

# Model View Presenter Mimarisel Örüntüsü

# MVC ve MVP

- **MVP** örüntüsü **MVC'nin bir türevidir**
- View tarafındaki **UI mantığı** UI bileşenlerinden tamamen çıkartılır ve Presenter'da toplanır
- UI bileşenlerinin görevi sadece **UI render** ile sınırlandırılır
- **İki farklı türü vardır**
  - Passive View
  - Supervising Controller

# MVP



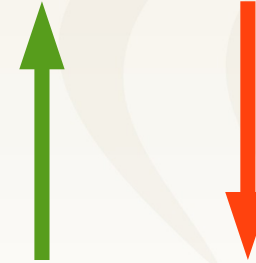
View

View ile Model  
arasında data  
binding'de  
mümkündür  
(Supervising controller)

UI event'leri uygulamaya özel business  
event'lere dönüştürülür

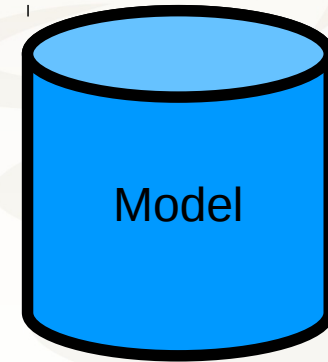


UI state değişiklikleri  
UI'a Presenter tarafından  
yansıtılır



Presenter  
üzerinden  
iş mantığı  
koşturulur

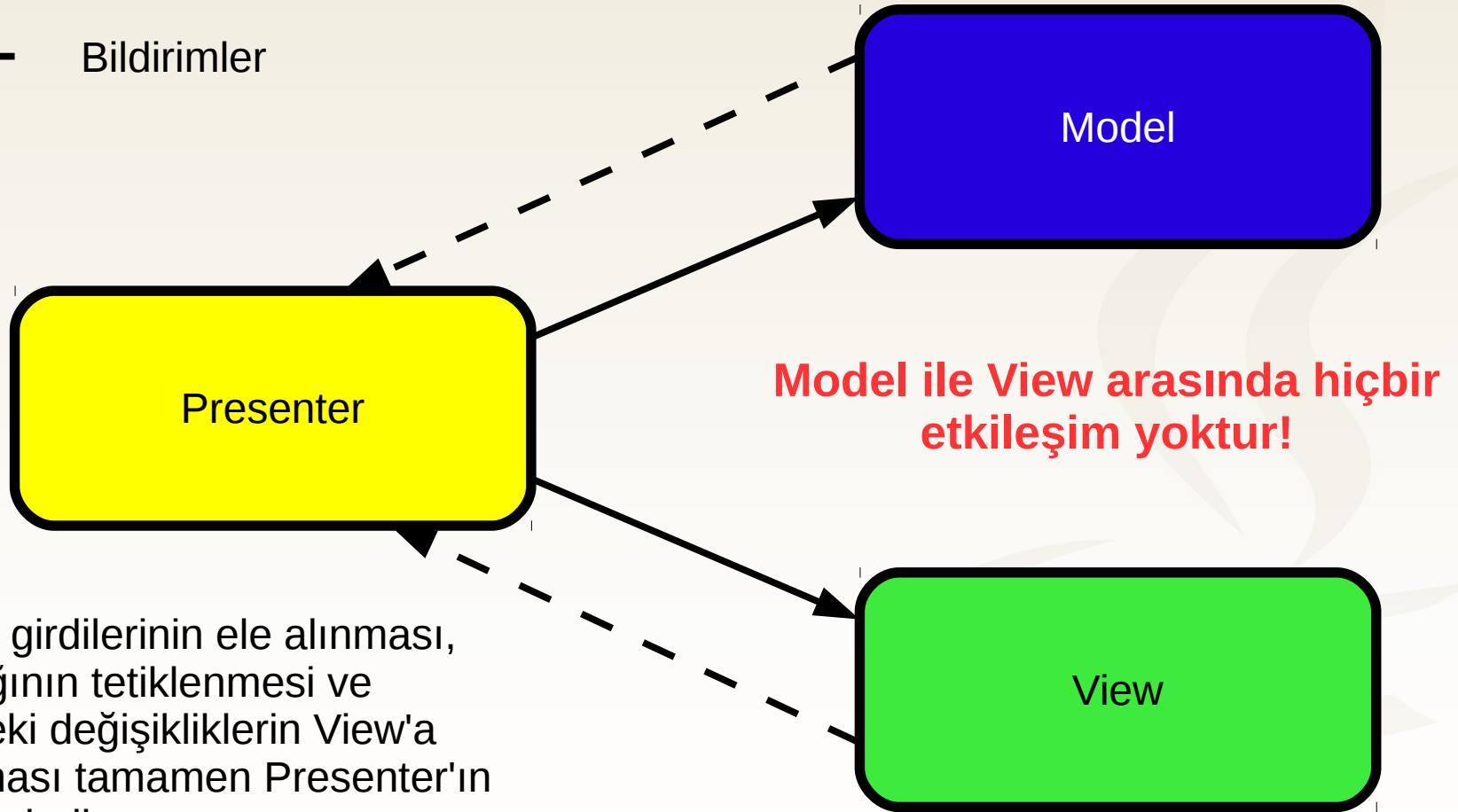
Model üzerindeki  
değişiklikler  
Presenter  
tarafından takip  
edilir



# MVP Türleri: Passive View

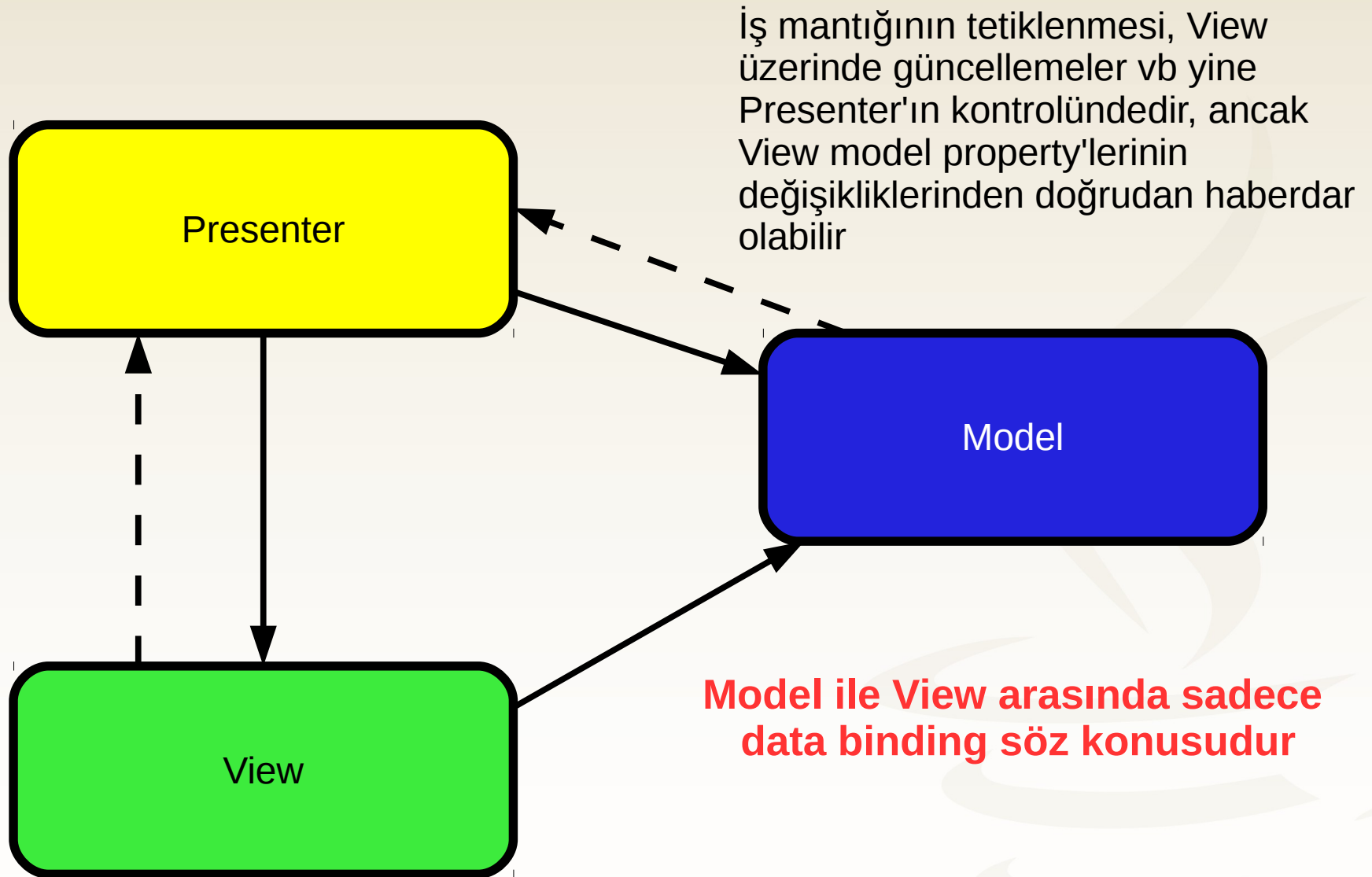
→ Etkileşimin yönü

← - - - Bildirimler



Kullanıcı girdilerinin ele alınması, iş mantığının tetiklenmesi ve model'deki değişikliklerin View'a yansıtılması tamamen Presenter'ın kontrolündedir

