lcl_sub.
 create object k1.
```

It will call first
**super class class_constructor**
then **subclass Class_constructor** and
then **Subclass instance Constructor**

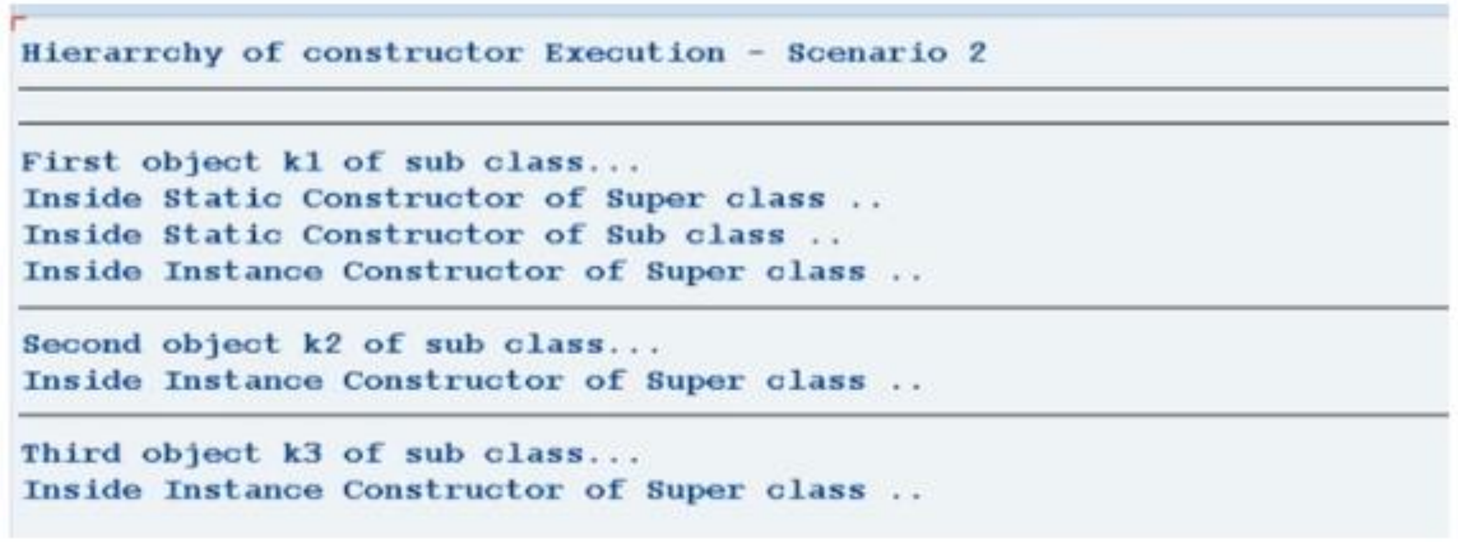
## Constructor Hierarchy

**Scenario 2 : When only Class Constructor is defined in Subclass** continued...



```
*uline.
*write :/ 'Second object ob2 of super class ...'.
*  data ob2 type ref to lcl_super.
*  create object ob2.

uline.
write :/ 'First object k1 of sub class...'.
data k1 type ref to lcl_sub.
create object k1.

uline.
write :/ 'Second object k2 of sub class...'.
data k2 type ref to lcl_sub.
create object k2.

uline.
write :/ 'Third object k3 of sub class...'.
data k3 type ref to lcl_sub.
create object k3.
```

Output:

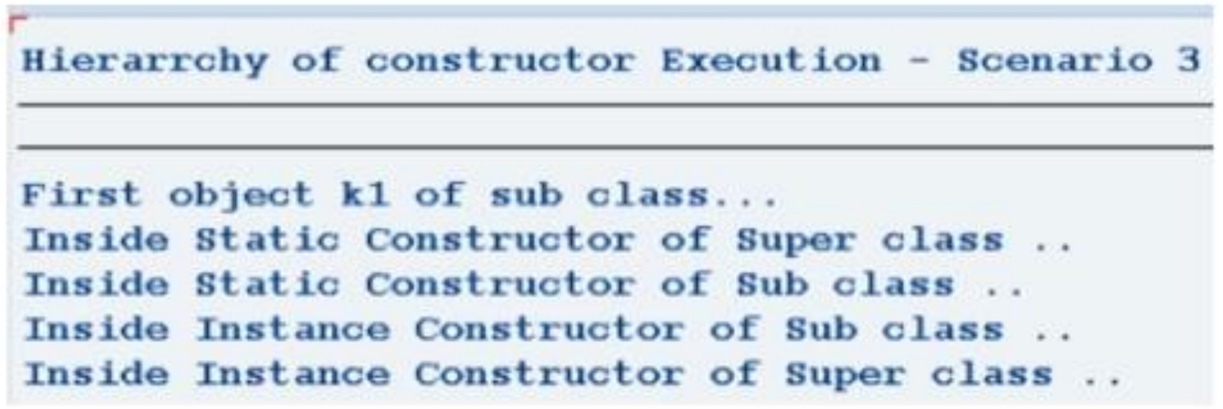


```
Hierarrchy of constructor Execution - Scenario 2
--------------------------------------------------------------
--------------------------------------------------------------
First object k1 of sub class...
Inside Static Constructor of Super class ..
Inside Static Constructor of Sub class ..
Inside Instance Constructor of Super class ..
--------------------------------------------------------------
Second object k2 of sub class...
Inside Instance Constructor of Super class ..
--------------------------------------------------------------
Third object k3 of sub class...
Inside Instance Constructor of Super class ..
```

## Constructor Hierarchy: Scenario 3

**Hierarchy of constructor execution-scenario 3**

- If a super class contains static and instance Constructor and subclass also with static and instance constructor, then it is mandatory for sub class instance constructor to call the super class instance constructor explicitly by using super keyword.
- In this case, if we instantiate first object of sub class before accessing any static components of super/sub class and before creating any objects for super class, then SAP first executes static constructors from super class to sub class (top to bottom) and then instance constructors from sub class to super class (bottom to top) and from second object of sub class onwards, only instance constructors will be executed from sub class to super class.



```
Hierarrchy of constructor Execution - Scenario 3

First object k1 of sub class...
Inside Static Constructor of Super class ..
Inside Static Constructor of Sub class ..
Inside Instance Constructor of Sub class ..
Inside Instance Constructor of Super class ..
```

**Note that:**
**Static from Top to bottom and Instance from bottom to top**

**1st Point Demo:->Code will be same as Scenario 2 only constructor definition and implementation will be added to the subclass.**

```
uline.
write :/ 'First object k1 of sub class...'.
data k1 type ref to lcl_sub.
create object k1.
```

## Constructor Hierarchy Scenario 5

**If Super class Instance Constructor is having mandatory parameter**
**and**
**Subclass is not having any constructor implementation**

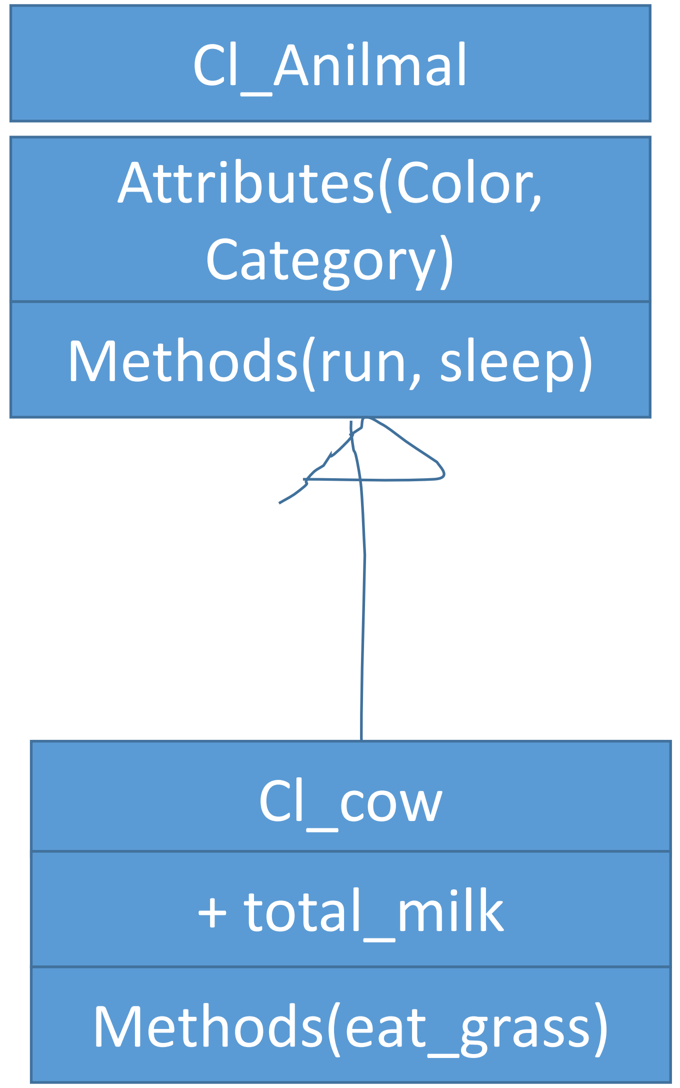
**In that Case**

**While Instantiating the sublclass we must have to pass the mandatory parameter becuase it implicitely calles the super class construtor.**

UML : Unified Modeling Language
Narrow Casting
Wide Casting

What is UML Diagram?

Ans. UML: This is standard pictorial/graphical representation of denoting the components of a class and its relationship between one or more than One Classes.

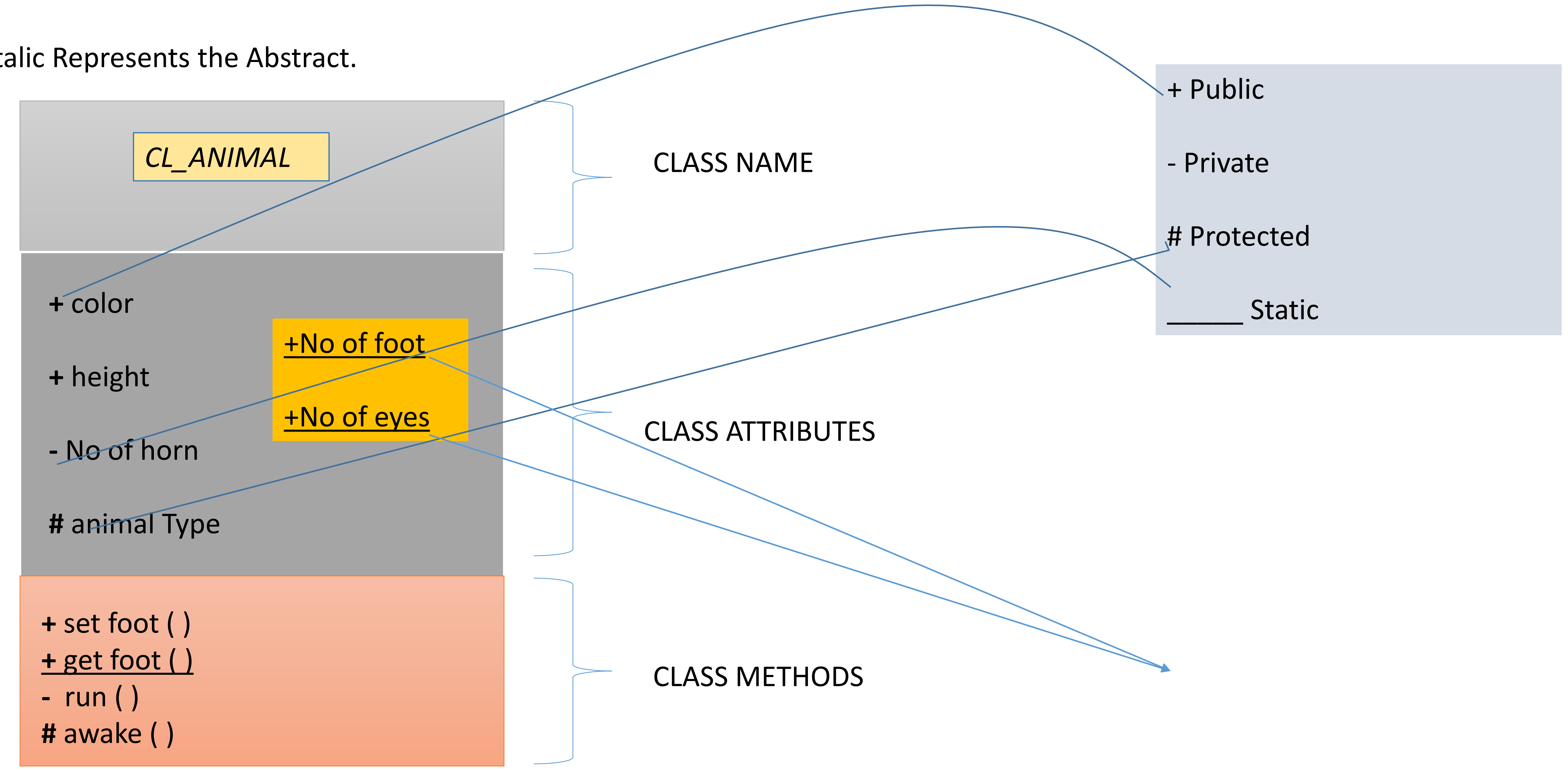


How to represent Class via UML diagram. What is the meaning of symbol "+", "-" and "#" and "________"

Ans – Below is the detail

Please watch out this video: https://www.youtube.com/watch?v=UI6lqHOVHic

Italic Represents the Abstract.



What is **association** relationship, **aggregation** relationship and **composition** relationship in OOPS

Ans. – **Association:** is a **USING** relationship where all objects have there own lifecycle and there is no owner. E.g. ( Relationship between Teacher and Student, Doctor and Patient etc.

**Aggregation:** is a **HAS A** relationship which is specialized form of Association. Objects have their own lifecycle but only one of the objects will be owner. E.g. Teacher and School, Company and employee, District and people(District Magistrate) etc.

**Composition:** is a **MUST HAVE** relationship which is again a specialized form of aggregation where one object life cycle is completely dependent on another object. E.g. Building and Room, Plane and Wings, Body and Mind etc.

ASSOCIATION
AGGREGATION
COMPOSITION

$$\begin{tikzcd}
 & \text{Casting} \arrow[dl] \arrow[dr] & \\
\text{Narrow Casting} & & \text{Wide casting}
\end{tikzcd}$$

Data: lv_age_i type int4.
Data: lv_age_c type char5.

Lv_age_c = ‘23’.

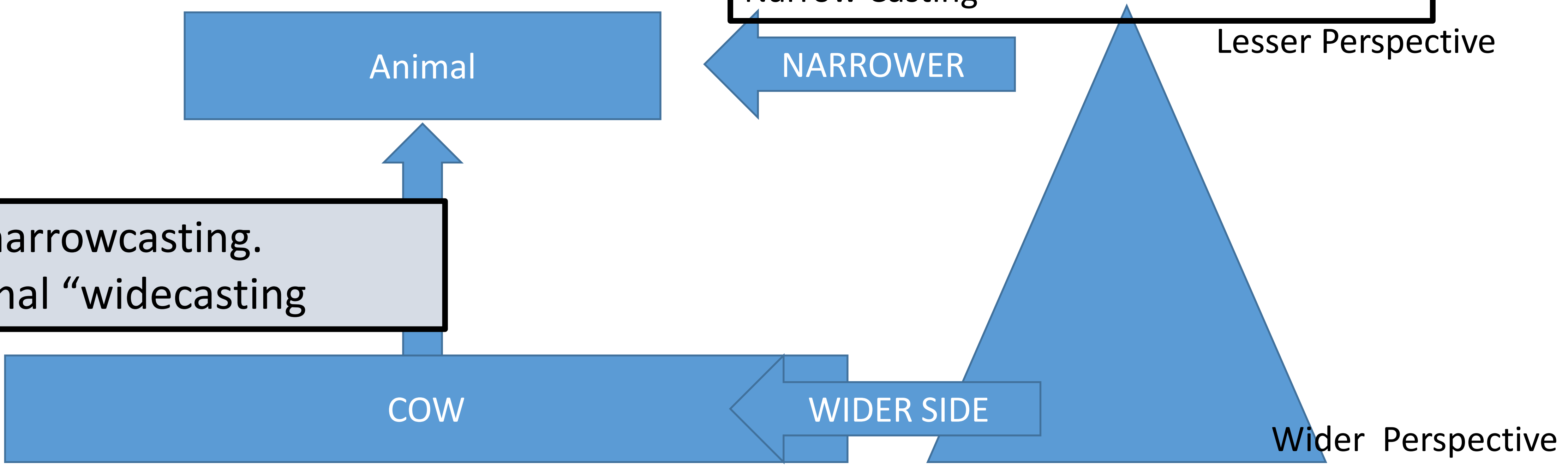
lv_age_i = Lv_age_c

Narrow Casting: Assigning the object of a sub class to The reference of the super class is known as Narrow Casting

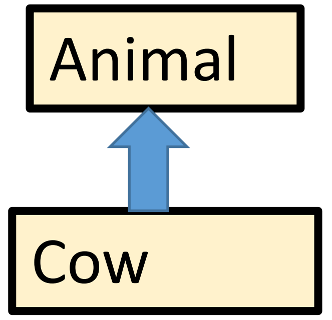
Wide Casting: Assigning the object of Super class to the sub class
Condition: Before performing the Wide Casting we must have to perform the Narrow Casting

?=

animal



Ob_animal = ob_cow”narrowcasting.
Ob_cow ?= re_ob_Animal “widecasting



**Q1**. What is the different types of casting in OOPS in terms of inheritance(i.e. Super Class vs Sub class)( VVI )

**Ans** – There are two types of casting –

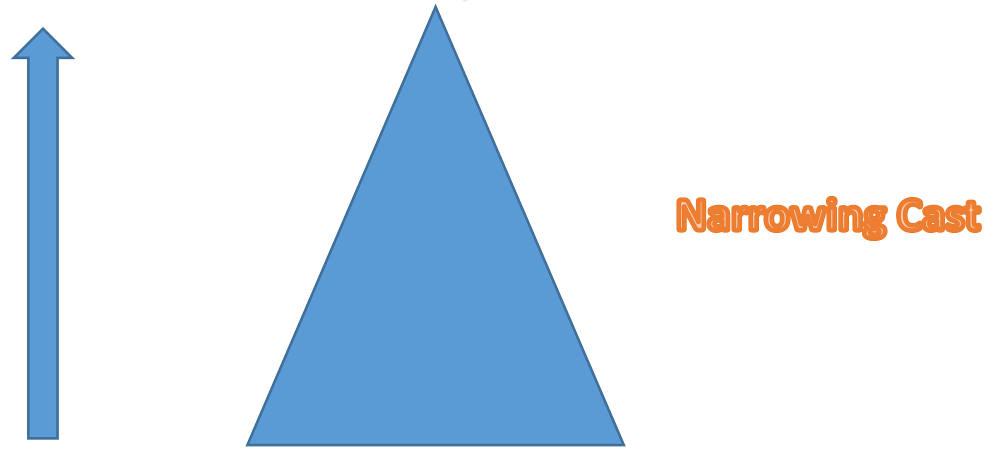
**1. Narrowing Cast and other is**

**2. Widening Cast**

**Q2**. What is narrowing Cast? Explain in detail ( VVI )

**Ans**. Assigning of instance of subclass to the reference of Superclass.

SUPER CLASS – Narrower/Lesser Perspective, General Detail



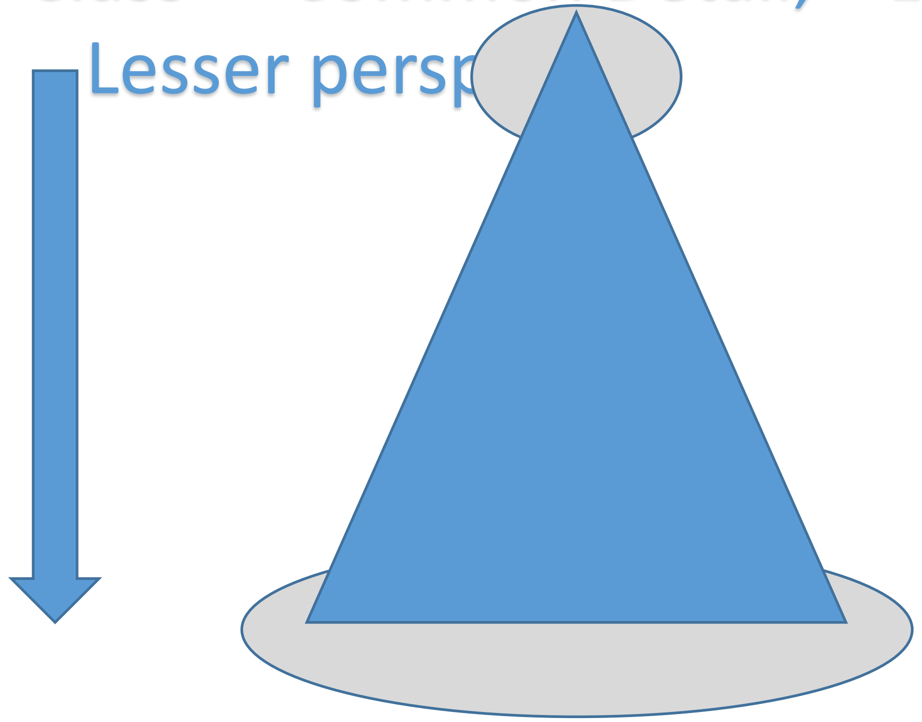
SUBCLASS – Wider Perspective, More Detail

## Q3. What is Widening Cast. Explain in Detail

Ans. Assigning the object instance of super class to the subclass. i.e. you are going from lesser perspective to wider perspective

Super Class -> Common Detail, Lesser information,

Lesser persp



Widening Cast

Sub Class– More information, More method, More properties, More Detail

How you can achieve the multiple inheritance? (VVI)

Ans.-> With the help of **Interface** we can achieve this.

## EVENTS in Procedural ABAP

Classical Report:
LOAD-OF-PROGRAM
INITIALIAZATION
AT SELECTION SCREEN
AT SELECTION SCREEN OUTPUT

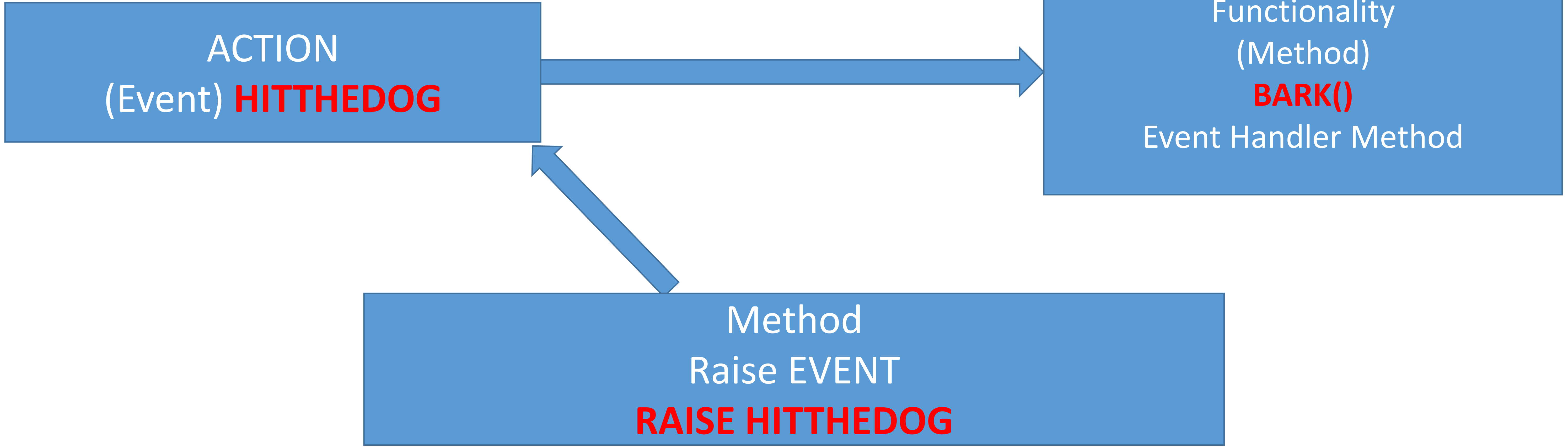
Interactive report:
**AT USER COMMAND**
**AT LINE SELECTION**

**<u>Module Pool:</u>**
PAI
PBO
POH
POV

## Events

An event is an action used for providing dynamic features to applications.

1. Create an Event
2. Create a Method to handle the Event( Event Handler Method)
3. Set up the linkage between Action and Event handler method
4. Register the Event
5. RAISE the Event in a separate Method
6. Create Program and from there call the method in step 5



$$\begin{tikzcd}\boxed{\text{Event}} \arrow[r] & \boxed{\text{Method}}\end{tikzcd}$$

Q. How a method is triggered when an event is fired in OOPS

Ans. -> We must have to register the method with the events

**SET HANDLER <METHOD> FOR <ALL INSTANCES/SPECIFIC INSTANCE)**

EVENTS : <EVENT_NAME>  EXPORTING  <PARAMETER NAME>.
METHOD:

## Summary for Events

## EVENTS

**What are events in SAP Classes?**

**Event is a mechanism by which method of one class can raise method of another class, without the hazard of instantiating that class.**

**What are the steps to create an event handler method?**

1. Define an event. ( Two types of Event -> 1.Instance Events(EVENTS) and 2.Static Events(CLASS-EVENTS)
2. Define a method. (This Method will be knwon as event handler method)
3. Link event and method and convert the method into event-handler method.
4. Create a triggering method which will raise the event.
5. Use set handler and register event handler method to a particular instance in the program.

