17-6-25. DESIGN PATTERNS:--7 A design pattern in programming is a reusable Solution to a common problem that occur during Software development of design -> These Patterns Provide structured approach to some design & development issues. I used to execute move maintainable, flexible & Scalable code Design Pattexns expoted of Provide Slobal Point of acress to it (Seational Patterns Structural Pattern Behavioral Pattern - singerton - Observe - Composite - Stratergy - Factoxy - Builder /- Decorator [1] (xeational dopattern: pointailarda xead til These pattern four on object execute mechanism. -> Provide ways to execute object in a mainner that is flexible and maintanable. -> Based on application specific requirements we choose a object exection mechanism. [2] Structural Pattern: -7 These Patterns focus on class & objects to execte large structures while keeping them flexible and officient. -> Helps us to ensure that classes and objects can work together efficiently to acheive a goal

[3] Behavisol Pattesn:-- Jused for intraction of communication between object and classed. -7 Provide solution for officiently manage flow of Control, behaviour between object. design a soverorment issued \* (Seational Pattern :tis Singerton Pattern: --7 Ensures that only one instance of a class is executed 4 provide global Point of access to it. -7 used when we want to have single shared instance of class throught the application. ways to create singleton class :- blind [1] Fages intialisation [4] Double chack [i] Laty intialisation [5] Bill Pugh solution. [3] Synchronized block. Four Mair class: - yunger sittones not milyan no bears Public class main mains do moistres soldo o Public static void main (String angs []) antida p sono DB Connection obj = DB Connection get Instancecy; es the estate that chiefe on and chiefe con week together efficiency to acherve a so

```
Eages Intialisation !-
Public class DB Connection
 Private Static DB Connection canobic ct = new DB Connections
   Private DB Connection () { }
   Private Static DB Connection get Instances)
    xeturn canobiecti
 Laty intialization:
 Public class DB Connection
 Private Static DB Connection conObject,
   private DB Connection C) [].
     Public static DB Connection get Instance()
              if (conobiat = = null)
                       conobsect = new DB Connections.
             return conobject
   Synchronization Block:-
  Public class DB Comeation
     Private static DB Connection condicate
       Private DB Connection CD & ].
       Synch sonized Public Static DB connection get Instances
              IP (conobject == nui)
                      Con Object = new DB Connection ()
             se toxn conobject
```

```
Double check locking system:-
 Public class DB Connection
    . Private static volatile DB Connection con=new DB connection
      Private DB Connection C) {}
    Public static DB Connection get Instance C)
            Synchronize d (DB Connection . class)
                  if (con = = nui)
                  Con = new DB connection cz.
            Seturn con.
  Big Pugh singleton: (uses cagex initialisation)
   Public class DBC
  Private DBC CD { }
   Private static class DBHelpes
        PSÍVATE Static final DB = INSTANCE_OBJECT= New DBC
   Public static DB get Instances
     Yeturn DBHERES. INSTANCE_OBJECT
   3
   ENUM:-
   Public enum DBC
    INSTANCE
      Public static do Some thing col
```

