**Day 12:**

**Design pattern :** Design pattern is best practise or solution for repeating problem.

Core Design pattern : GOF

23 design pattern.

These design pattern mainly divided into 3 types.

1. **Creational design pattern**: these design pattern deal with object creation.
2. Singleton design pattern:
3. Builder design pattern
4. **Factory pattern**
5. **Abstract factory pattern**
6. Prototype design pattern
7. **Structural design pattern:** structure of class and interfaces.
8. Adapter design pattern
9. Composite
10. Flyweight design pattern
11. Proxy design pattern
12. Decorator design pattern
13. Bridge design pattern
14. Façade design pattern

1. **Behavioural design pattern:** these design patter deal with object interaction.
2. Chain of responsibility:
3. Command design pattern
4. Visitor design pattern
5. Template design pattern
6. Interpreter design pattern
7. Mediator design pattern
8. Memento design pattern
9. Iterator design pattern
10. Observable design pattern
11. State design pattern
12. Strategy design pattern

**Singleton design pattern:**

According to this design pattern we need to create only one object of that particular class.

**Factory design pattern :** Defines an interface for creating an object but let subclasses alter the type of object that will be created.

Payment -🡪 interface

UPIPayment 🡪 classes

CreditCartdPayment 🡪 classes

DebitCardPaymment 🡪 classes

PaymentFactory -🡪 this class contains factory method which is responsible to create the object of respective class and call the payment method part Payment interface.

**Abstract factory:** Provide an interface for creating families of related or dependent object without specifying their concreate class.

Payment -🡪 interface

UPIPayment 🡪 classes

CreditCartdPayment 🡪 classes

DebitCardPaymment 🡪 classes

PaymentFactory -🡪 this class contains factory method which is responsible to create the object of respective class and call the payment method part Payment interface.

Loan 🡪 interface

HomeLoan 🡪

PersonalLoan -🡪

LoanFactory 🡪

BankFactory : which is responsible to provide Factory reference like LoandFactory or PaymentFactory

**Interface BankFactory {**

**PaymentFactory createPayment(String type);**

**LoanFactory processLoan(String type);**

**}**

**Builder pattern:** Separate the construction of a complex object from its representation.

User or Person or Customer

-🡪 name, age, emailid, phonenumber, city and state etc

Empty constructor,

Parameterized constructor with different parameter

1. Makes object construction clear and readable
2. Avoid many constructor overloading concept.
3. Easy to make which field mandatory and optional
4. This is immutable object.

Prototype : creating clone of the user defined or pre defined object is known prototype design pattern.

**Adapter design pattern:** Convert the interface of a class into another interface base upon client expect.

**Decorator design pattern:** Attach the additional responsibility to an object dynamically.