```
SET HANDLER <Class/Object>-<EVT-HDLER-METHOD> FOR <ClassINSTANCE>. "Syntax
```

### How to Raise and Event

```
RAISE EVENT <EVENT NAME>
```

## Events with Parameter and Event handler Method

- An Event can have only Export Parameter.
- It will be only pass by value. Pass by reference is not supported.
- All the export parameter of the events can be there as import parameter of the EVENT HANDLER method
- Event handler method importing parameters can't have types

**SYNTAX(EVENT):**

```
EVENTS: hit_the_dog EXPORTING VALUE(im_event) TYPE i.
```

**SYNTAX(EVENT HANDLER METHOD)**

```
METHODS: barking FOR EVENT hit_the_dog OF cl_event_raiser
            IMPORTING im_event [sender] .
                                "Here type for im_event is not required.
```

**Note: 1. Here sender is only possible in case of instance method.**

**2. Sender parameter contains the reference of triggering Class**

## EVENTS – FOR ALL INSTANCES

SYNTAX:

```
 SET HANDLER go_event_handler->weeping FOR ALL INSTANCES .
```

**Note: FOR ALL INSTANCE addition is only valid for INSTANCE EVENT**

### Sender parameter

1. **Here sender is only possible in case of instance method.**
2. **Sender parameter contains the reference of triggering Class**

## EVENTS – DEREGISERATION AND REREGISTERATION

How to Deregister:

```
SET HANDLER go_event_handler->barking FOR ALL INSTANCES ACTIVATION abap_false.
      OR
SET HANDLER go_event_handler->barking FOR ALL INSTANCES ACTIVATION space..
```

How to Register:

```
 SET HANDLER go_event_handler->barking FOR ALL INSTANCES ACTIVATION abap_true.
```

**Note:** If you don’t specify ACTIVATION abap_true then also no problem. By default it takes.

## One Event : Multiple Handler Method

- One event can have multiple handler method.
- It gets executed in the same sequence in which it is registered.

```
METHODS: barking FOR EVENT hit_the_dog OF cl_event_raiser IMPORTING
         im_event sender  ."barking is event handler method

METHODS: weeping FOR EVENT hit_the_dog OF cl_event_raiser.
```

What syntax/command need to be used to trigger the method?
Ans. -> RAISE <EVENT NAME>

method1

Click

Method2

## STATIC EVETNS

Static Events Can be raised inside the Instance Method or static Method.

Main Difference:
While registering the handler we have to use only

**SET HANDLER** <HANDLER METHOD>.

We don’t have to give the “FOR” because it’s static event.
Static Event can be raised from instance Method as well as Class Methods

Note**: Inside the Static Method, Instance Event can’t be raised.**

## QUESTIONS: Give the Answers

Q1. Can u raise the static event inside the instance method? **Yes**

Q1. Can u raise the Instance event inside the instance method? **Yes**

Q2. Can u raise the instance event inside the static method? **NO**

Q3. Can a handler method be Static or Instance? **Yes**

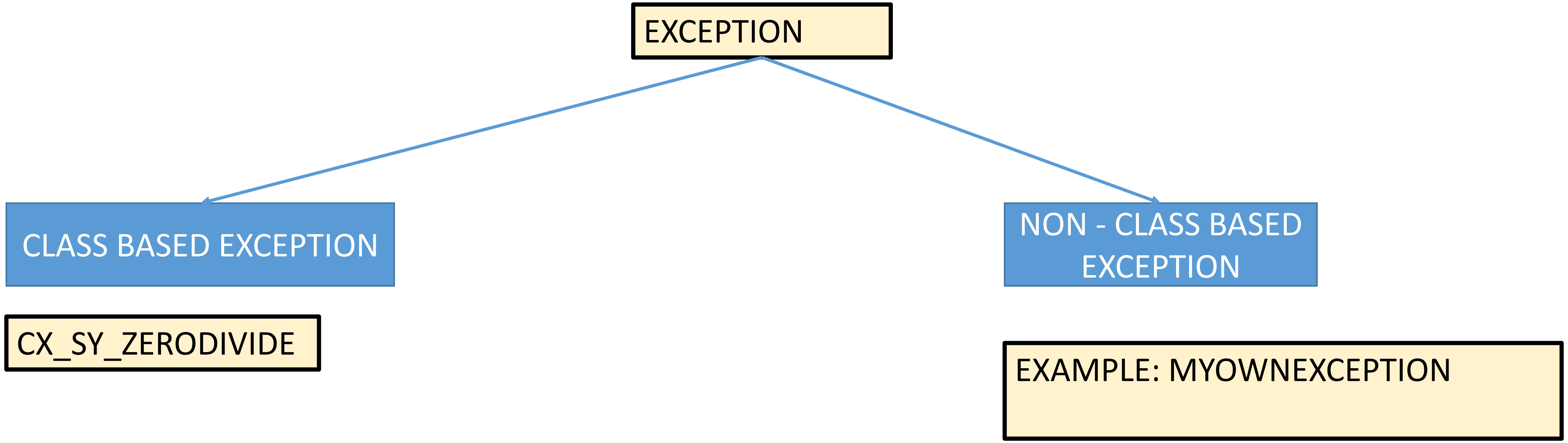
Q4. Does the handler method depend on Event type( Static Event or Instance Event ). **No**

## EVENTS

| Parameters: | INSTANCE EVENT | STATIC EVENTS |
|---|---|---|
| Registration | 1. **SET HANDLER <HANDLER Method> for <object> /<ALL INSTANCES>** | 1. **SET HANDLER <HANDLER METHOD>** |
| Method In which type of the method event will be raised? | 2. **Only inside the instance Method** | 2. **Inside the instance method and Static method** |
| Sender reference | 3. **Supported** | 3. **Not Supported** |

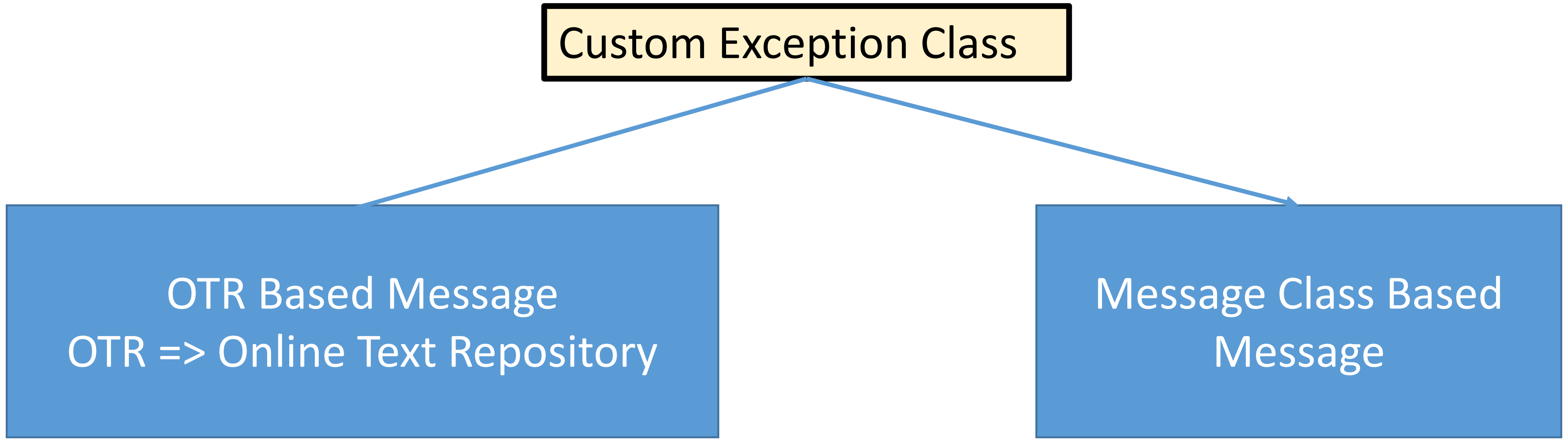
## What are the Different types of Exception?

Ans. There are two types of Exceptions -



What is the very first class/root class for exception Class?
Ans. -> **CX_ROOT.**

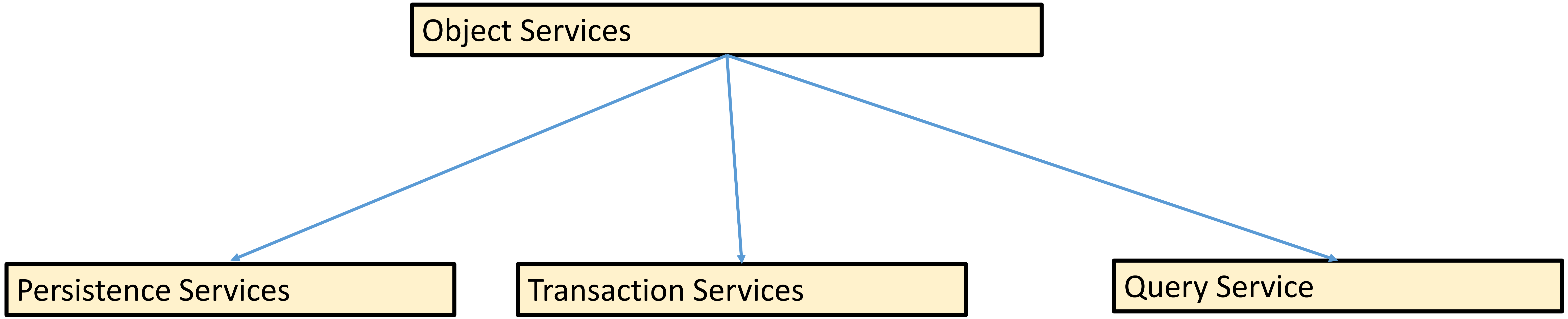
What are the different types of Custom Exception Class?



What is the use of CLEANUP Block? (VI)
Ans. To **free** the memory / internal / external resources.

```
CLEANUP.

AJK

AJKLDSFJL
LJKALDSF




ENDCLEANUP.
```

What are the different types of Object Services ?



## Persistence Service/object

VVI

Persistence data → Transient Data (Non-Persistent Data)

The data which persists permanently in the system is known as persistent data.

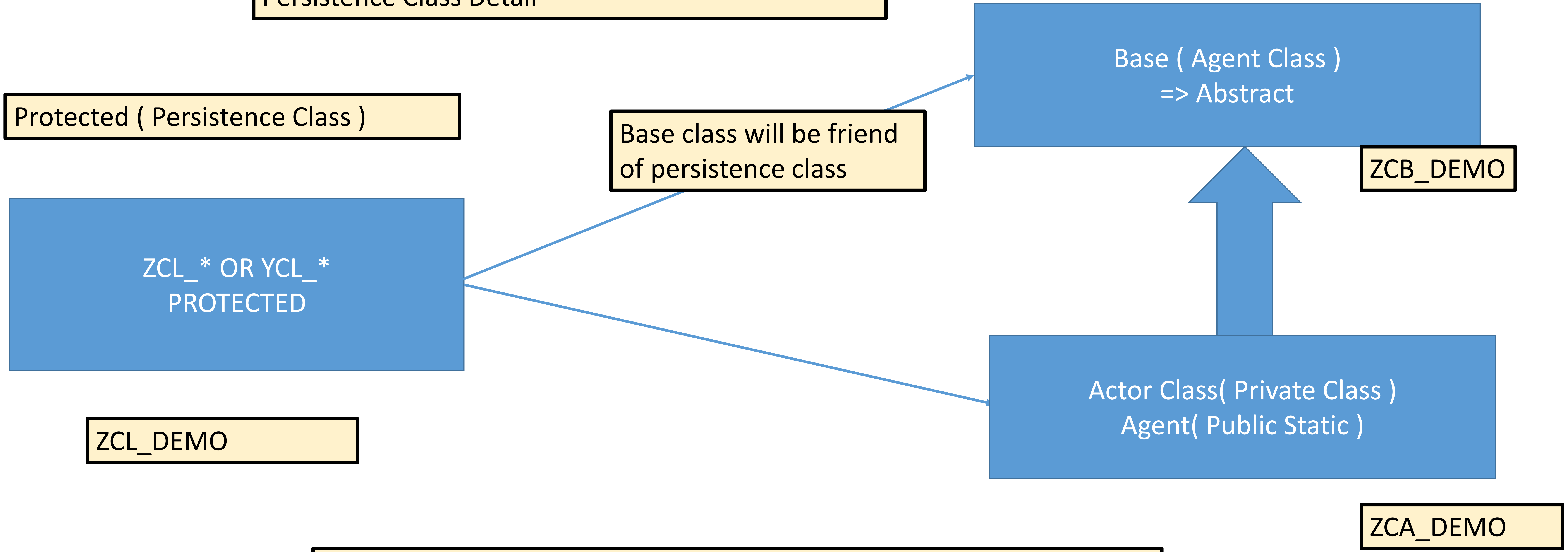
DML
MODIFY  <DB> FROM <WA>/ <TABLE table>
INSERT <DB> from <WA>/ <TABLE table>

If you want to store the state of the object(data object/Class object => wa/internal/var/Instance object) permanently in the db in the OOPS way we have the Persistence service

The data which is going to be held only during the runtime of the program that is known as Transient Data.
e.g. Internal table, variable, work area

Report ZPKTEST.

## Persistence Class Detail


Protected ( Persistence Class )
ZCL_* OR YCL_*
PROTECTED
ZCL_DEMO
Base class will be friend of persistence class
Base ( Agent Class )
=> Abstract
ZCB_DEMO
Actor Class( Private Class )
Agent( Public Static )
ZCA_DEMO

You require a database which is going to save the data.
DML

For Detail : https://youtu.be/Erh5EuPCtt4

## Singleton Design Pattern

VVVVI -

Use Case / Problem Statement -> How to deal with a logger Class

**Singleton Design Pattern –**
1. It will help you get the same object every time. It will not allow you to instantiate a class multiple time.

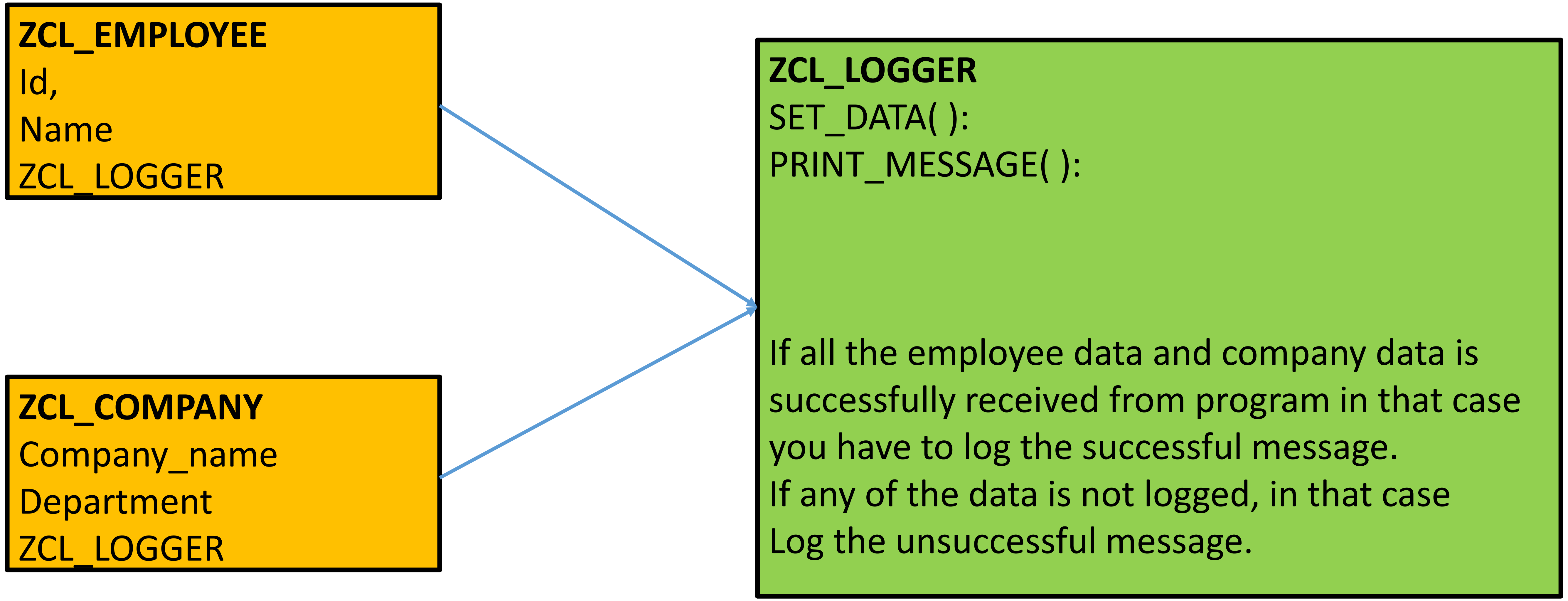
Single(Only one) + Ton( Instance )

**Steps to create the Singleton Design Pattern:-**

1. Define the class as a **Private**
2. Define one Method : **Public** + Static with a **returning parameter** of type ref to the same class.
3. Define the Attribute of type ref to the same class: **Private** + Static – which will hold the instance of the class
4. Write the logic inside the method