Buildes Pattern: -- Buildes Pattern is a excational design Pattern that allows you to Construct Complex objects step - by-step Serexating the Const xuction logic from the representation -7 Useful when objects have many optional fields -> Avoid telescoping Constructors. Without Builder Pattern with builder Pattern Person. Java. Pexson. Java: -Public class Person Public class Person E private string name; private String name; Psivate int ago; Private int age; Private Person (Builder builder) this name = builder name Public Pesson (String name, int age) this age = builder age this name = name; o wold Public Void displayes [ this. age = age; 5.05 (name+ . . .) Public void display c)
Public Static class Buildre S.O.P ("Name"+ name +. ), so work | Private int asc. Private String name the most Public Buildon set Name (String ) this name = name; Ost. Java: return this. Public class Test Public Buider set Age (int age) 5 2 this. ase = age: Public static void main (String angs []) setusn this Person P= new Person ("Vikat", 35). Public Person buildent P. display co. return new Person (this) IT We were now Chiclosof 3 Inside main oped. countil move (and ("count)) Person Penew Person. Buildonc) · setName ("Kohli") -· Set ADe(

```
Factory Pattern:
  - A coartional design pattern that provides an interface
  to execute objects in a supexclass but allows
  Subclass to altex the type of objects that
   will be executed.
  - Avoid using new directly in the client
   -> The will use factory Pattern when we need to execut
  object based on input ox condition
  Code :-
   Public intextace shape 2
    void dxawcz;
Private Person (Builder English)
Public class circle implements share?
this age = boilder. age.
          Public void draw co of sman sman
VOIA displayed !!
  System.out. Println ("Drawing a circle).
Public sclass square implements shape of
           Public void draw () ( ) + smort small ) 903
(mone) and system out. Println ("Drawing a senare").
   theis name - James
Public class shape Factory
     Public Shape get Shape (String type)
   E if (type. equal s. Ignove (ase ("cixcle")) {
               * Ketush new Circle();
          3 else if (type. equals Ignore Case (square)) {
            Seturn new Squarecy.
            Yeturn null;
```

```
Main Java
 Rublic class Main
      Public static void main (String [] args)
       Shape Factory factory = new Shape Factory C2.
           Shape si = factory. gershape ("circle").
    ( Verning 1. 31. d & aw Co, ) your A resulted
           shape se = factory getshape ('square').
            52.d8awc7
Structral Pattern:
 Adaptex:
-7 Adapter Pattern is a structural Pattern that allows
   object with in compatible intexfaces to work together
   by convexting one intexface into another.
-> Bridge the gap between two in compatible interferes.
          Client -> Target' (expected interface)
          boo assolo LAdaptes among
                     A dapter (incompatible class)
Code:
class old Printer 2
        Public void Pxin+Old C) {
          System . out . Paintln ( old painter);
        3
 3
```

```
intexface Printer 2
       Void Print (7)
  class PrinterAdapter implements Printer &
        Psivate OldPrinter oldPrinter
        Public Printer Adapter (Old Printer) {
       this old Pxintex = old Pxintex
       Public Void Print co {
             old Printer. Printold co;
   Public class Test {
       Public Static void main (String [] 20195) {
old princes old = new old Printercy:
       Printer adapter = new printer Adapter (013)
 adaptes. Printc).
 [2] Composite Pattern:-
  — used to treate individual objects and groups of
     objects in a uniform way.
        Component (intexface)
          leaf Composite & sounds & blo
                  Sall place play blow silder
      Cointains List < Component>
```

```
public interface Employee 2
        void show Details CJ.
3
Public class Developer implements Employee {
        Private String name; Commercial modern was constituted
        Public Developer (String name) {
       this. name = name;
        Public void showPetails a) &
             System. out . println ("Developer:" + name);
 Public class Manages implements Employee 2
       Private String name;
       Private list < Employee7 team = new Assay List < 7 (2)
       Public Manages (String name) {
               this. name = name;
       3
       Public void add (Employee emp) of
               team. add (emp);
       3
       Public void show Petails () &
             System . Out . Println ("Manager: ", +name);
             for (Employee e: team) {
                c. Show Details Co.
             class Test &
    Public
         Public Static void main (Sering CJ ax95) {
              Developes devi = new Developer ("Rahui"):
              Developer dev2 = new Developer (" Kixan");
```

Manager ("visat"). Show Detailsco, = new (devi) manager add add manages. Manager manager Behar

Paray Listery dePendents observer & message list of class channel implements subjectly automatically when 6 List Lobsenven> observend = new Public intestace observer E noti Fyobsexver (String msg); addobserver (observer add Observer (Observen 3); Follower (String name) & void UPdate (Sexing message) xemove Observer (6 bscnver System. out , Printly (name) ( String Followex implements maintains a this name = name, nome. observs. add (0); Subject of update them String - where an object java . U#1. \* , Void void intexface Pattern Pattern and notified class Psivate Public Public Voi d Void Void Private changes. Public Behavi 801 2 Public Public Public observe imPoxt

"Vivae) 2(58x0) 2 channel (2). channel . Upload (" +Highiights). String [7] Described (Sexual Follower channel . add Observer (FD) stop day maj n chanel =new = new Void Box42 Follower lest Static channel class Public Public ner