Monday
1-06-2020

DP Assignment-3

UNIT-3

N.V.S.K.kalyani
174N1A0584
3rd B.Tech CSE-C

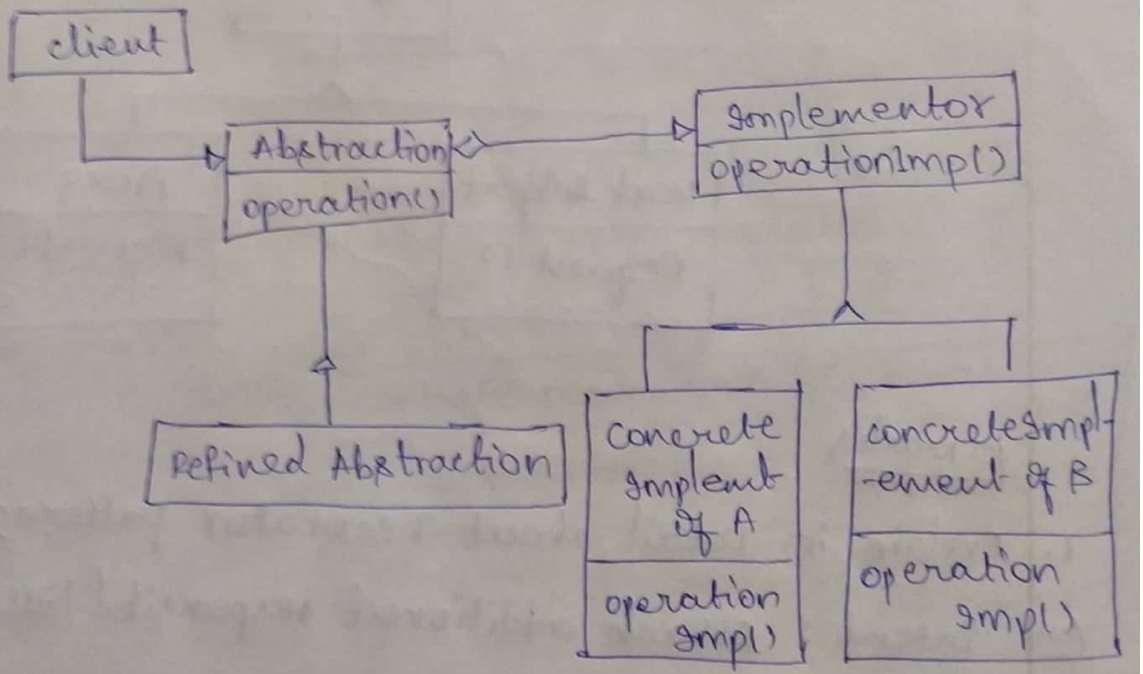
2 Marks

1. Define the Implementation Adapter pattern?

Ans 1. Implementing class adapters
2. pluggable adapter
   - => using abstract operations
   - => using delegate objects
   - => parameterized adapters

2. Sketch the structure of Bridge pattern?

Ans



3. what is the Intent and Applicability of composite pattern?

Ans Intent: compose objects into tree structures to represent part-whole hierarchies.

Applicability:

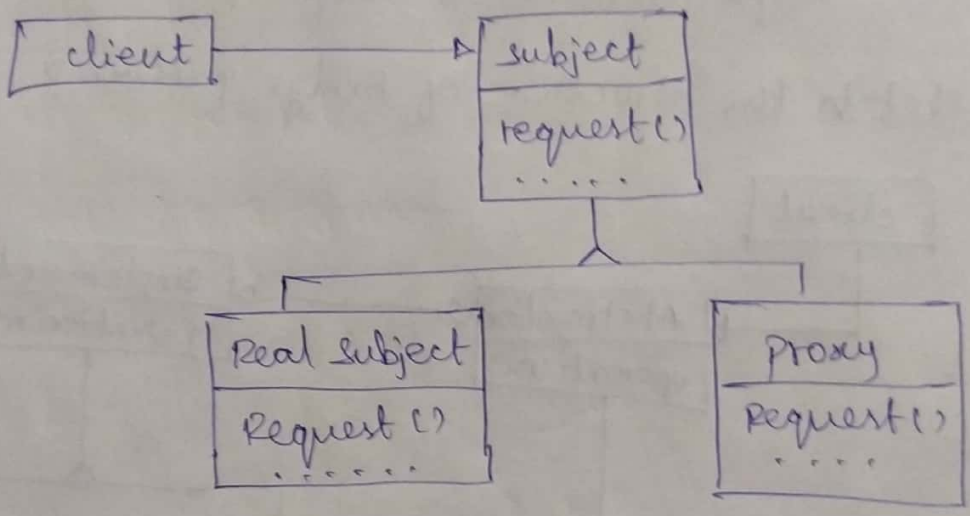
* you want to represent part-whole hierarchies of objects.

