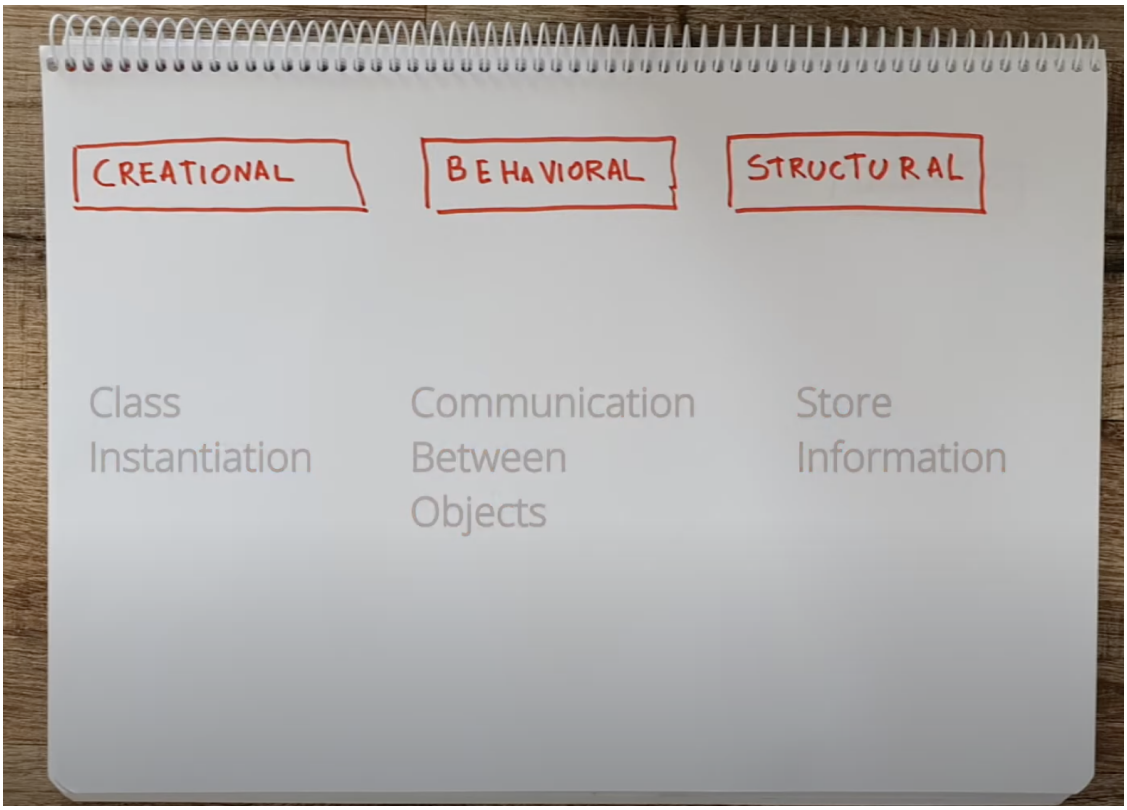
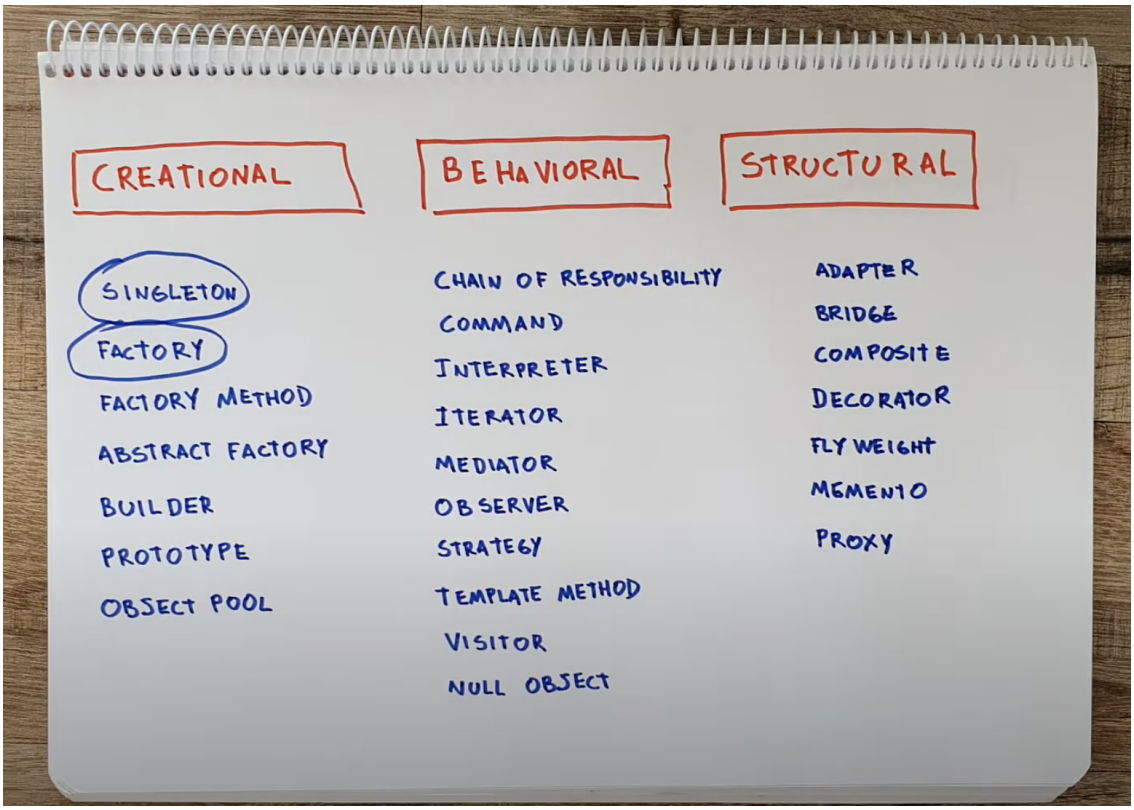
Design Patterns

