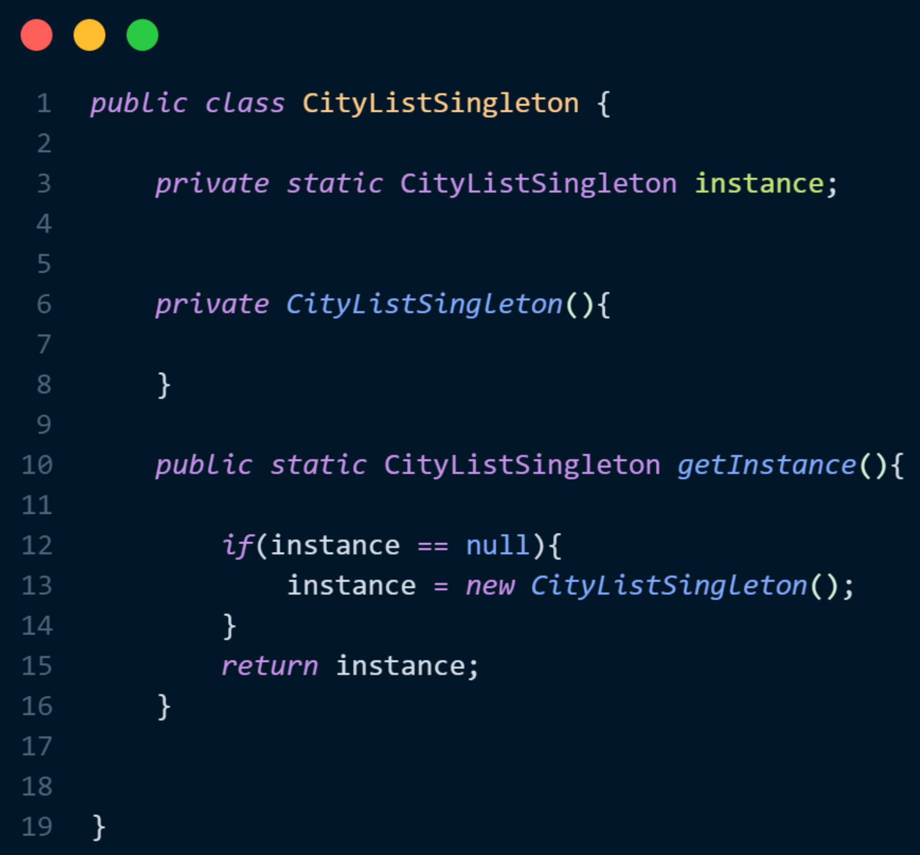
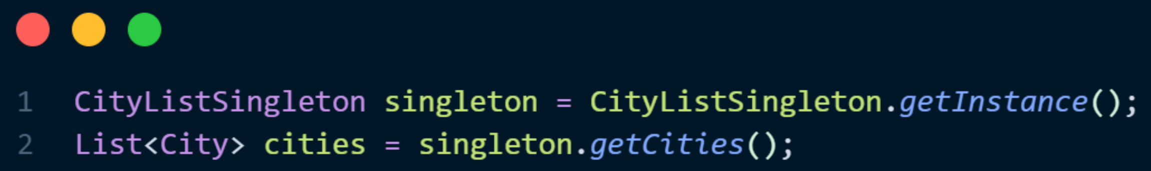
YAZILIM MİMARİSİ VE TASARIMI

Singleton Pattern



```java
public class CityListSingleton {

    private static CityListSingleton instance;


    private CityListSingleton(){

    }

    public static CityListSingleton getInstance(){

        if(instance == null){
            instance = new CityListSingleton();
        }
        return instance;
    }


}
```



```java
CityListSingleton singleton = CityListSingleton.getInstance();
List<City> cities = singleton.getCities();
```

Factory Pattern

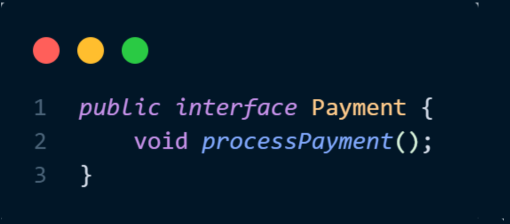
```java
public interface Payment {
    void processPayment();
}
```

```java
public class PaymentFactory {

    public Payment createPayment(String paymentType){

        if(paymentType.equalsIgnoreCase("CreditCard")){
            return new CreditCartPayment();
        }
        else if (paymentType.equalsIgnoreCase("BankTransfer")){
            return new BankTransferPayment();
        }
        else if (paymentType.equalsIgnoreCase("PayPal")){
            return new PayPalPayment();
        }
        else if (paymentType.equalsIgnoreCase("Dijital")){
            return new DijitalPayment();
        }


        return null;
    }

}
```

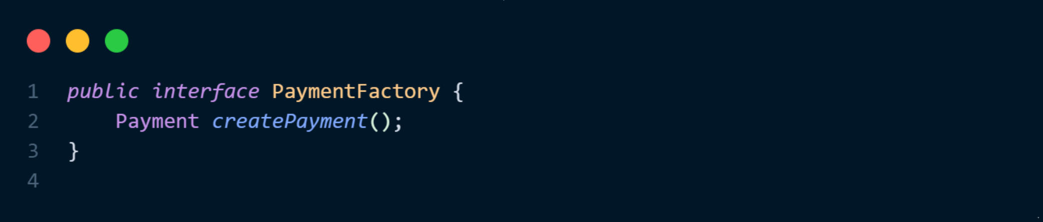
```java
public class CreditCartPayment implements  Payment{
    @Override
    public void processPayment() {
        System.out.println("Kredi Kartı ile Ödemeniz Gerçekleşti.");
    }
}
```

```java
PaymentFactory paymentFactory = new PaymentFactory();
Payment creditCartPayment = paymentFactory.createPayment("CreditCard");
creditCartPayment.processPayment();
```

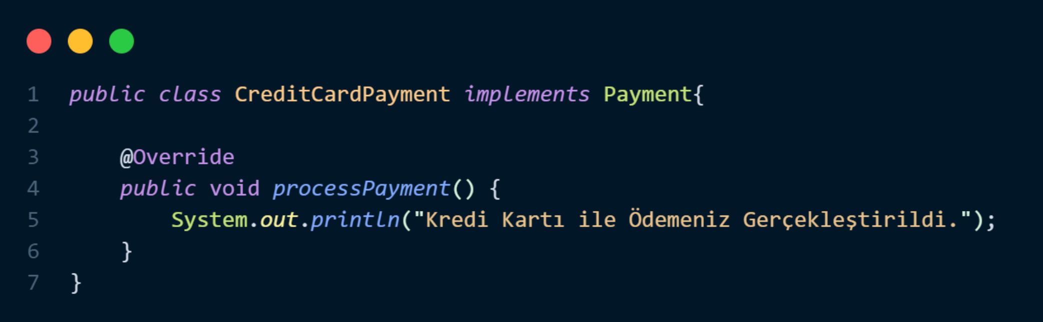
Abstract Factory Pattern



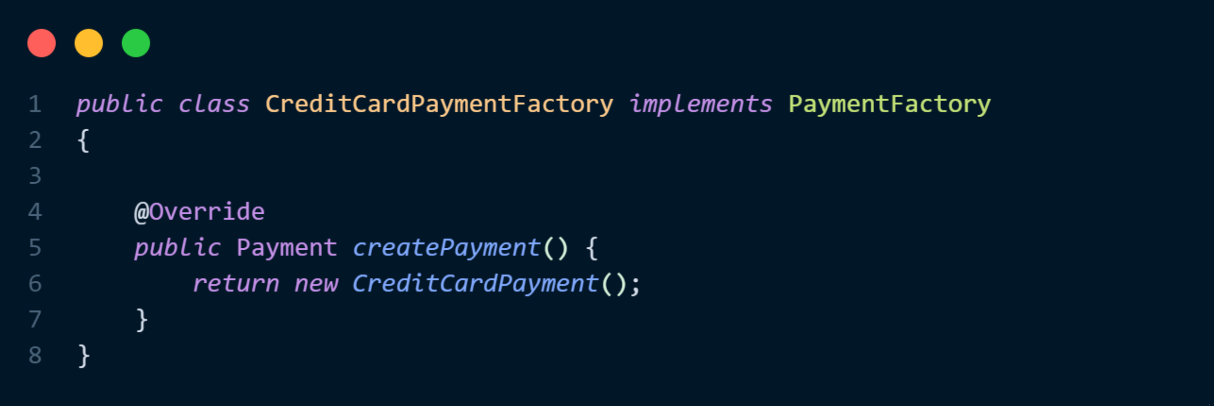
```java
public interface Payment {
    void processPayment();
}
```



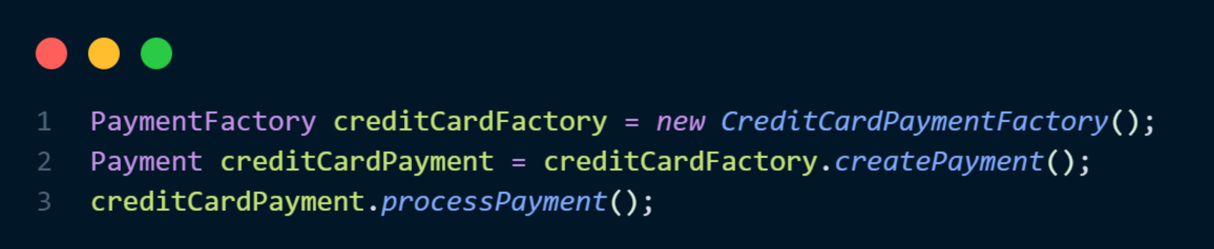
```java
public interface PaymentFactory {
    Payment createPayment();
}

```



```java
public class CreditCardPayment implements Payment{

    @Override
    public void processPayment() {
        System.out.println("Kredi Kartı ile Ödemeniz Gerçekleştirildi.");
    }
}
```