```
    IF <ATTRIBUTE> is bound.
            <RETUNING_PARAM_OF_METHOD> =  <ATTRIBUTE>.
    ELSE.
            CREATE OBJECT <ATTRIBUTE>
            <RETUNING_PARAM_OF_METHOD> =  <ATTRIBUTE>.
    ENDIF.
```

SINGLETON PATTERN – Logger Class Example

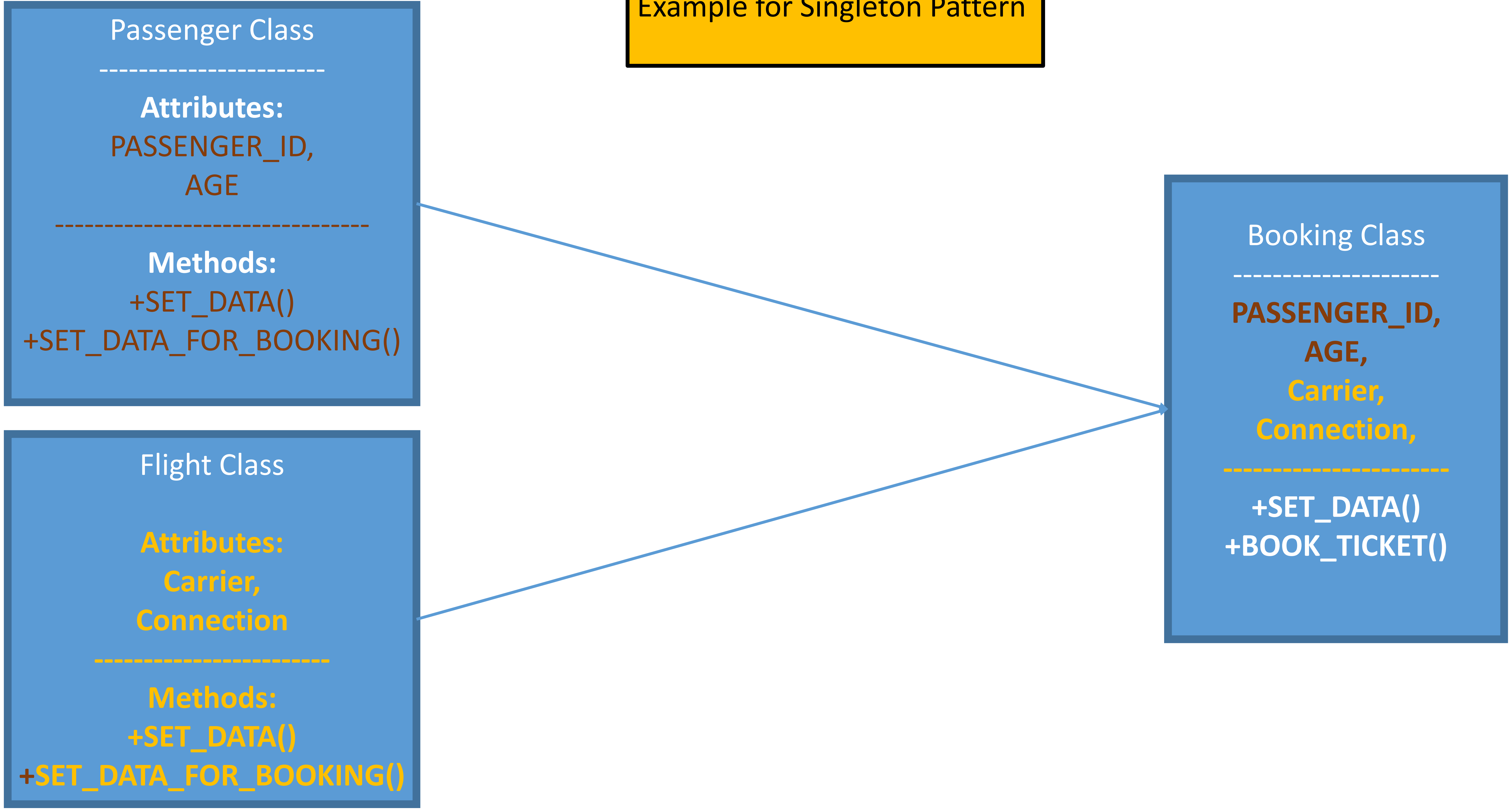


If all the employee data and company data is successfully received from program in that case you have to log the successful message.
If any of the data is not logged, in that case
Log the unsuccessful message.

Report ZLOGGER

O_LOGGER->PRINTLOG()

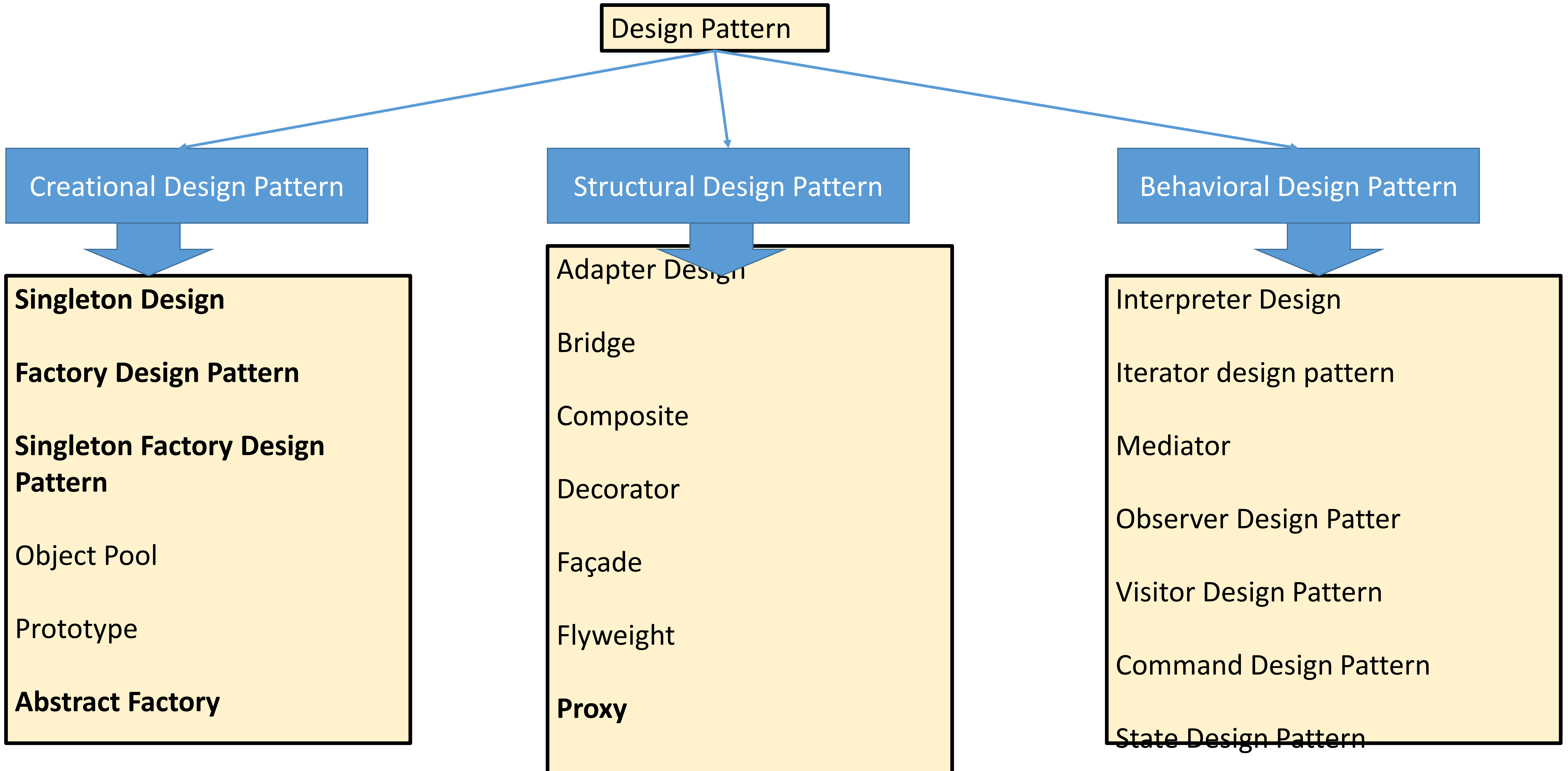
## Example for Singleton Pattern



## Factory Design Pattern

This comes under the category of creational Design Pattern

It is used to hide the complexity of object creation and produces the object on demand


Design Pattern
Creational Design Pattern
Structural Design Pattern
Behavioral Design Pattern
**Singleton Design**
**Factory Design Pattern**
**Singleton Factory Design Pattern**
Object Pool
Prototype
**Abstract Factory**
Adapter Design
Bridge
Composite
Decorator
Façade
Flyweight
**Proxy**
Interpreter Design
Iterator design pattern
Mediator
Observer Design Patter
Visitor Design Pattern
Command Design Pattern
State Design Pattern

## MVC – Model view Controller

**Model**: Data Provider

**View**: Interface

**Controller**: Actual Logic/Business Logic/ Data Manipulation

Spring -> MVC Design pattern
SAP UI5 -> MVC Design Pattern
FIORI -> MVC Design Pattern

## Proxy Design Pattern

- Sometimes you need to delay the instantiation of the object because it may be very costly to instantiate as it may take large amount of resources as memory and CPU without having any fruitful result. It will take extra time as well

- Why we don't get fruitful result because we may have bad data, which may further stop processing.

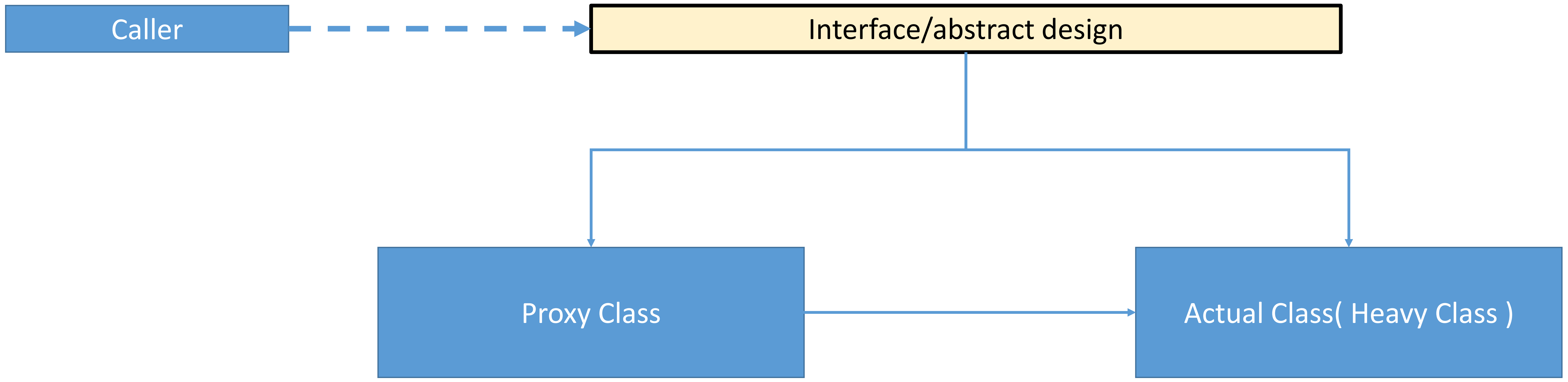
- Hence it's always better to first validate all the required data in a separate class

- This another class is known as proxy class because it gets executed first, in stead of actual class.

- ➢ **Proxy class is light weight class.**
- ➢ **It hold the implementation of validation logic**
- ➢ **If validation is successful then only go and create the actual/heavy class object and execute the functionality.**

Very- very heavy class- excel,
outlook,
sap heavy class.

Proxy Design Pattern Implementation



Note:-

**Generally proxy design pattern is used in ABAP when we have to instantiate the NON ABAP object.**

**Because, NON-ABAP objects are very heavy ( e.g. EXCEL object and OUTLOOK object )**

## **Adapter** Design Pattern

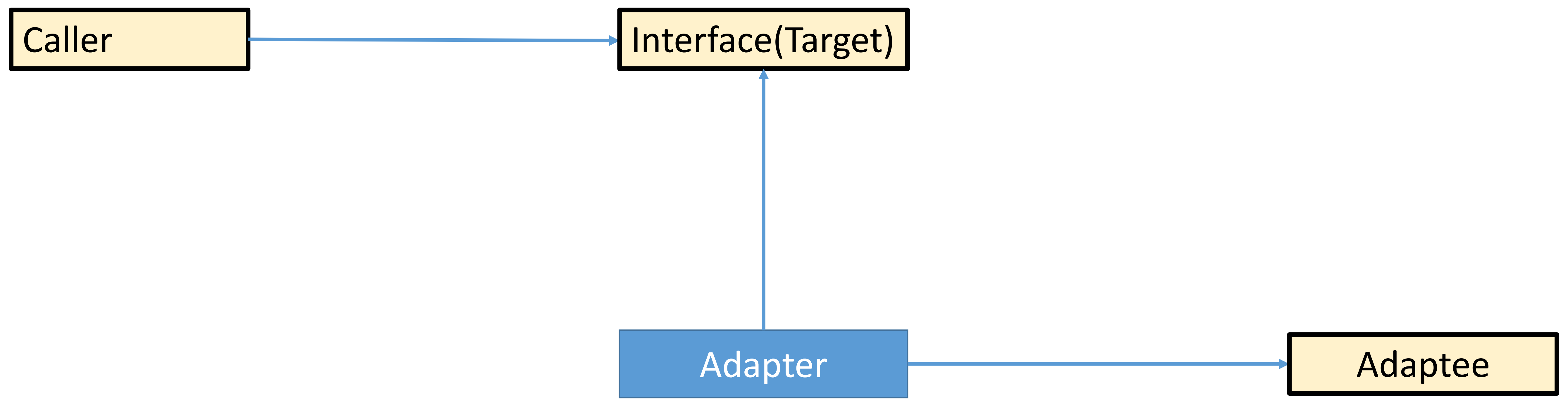
1. Adapter converts the objects which are incompatible due to the difference in the interface. By Implementing the Adapter, we can allow classes to work together, which can't work without Adapter.
2. Sometimes we come across the scenario where client expects to have certain interface. Already we have an object which is satisfying the client need then why to create the another object. We can create a wrapper and use it.