# MVP Türleri: Supervising Controller



# MVP ve TDD

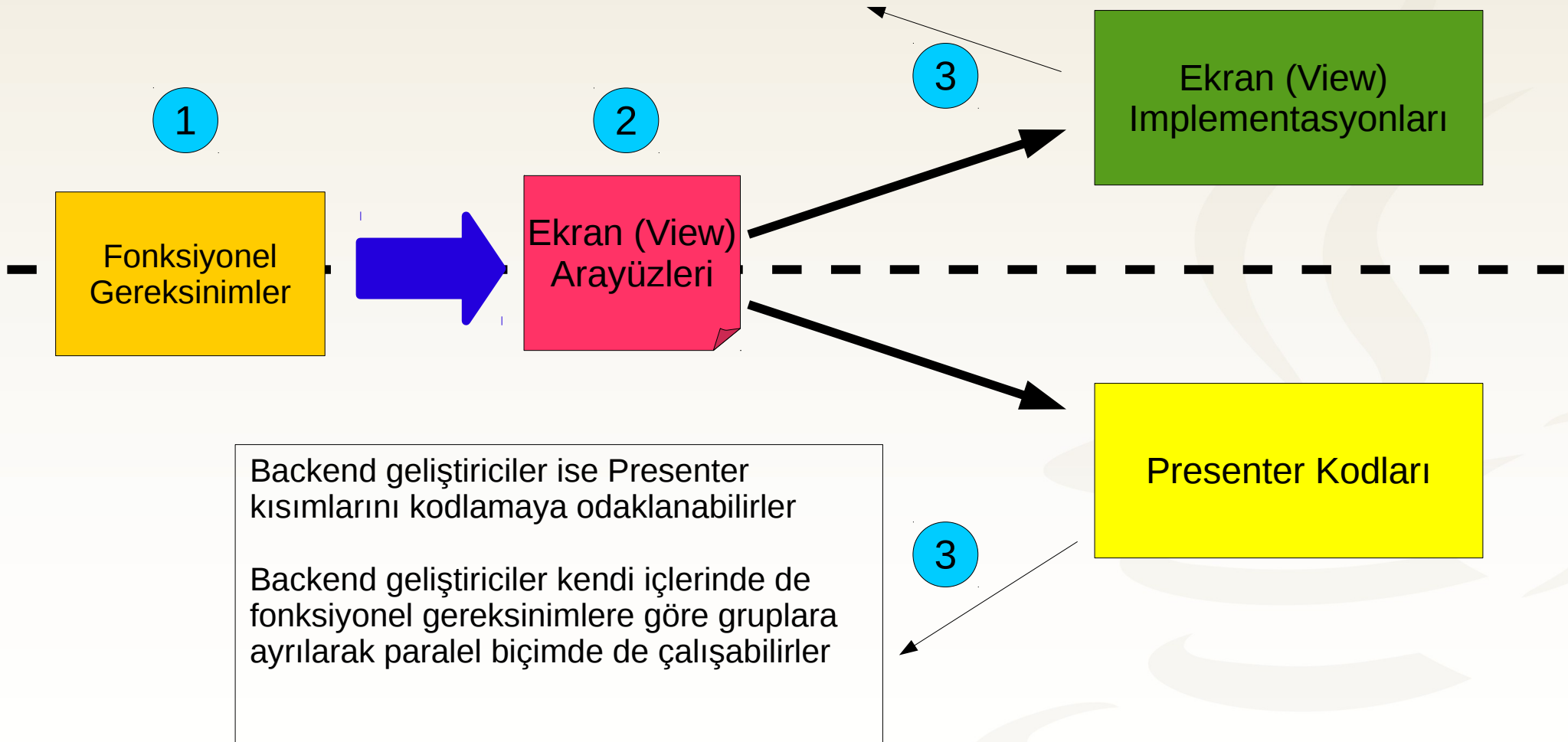
- MVP örüntüsü, **TDD** ile yazılım geliştirmeyi de kolaylaştırmaktadır
- Presenter sınıflarının **UI bileşenlerinden bağımsız** biçimde geliştirilmesi mümkündür
- Kullanıcı senaryoları ve gereksinimleri bire bir **Presenter içindeki fonksiyonlara** karşılık gelmektedir

# MVP ve TDD

- Presenter kodları geliştirilirken ilgili **model** ve **view** nesnelerine veya **servis bileşenlerine** ihtiyaç duyulur
- Bu nesnelerin gerçek implemantasyonları yerine sahteleri (**mock**) Presenter'a sunulabilir
- Böylece Presenter kodları **diğer katmanlardan bağımsız biçimde** kendi başına geliştirilebilir

# MVP ve TDD

UI geliştiriciler tamamen GUI geliştirmeye odaklanabilirler. View içerisinde sadece UI widget'ların oluşturulması ve sayfalara yerleştirilmesi söz konusudur