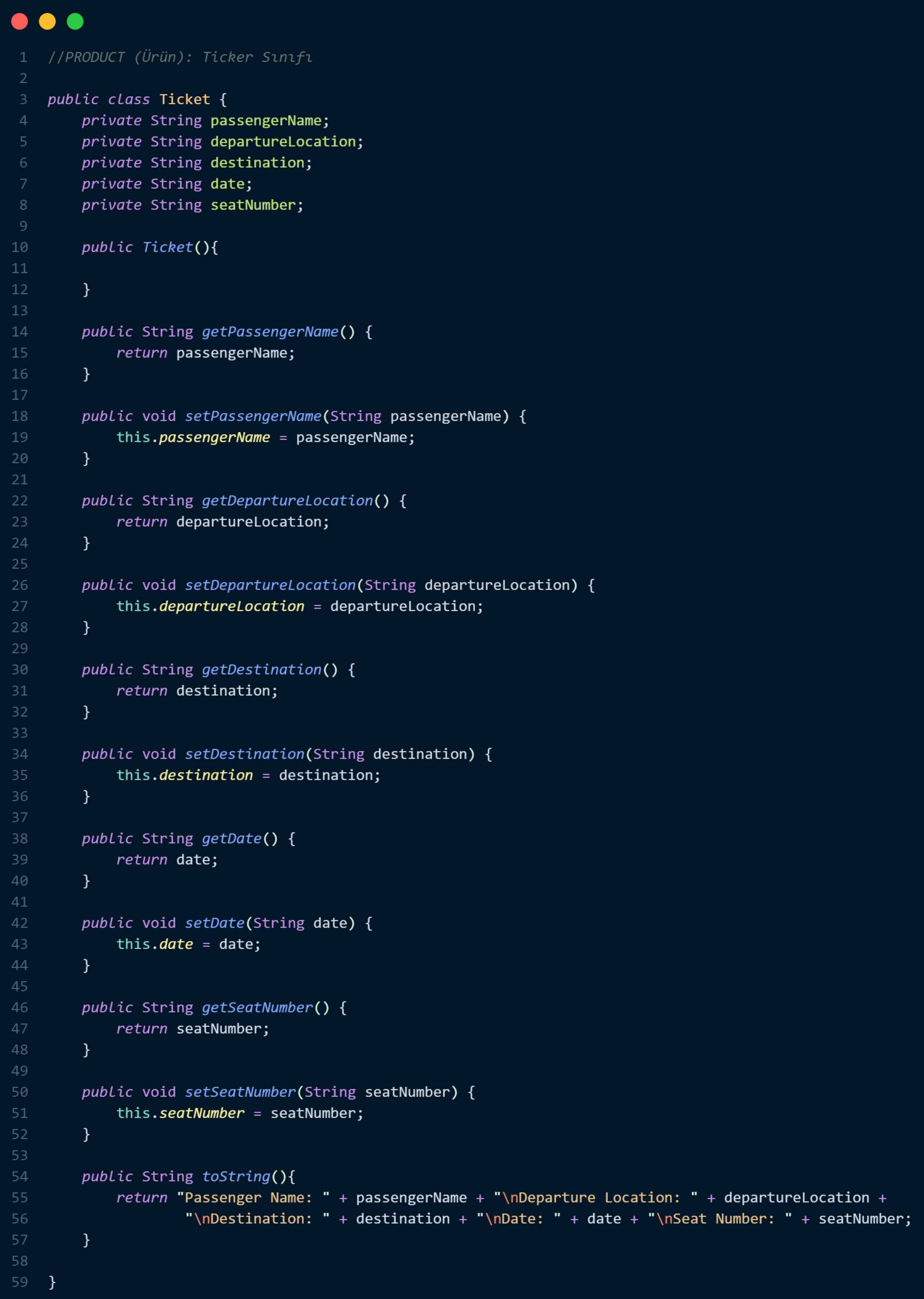


```java
public class CreditCardPaymentFactory implements PaymentFactory
{

    @Override
    public Payment createPayment() {
        return new CreditCardPayment();
    }
}
```

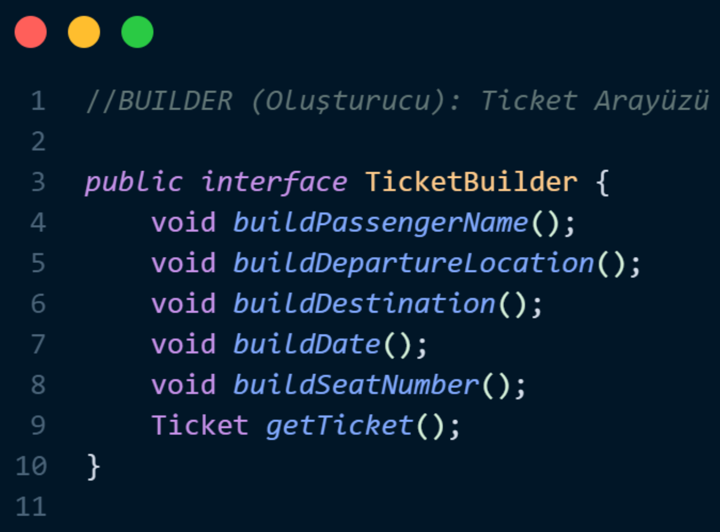


```java
PaymentFactory creditCardFactory = new CreditCardPaymentFactory();
Payment creditCardPayment = creditCardFactory.createPayment();
creditCardPayment.processPayment();
```

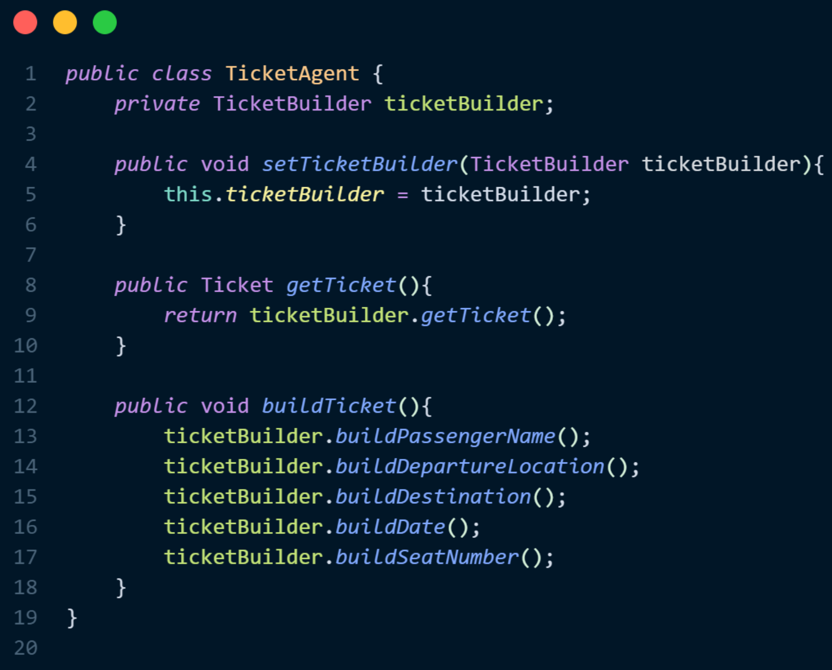
Builder Pattern



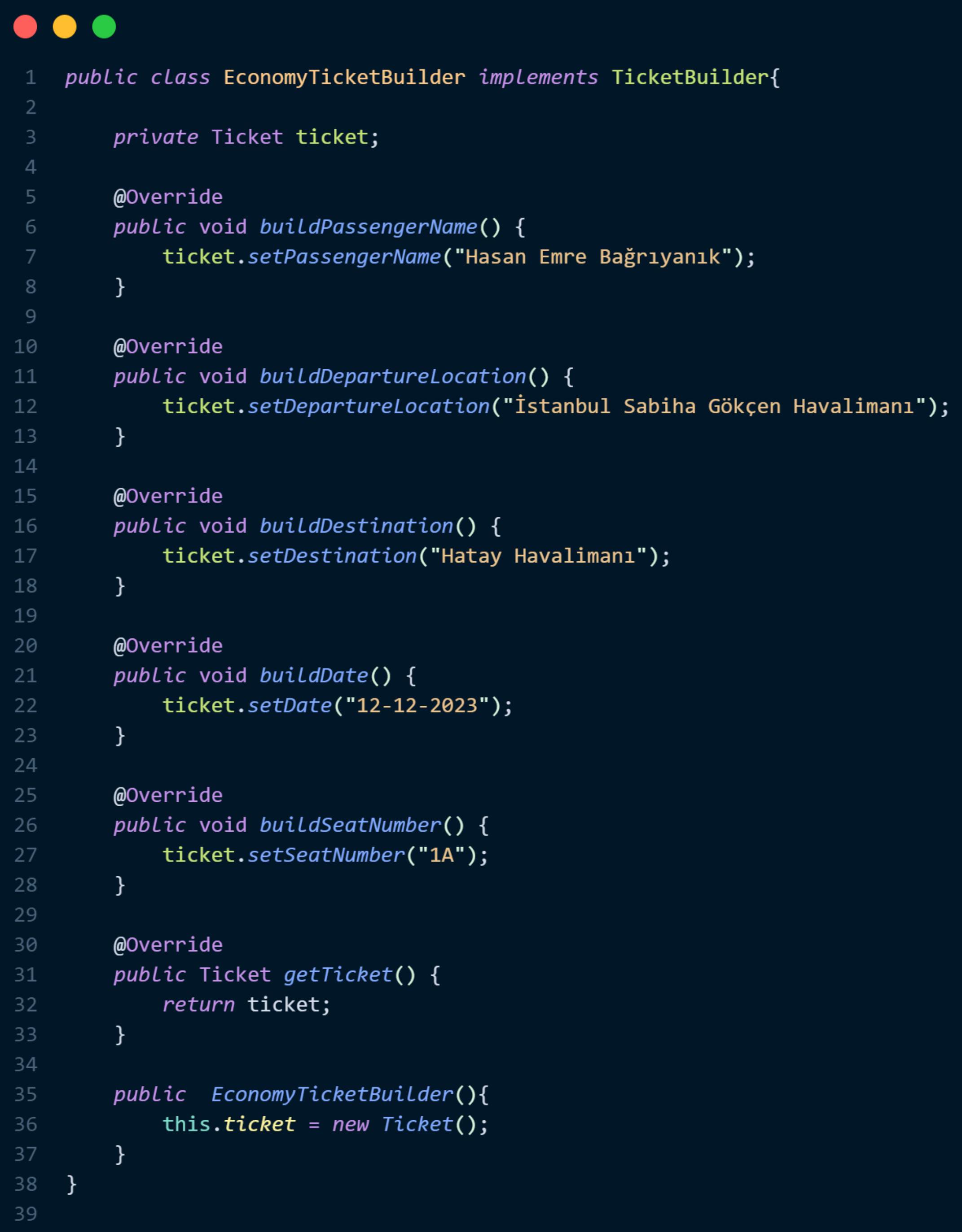
```java
//PRODUCT (Ürün): Ticker Sınıfı

public class Ticket {
    private String passengerName;
    private String departureLocation;
    private String destination;
    private String date;
    private String seatNumber;

    public Ticket(){

    }

    public String getPassengerName() {
        return passengerName;
    }

    public void setPassengerName(String passengerName) {
        this.passengerName = passengerName;
    }

    public String getDepartureLocation() {
        return departureLocation;
    }

    public void setDepartureLocation(String departureLocation) {
        this.departureLocation = departureLocation;
    }

    public String getDestination() {
        return destination;
    }

    public void setDestination(String destination) {
        this.destination = destination;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getSeatNumber() {
        return seatNumber;
    }

    public void setSeatNumber(String seatNumber) {
        this.seatNumber = seatNumber;
    }

    public String toString(){
        return "Passenger Name: " + passengerName + "\nDeparture Location: " + departureLocation +
                "\nDestination: " + destination + "\nDate: " + date + "\nSeat Number: " + seatNumber;
    }

}
```