3. In real time we have USB Adapter, Wi-Fi Adapter etc.

How to run a function module in Batch Job?

You already have a function module to sum( say ), and your client came up with need to pass the parameters – How to do?

Adapter Design Pattern Implementation



## Facade Design Pattern

Facade is a French word meaning – face or frontage

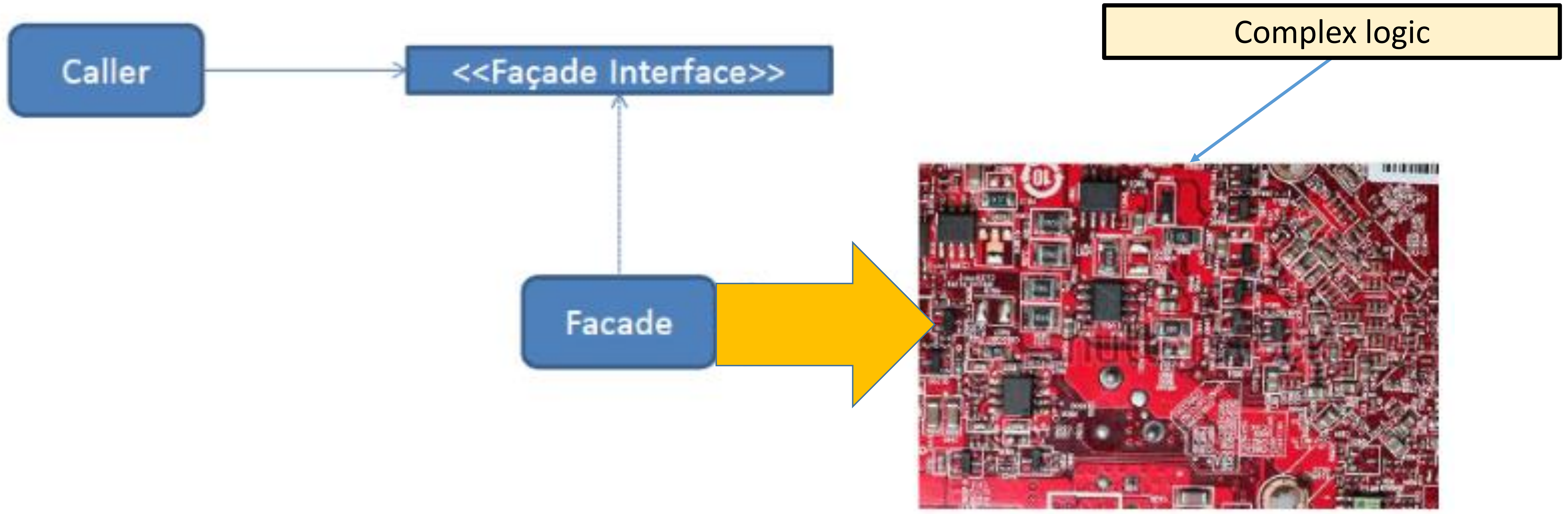
Consider a scenario where your team is developing a big application. Application have been divided into different parts and assigned to different DEVELOPERS.

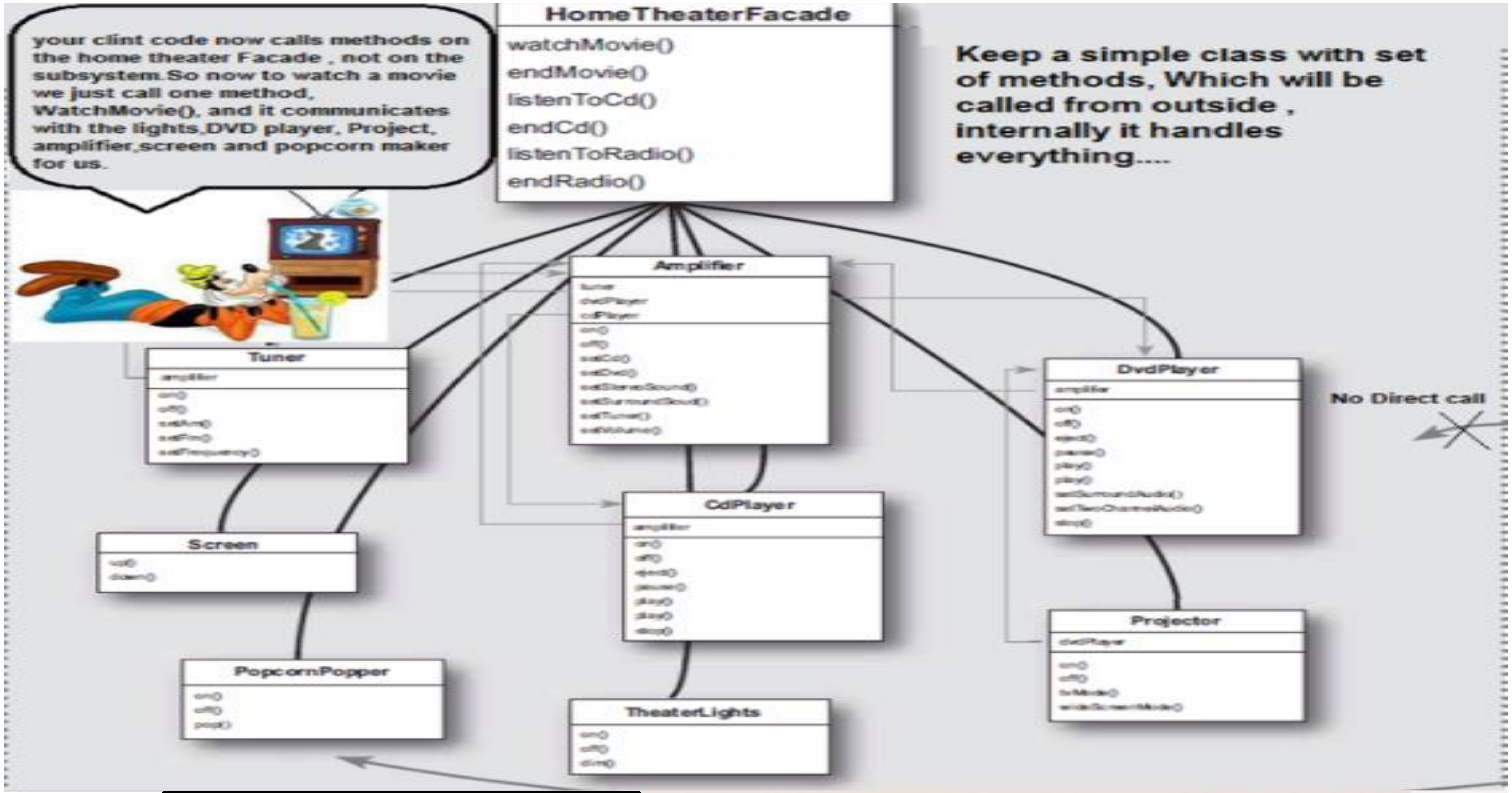
Now suppose , every one has to handle the message in that application different languages.

So here everyone would end up with there own function module or methods for displaying the message or capturing the message.

So what is solution: Façade : So Team can decide to create a single function module / Method which will handle the message and this FM/method can be used by everyone. Now tomorrow if some extra addition need to be done related with message, that single FM will be the place where one can go and change and communicate to other team members.

## Façade Design Pattern



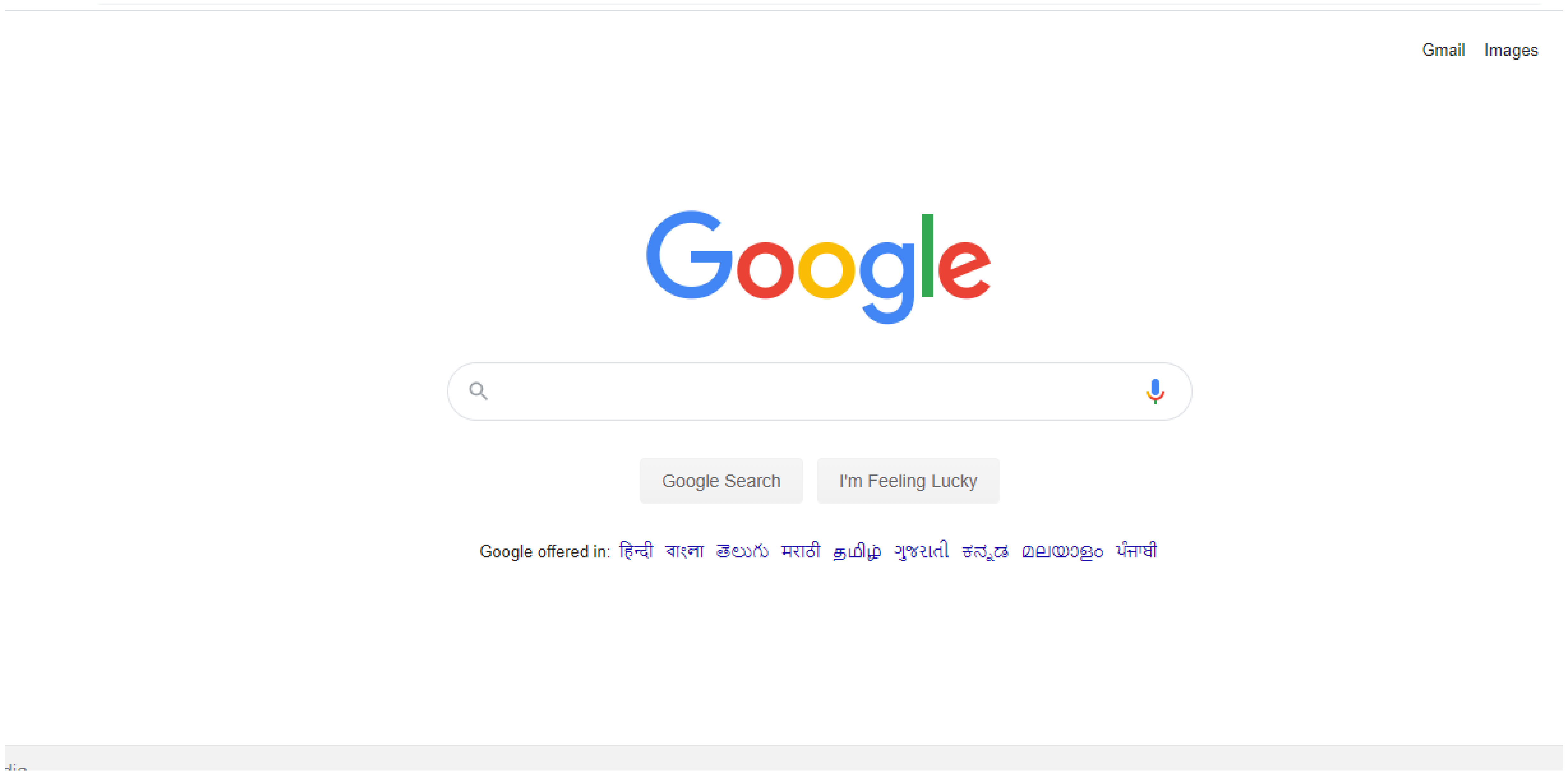


your clint code now calls methods on the home theater Facade , not on the subsystem. So now to watch a movie we just call one method, WatchMovie(), and it communicates with the lights, DVD player, Project, amplifier, screen and popcorn maker for us.

Keep a simple class with set of methods, Which will be called from outside , internally it handles everything....

Source: Internet(sapignite.com)

Façade Design Pattern -> Google.come

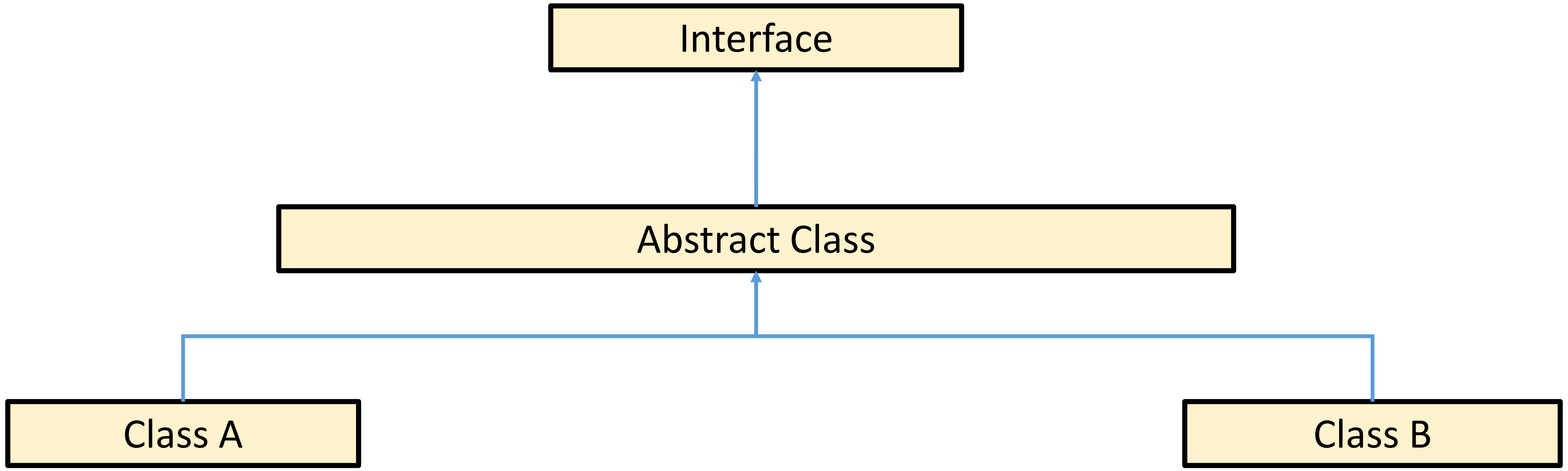


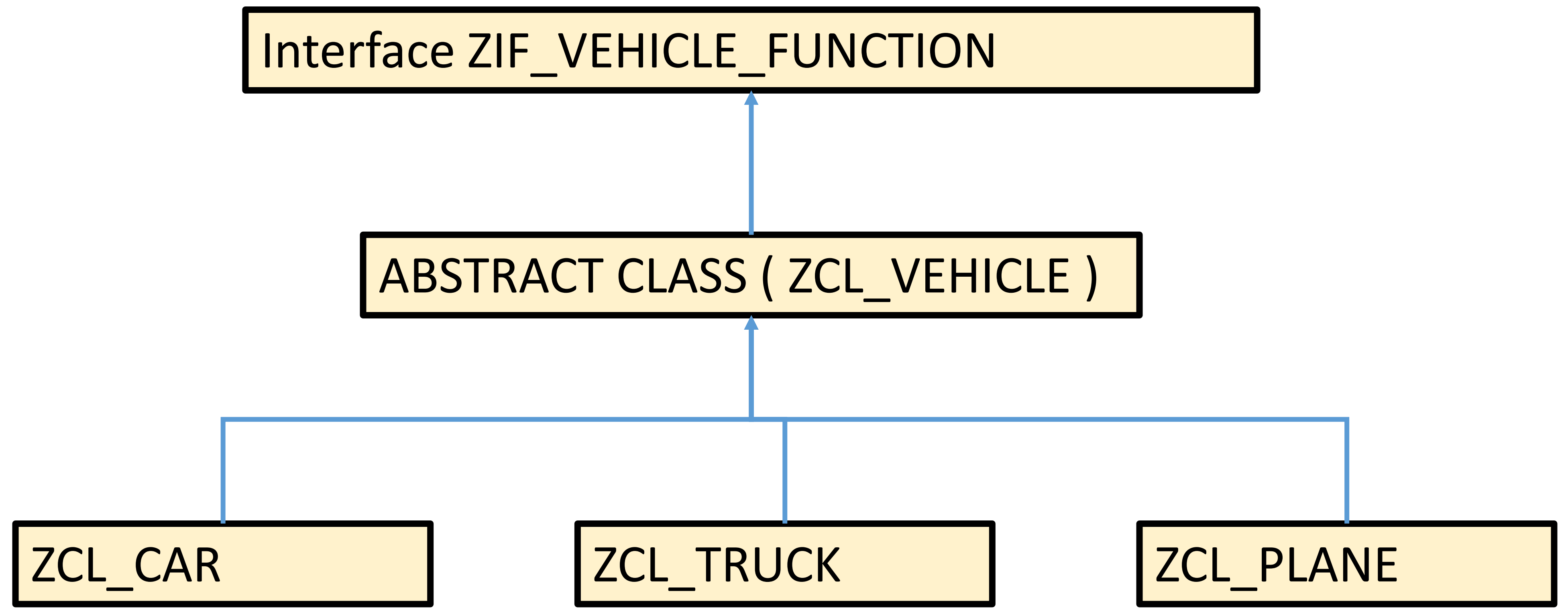
Gmail Images

Google
Google Search
I'm Feeling Lucky
Google offered in: हिन्दी বাংলা తెలుగు मराठी தமிழ் ગુજરાતી ಕನ್ನಡ മലയാളം ਪੰਜਾਬੀ

## Template Design Pattern

This pattern is required if two different components have significant similarities




Interface ZIF_VEHICLE_FUNCTION
ABSTRACT CLASS ( ZCL_VEHICLE )
ZCL_CAR
ZCL_TRUCK
ZCL_PLANE

Template Design Pattern

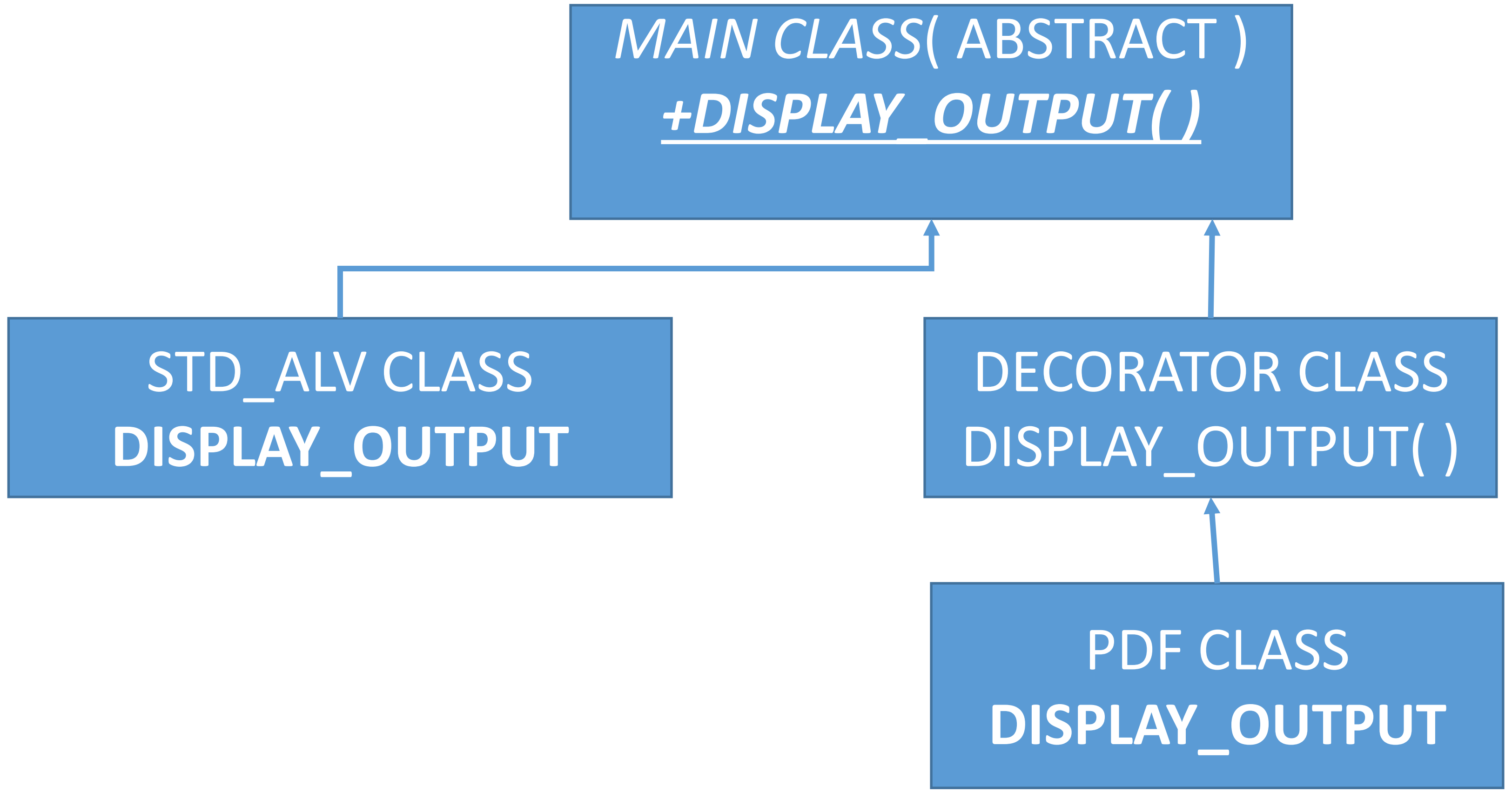
ZCL_VEHICLE: ( **ABSTRACT CLASS**, CONCRETE CLASS OR **INTERFACE** )

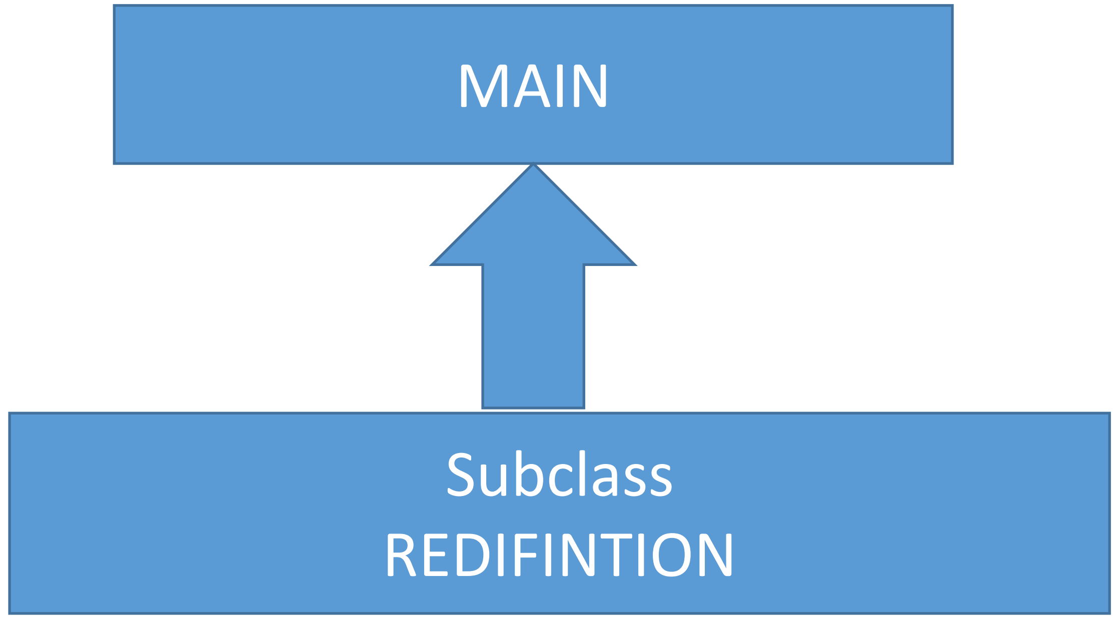
1. RUN( )
2. ACCELARATE( )
3. SPEED( )
4. SOUND( )

## Decorator design Pattern

Decorator ( Decoration )

1. In Decorator design pattern we use the RDEFINITION of Existing method





ALV OUTPUT
PDF OUTPUT

## Decorator Design Pattern

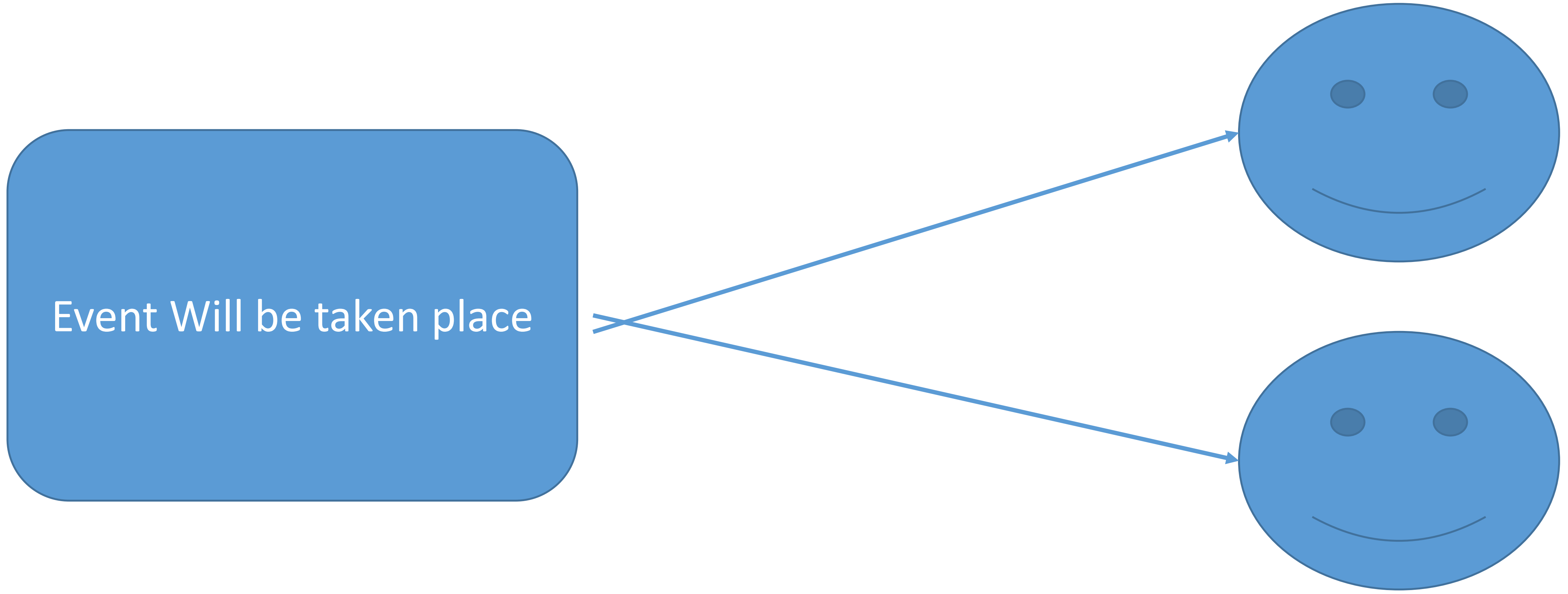
1. When we require to extend the behavior of certain objects at run time then we use the **Decorator Design Pattern**
2. This can be separate than the existing instances of the same class
3. This can be achieved by creating certain environment( Infrastructure ).

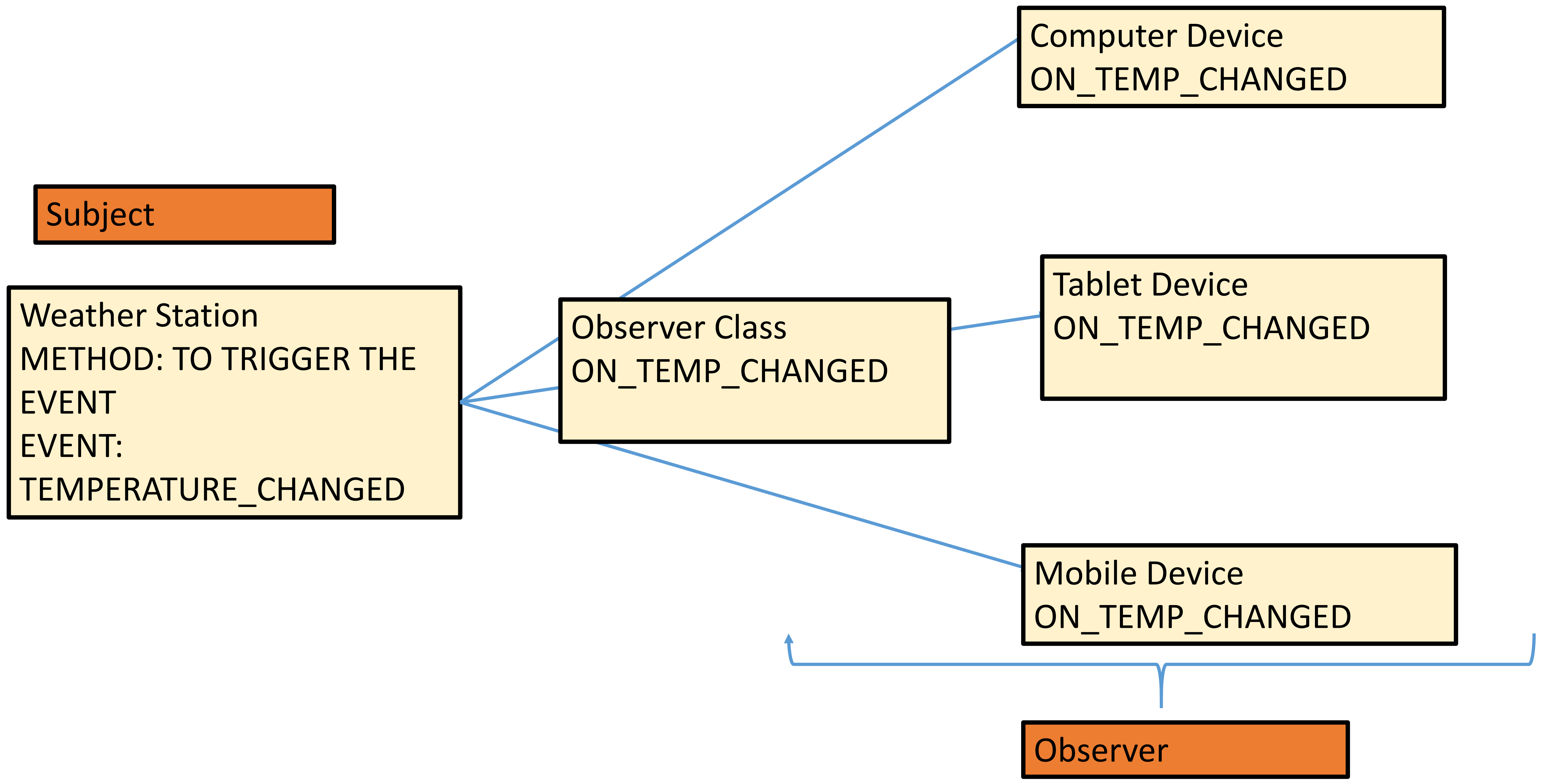
## Steps to Follow to adopt the Decorator Design Pattern

1. Identify the common method and put inside the Abstract Class