Types



Design Patterns



Patterns

Builder Pattern

Pattern that facilitates creation of complex objects.

```
public class BankAccount {

    public static class Builder {

        private long accountNumber; //This is important, so we'll pass it to the constructor.
        private String owner;
        private String branch;
        private double balance;
        private double interestRate;

        public Builder(long accountNumber) {
            this.accountNumber = accountNumber;
        }

        public Builder withOwner(String owner){
            this.owner = owner;

            return this;  //By returning the builder each time, we can create a fluent interface.
        }

        public Builder atBranch(String branch){
            this.branch = branch;

            return this;
        }

        public Builder openingBalance(double balance){
            this.balance = balance;

            return this;
        }

        public Builder atRate(double interestRate){
            this.interestRate = interestRate;

            return this;
        }

        public BankAccount build(){
            //Here we create the actual bank account object, which is always in a fully initialized state when
it's returned.
            BankAccount account = new BankAccount();  //Since the builder is in the BankAccount class, we can
invoke its private constructor.
            account.accountNumber = this.accountNumber;
            account.owner = this.owner;
            account.branch = this.branch;
            account.balance = this.balance;
            account.interestRate = this.interestRate;

            return account;
        }
    }

    //Fields omitted for brevity.
    private BankAccount() {
        //Constructor is now private.
    }

    //Getters and setters omitted for brevity.

}
```

```
BankAccount account = new BankAccount.Builder(1234L)
            .withOwner("Marge")
            .atBranch("Springfield")
            .openingBalance(100)
            .atRate(2.5)
            .build();
```

Anti-Patterns

An anti-pattern is a common response to a recurring problem that is **usually ineffective**, and sometimes **counterproductive**.

Telescoping Constructor Pattern

- overloading constructor with various parameters

References

Reference	URL
OO Design Patterns Explained	https://www.youtube.com/watch?v=aiSAO2AXa9g