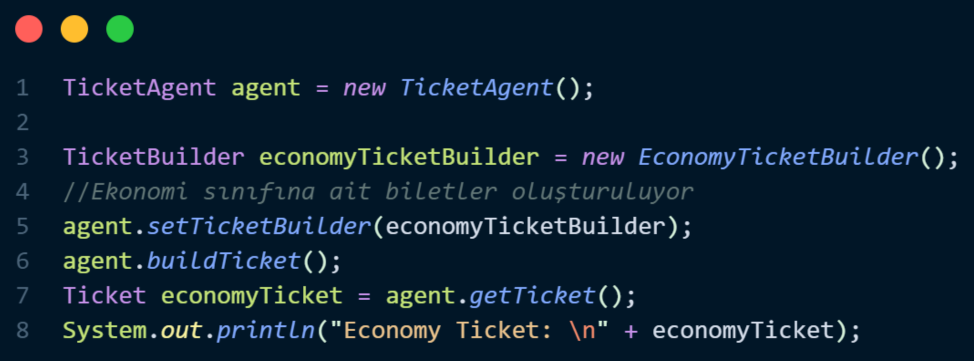
```java
//BUILDER (Oluşturucu): Ticket Arayüzü

public interface TicketBuilder {
    void buildPassengerName();
    void buildDepartureLocation();
    void buildDestination();
    void buildDate();
    void buildSeatNumber();
    Ticket getTicket();
}
```



```java
public class TicketAgent {
    private TicketBuilder ticketBuilder;

    public void setTicketBuilder(TicketBuilder ticketBuilder){
        this.ticketBuilder = ticketBuilder;
    }

    public Ticket getTicket(){
        return ticketBuilder.getTicket();
    }

    public void buildTicket(){
        ticketBuilder.buildPassengerName();
        ticketBuilder.buildDepartureLocation();
        ticketBuilder.buildDestination();
        ticketBuilder.buildDate();
        ticketBuilder.buildSeatNumber();
    }
}
```



```java
public class EconomyTicketBuilder implements TicketBuilder{

    private Ticket ticket;

    @Override
    public void buildPassengerName() {
        ticket.setPassengerName("Hasan Emre Bağrıyanık");
    }

    @Override
    public void buildDepartureLocation() {
        ticket.setDepartureLocation("İstanbul Sabiha Gökçen Havalimanı");
    }

    @Override
    public void buildDestination() {
        ticket.setDestination("Hatay Havalimanı");
    }

    @Override
    public void buildDate() {
        ticket.setDate("12-12-2023");
    }

    @Override
    public void buildSeatNumber() {
        ticket.setSeatNumber("1A");
    }

    @Override
    public Ticket getTicket() {
        return ticket;
    }

    public  EconomyTicketBuilder(){
        this.ticket = new Ticket();
    }
}

```



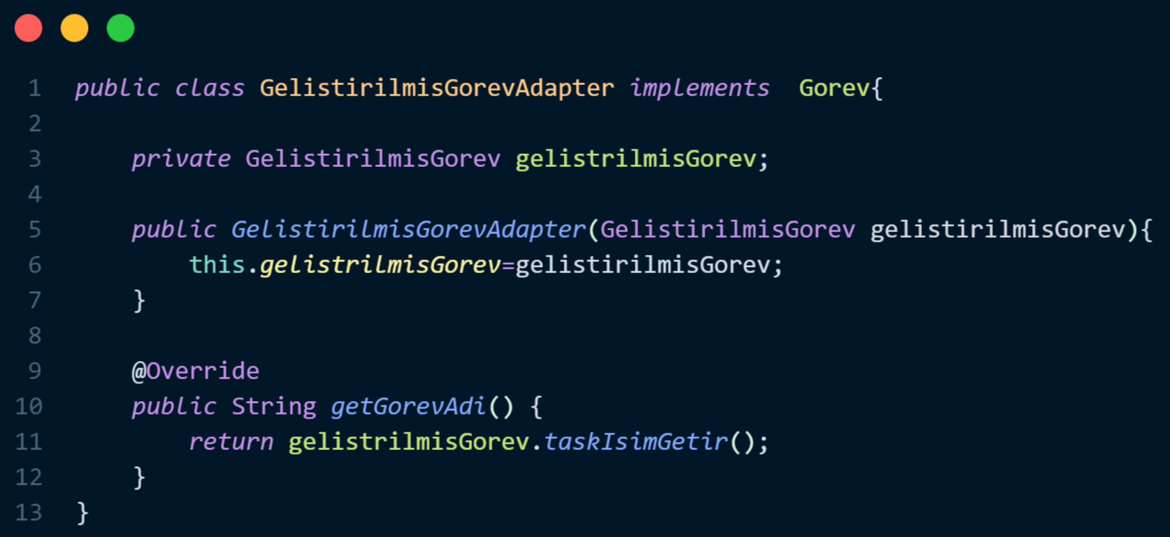
```java
TicketAgent agent = new TicketAgent();

TicketBuilder economyTicketBuilder = new EconomyTicketBuilder();
//Ekonomi sınıfına ait biletler oluşturuluyor
agent.setTicketBuilder(economyTicketBuilder);
agent.buildTicket();
Ticket economyTicket = agent.getTicket();
System.out.println("Economy Ticket: \n" + economyTicket);
```

Adapter Pattern

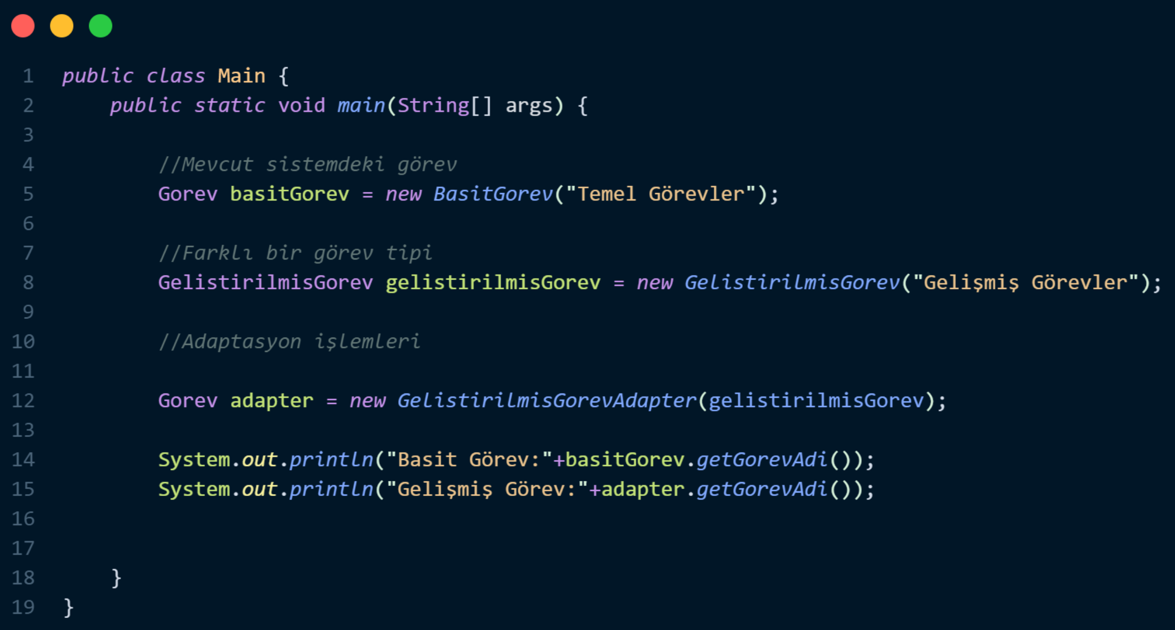
```java
public interface Gorev {
    String getGorevAdi();
}
```

```java
public class BasitGorev implements Gorev{

    private String gorevAdi;

    public BasitGorev(String gorevAdi){
        this.gorevAdi=gorevAdi;
    }

    @Override
    public String getGorevAdi() {
        return gorevAdi;
    }
}
```

```java
public class GelistirilmisGorev {
    private String taskName;
    public GelistirilmisGorev(String taskName){
        this.taskName=taskName;
    }
    public String taskIsimGetir(){
        return taskName;
    }
}
```

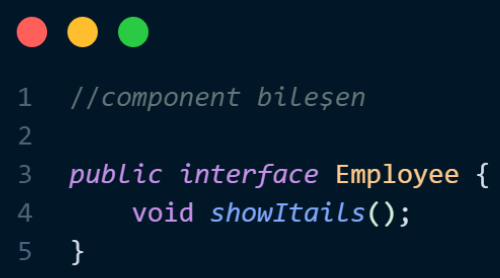


```java
public class GelistirilmisGorevAdapter implements  Gorev{

    private GelistirilmisGorev gelistrilmisGorev;

    public GelistirilmisGorevAdapter(GelistirilmisGorev gelistirilmisGorev){
        this.gelistrilmisGorev=gelistirilmisGorev;
    }

    @Override
    public String getGorevAdi() {
        return gelistrilmisGorev.taskIsimGetir();
    }
}
```

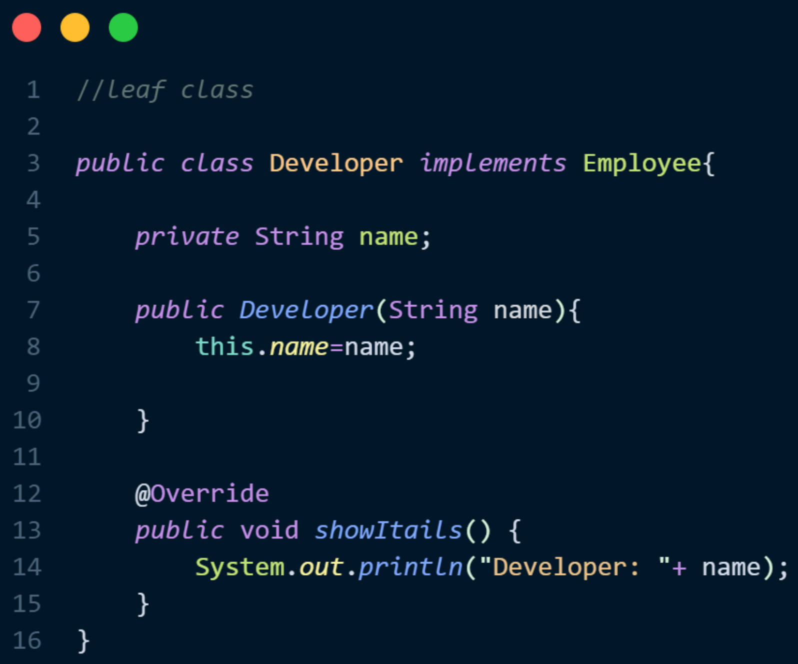


```java
public class Main {
    public static void main(String[] args) {

        //Mevcut sistemdeki görev
        Gorev basitGorev = new BasitGorev("Temel Görevler");

        //Farklı bir görev tipi
        GelistirilmisGorev gelistirilmisGorev = new GelistirilmisGorev("Gelişmiş Görevler");

        //Adaptasyon işlemleri

        Gorev adapter = new GelistirilmisGorevAdapter(gelistirilmisGorev);

        System.out.println("Basit Görev:"+basitGorev.getGorevAdi());
        System.out.println("Gelişmiş Görev:"+adapter.getGorevAdi());


    }
}
```