2. Create a Decorator Class – This is nothing but subclass of the main class. This is also called component
3. In the **subclass**( i.e. in **Decorator Class** ), create an attribute with type ref to Main Class
4. Create a constructor in Decorator Class with importing parameter REF TO Main Class. Set the attribute from this parameter
5. Redefine all other methods in decorator class inherited from Main Class. Just call the same method using the reference of the main class
6. Create **subclass of the decorator** when a new behavior is required
7. Redefine the **methods**
8. Write the main logic using a helper variable of type ref to Main Super Class

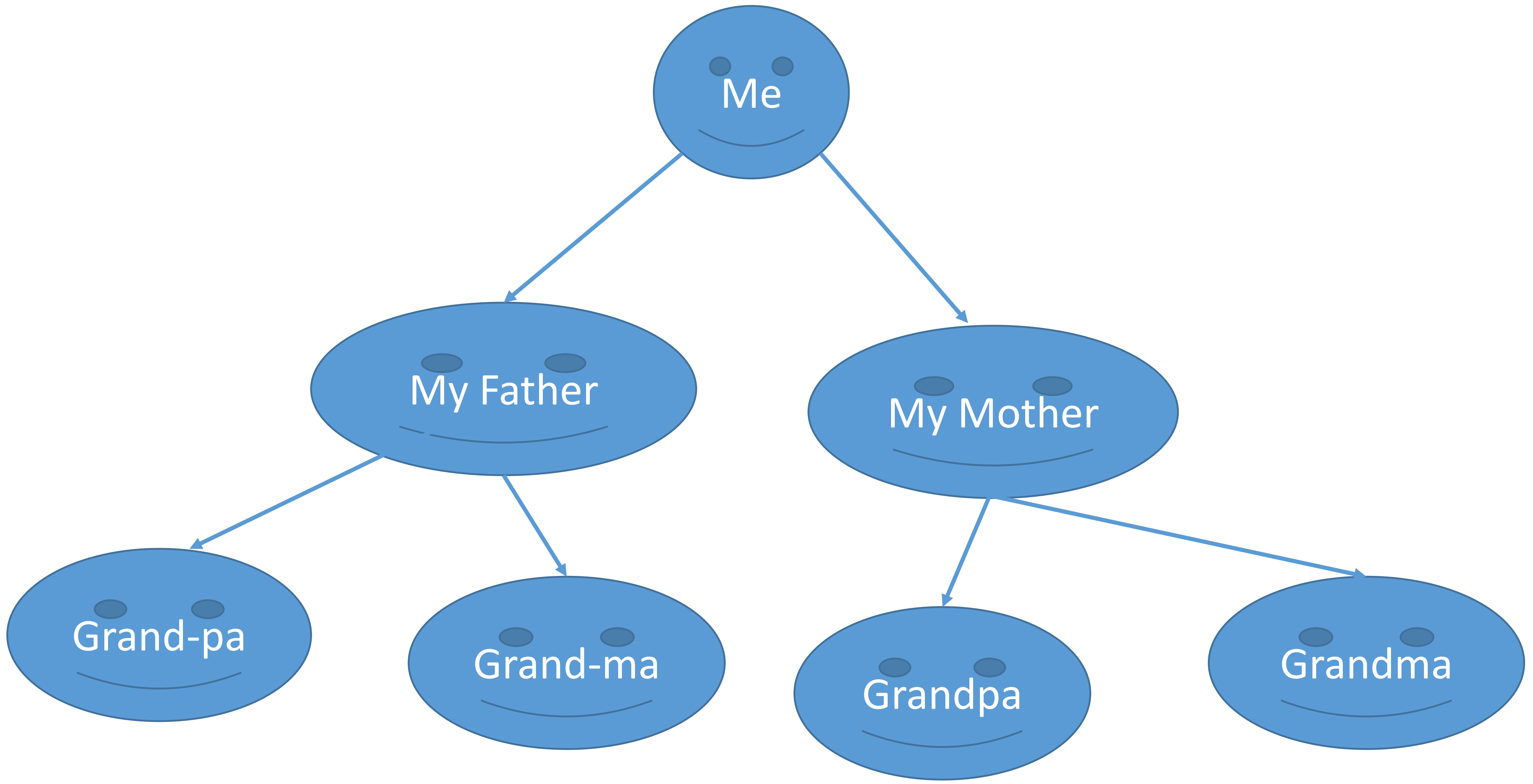
## Observer Design Pattern

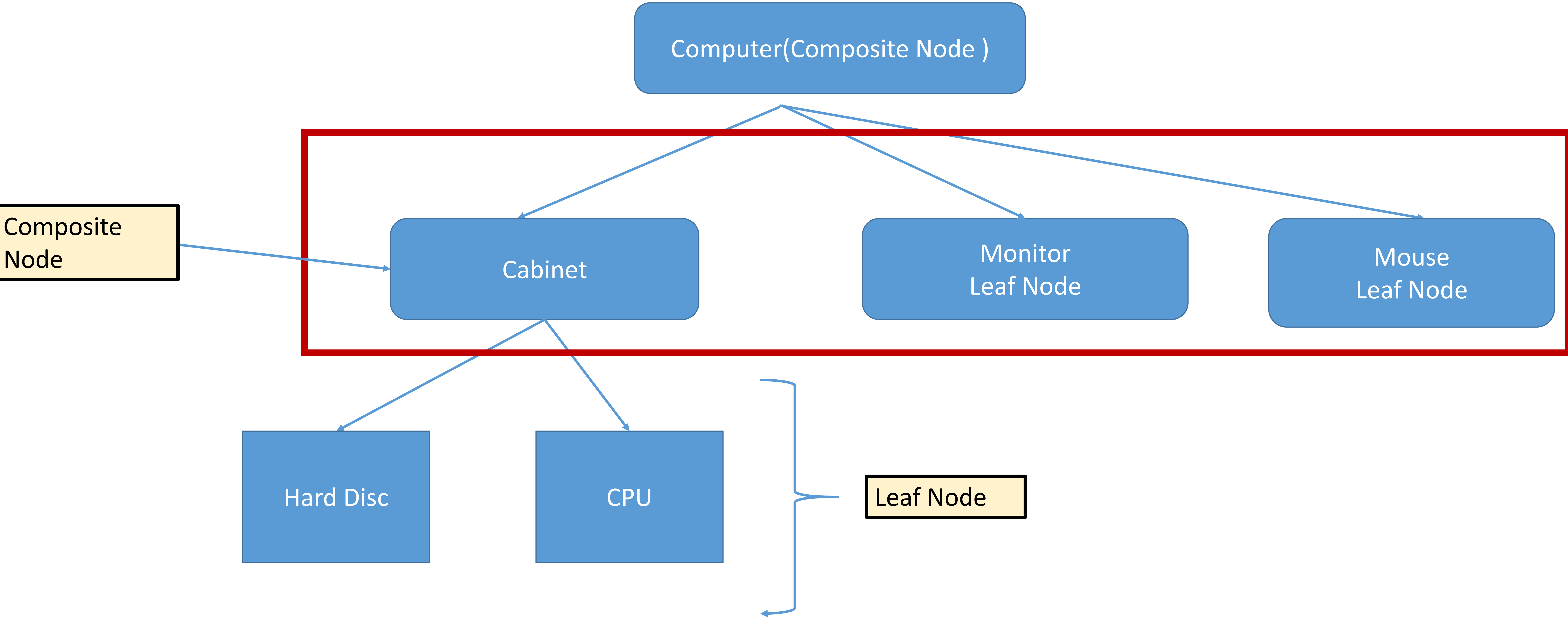
Observer: One who observes

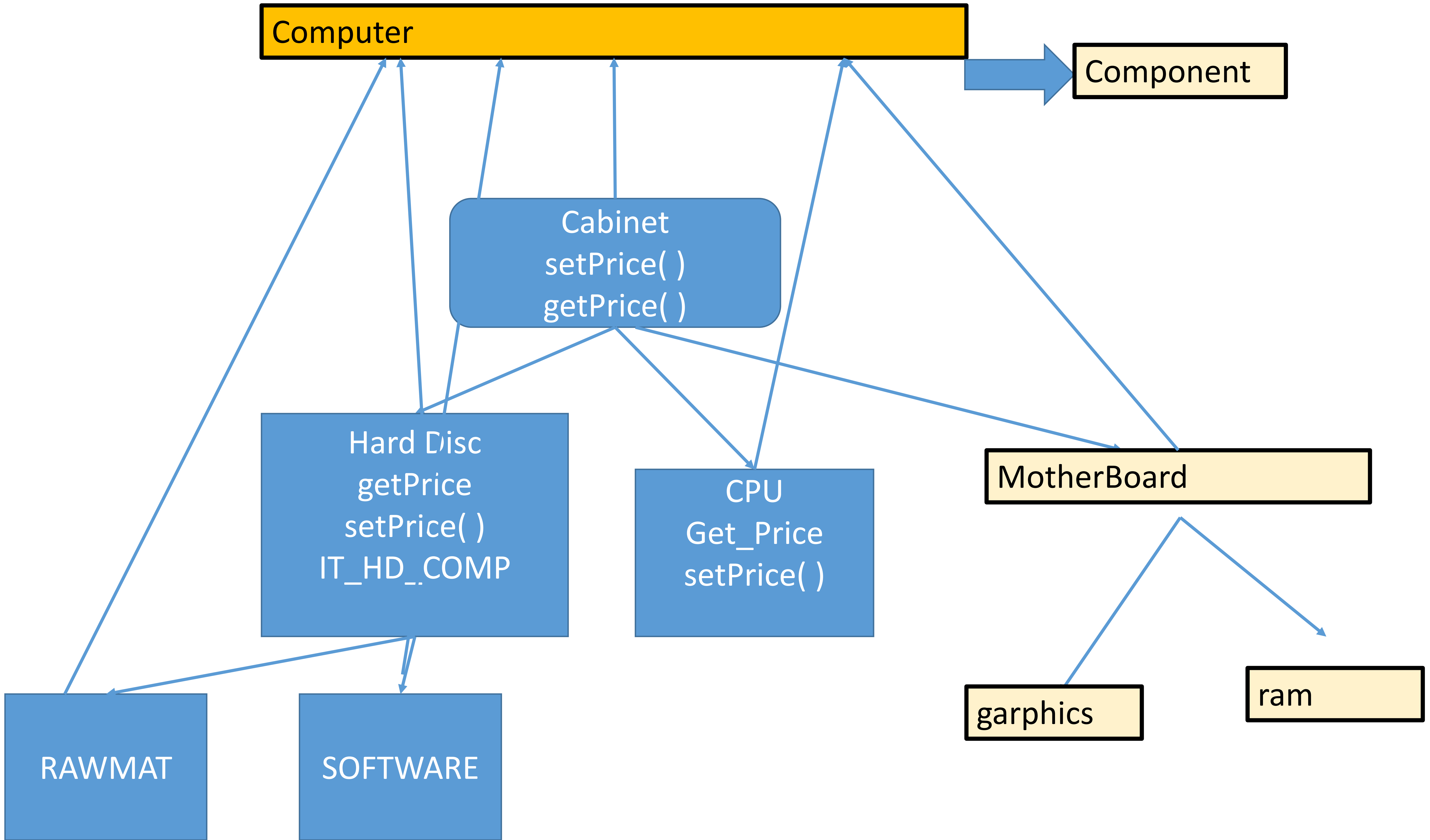



Subject
Weather Station
METHOD: TO TRIGGER THE EVENT
EVENT: TEMPERATURE_CHANGED
Observer Class
ON_TEMP_CHANGED
Computer Device
ON_TEMP_CHANGED
Tablet Device
ON_TEMP_CHANGED
Mobile Device
ON_TEMP_CHANGED
Observer

**Composite** Design Pattern( Structural Design Pattern)




Computer(Composite Node )
Composite Node
Cabinet
Monitor Leaf Node
Mouse Leaf Node
Hard Disc
CPU
Leaf Node


Computer
Component
Cabinet
setPrice( )
getPrice( )
Hard Disc
getPrice
setPrice( )
IT_HD_COMP
CPU
Get_Price
setPrice( )
MotherBoard
garphics
ram
RAWMAT
SOFTWARE

What is Strategy design pattern?

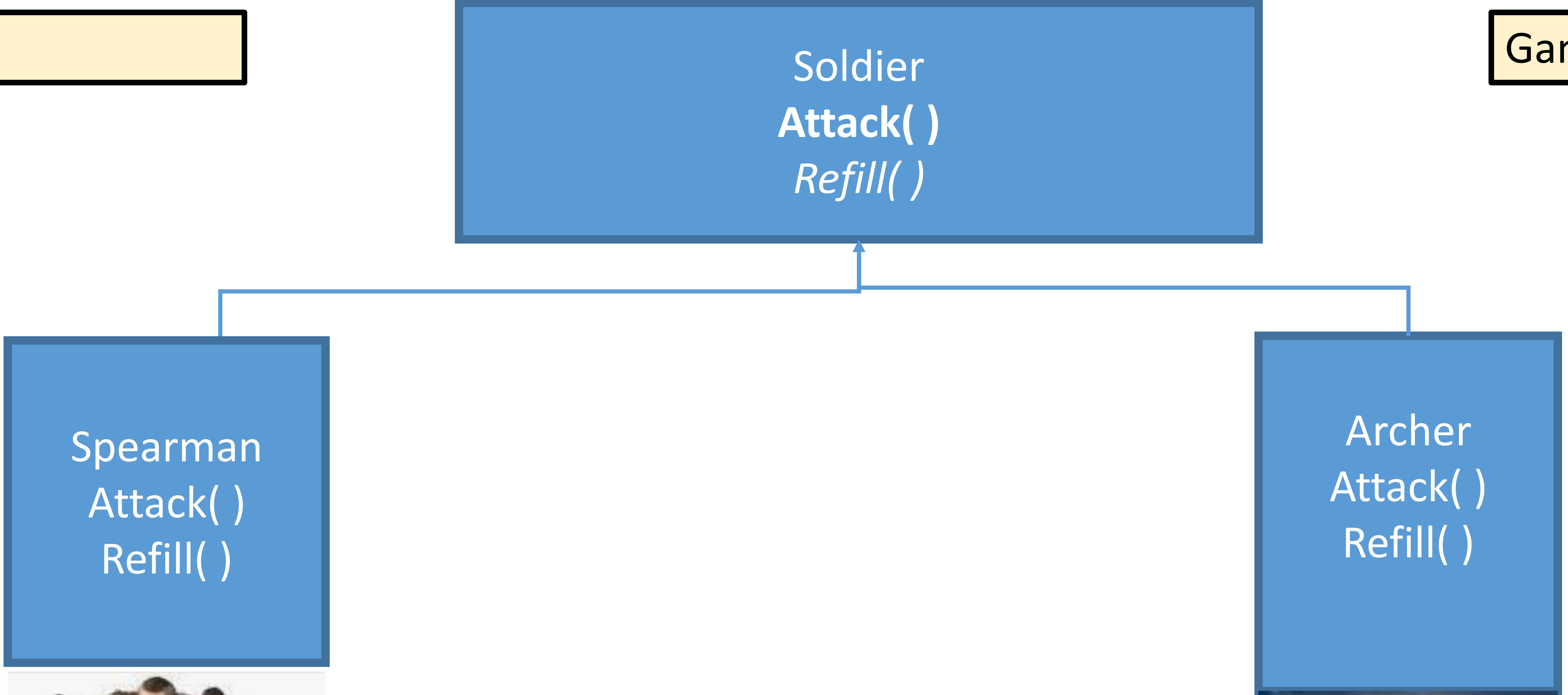
## **Strategy** Design Pattern

As per Wikipedia definition

In computer programming, the **strategy pattern** (also known as the **policy pattern**) is a behavioral software design pattern that enables selecting an algorithm at runtime. Instead of implementing a single algorithm directly, code receives run-time instructions as to which in a family of algorithms to use.

Strategy Pattern

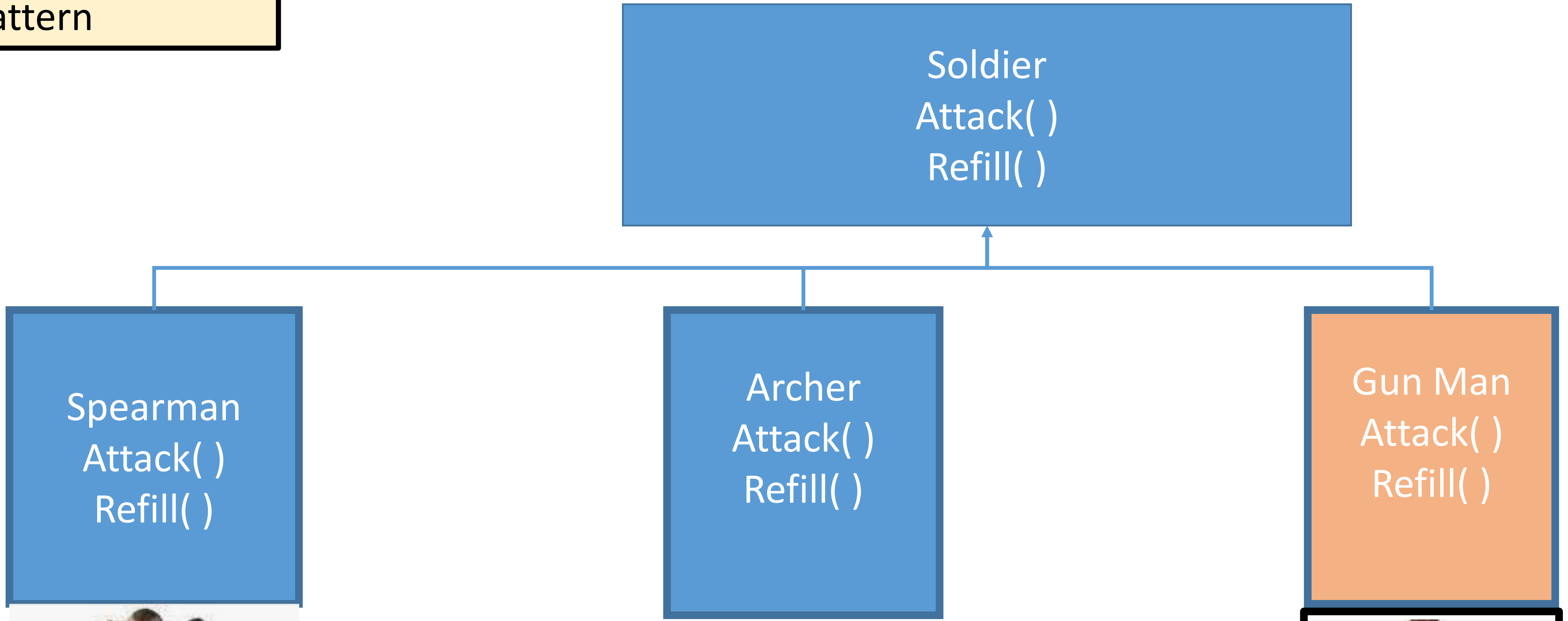
Game Development



Refill-> Pause for 5 seconds

Strategy Pattern

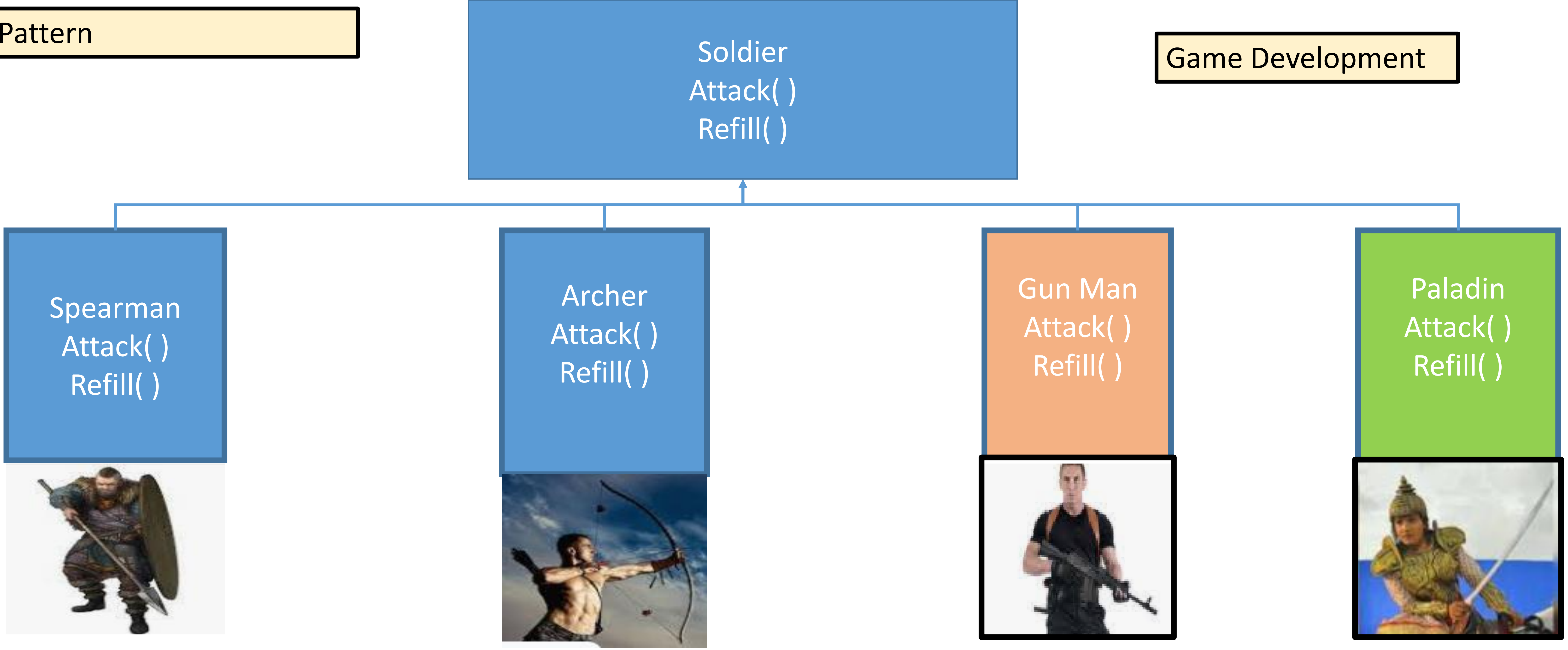
Game Development



Refill-> Pause for 5 seconds

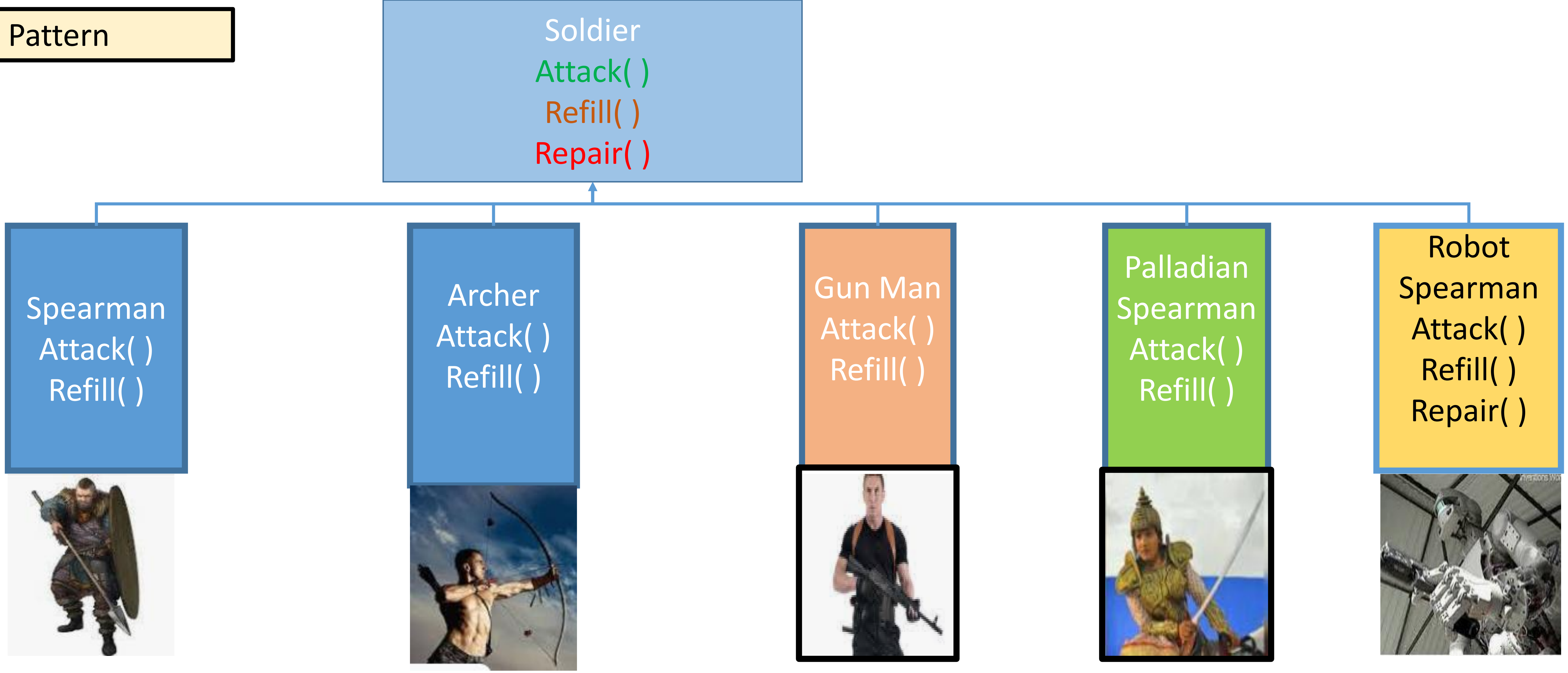
Strategy Pattern

Game Development



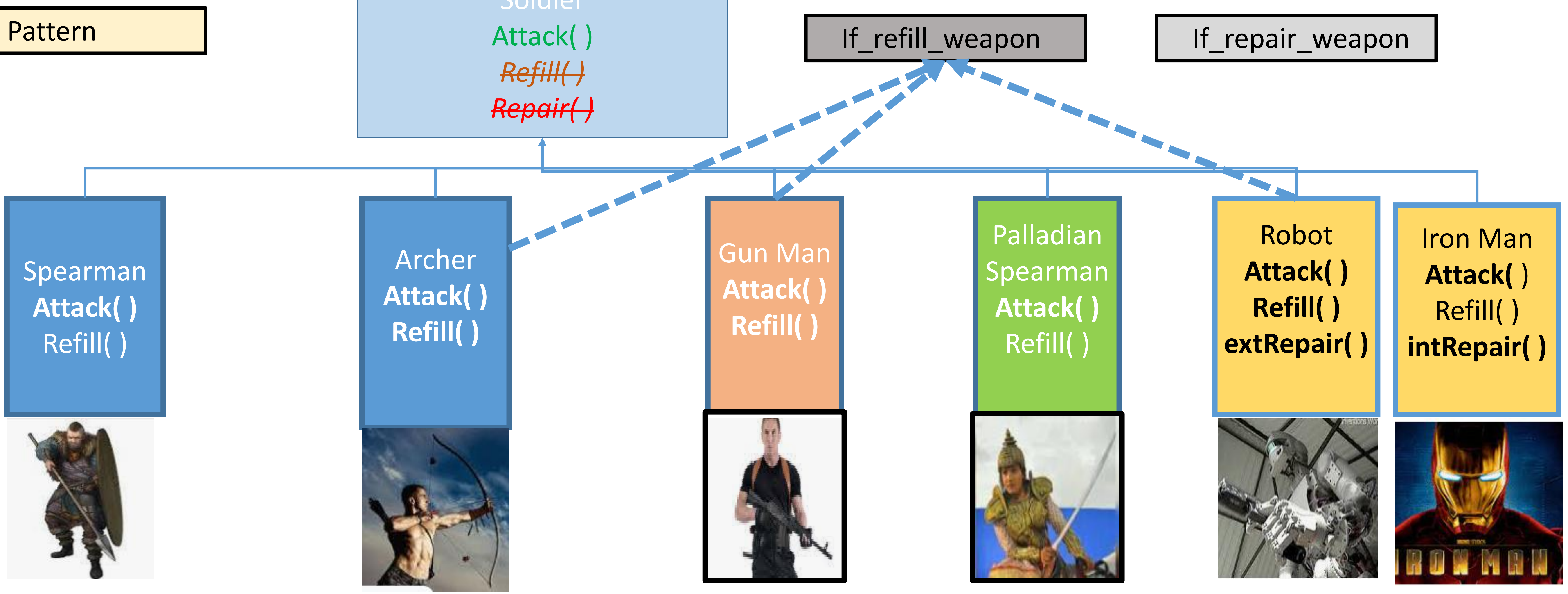
Refill-> Pause for 5 seconds

Strategy Pattern



Refill-> Pause for 5 seconds

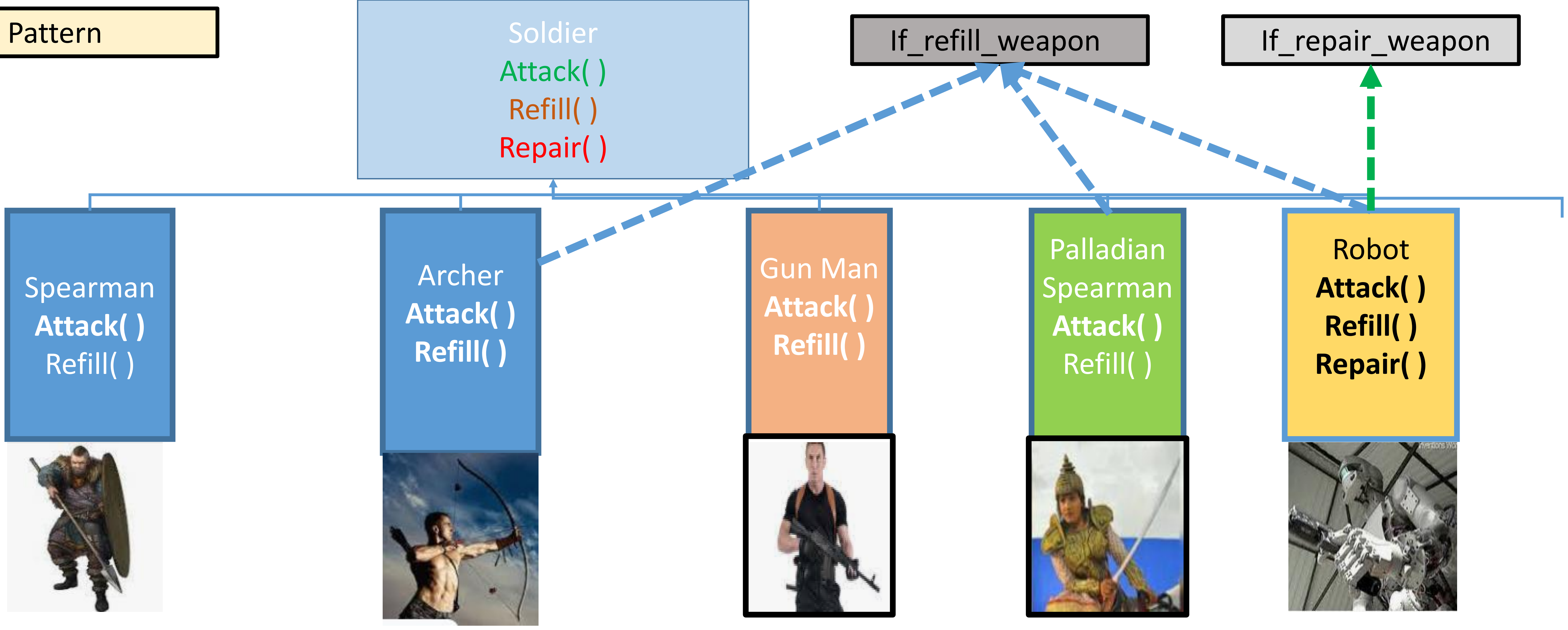
## Strategy Pattern



| | |
|---|---|
| Refill | -> Pause for 5 seconds |
| Gunman | –> **Time** based feeling |
| Archer | -> **Count** based feeling |

| | |
|---|---|
| Repair | -> Repair the Weapon |
| Robot | –> **External Repair** |
| Iron Man | -> **Internal Repair** |

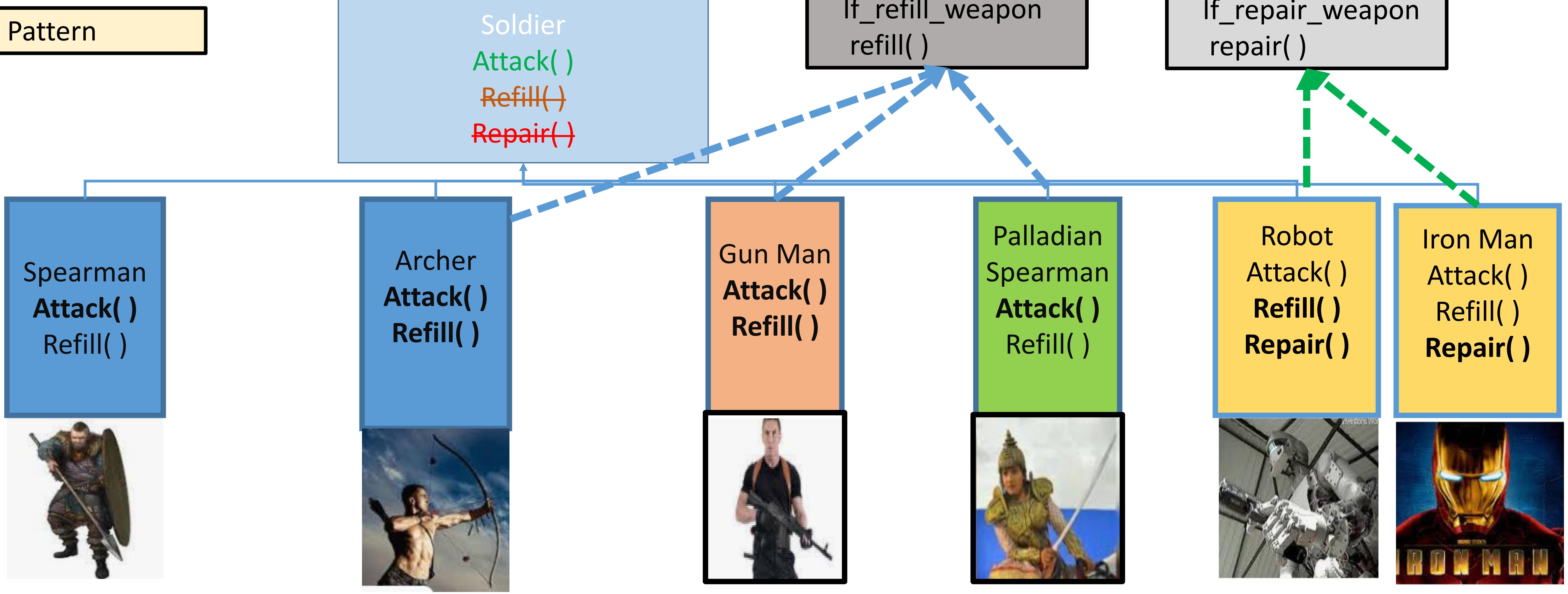
## Strategy Pattern



Refill -> Pause for 5 seconds
Gunman –> **Time** based feeling
Archer -> **Count** based feeling

Repair -> Repair the Weapon
Robot –> **External Repair**
Iron Man -> **Internal Repair**

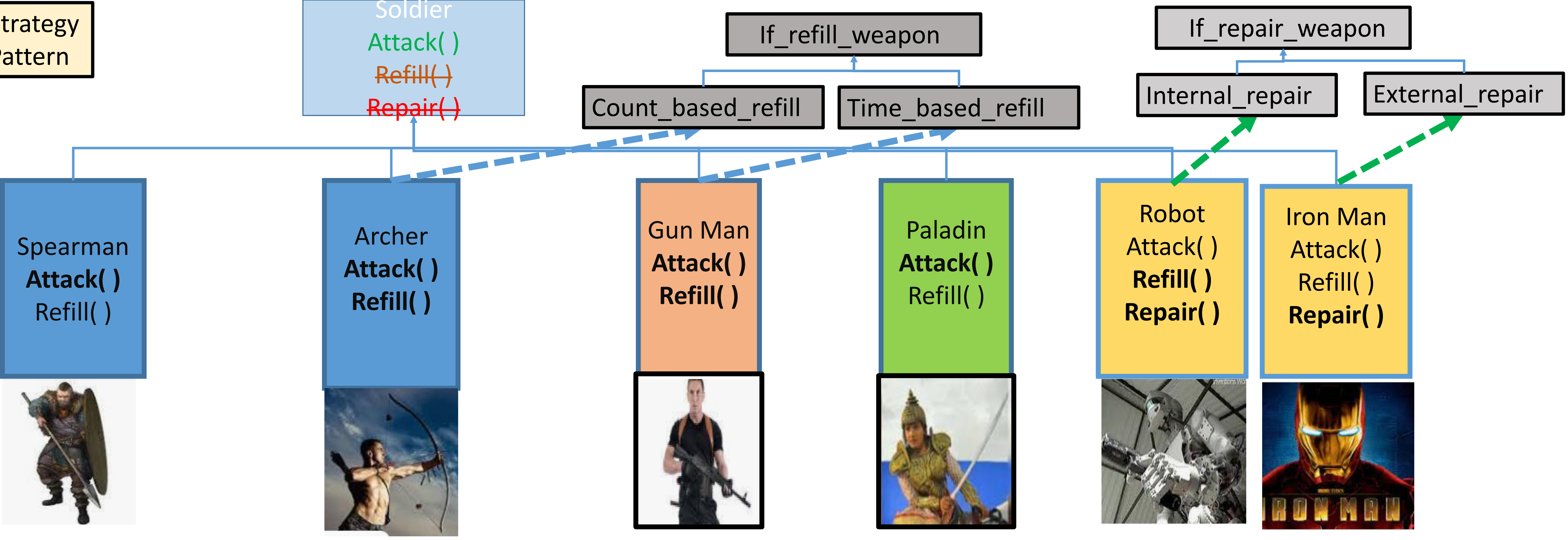
## Strategy Pattern



| | |
|---|---|
| Refill | -> Pause for 5 seconds |
| Gunman | –> **Time** based feeling |