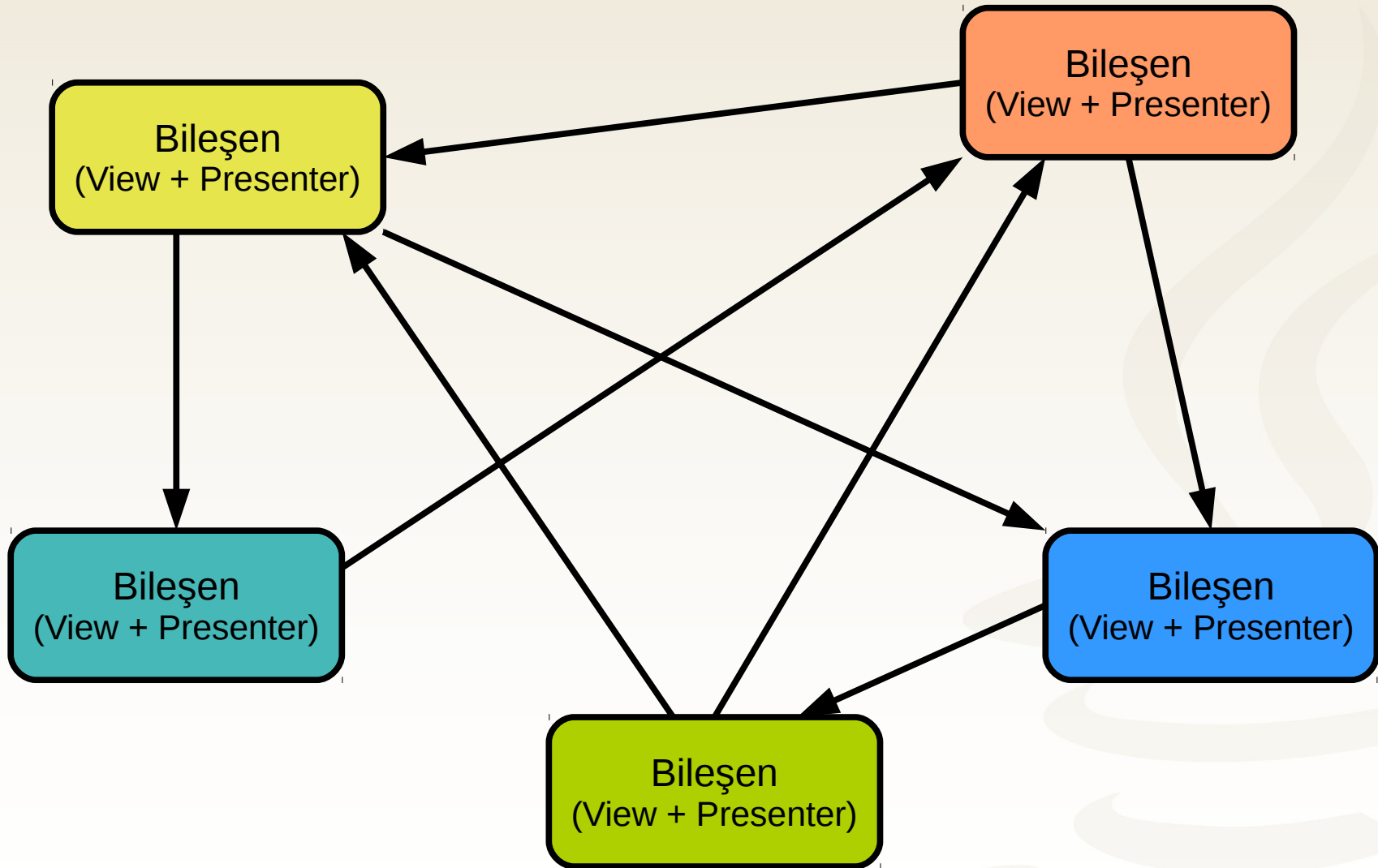


# Yalnız Başına MVP

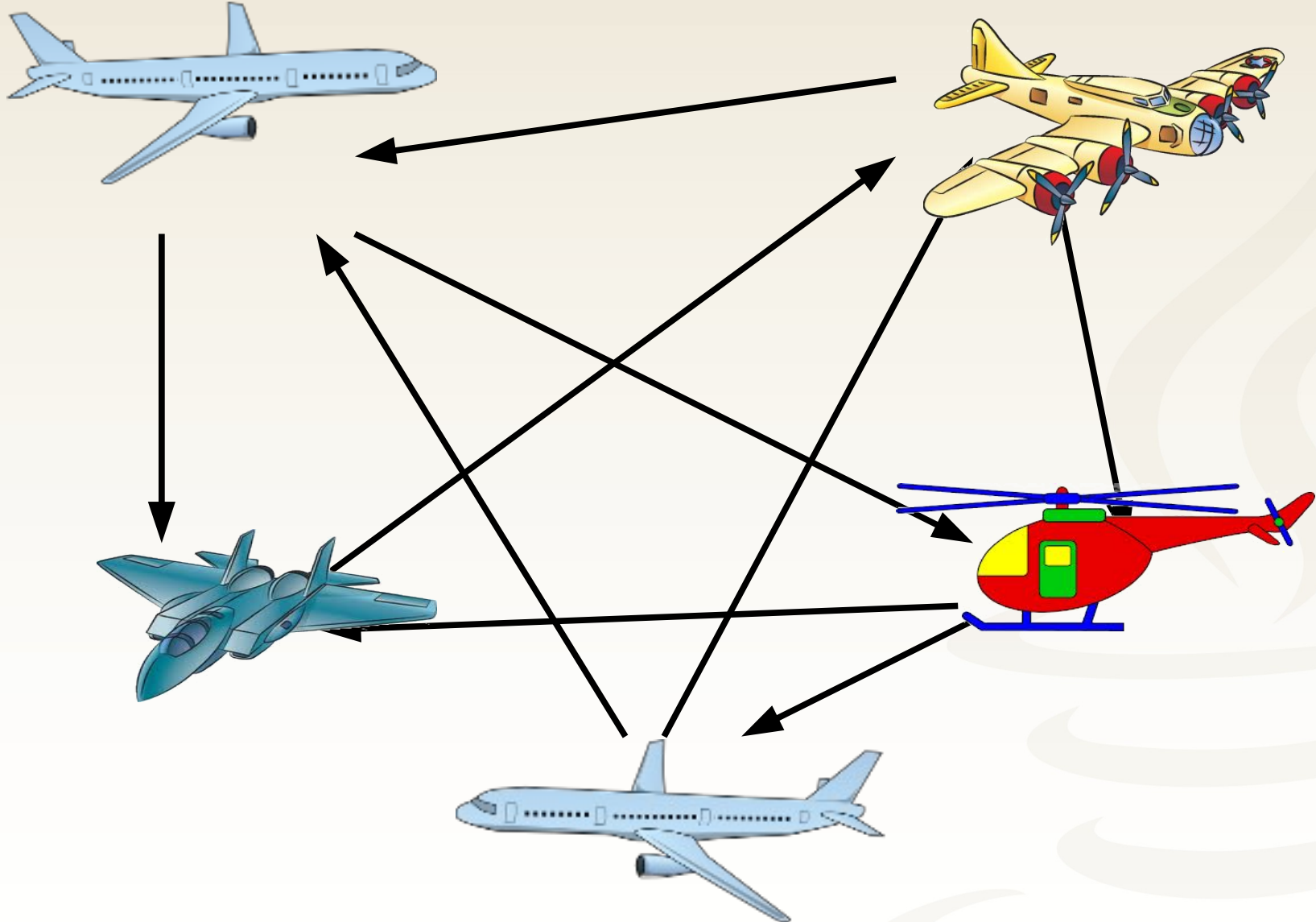
- Karmaşık bir ekranda UI bileşenleri ve Presenter'lar kolayca **birbirlerine bağımlı** hale gelebilmektedirler
- Bu durum kodu **karmaşık** hale getirmektedir
- Ayrıca UI bileşenlerinin ve Presenter'ların **farklı senaryo'larda yeniden kullanılmalarını** da engellemektedir
- Bu sorunlar MVP'nin **Mediator örüntüsü ile birlikte kullanılması** ile aşılabılır