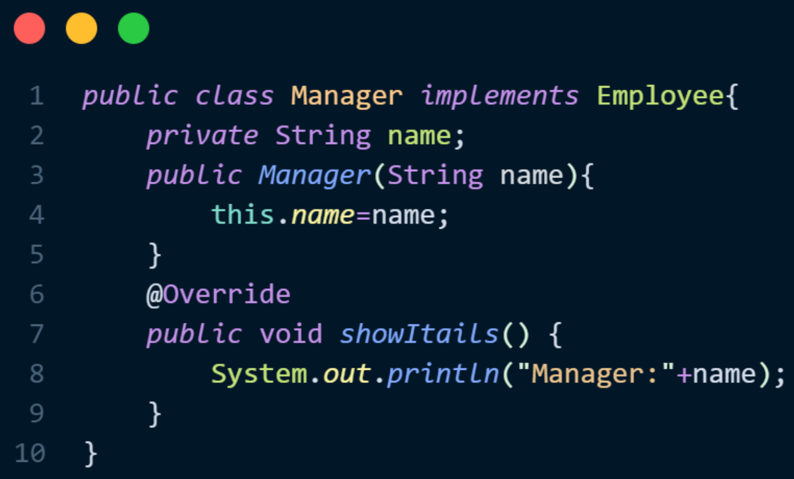
Composite Pattern



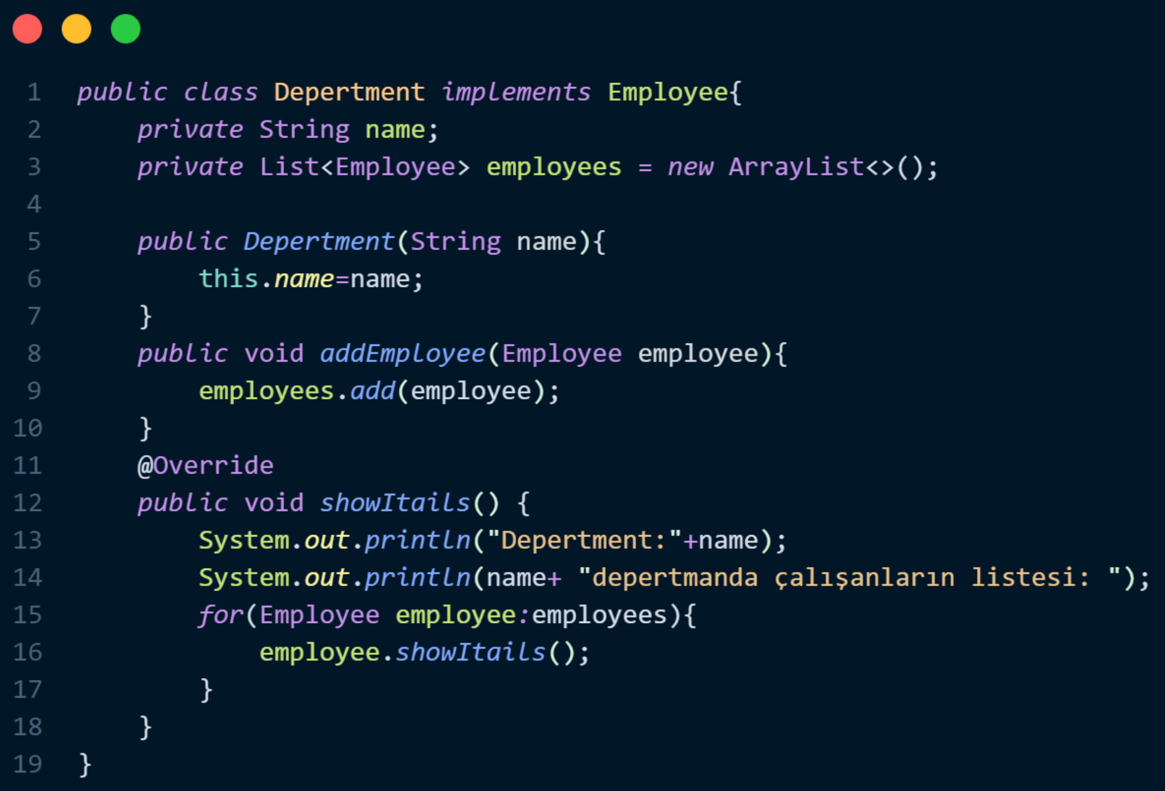
```java
//component bileşen

public interface Employee {
    void showItails();
}
```



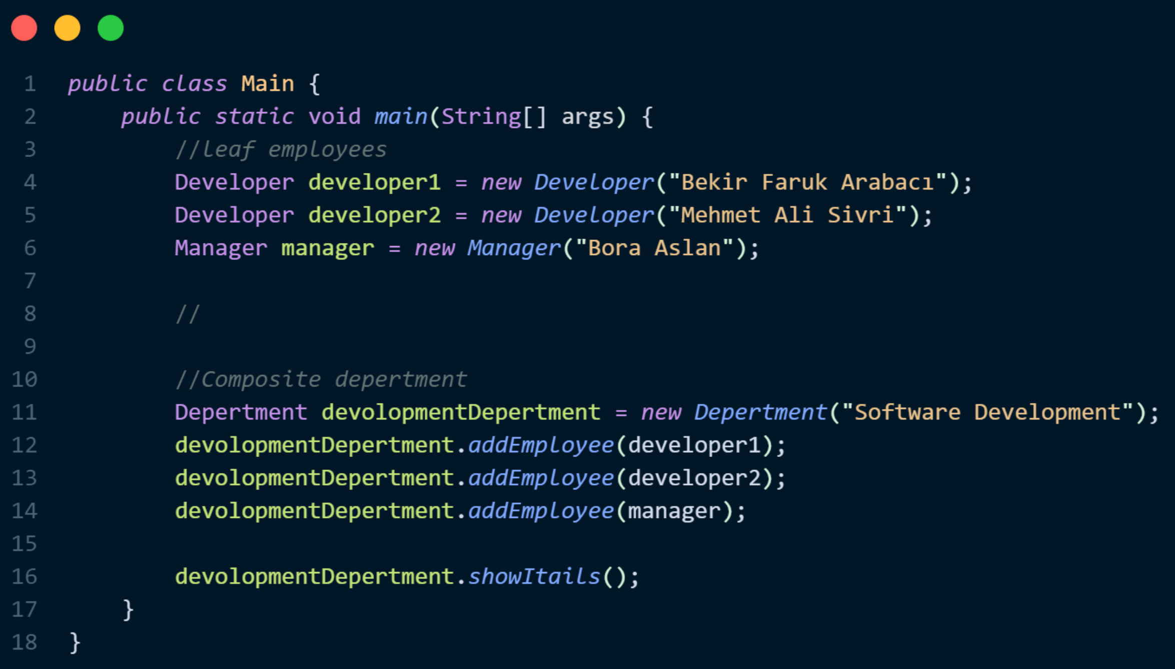
```java
//leaf class

public class Developer implements Employee{

    private String name;

    public Developer(String name){
        this.name=name;

    }

    @Override
    public void showItails() {
        System.out.println("Developer: "+ name);
    }
}
```



```java
public class Manager implements Employee{
    private String name;
    public Manager(String name){
        this.name=name;
    }
    @Override
    public void showItails() {
        System.out.println("Manager:"+name);
    }
}
```

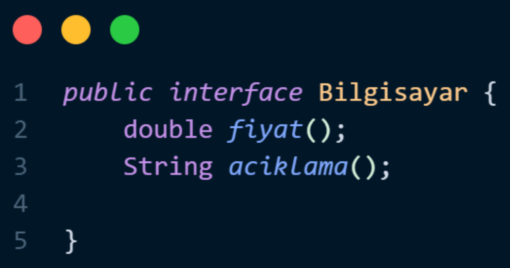


```java
public class Depertment implements Employee{
    private String name;
    private List<Employee> employees = new ArrayList<>();

    public Depertment(String name){
        this.name=name;
    }
    public void addEmployee(Employee employee){
        employees.add(employee);
    }
    @Override
    public void showItails() {
        System.out.println("Depertment:"+name);
        System.out.println(name+ "depertmanda çalışanların listesi: ");
        for(Employee employee:employees){
            employee.showItails();
        }
    }
}
```

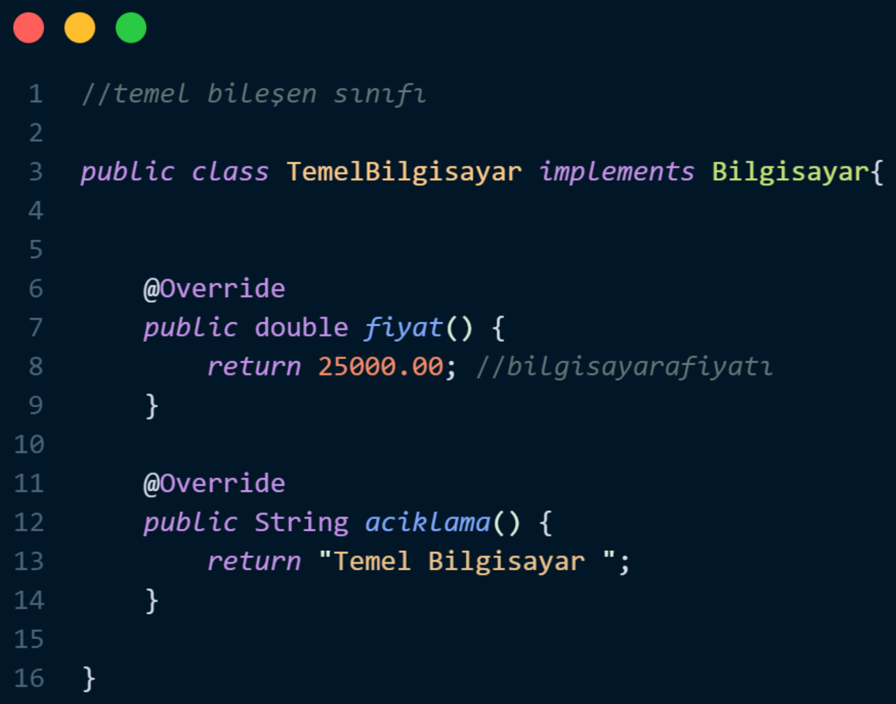


```java
public class Main {
    public static void main(String[] args) {
        //leaf employees
        Developer developer1 = new Developer("Bekir Faruk Arabacı");
        Developer developer2 = new Developer("Mehmet Ali Sivri");
        Manager manager = new Manager("Bora Aslan");

        //

        //Composite depertment
        Depertment devolopmentDepertment = new Depertment("Software Development");
        devolopmentDepertment.addEmployee(developer1);
        devolopmentDepertment.addEmployee(developer2);
        devolopmentDepertment.addEmployee(manager);

        devolopmentDepertment.showItails();
    }
}
```