| Archer | -> **Count** based feeling |

| | |
|---|---|
| Repair | -> Repair the Weapon |
| Robot | –> **External Repair** |
| Iron Man | -> **Internal Repair** |

trategy
Pattern



Refill -> Pause for 5 seconds
Gunman –> **Time** based feeling
Archer -> **Count** based feeling

Repair -> Repair the Weapon
Robot –> **External Repair**
Iron Man -> **Internal Repair**

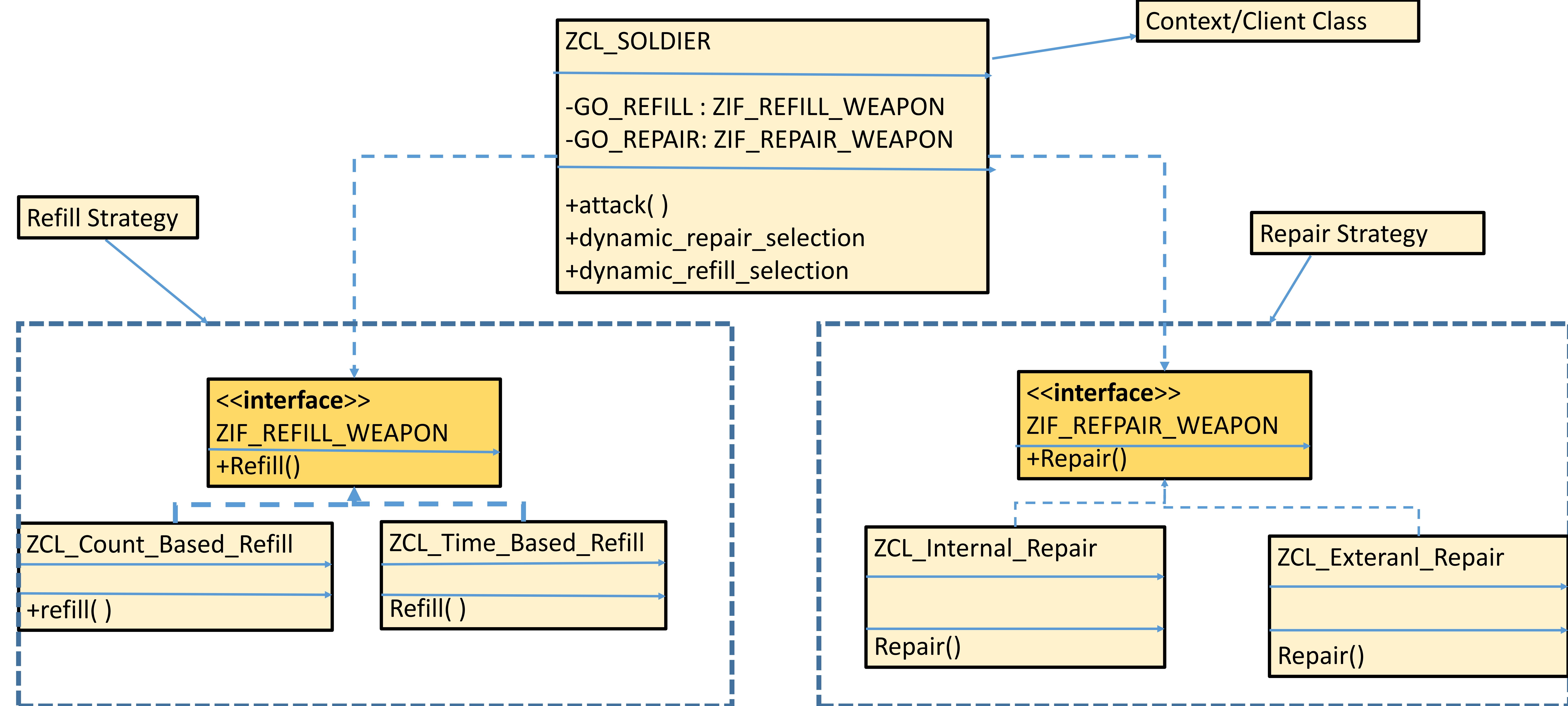
## Strategy Pattern: Conclusion

1. This comes under the Behavioral Pattern
2. In this Pattern Class Behavior/algorithm Changes at run time

**Hope you enjoyed ☺**

Approach

1. Identify the changeable variable and take it out and put in an interface or Abstract Class
2. Encapsulate in family of algorithm separately in concrete classes
3. Client should be using these interface as references. Client should not implement it


ZCL_SOLDIER
-GO_REFILL : ZIF_REFILL_WEAPON
-GO_REPAIR: ZIF_REPAIR_WEAPON
+attack( )
+dynamic_repair_selection
+dynamic_refill_selection
Context/Client Class
Refill Strategy
Repair Strategy
<<interface>>
ZIF_REFILL_WEAPON
+Refill()
<<interface>>
ZIF_REFPAIR_WEAPON
+Repair()
ZCL_Count_Based_Refill
+refill( )
ZCL_Time_Based_Refill
Refill( )
ZCL_Internal_Repair
Repair()
ZCL_Exteranl_Repair
Repair()

What is SOLID ?

**S**ingle-responsibility principle
**O**pen-closed principle
**L**iskov substitution principle
**I**nterface segregation principle
**D**ependency inversion principle

subclass

Super class

**SOLID**

This was given by author - **Robert C. Martin ( Uncle BOB )** in Year 2000.

**Definition:**
This is very popular set of **design principles** which guides while developing a software in OOPS, what are the important guideline need to be followed for better **maintainability** and **reusability and flexibility.**
It is mnemonic acronym for below set of principles -

INTERFACE
METHOD A,

A

**S** -> Single Responsibility Principle (SRP )
A class should **have one and only one** reason to change

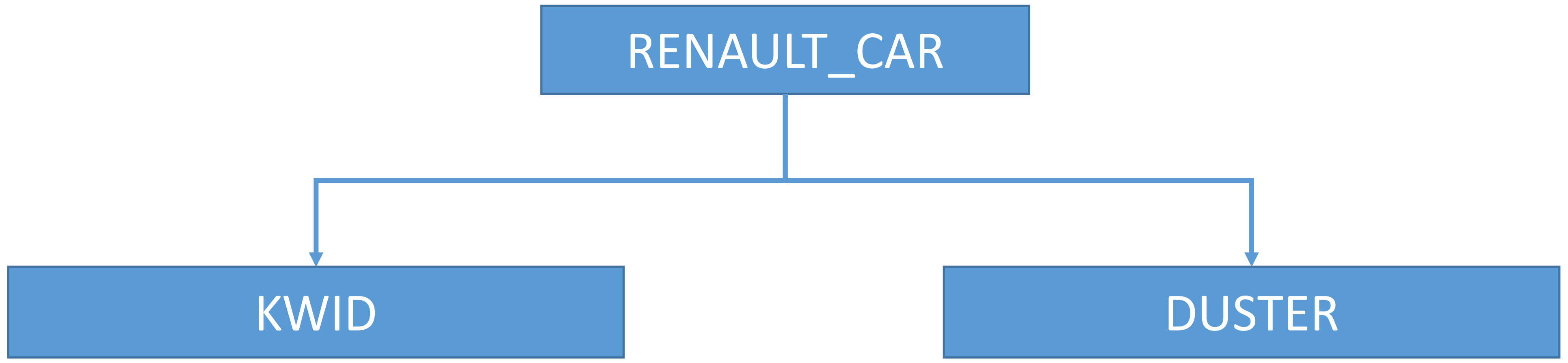
**O** -> Open-Closed Principle
Open for Extension and Closed for Modification

**L** -> **LISKOV** Substitution Principle
Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program

**I** -> Interface segregation Principle
In stead of generic Interfaces, we should have the specific Interface

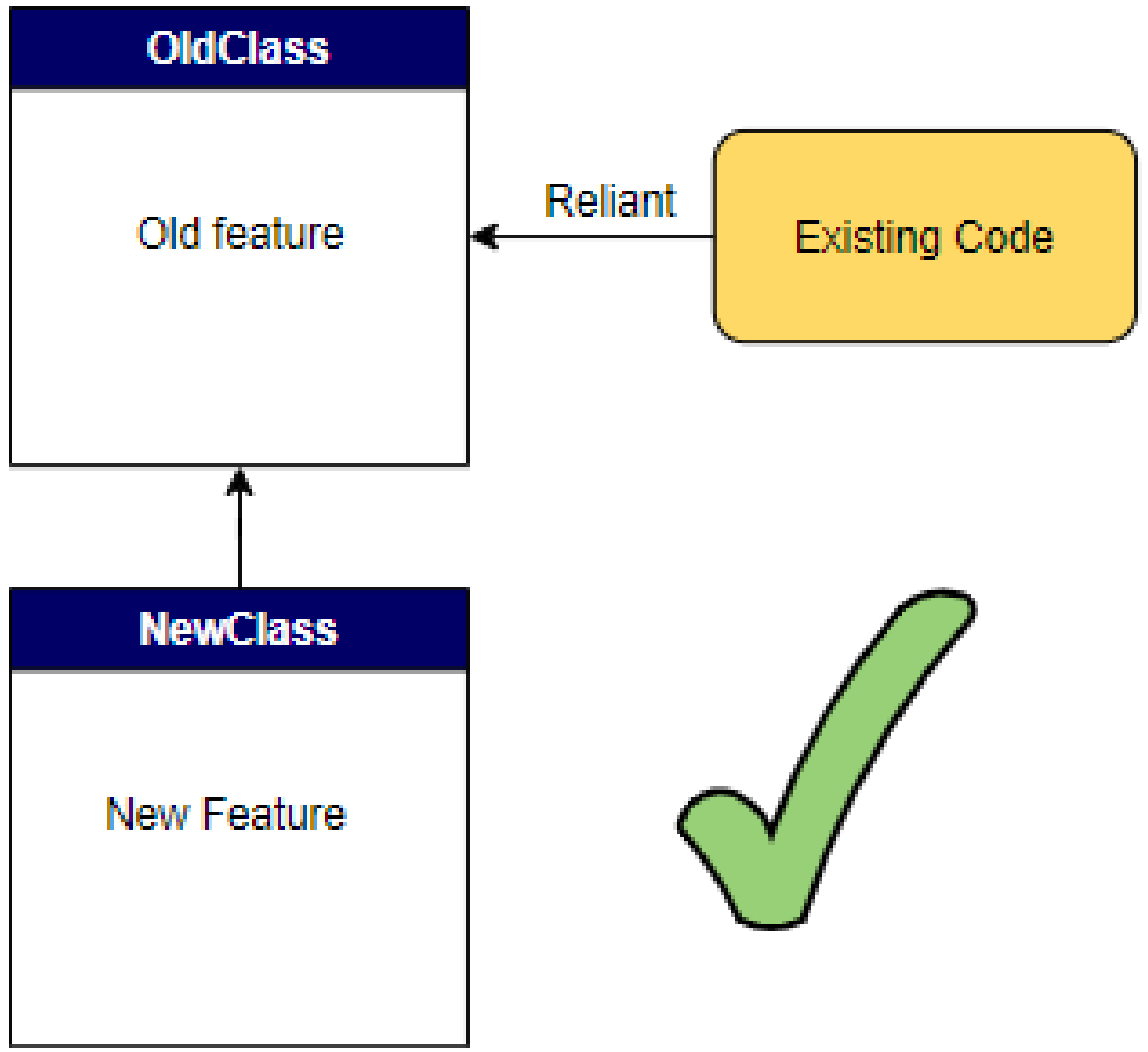
**D** -> Dependency Inversion Principle => Depend on abstraction, not on Concretion

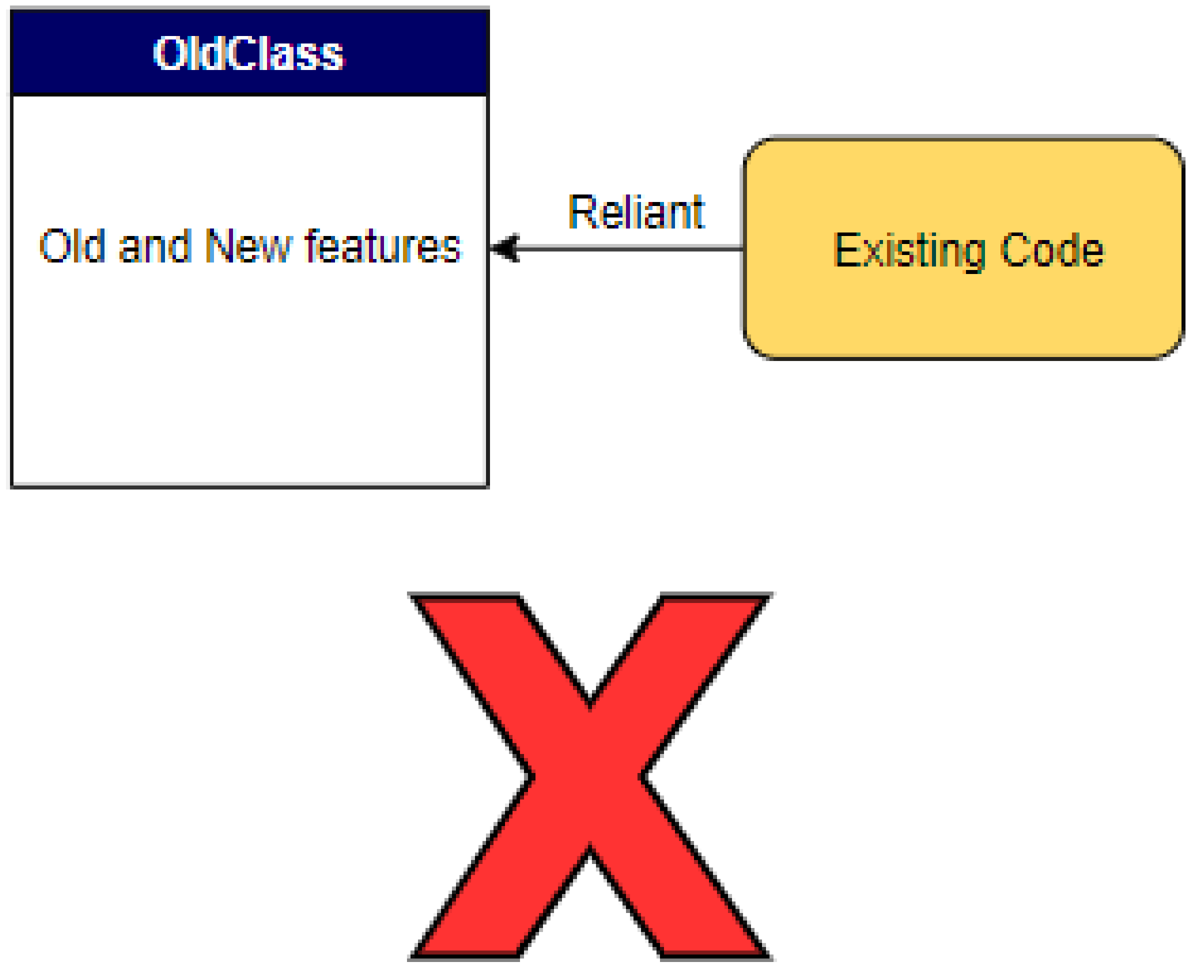
S->**Single Responsibility Principle**



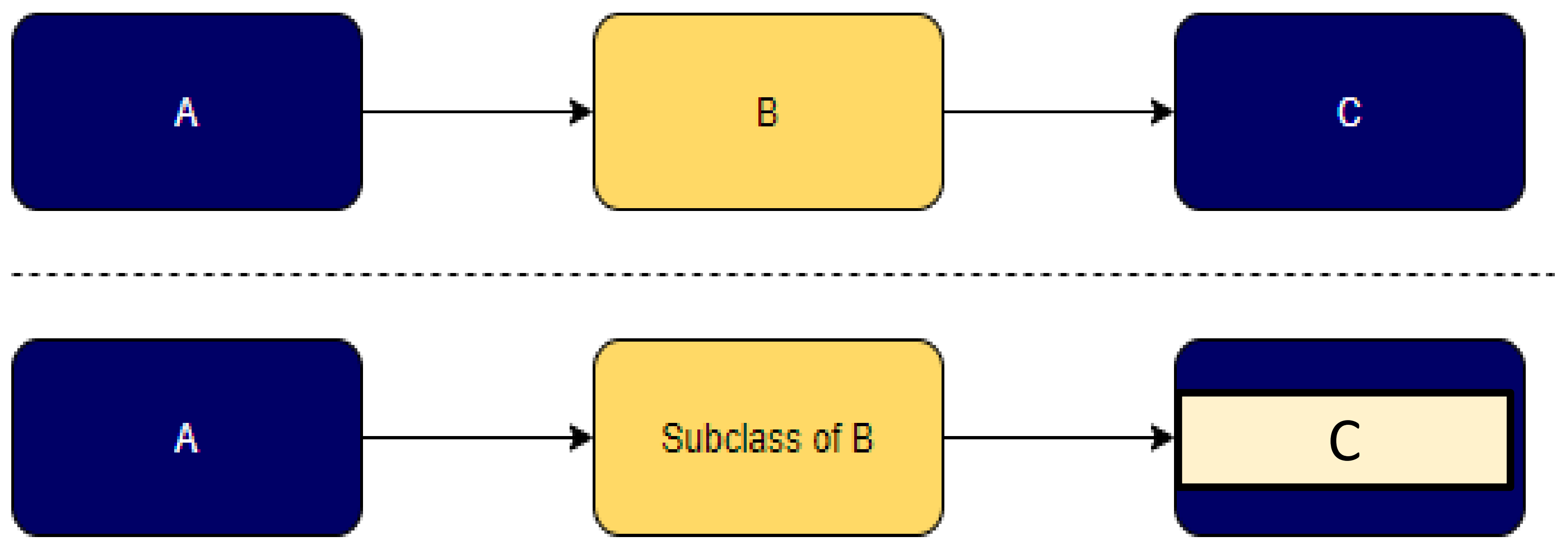
Open Closed Principle

Open for extension but closed for modification



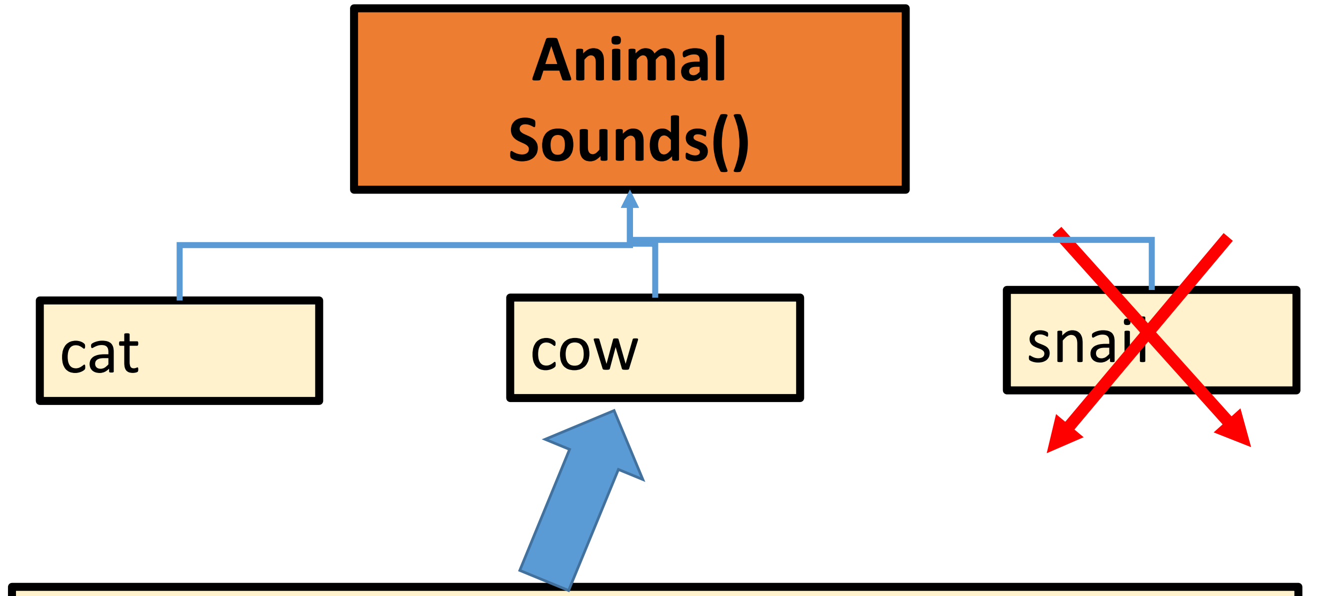


## Liskov substitution principle



B

LSP: seamless substitution for the B class in existing code

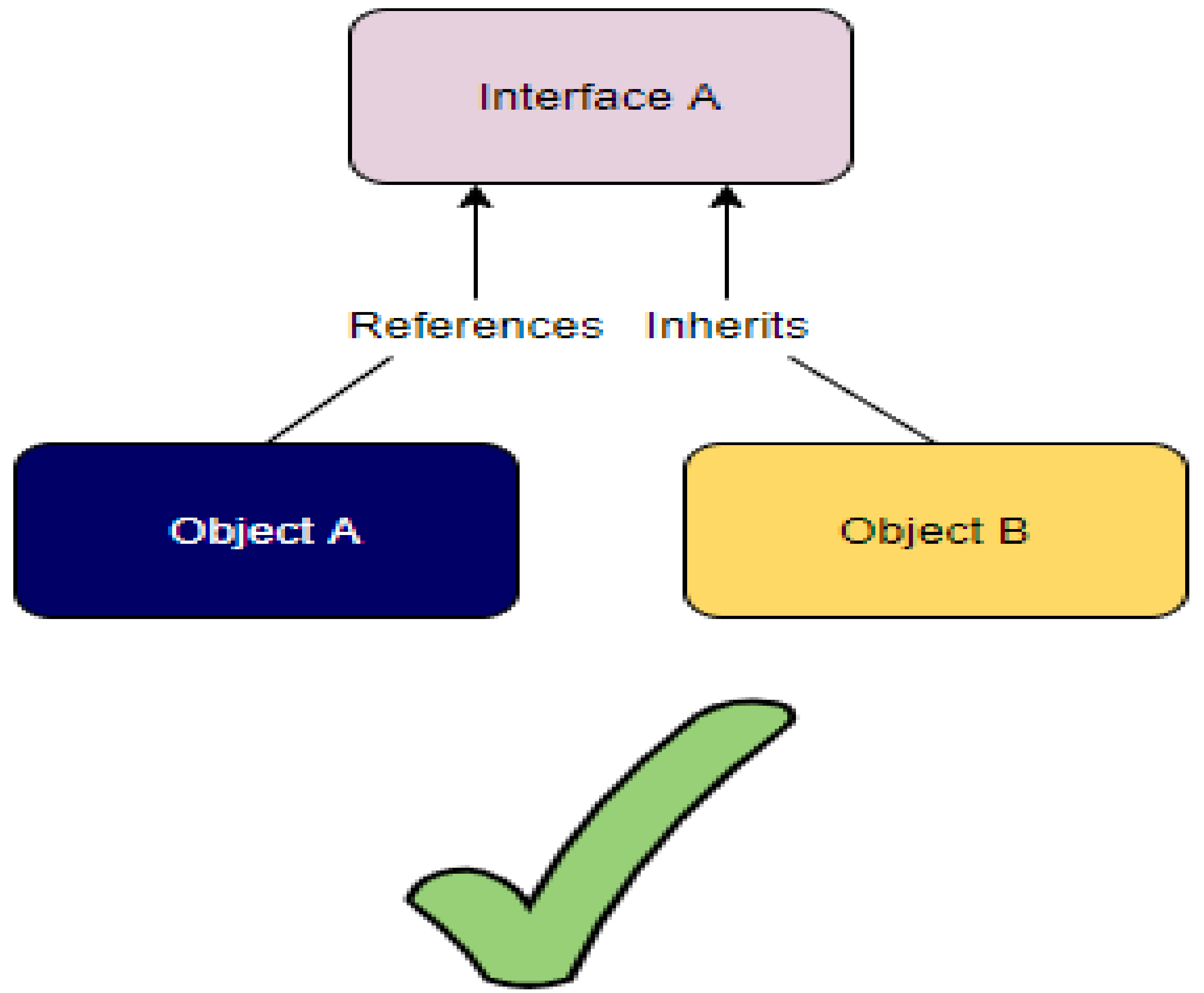


Hence in the above example as per LISKOV'S principle rules are violating

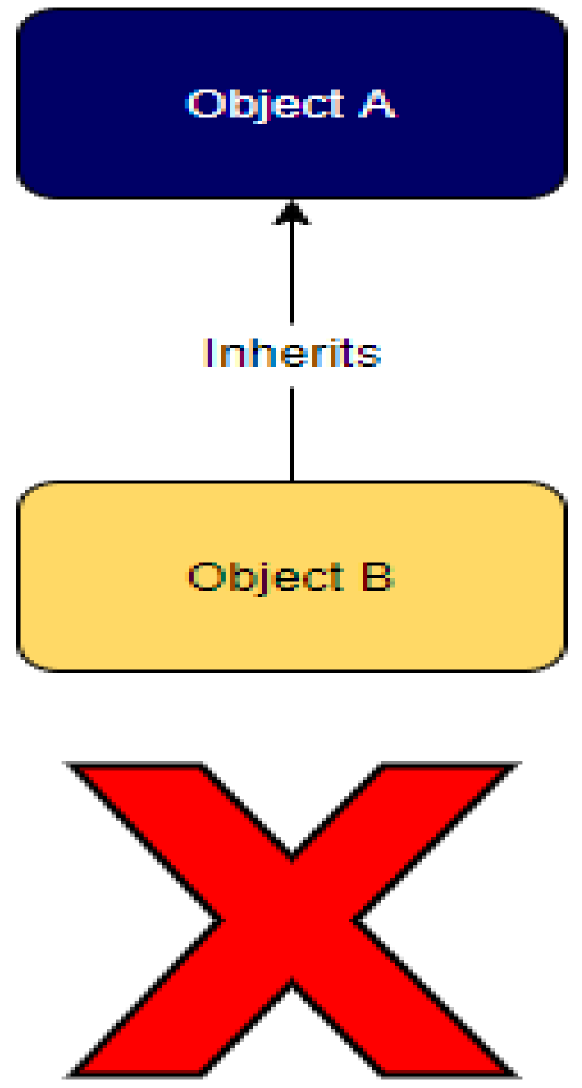
The principle says that any class must be directly replaceable by any of its subclasses without any error.
