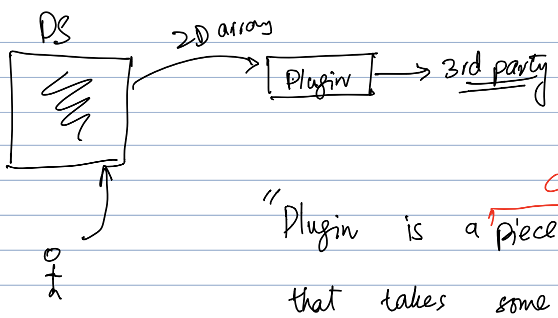


class

"Plugin is a piece of code that takes some input image and modifies it"!

operation → "in-place"

Photoshop

```
class    Plugin {
    public:
virtual void   apply_filter ( int    a[5][5] );


};
```

Third party

```
class    Make It Black   :  public    Plugin {
    public:
    void    apply_filter ( int    a[5][5] ) ;

};
```

```
main ( ) {
         int      a[5][5]    ;
            // init
         Plugin    p;                        |  MakeItBlack  mib ;
         p.apply_filter  ( a ) ;             |  mib.apply_filter( a );
         // output    a                      |   // output   a.
                                             |

}
```

Adobe

① void Plugin:: apply_filter (int a[5][5]) {
cout << "Doing nothing" ; ←

Third party.

② void MakeItBlack :: apply_filter (int a[5][5]) {
// loop ← all values 0 ;

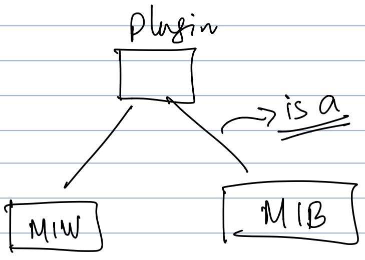
}

Plugin *p; → reference variable → object

p → "Plugin"

~~Plugin~~ p1;

p = &p1;



MakeItBlack mib;

p = &mib; ?

"A reference variable can store address of its own class and address of instance of any of its children"!

→ p = get_filter(); ~~;~~

→ p -> apply_filter (a) ;

↳ after virtual

a → black image

POLYMORPHISM

call Parent's fn.

execution is of some child depending on what the ref. var is pointing to ! ////

→ function virtual
→ Ref. var → of Parent
→ object → of child —
→ function overridden.

⓪ → call → def. of child

```
Plugin * get_filter ( ) {
        if ( day is monday ( ) ) {
            Make It Black   mib ;

            return   &mib ;
        }
    {  else
        {   Make It white   miw ;

            return   &miw ;
        }
}
```

Virtual fn

pure virtual fn $\longrightarrow$ <u>abstract</u>

if one method in a class is abstract

$\longrightarrow$ class is <u>abstract</u>

$\rightarrow$ Plugin is an <u>abstract class</u>

$\rightarrow$ You cannot instantiate an abstract class!