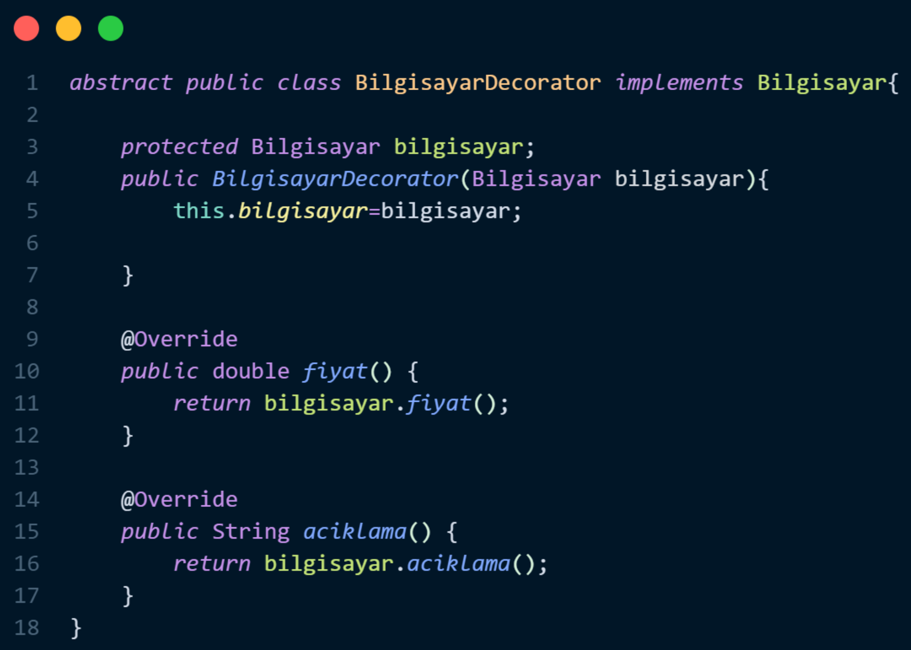
Decorator Pattern



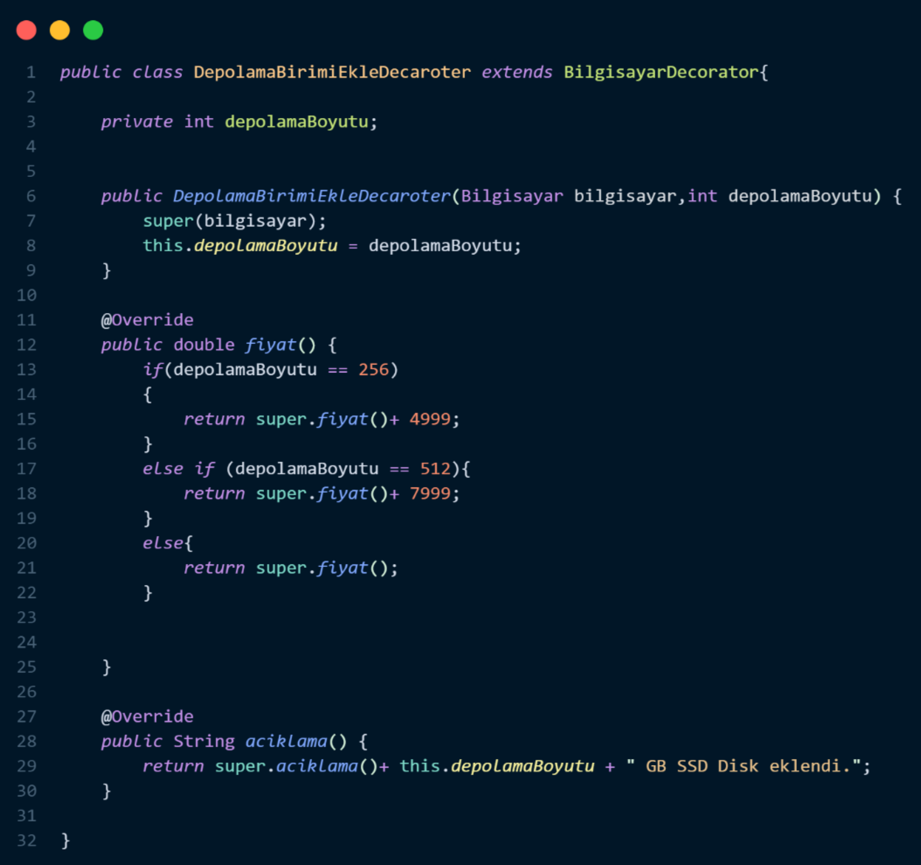
```java
public interface Bilgisayar {
    double fiyat();
    String aciklama();

}
```



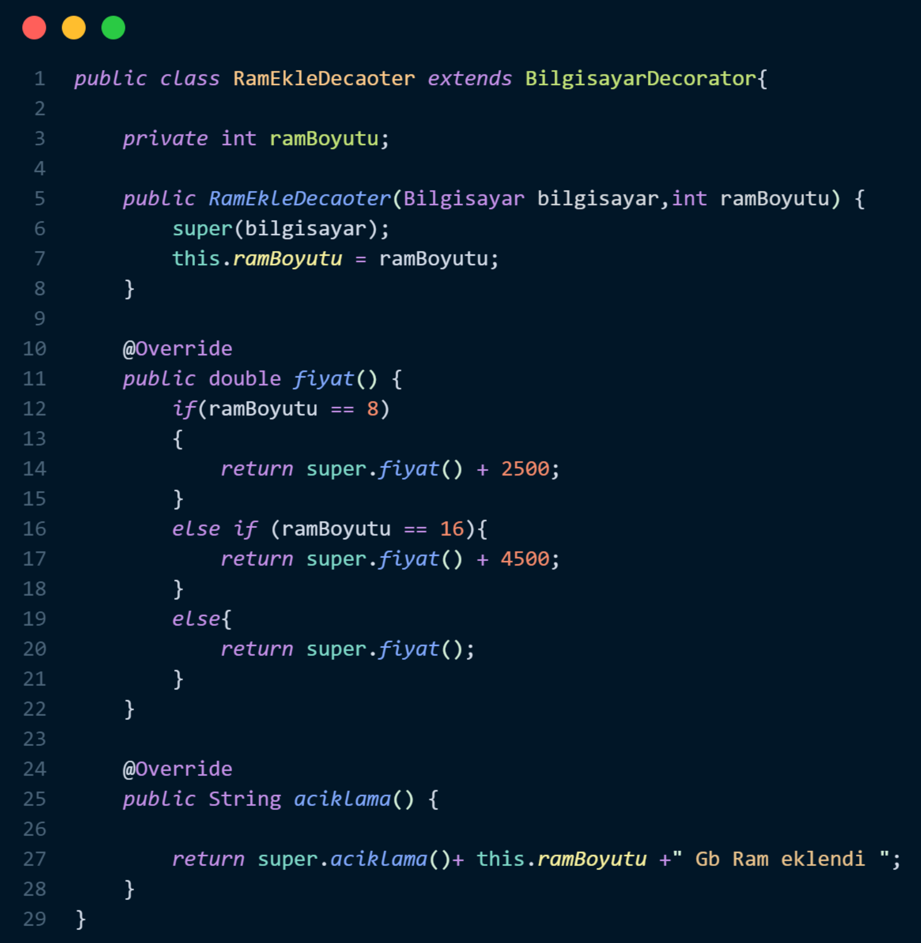
```java
//temel bileşen sınıfı

public class TemelBilgisayar implements Bilgisayar{


    @Override
    public double fiyat() {
        return 25000.00; //bilgisayarafiyatı
    }

    @Override
    public String aciklama() {
        return "Temel Bilgisayar ";
    }

}
```



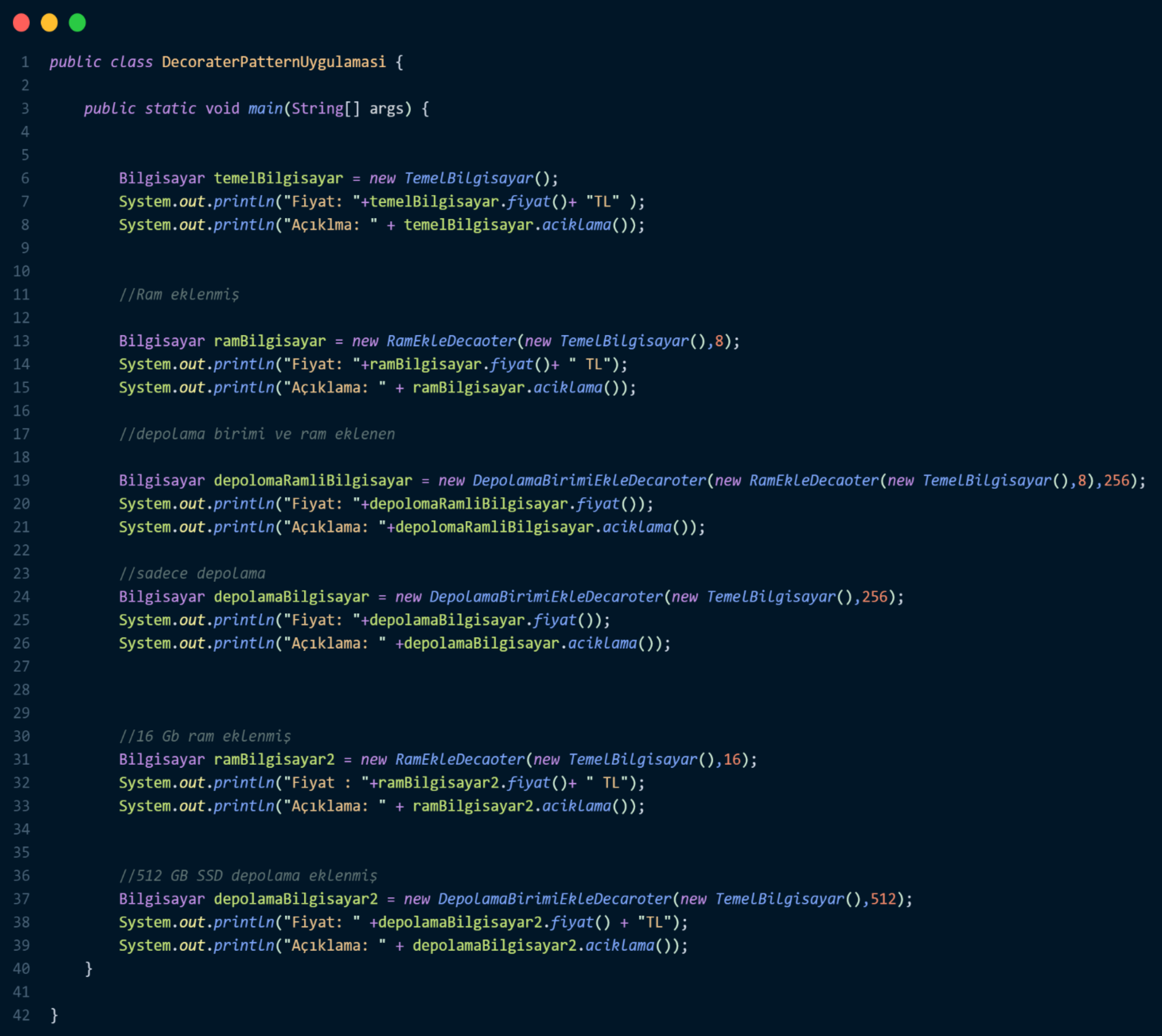
```java
abstract public class BilgisayarDecorator implements Bilgisayar{

    protected Bilgisayar bilgisayar;
    public BilgisayarDecorator(Bilgisayar bilgisayar){
        this.bilgisayar=bilgisayar;

    }

    @Override
    public double fiyat() {
        return bilgisayar.fiyat();
    }

    @Override
    public String aciklama() {
        return bilgisayar.aciklama();
    }
}
```



```java
public class DepolamaBirimiEkleDecaroter extends BilgisayarDecorator{

    private int depolamaBoyutu;


    public DepolamaBirimiEkleDecaroter(Bilgisayar bilgisayar,int depolamaBoyutu) {
        super(bilgisayar);
        this.depolamaBoyutu = depolamaBoyutu;
    }

    @Override
    public double fiyat() {
        if(depolamaBoyutu == 256)
        {
            return super.fiyat()+ 4999;
        }
        else if (depolamaBoyutu == 512){
            return super.fiyat()+ 7999;
        }
        else{
            return super.fiyat();
        }


    }

    @Override
    public String aciklama() {
        return super.aciklama()+ this.depolamaBoyutu + " GB SSD Disk eklendi.";
    }

}
```

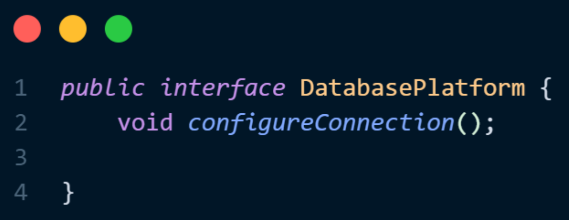


```java
public class RamEkleDecaoter extends BilgisayarDecorator{

    private int ramBoyutu;

    public RamEkleDecaoter(Bilgisayar bilgisayar,int ramBoyutu) {
        super(bilgisayar);
        this.ramBoyutu = ramBoyutu;
    }

    @Override
    public double fiyat() {
        if(ramBoyutu == 8)
        {
            return super.fiyat() + 2500;
        }
        else if (ramBoyutu == 16){
            return super.fiyat() + 4500;
        }
        else{
            return super.fiyat();
        }
    }

    @Override
    public String aciklama() {

        return super.aciklama()+ this.ramBoyutu +" Gb Ram eklendi ";
    }
}
```