Link: [https://www.youtube.com/tch?v=ObHQHszbIcE](https://www.youtube.com/tch?v=ObHQHszbIcE)

The **Liskov substitution principle** (LSP) is a specific definition of a subtyping relation created by ***Barbara Liskov*** and ***Jeannette Wing***.

## D: Dependency inversion principle



*One should depend upon abstractions, [not] concretions."*



Two points ->

- **High-level** modules should not depend on **low-level** modules. Instead, both should depend on abstractions (interfaces)
- Abstractions should not depend on details. Details (like **concrete implementations**) should depend on abstractions.

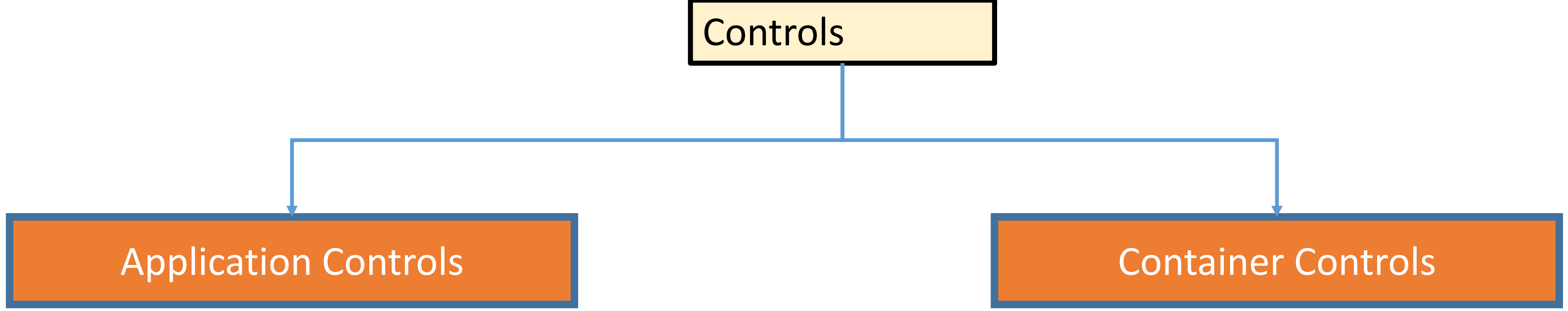
## Control Framework

**Control in general:**

- ✓ Controls are independent software component embedded in SAP GUI.
- ✓ Controls can be used to achieve extra flexible functionality.
- ✓ SAP supports **ACTIVEX Controls** for MS Windows platforms and JavaBeans for the SAP GUI in the Java environment.
- ✓ Control helps to improve the performance by –
  having the data on the presentation server instead of application server( e.g. during scrolling and editing the texts etc. )
- ✓ Frequent data exchange on from Backend to Frontend and vice versa increase the load on the network and gives the bad experience for user due to slowdown of the application
- ✓ Hence if most of the work is going to done at frontend only then control becomes useful.

**Controls framework:**

- ✓ It was introduced in Release 4.6
- ✓ CFW encapsulates all the controls in global classes of ABAP objects.



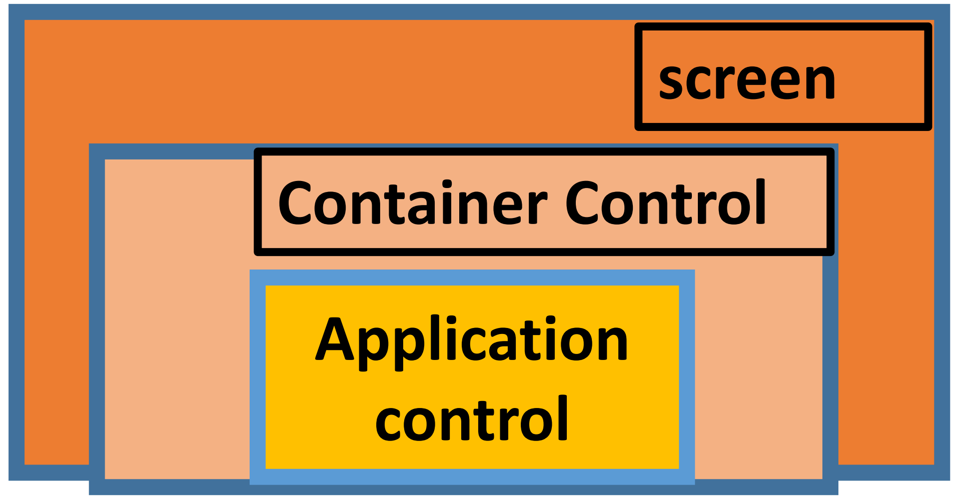
**Application Controls**

This component is used inside the container controls to achieve a specific functionality like displaying the html page, Text etc.

**Container Controls**

This component is used to hold the application controls and provides the standard layout on the screen

Each container controls must be associated with the screen.



## Container Controls

### CL_GUI_CUSTOM_CONTAINER

How to use:-
1. Create custom container in screen painter (say **CONTAINER** )
2. Pass the custom container name(**CONTAINER**) to create the object of the class CL_GUI_CUSTOM_CONTAINER. Say object name is **O_CUST_CONTAINER**
3. Call the application control object method by passing the custom control object(**O_CUST_CONTAINER)** as parent

Note: do all this in PBO of the SCREEN

### CL_GUI_DOCKING_CONTAINER

How to Use:-
1. While creating the object of this class must need to be linked with any one of the edge of the screen.
2. That edge is now become Docker and can be used for the purposed of any application control

Note: No Screen painter is required.

### CL_GUI_SPLITTER_CONTAINER

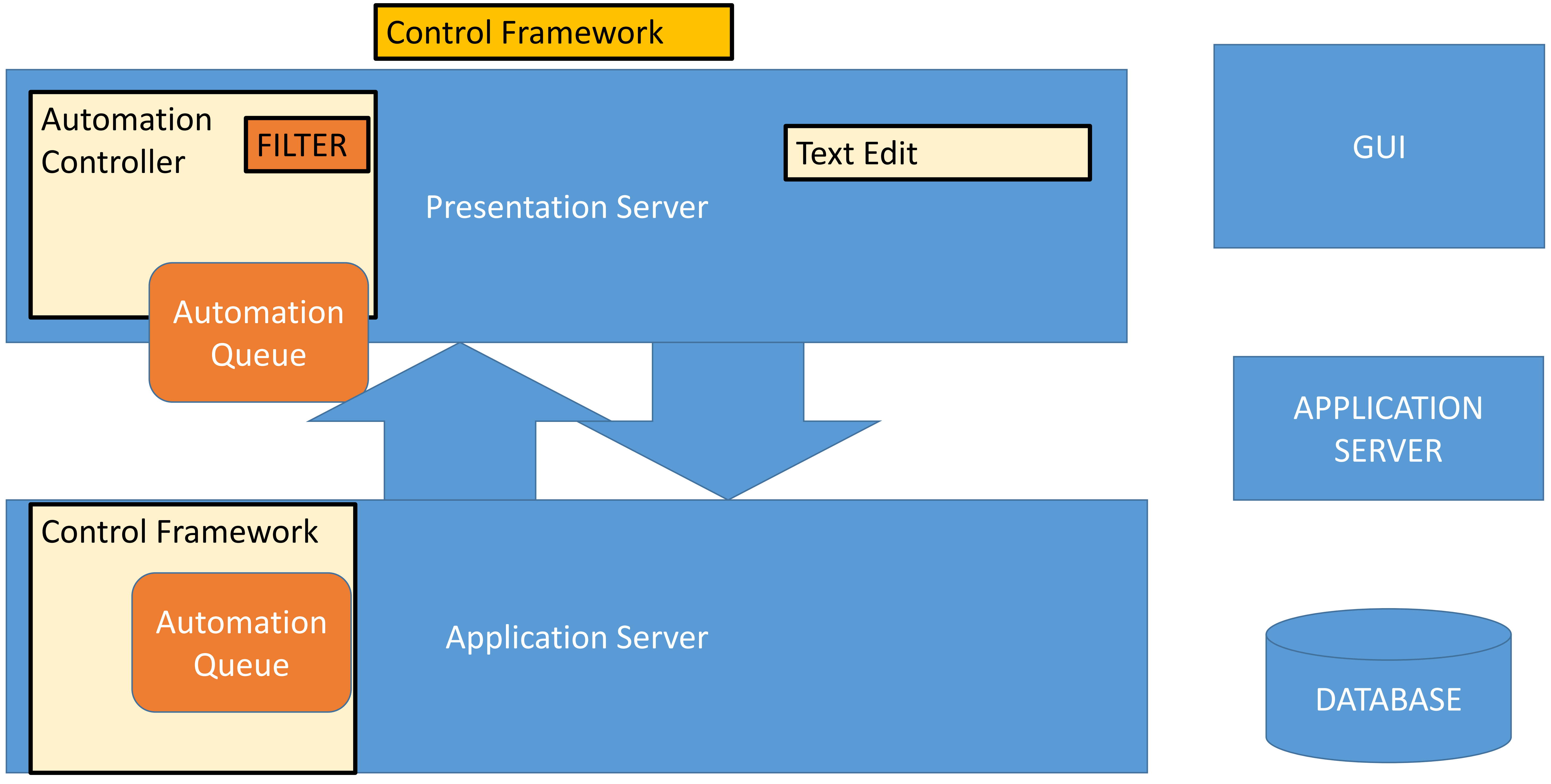
Splitting the screen area into vertical and horizontal section
That means splitting the screen into multiple parts

**CL_GUI_CFW=>FLUSH**

When you want to pass the message or instruction **before the PBO** call of the screen, we can use the CL_GUI_CFW=>FLUSH.