# Mediator Öncesi

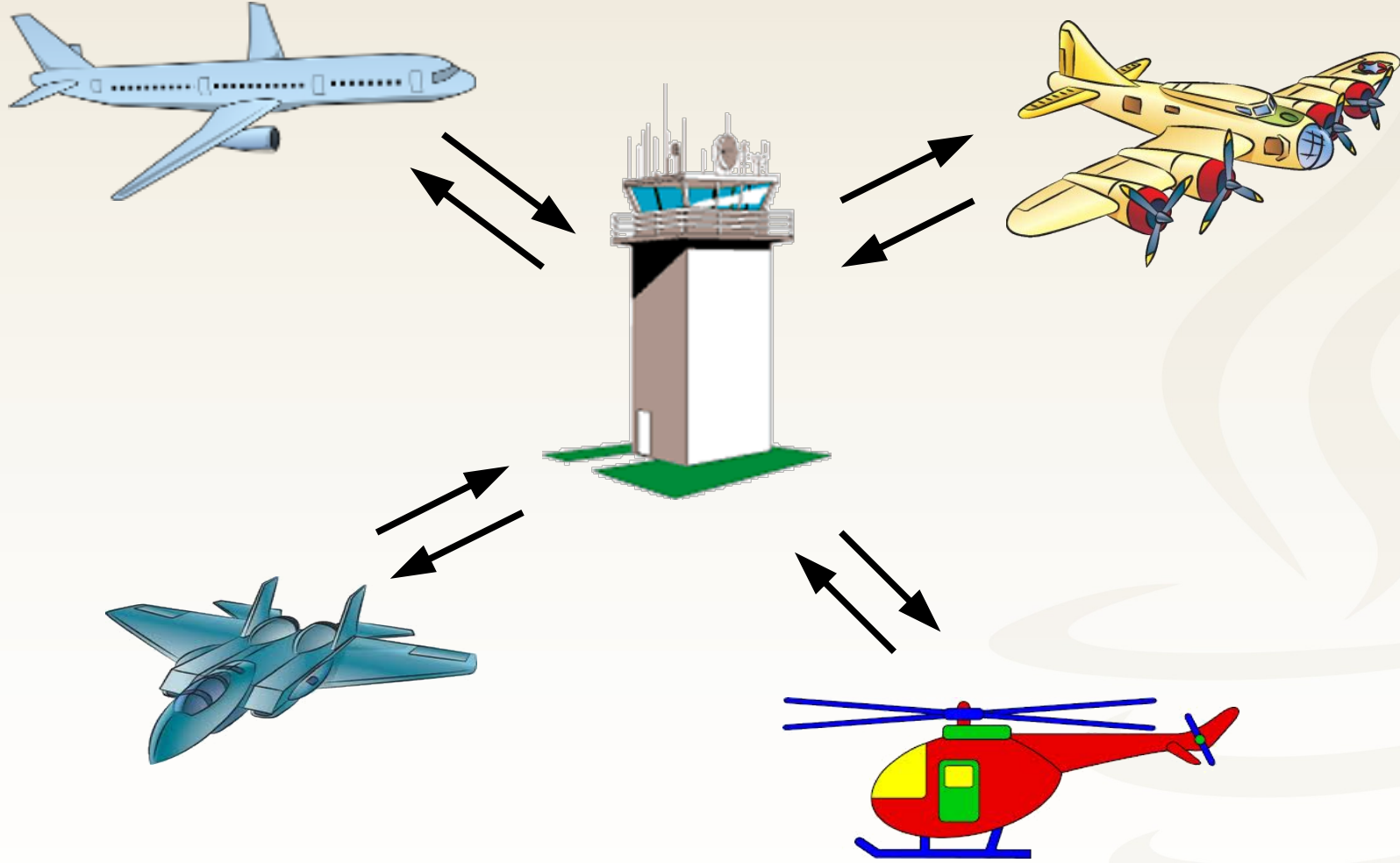
## Bileşenler Arası Etkileşim



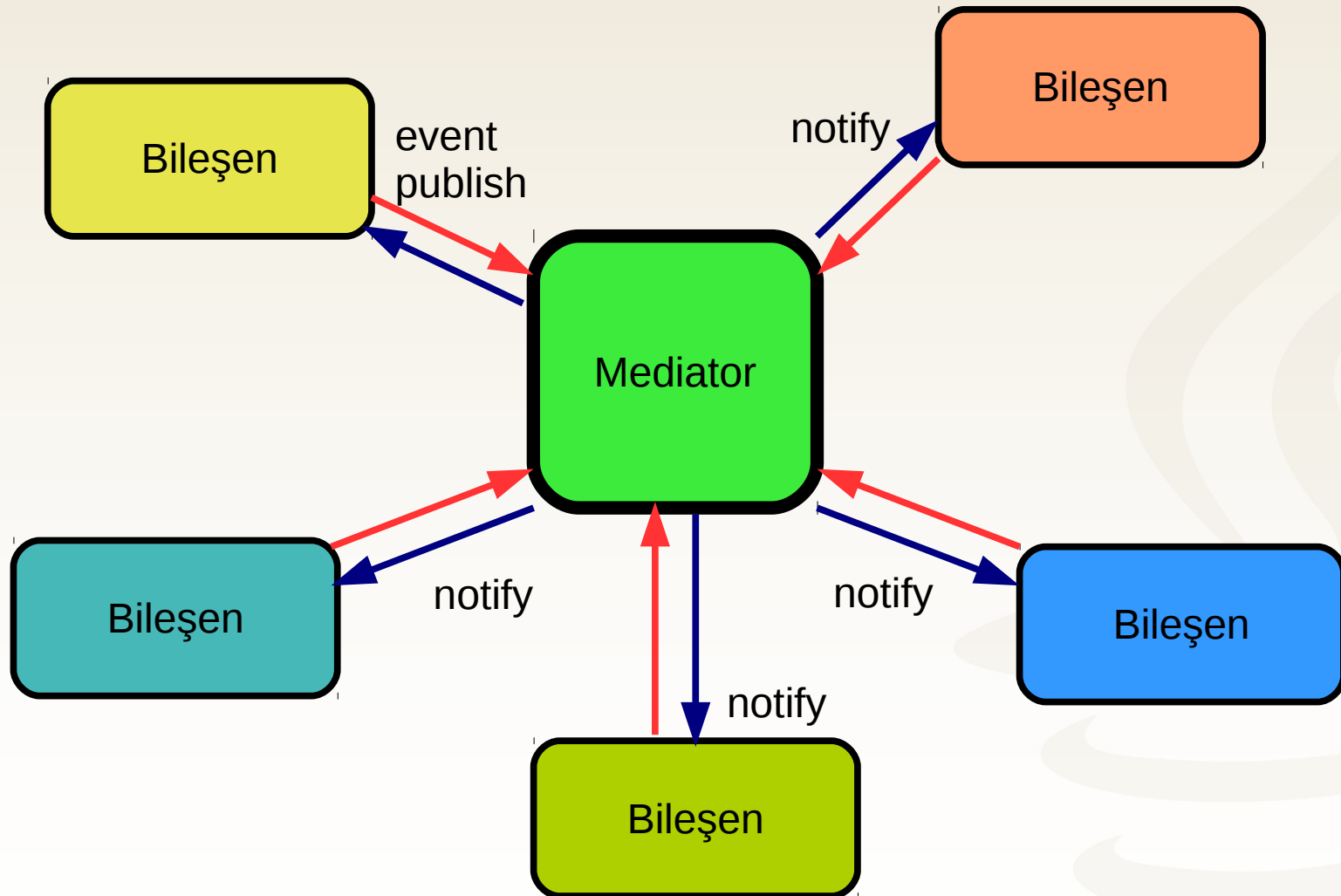
# Mediator Öncesi Bileşenler Arası Etkileşim



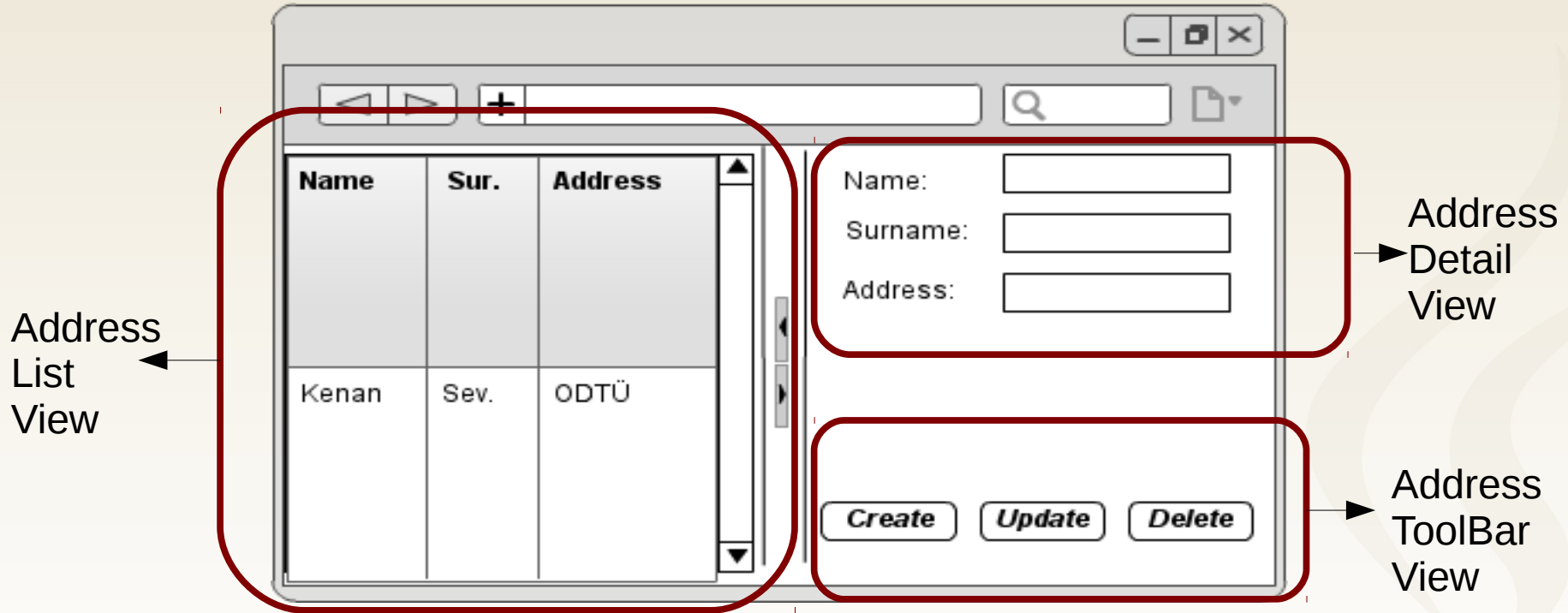
# Mediator Sonrası Bileşenler Arası Etkileşim



# Mediator Sonrası Bileşenler Arası Etkileşim



# LAB ÇALIŞMASI: MVP



Adres bilgilerinin yönetildiği bir ekran geliştirilecektir. ListView'da listelenen adreslerden birisi seçildiği vakit detayları DetailView'da görüntülenecek, ToolBarView'daki buton'larda buna göre aktif veya pasif olacaktır. Seçilen adres güncellenebilecek veya silinebilecek, buna göre değişiklikler ListView'a yansıtılacak, DetailView ve ToolBarView'da da gerekli değişiklikler yapılacaktır.