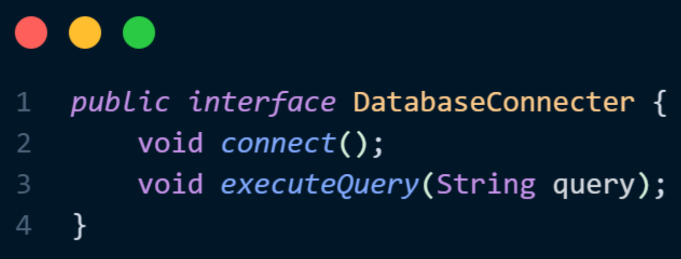


```java
public class DecoraterPatternUygulamasi {

    public static void main(String[] args) {


        Bilgisayar temelBilgisayar = new TemelBilgisayar();
        System.out.println("Fiyat: "+temelBilgisayar.fiyat()+ "TL" );
        System.out.println("Açıklma: " + temelBilgisayar.aciklama());


        //Ram eklenmiş

        Bilgisayar ramBilgisayar = new RamEkleDecaoter(new TemelBilgisayar(),8);
        System.out.println("Fiyat: "+ramBilgisayar.fiyat()+ " TL");
        System.out.println("Açıklama: " + ramBilgisayar.aciklama());

        //depolama birimi ve ram eklenen

        Bilgisayar depolomaRamliBilgisayar = new DepolamaBirimiEkleDecaroter(new RamEkleDecaoter(new TemelBilgisayar(),8),256);
        System.out.println("Fiyat: "+depolomaRamliBilgisayar.fiyat());
        System.out.println("Açıklama: "+depolomaRamliBilgisayar.aciklama());

        //sadece depolama
        Bilgisayar depolamaBilgisayar = new DepolamaBirimiEkleDecaroter(new TemelBilgisayar(),256);
        System.out.println("Fiyat: "+depolamaBilgisayar.fiyat());
        System.out.println("Açıklama: " +depolamaBilgisayar.aciklama());



        //16 Gb ram eklenmiş
        Bilgisayar ramBilgisayar2 = new RamEkleDecaoter(new TemelBilgisayar(),16);
        System.out.println("Fiyat : "+ramBilgisayar2.fiyat()+ " TL");
        System.out.println("Açıklama: " + ramBilgisayar2.aciklama());


        //512 GB SSD depolama eklenmiş
        Bilgisayar depolamaBilgisayar2 = new DepolamaBirimiEkleDecaroter(new TemelBilgisayar(),512);
        System.out.println("Fiyat: " +depolamaBilgisayar2.fiyat() + "TL");
        System.out.println("Açıklama: " + depolamaBilgisayar2.aciklama());
    }

}
```

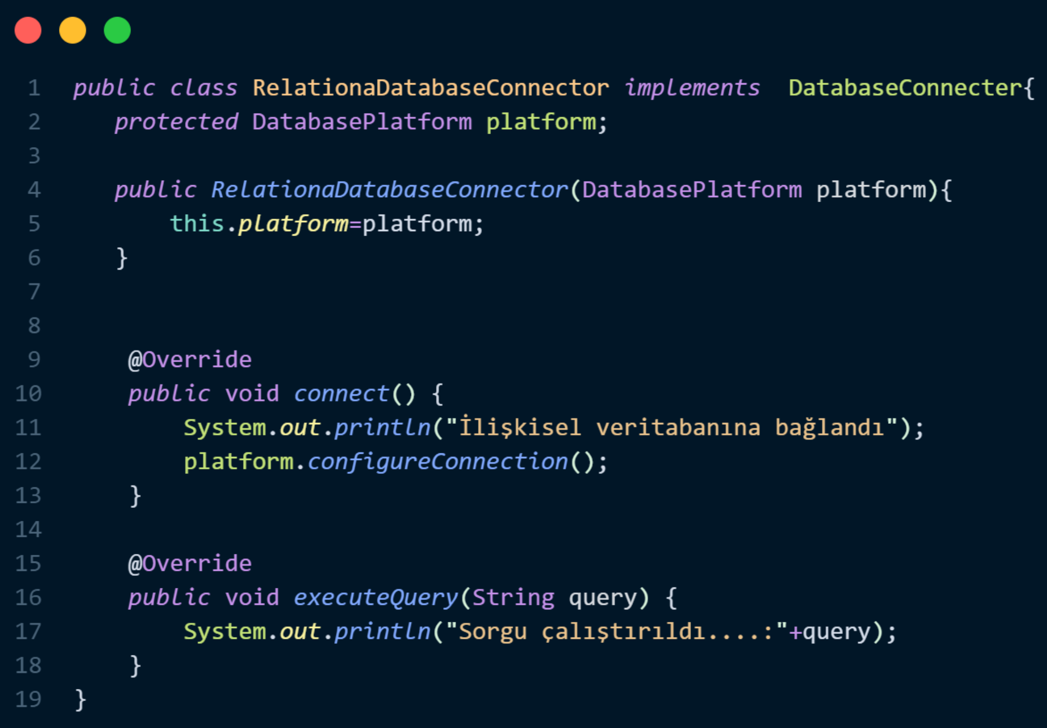
Bridge Pattern



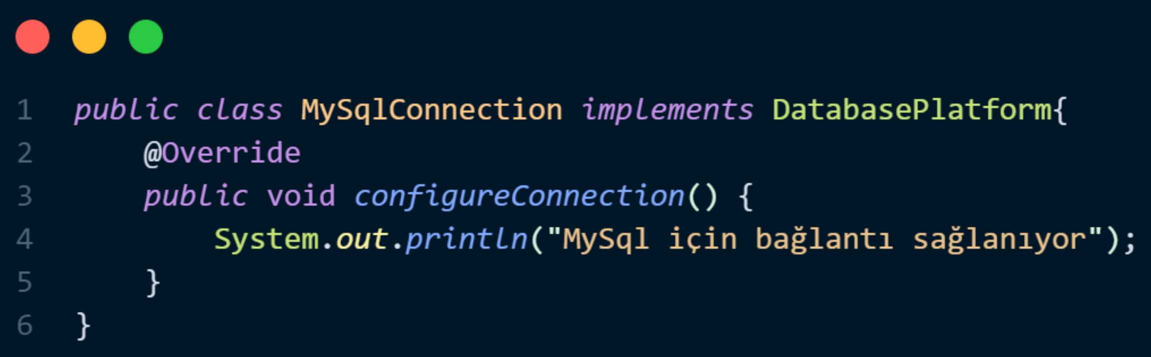
```java
public interface DatabasePlatform {
    void configureConnection();

}
```



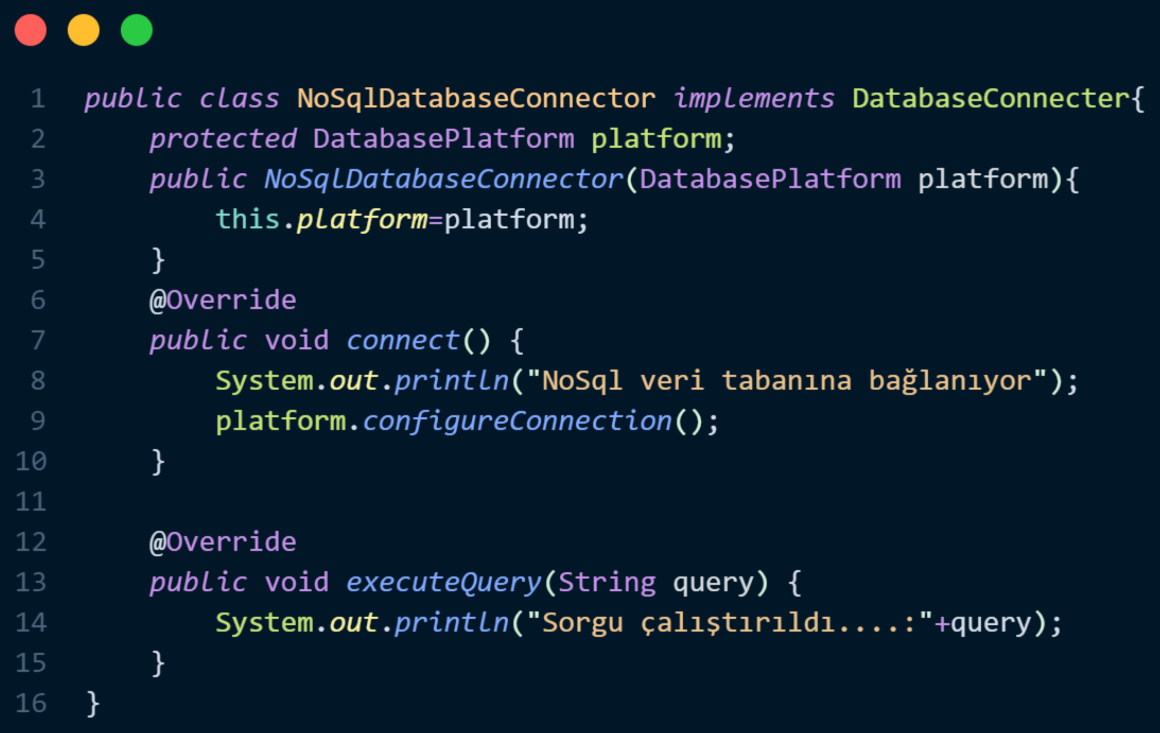
```java
public interface DatabaseConnecter {
     void connect();
     void executeQuery(String query);
}
```



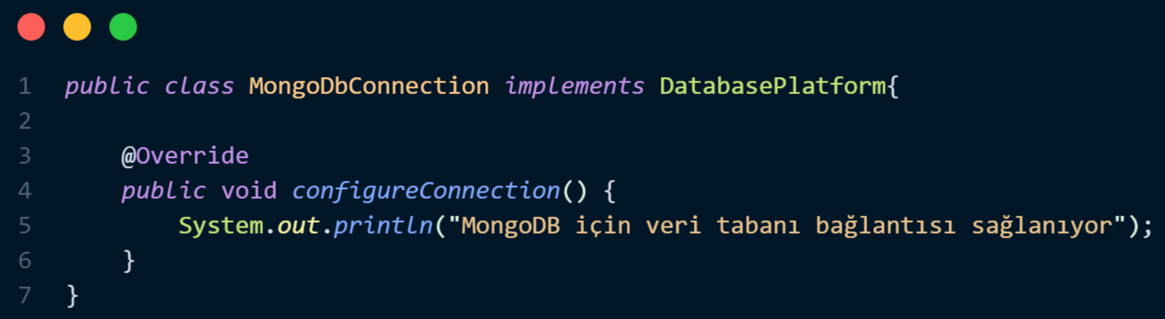
```java
public class RelationaDatabaseConnector implements  DatabaseConnecter{
   protected DatabasePlatform platform;

   public RelationaDatabaseConnector(DatabasePlatform platform){
       this.platform=platform;
   }


    @Override
    public void connect() {
        System.out.println("İlişkisel veritabanına bağlandı");
        platform.configureConnection();
    }

    @Override
    public void executeQuery(String query) {
        System.out.println("Sorgu çalıştırıldı....:"+query);
    }
}
```