4. Explain the consequences of Facade pattern?

Ans 1. It shields clients from subsystem components.
2. It promotes weak coupling between the sub-system and its clients.
3. It doesn't prevent applications from using subsystem.

5. Sketch the structure of proxy pattern?

Ans



10 Marks

1. Explain in Detail about Decorator pattern?

Ans Intent: Attach additional responsibilities to an dynamically.

Also known as: Wrapper

Motivation:

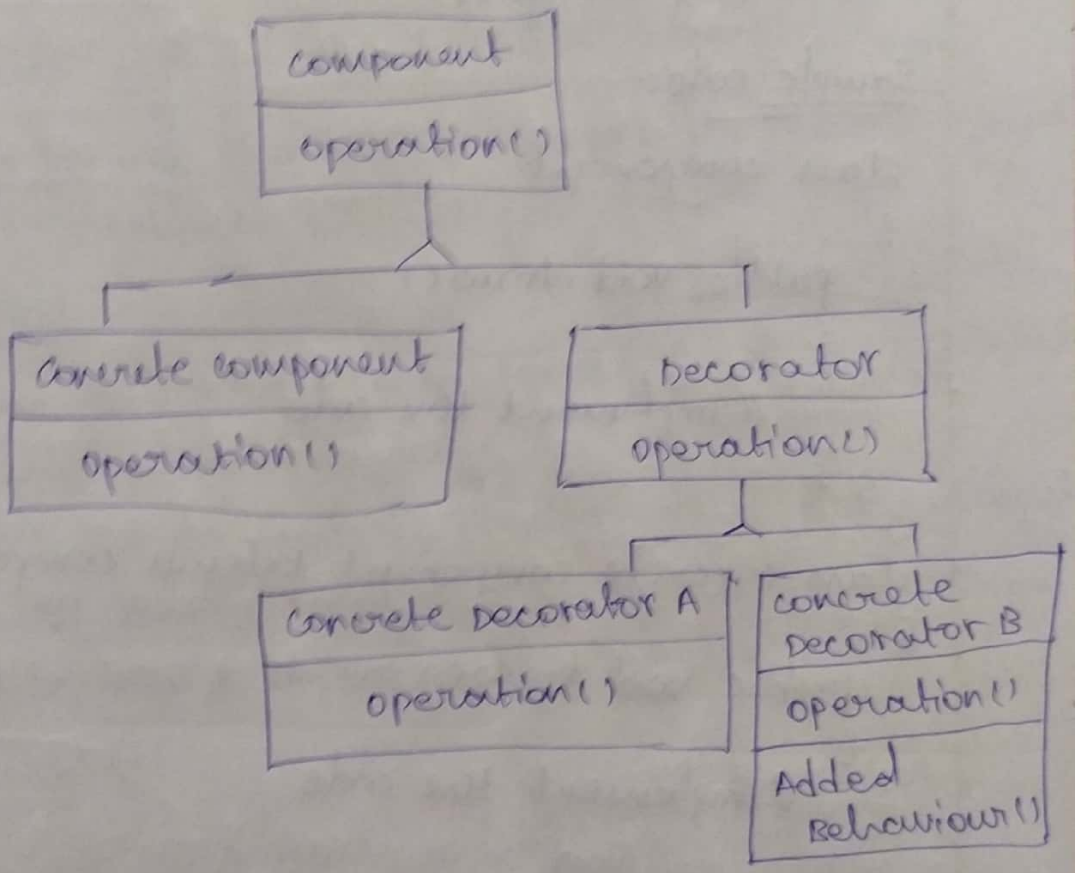
* Sometimes we want to add responsibilities to individual objects not to an entire class A.

Graphical use toolkit.

Ex: should let you add properties of borders or behaviour like scrolling to any user Interface.

17UNIA05824

* Visual component is the abstract class for visual it defines their drawing and event handling interface Applicability.
* To add responsibilities to individual objects dynamically and Transparently.
* The responsibilities that can be withdrawn.
* when extension by subclassing is impractical



Participants:-

component (visual component)
concrete component (text view)
Decorator
concrete decorator (Border decorator, scroll decorator)

Collaboration

Decorator forwards requests to its component

object. It may optionally perform additional operations before and after forwarding the request.

consequences :-

1. Interface conformance
2. keeping component classes light weight.
3. changing the skins of an object verses changing its guts.