# MVP ve Mediator

The screenshot shows a web application in Mozilla Firefox. The browser address bar displays `http://localhost:8080/mvp/addressbook`. The application interface is divided into three main sections, each highlighted with a red rounded rectangle and labeled with an arrow:

- Address List View:** A table with three columns: NAME, SURNAME, and ADDRESS. It contains six rows of data.
- Address Detail View:** A form with three input fields labeled Name, Surname, and Address.
- Address ToolBar View:** A horizontal bar containing three buttons: Create, Update, and Delete.

Arrows indicate the relationships between these views: an arrow points from the Address List View to the Address Detail View, and another arrow points from the Address ToolBar View to the Address Detail View.

NAME	SURNAME	ADDRESS
Kenan	Sevindik	ODTÜ MET No:D-4/A Ankara
Muammer	Yücel	Ankara
Murat	Öksüzer	Adana
Ali	Güçlü	İstanbul
Veli	Saygılı	İzmir
Ahmet	Tok	Bursa

Name

Surname

Address

Create Update Delete

# Mediator.java

```
public class Mediator {
    private Collection<Presenter> listeners = new
        ArrayList<Presenter>();
    public void addListener(Presenter listener) {
        listeners.add(listener);
    }

    public void removeListener(Presenter listener) {
        listeners.remove(listener);
    }

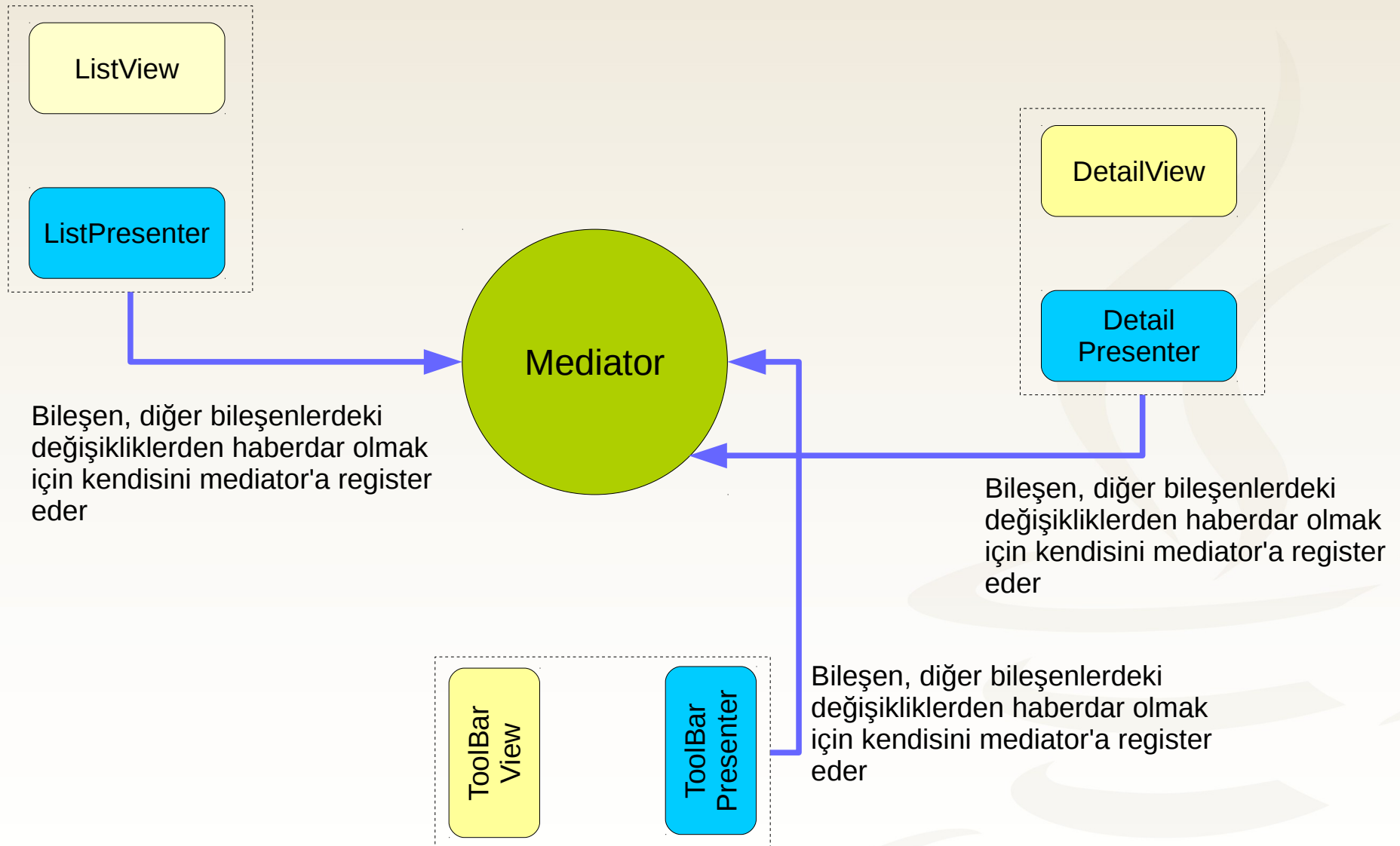
    public void publish(BusinessEvent event) {
        for(Presenter listener:listeners) {
            listener.handle(event);
        }
    }
}
```



# Presenter.java

```
public interface Presenter {  
    public void handle(BusinessEvent event);  
}
```

# Adım 1: Mediator Registration



# Address List Presenter

```
public class AddressListPresenter implements Presenter {

    private AddressListView view;

    public AddressListPresenter(AddressListView view,
        Mediator mediator) {

        this.view = view;
        mediator.addListener(this);
    }

    @Override
    public void handle(BusinessEvent event) {
        ...
    }
}
```