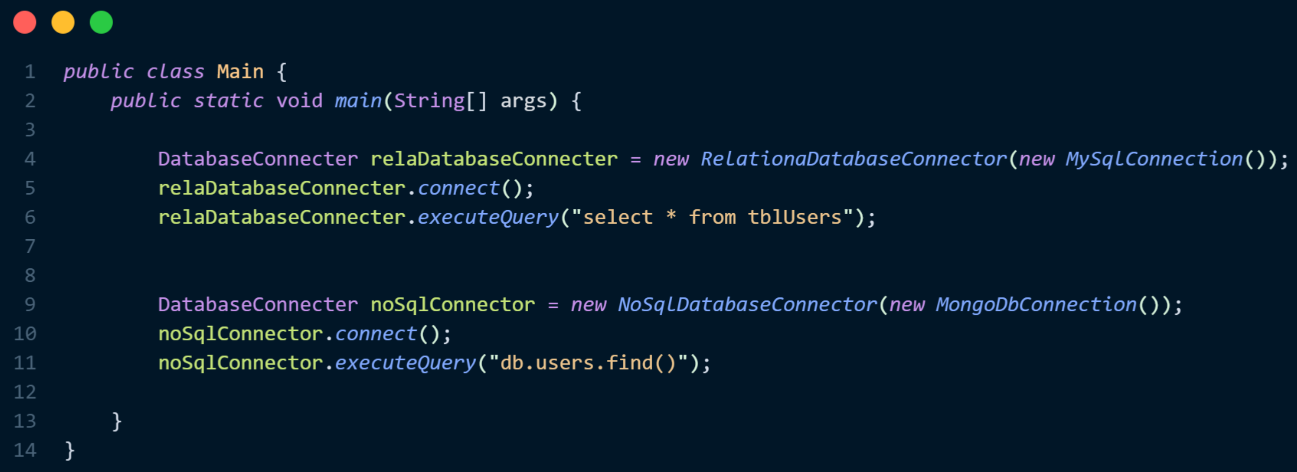
```java
public class MySqlConnection implements DatabasePlatform{
    @Override
    public void configureConnection() {
        System.out.println("MySql için bağlantı sağlanıyor");
    }
}
```



```java
public class NoSqlDatabaseConnector implements DatabaseConnecter{
    protected DatabasePlatform platform;
    public NoSqlDatabaseConnector(DatabasePlatform platform){
        this.platform=platform;
    }
    @Override
    public void connect() {
        System.out.println("NoSql veri tabanına bağlanıyor");
        platform.configureConnection();
    }

    @Override
    public void executeQuery(String query) {
        System.out.println("Sorgu çalıştırıldı....:"+query);
    }
}
```

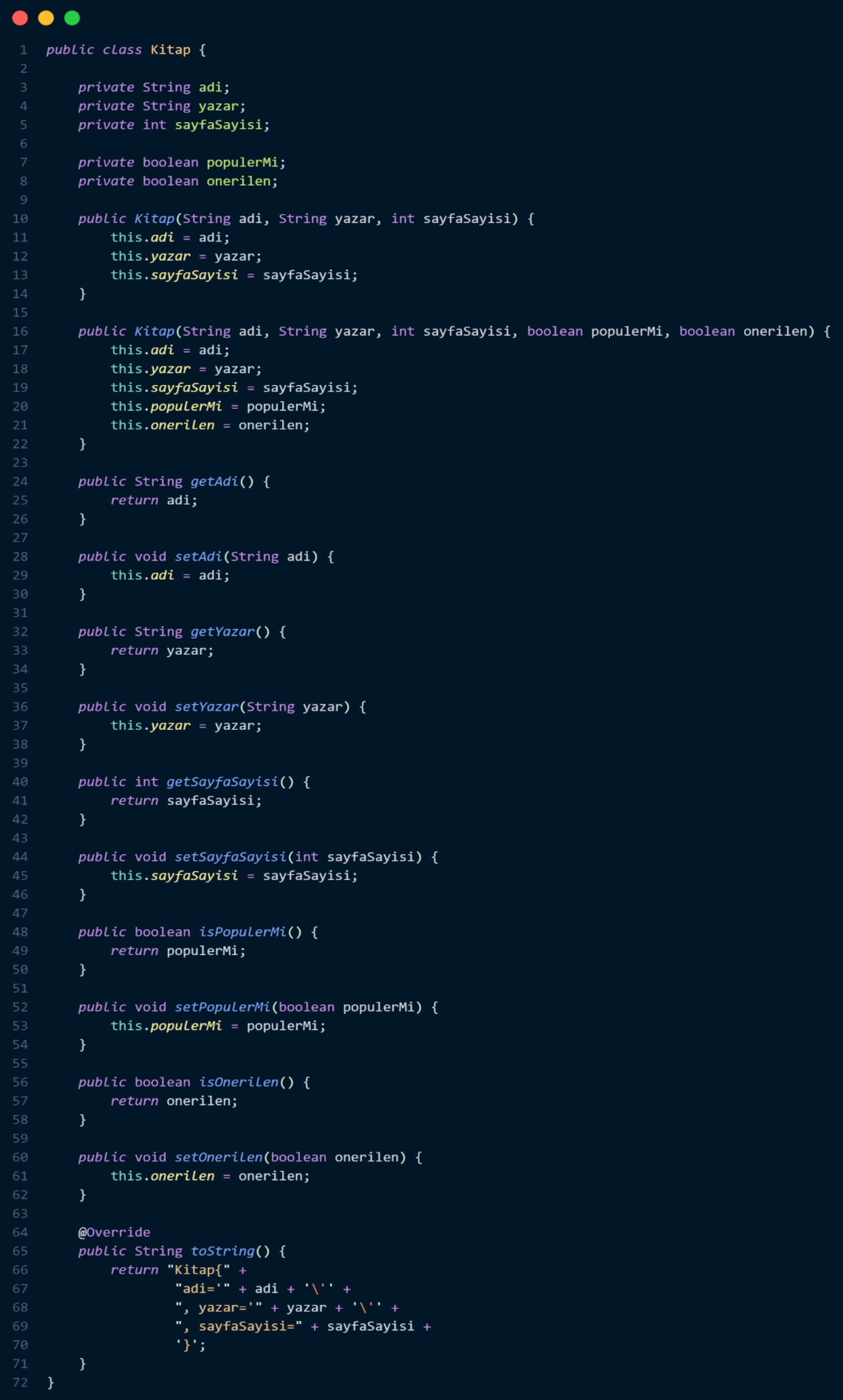


```java
public class MongoDbConnection implements DatabasePlatform{

    @Override
    public void configureConnection() {
        System.out.println("MongoDB için veri tabanı bağlantısı sağlanıyor");
    }
}
```

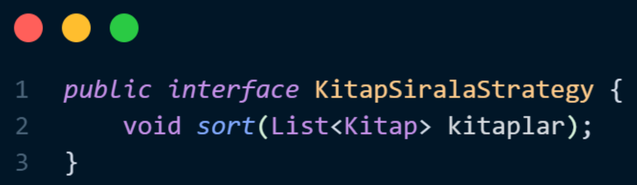


```java
public class Main {
    public static void main(String[] args) {

        DatabaseConnecter relaDatabaseConnecter = new RelationaDatabaseConnector(new MySqlConnection());
        relaDatabaseConnecter.connect();
        relaDatabaseConnecter.executeQuery("select * from tblUsers");


        DatabaseConnecter noSqlConnector = new NoSqlDatabaseConnector(new MongoDbConnection());
        noSqlConnector.connect();
        noSqlConnector.executeQuery("db.users.find()");

    }
}
```

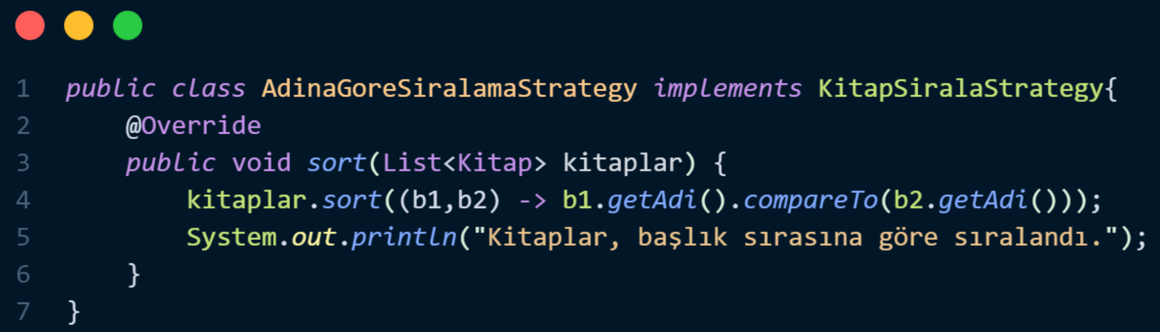
Strategy & State Pattern



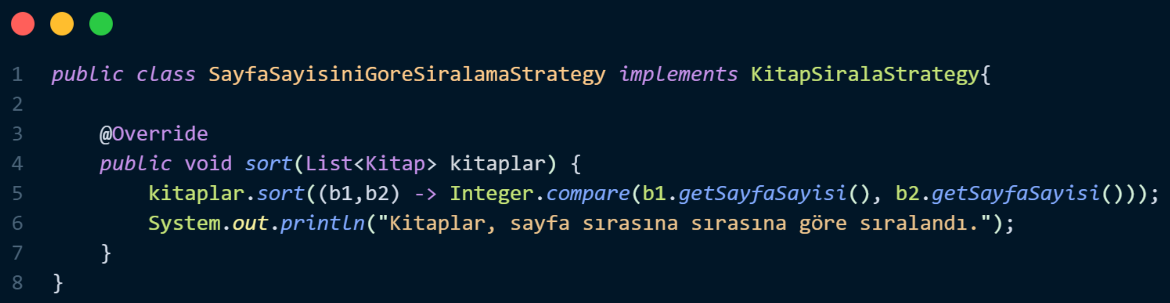
```java
public class Kitap {

    private String adi;
    private String yazar;
    private int sayfaSayisi;

    private boolean populerMi;
    private boolean onerilen;

    public Kitap(String adi, String yazar, int sayfaSayisi) {
        this.adi = adi;
        this.yazar = yazar;
        this.sayfaSayisi = sayfaSayisi;
    }

    public Kitap(String adi, String yazar, int sayfaSayisi, boolean populerMi, boolean onerilen) {
        this.adi = adi;
        this.yazar = yazar;
        this.sayfaSayisi = sayfaSayisi;
        this.populerMi = populerMi;
        this.onerilen = onerilen;
    }

    public String getAdi() {
        return adi;
    }

    public void setAdi(String adi) {
        this.adi = adi;
    }

    public String getYazar() {
        return yazar;
    }

    public void setYazar(String yazar) {
        this.yazar = yazar;
    }

    public int getSayfaSayisi() {
        return sayfaSayisi;
    }

    public void setSayfaSayisi(int sayfaSayisi) {
        this.sayfaSayisi = sayfaSayisi;
    }

    public boolean isPopulerMi() {
        return populerMi;
    }

    public void setPopulerMi(boolean populerMi) {
        this.populerMi = populerMi;
    }

    public boolean isOnerilen() {
        return onerilen;
    }

    public void setOnerilen(boolean onerilen) {
        this.onerilen = onerilen;
    }

    @Override
    public String toString() {
        return "Kitap{" +
                "adi='" + adi + '\'' +
                ", yazar='" + yazar + '\'' +
                ", sayfaSayisi=" + sayfaSayisi +
                '}';
    }
}
```