Sample code :-

```
class component
{
  public void draw()
  {
    // implement the code
  }
}
class concrete component Extends component
{
  public void draw()
  {
    // implement the code
  }
}
class decorator extends component
{
  public void drawborder()
  public void scroll bar()
}
class concrete decorator Extends decorator
{
  // implement the code
}
```

known uses:-

1. Any text view document
2. Interfacing

Related patterns:

1. composite
2. Adapter

2. Discuss in detail about Flyweight pattern?

Ans Intent:- use sharing to support large no. of gained objects efficiently.

Motivation:-

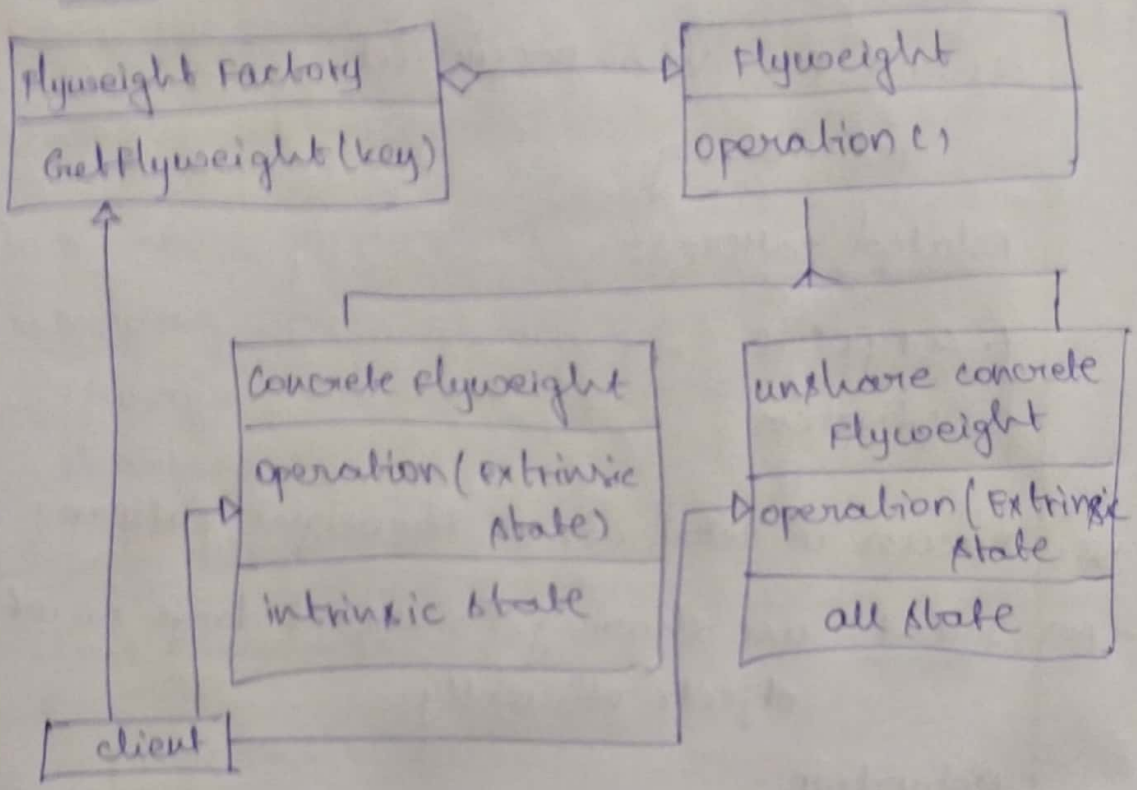
* Some applications could benefit from using object through out their design.
* A Flyweight contents simultaneously shared object that can be maptiple. The flyweight is an independent object in each context.

Applicability:-

* An Application uses a large no. of objects
* Most object state can be made extrinsic.
* Many groups of objects may be replaced by relatively few shared objects.

participants: flyweight
concrete flyweight (character)
unshared concrete flyweight (row, column)
flyweight factor
client

Structure:



Collaboration:

clients should not instantiate concrete fly weight directly. clients must obtain concrete Flyweight objects exclusively from the Flyweight factor object to ensure they are shared proper

Consequences:

1. The reduction in the total no. of instances that comes from sharing.
2. The amount of intrinsic state per object

Implementation:-

1. Removing Extensic state
2. Managing shared objects

known uses

1. Resource sharing
2. Network

Related patterns:-

1. Composite 2. State 3. Strategy.