# Address Detail Presenter

```
public class AddressDetailPresenter implements Presenter {

    private AddressDetailView view;

    public AddressDetailPresenter(AddressDetailView view,
        Mediator mediator) {

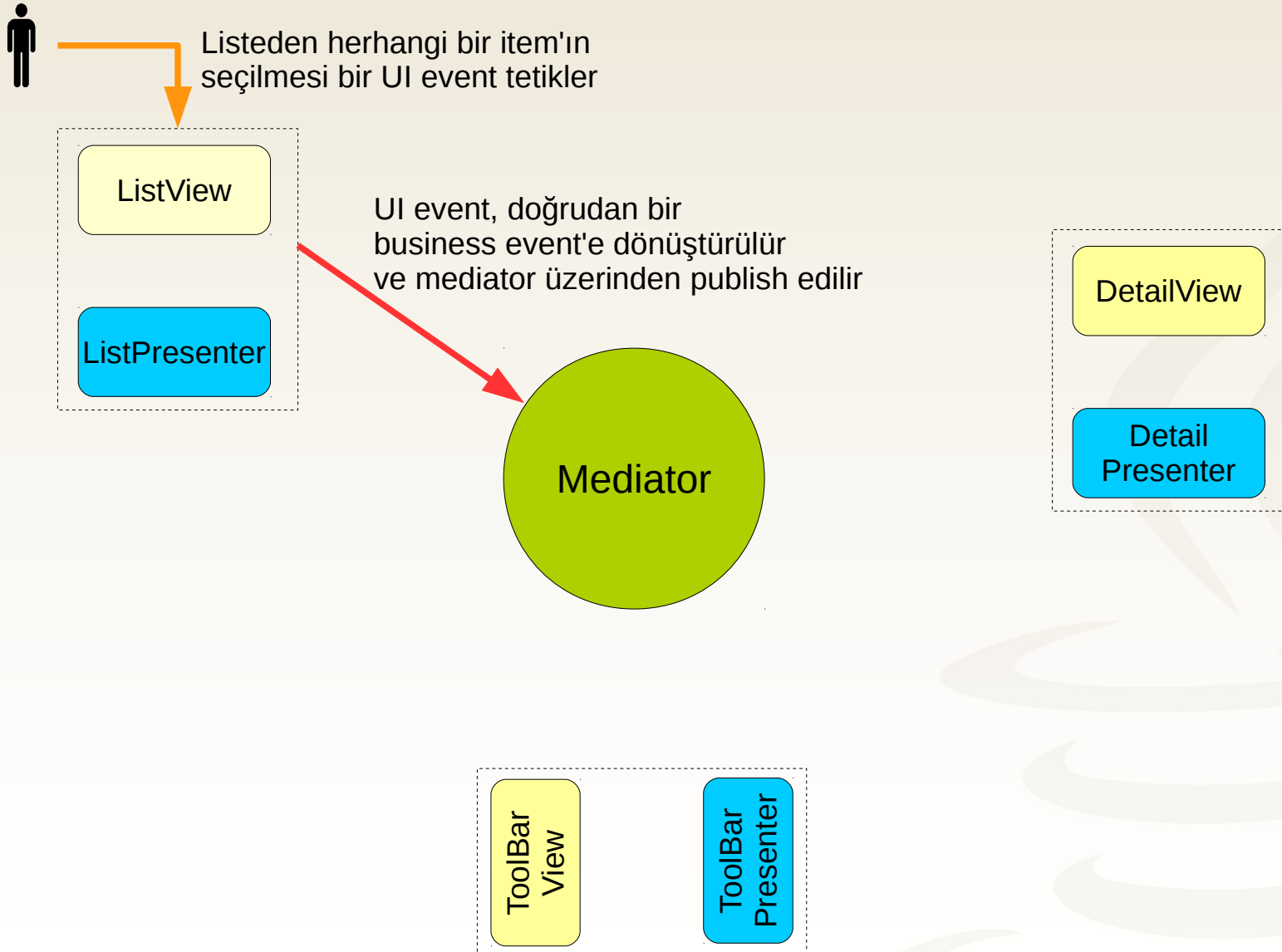
        this.view = view;
        mediator.addListener(this);
    }

    @Override
    public void handle(BusinessEvent event) {
        ...
    }
}
```

# Address ToolBar Presenter

```
public class AddressToolBarPresenter implements Presenter {  
  
    private AddressToolBarView view;  
  
    public AddressToolBarPresenter(AddressToolBarView view,  
        Mediator mediator) {  
  
        this.view = view;  
        mediator.addListener(this);  
    }  
  
    @Override  
    public void handle(BusinessEvent event) {  
        ...  
    }  
}
```

# Adım 2: UI Interaction (Item Select)



# Address List View

```
public class AddressListView implements ValueChangeListener {

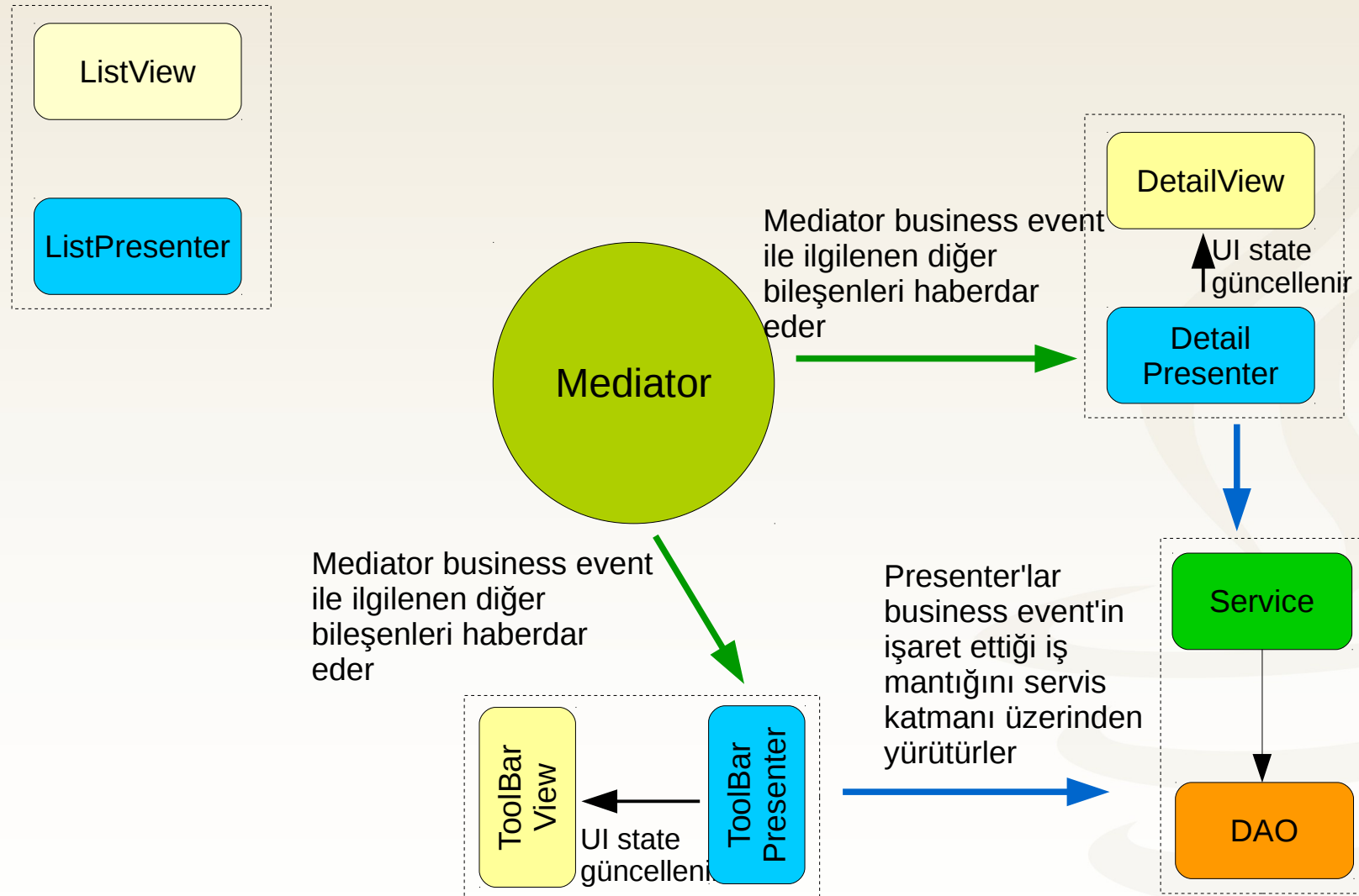
    public AddressListView(Mediator mediator) {
        this.mediator = mediator;
    }

    @Override
    public void valueChange(ValueChangeEvent event) {
        Address address = (Address) table.getValue();

        AddressSelectedEvent selectedEvent = new
            AddressSelectedEvent(address);
        mediator.publish(selectedEvent);
    }

    ...
}
```

# Adım 3: Event Notification (Address Selected)





# Address Detail Presenter

```
public class AddressDetailPresenter implements Presenter {
    @Override
    public void handle(BusinessEvent event) {
        if(event instanceof AddressSelectedEvent) {

            AddressSelectedEvent selectedEvent =
                (AddressSelectedEvent)event;

            Address address =
                selectedEvent.getSelectedAddress();

            view.displayAddress(address);
        }
    }
    ...
}
```