**CL_GUI_CFW=>DISPATCH**

When you want to trigger your Application EVENT HANDLER METHOD **before PBO** call then you can use this CL_GUI_CFW=>DISPATCH.



What is the use of CL_GUI_CFW=>Dispatch?

**CL_GUI_CFW=>DISPATCH**

When you want to trigger your Application EVENT HANDLER METHOD **before PBO** call then you can use this CL_GUI_CFW=>DISPATCH.

VVI

What is the use of CL_GUI_CFW=>FLUSH

VVI

**CL_GUI_CFW=>FLUSH**

When you want to pass the message or instruction **before the PBO** call of the screen, we can use the CL_GUI_CFW=>FLUSH.

Q . What are the steps need to followed to make a ALV editable?

VVI

**Ans.** -> There are two scenario –

Case 1. When you have to make complete ALV editable -

For making the full ALV editable we have to pass **LAYOUT-EDIT** = **‘X’.**

**There is no setting required at field catalogue level.**

Case 2: When you have to make a specific column editable.

But when you need to make a specific **field/column** editable then

We need to specify that column at field catalogue level and set EDIT property as ‘X’ like

**Fieldcatalogue-EDIT = ‘X’.**

## Q . What are the steps need to be followed to color an ALV?

VI

Steps need to be taken based on scenario –

**Case 1. setting color for specific column.**

When a specific column need to be colored then setting need to be done at FIELDCATALOGUE level.

FIELDCATALOGUE-**EMPHASIZE** = Cxyz where **x** = Color Code,
**y** = Intensified On(1) or Off(0)
and **z** = Inverse on(1) or off(0).

**Case 2. When a row need to be colored in ALV**

Below steps need to be followed –

**Step 1.** Include a field(SAY ROW_COLOR) of type CHAR4 at last in structure which will hold the color attribute.

**Step 2**. Make system aware that this field is to store the attribute for color by specifying setting attribute INFO_FIELDNAME at layout level.

GD_LAYOUT-INFO_FIELDNAME = ‘ROW_COLOR'.

**Case 3**. **Check the next Slide**

Case 3. **Color at cell Level**

step 1. Include a field **CELLCOLOR** of type **LVC_T_SCOL** at structure in last.

Step 2. Inform to the program by setting the **COLTAB_FIELDNAME** attribute of LAYOUT that the extra field **CELLCOLOR** is for the purpose of setting the color at cell level.

Do as below -

layout-COLTAB_FIELDNAME = **'CELLCOLOR'.**

**Step 3.** Populate the final internal table where corresponding CELLCOLOR field need to populated accordingly.

1. What is the Priority of the color for below Row Color, Column Color and Cell Color

Ans. -> Below is the priority sequence -

1. Cell Color
2. RowColor
3. Column Color

| MANDT | MATNR | MTART | MATKL | MBRSH |
|---|---|---|---|---|
| | 🟧 | | | |
| 🟩 | 🟥 | 🟩 | 🟩 | 🟩 |
| | 🟧 | | | |
| | 🟧 | | | |

Q. Suppose you have to display the check box column for selecting the row from output ALV

Step1. Define one extra column(CHECKBOX_FIELD) of CHAR1 which will be used for check box

Step2: - At fieldcatalogue level you have one attribute called 'CHECKBOX'

```
FIELDCATALOGUE-CHECKBOX = 'X'.
FIELDCATALOGUE-FIELDNAME =  'CHECKBOX_FIELD'
```