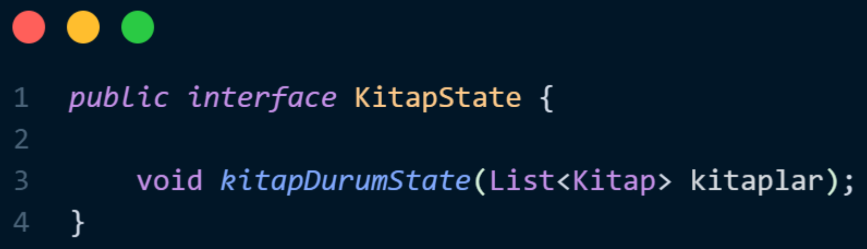
```java
public interface KitapSiralaStrategy {
    void sort(List<Kitap> kitaplar);
}
```



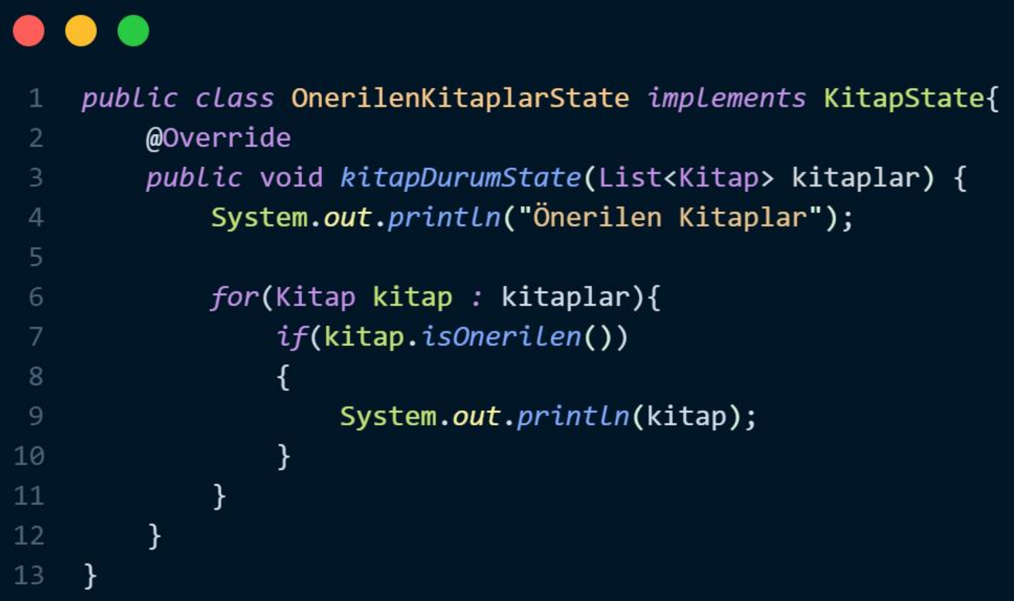
```java
public class AdinaGoreSiralamaStrategy implements KitapSiralaStrategy{
    @Override
    public void sort(List<Kitap> kitaplar) {
        kitaplar.sort((b1,b2) -> b1.getAdi().compareTo(b2.getAdi()));
        System.out.println("Kitaplar, başlık sırasına göre sıralandı.");
    }
}
```



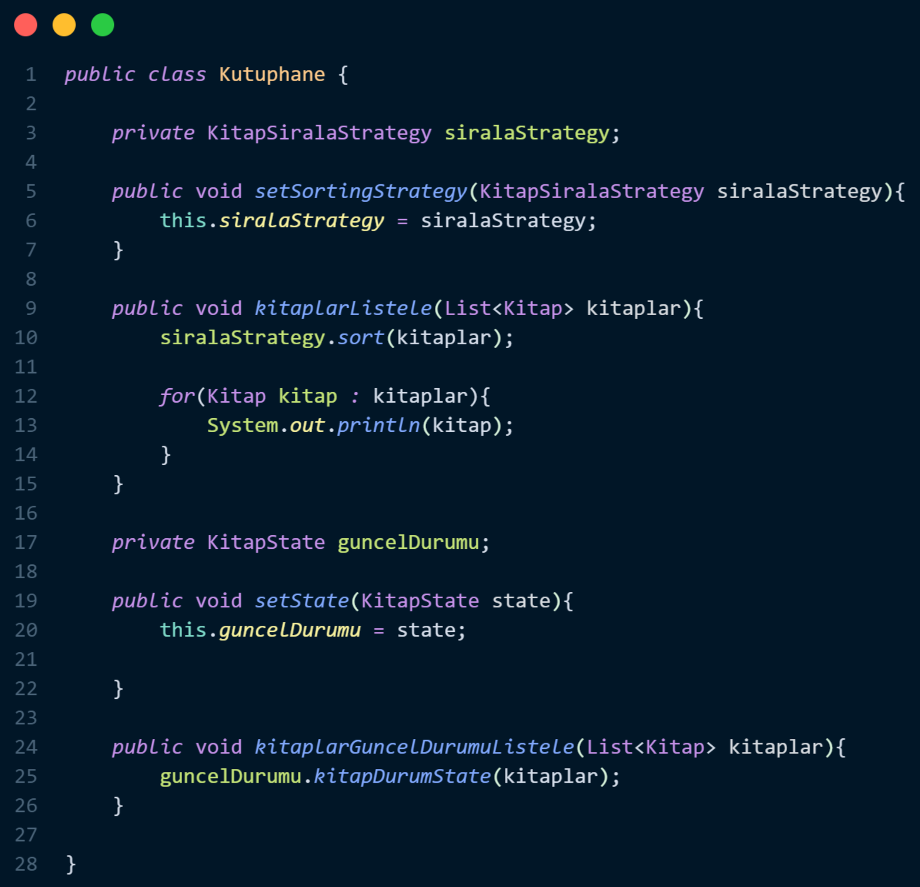
```java
public class SayfaSayisiniGoreSiralamaStrategy implements KitapSiralaStrategy{

    @Override
    public void sort(List<Kitap> kitaplar) {
        kitaplar.sort((b1,b2) -> Integer.compare(b1.getSayfaSayisi(), b2.getSayfaSayisi()));
        System.out.println("Kitaplar, sayfa sırasına sırasına göre sıralandı.");
    }
}
```



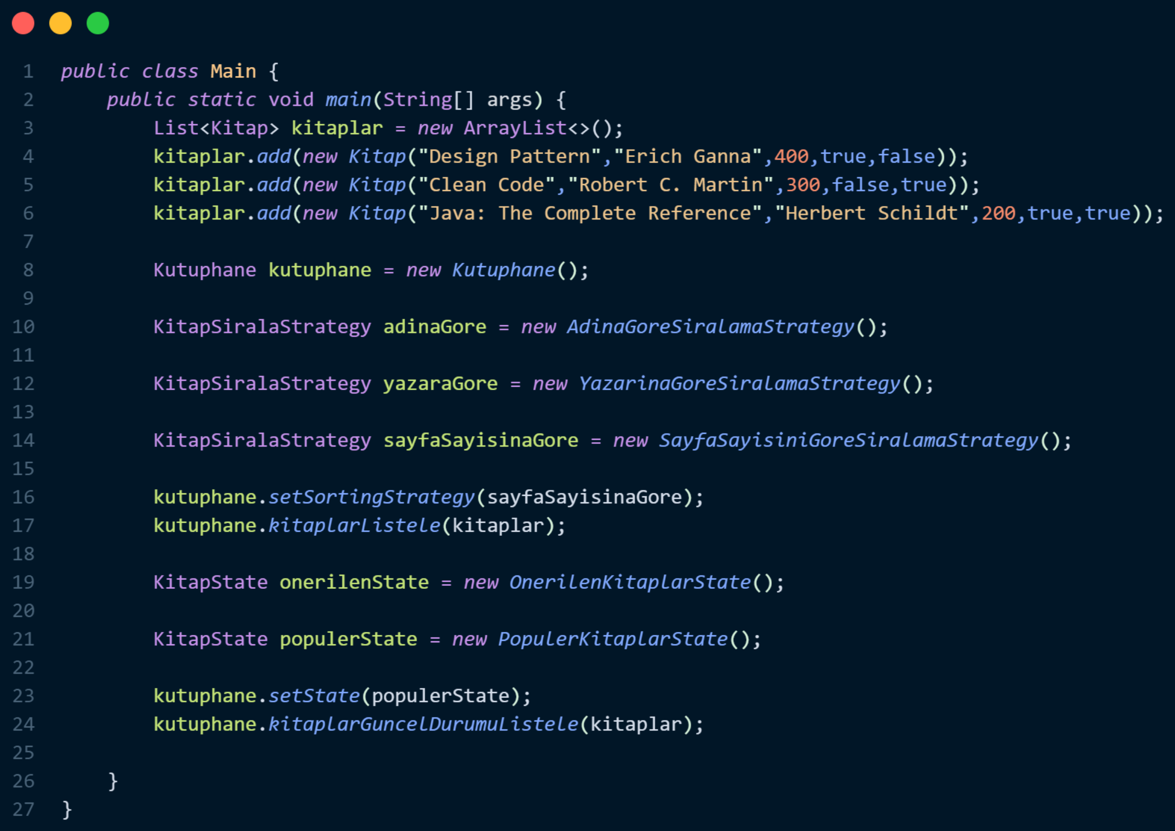
```java
public interface KitapState {

    void kitapDurumState(List<Kitap> kitaplar);
}
```



```java
public class OnerilenKitaplarState implements KitapState{
    @Override
    public void kitapDurumState(List<Kitap> kitaplar) {
        System.out.println("Önerilen Kitaplar");

        for(Kitap kitap : kitaplar){
            if(kitap.isOnerilen())
            {
                System.out.println(kitap);
            }
        }
    }
}
```



```java
public class Kutuphane {

    private KitapSiralaStrategy siralaStrategy;

    public void setSortingStrategy(KitapSiralaStrategy siralaStrategy){
        this.siralaStrategy = siralaStrategy;
    }

    public void kitaplarListele(List<Kitap> kitaplar){
        siralaStrategy.sort(kitaplar);

        for(Kitap kitap : kitaplar){
            System.out.println(kitap);
        }
    }

    private KitapState guncelDurumu;

    public void setState(KitapState state){
        this.guncelDurumu = state;

    }

    public void kitaplarGuncelDurumuListele(List<Kitap> kitaplar){
        guncelDurumu.kitapDurumState(kitaplar);
    }

}
```



```java
public class Main {
    public static void main(String[] args) {
        List<Kitap> kitaplar = new ArrayList<>();
        kitaplar.add(new Kitap("Design Pattern","Erich Ganna",400,true,false));
        kitaplar.add(new Kitap("Clean Code","Robert C. Martin",300,false,true));
        kitaplar.add(new Kitap("Java: The Complete Reference","Herbert Schildt",200,true,true));

        Kutuphane kutuphane = new Kutuphane();

        KitapSiralaStrategy adinaGore = new AdinaGoreSiralamaStrategy();

        KitapSiralaStrategy yazaraGore = new YazarinaGoreSiralamaStrategy();

        KitapSiralaStrategy sayfaSayisinaGore = new SayfaSayisiniGoreSiralamaStrategy();

        kutuphane.setSortingStrategy(sayfaSayisinaGore);
        kutuphane.kitaplarListele(kitaplar);

        KitapState onerilenState = new OnerilenKitaplarState();

        KitapState populerState = new PopulerKitaplarState();

        kutuphane.setState(populerState);
        kutuphane.kitaplarGuncelDurumuListele(kitaplar);

    }
}
```