# Address ToolBar Presenter

```
public class AddressToolBarPresenter implements Presenter {
    @Override
    public void handle(BusinessEvent event) {
        if(event instanceof AddressSelectedEvent) {

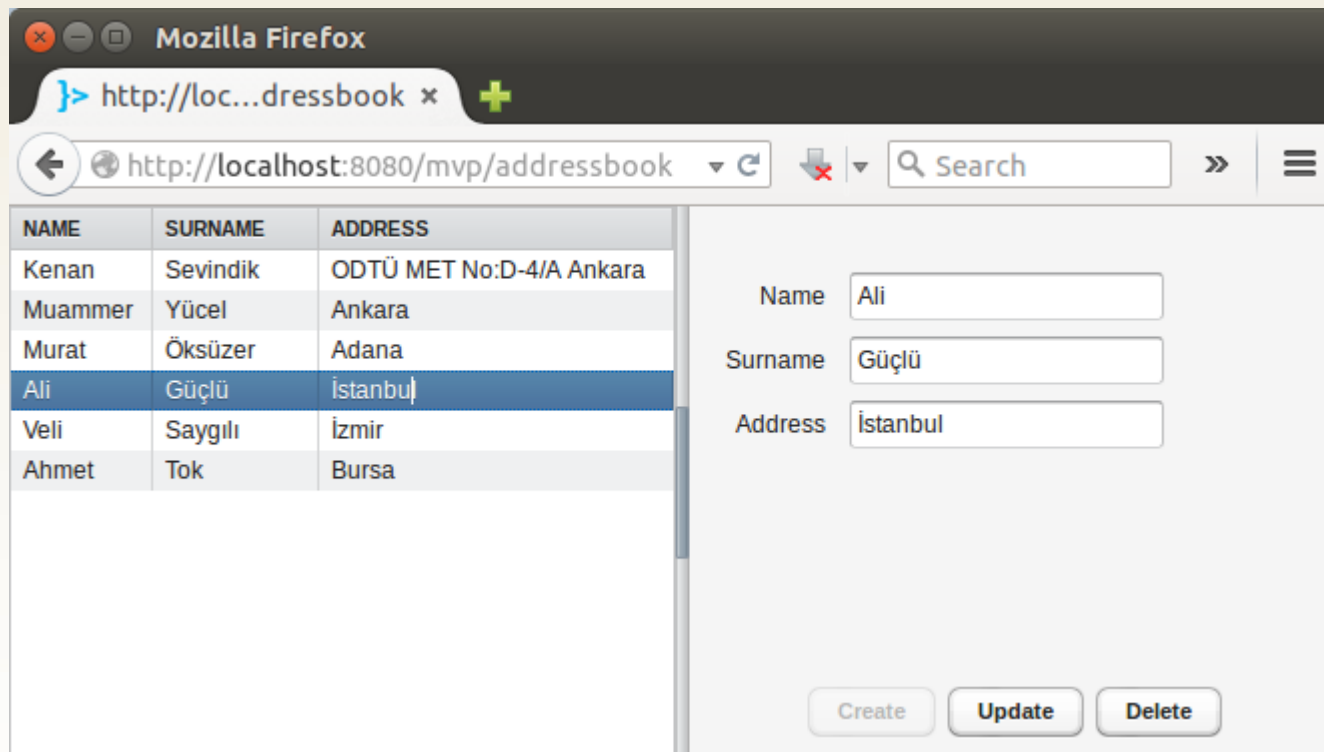
            AddressSelectedEvent selectedEvent =
                (AddressSelectedEvent)event;

            Address address =
                selectedEvent.getSelectedAddress();

            view.switchToUpdateMode();
            view.setAddress(address);

        }
        ...
    }
}
```

# Address Selected



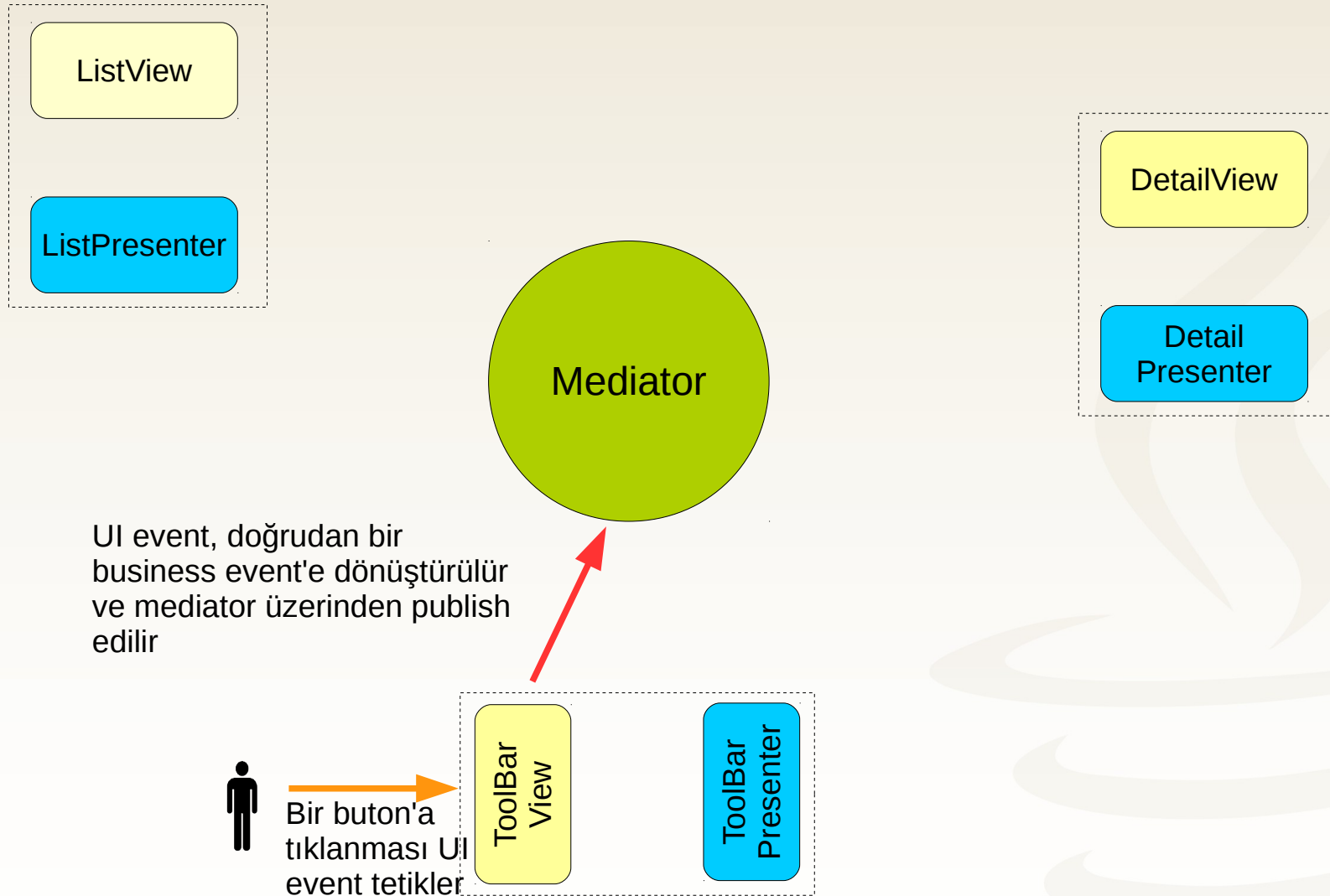
NAME	SURNAME	ADDRESS
Kenan	Sevindik	ODTÜ MET No:D-4/A Ankara
Muammer	Yücel	Ankara
Murat	Öksüzer	Adana
Ali	Güçlü	İstanbul
Veli	Saygılı	İzmir
Ahmet	Tok	Bursa

Name:

Surname:

Address:

# Adım 2: UI Interaction (Button Click)

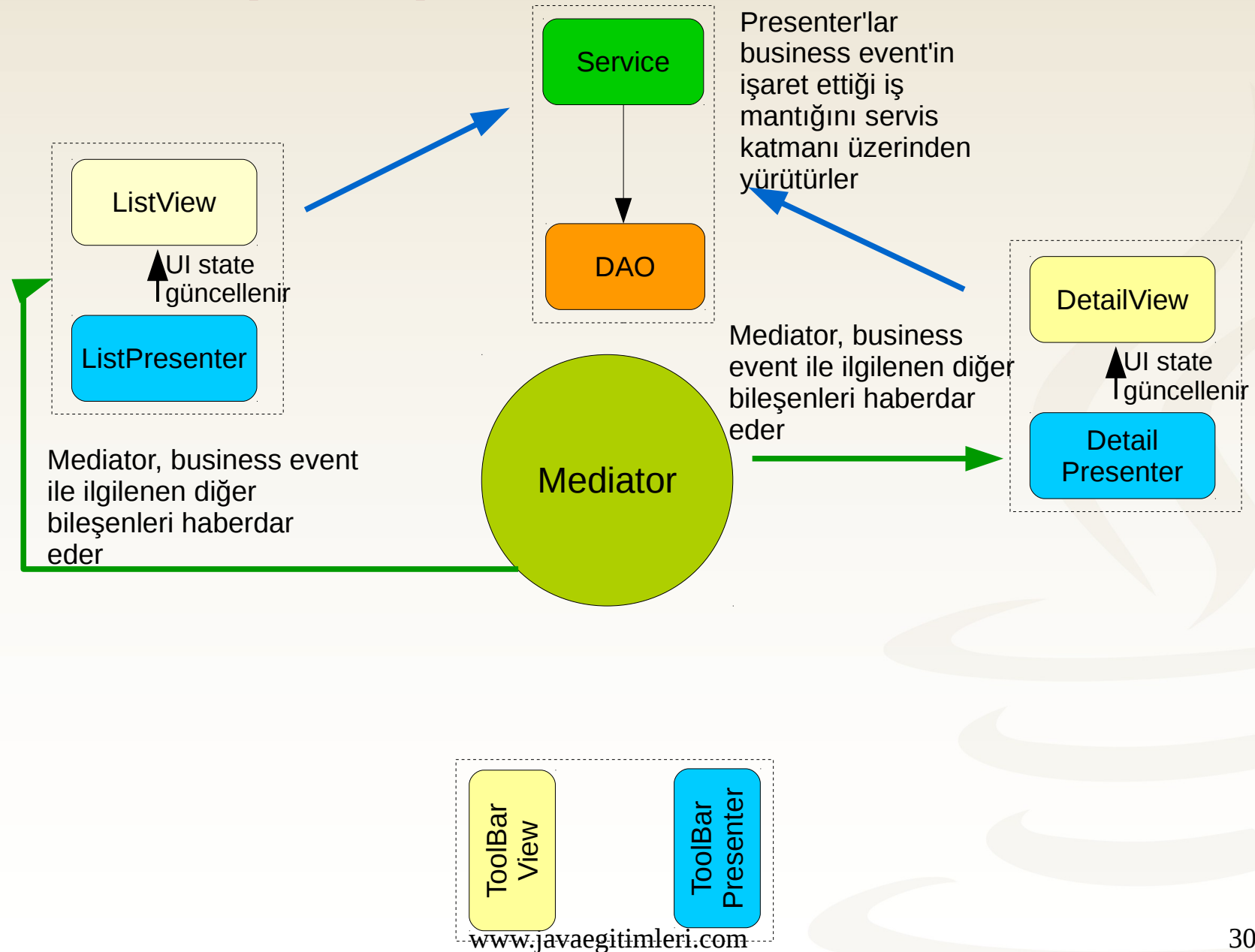


# Address ToolBar View

```
public class AddressToolBarView implements ClickListener {
    public AddressToolBarView(Mediator mediator) {
        this.mediator = mediator;
    }

    @Override
    public void buttonClick(ClickEvent event) {
        if(event.getButton() == updateButton) {
            AddressUpdateEvent updateEvent =
                new AddressUpdateEvent(address);
            mediator.publish(updateEvent);
        }
    }
    ...
}
```

# Adım 3:Event Notification (Address Update)



# Address List Presenter

```
public class AddressListPresenter implements Presenter {
    @Override
    public void handle(BusinessEvent event) {
        if(event instanceof AddressUpdateEvent) {

            AddressUpdateEvent updateEvent =
                (AddressUpdateEvent)event;

            Address address = updateEvent.getAddress();

            view.reloadAddress(address);

        }
        ...
    }
}
```

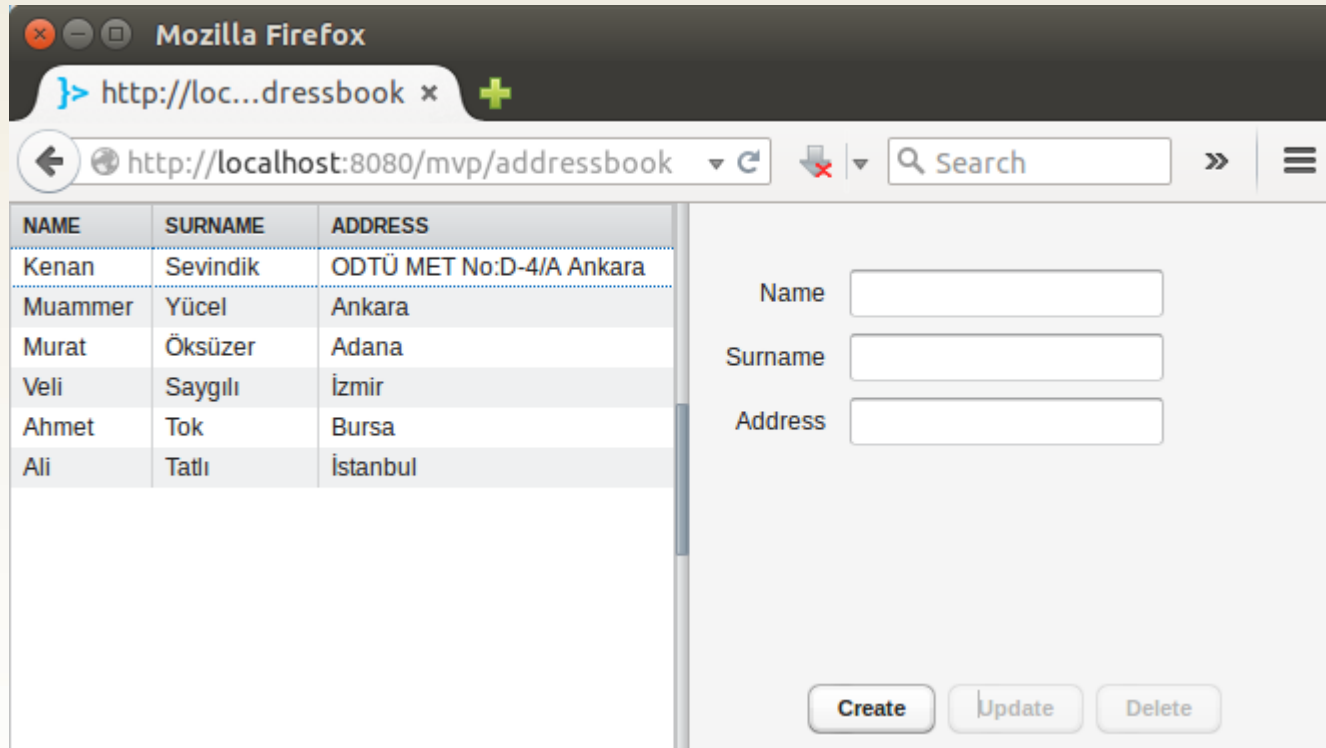
# Address ToolBar Presenter

```
public class AddressToolBarPresenter implements Presenter {
    @Override
    public void handle(BusinessEvent event) {
        if(event instanceof AddressSelectedEvent) {
            AddressSelectedEvent selectedEvent =
                (AddressSelectedEvent)event;
            Address address =
                selectedEvent.getSelectedAddress();
            view.switchToUpdateMode();
            view.setAddress(address);

        } else if(event instanceof AddressUpdateEvent) {
            view.switchToSelectionMode();
        }
    }
    ...
}
```



# Address Updated



Mozilla Firefox

http://loc...dressbook x

http://localhost:8080/mvp/addressbook

NAME	SURNAME	ADDRESS
Kenan	Sevindik	ODTÜ MET No:D-4/A Ankara
Muammer	Yücel	Ankara
Murat	Öksüzer	Adana
Veli	Saygılı	İzmir
Ahmet	Tok	Bursa
Ali	Tatlı	İstanbul

Name

Surname

Address

Create Update Delete

# İletişim



[www.harezmi.com.tr](http://www.harezmi.com.tr)

[www.java-egitimleri.com](http://www.java-egitimleri.com)



[info@harezmi.com.tr](mailto:info@harezmi.com.tr)

[info@java-egitimleri.com](mailto:info@java-egitimleri.com)



[@HarezmiBilisim](https://twitter.com/HarezmiBilisim)

[@JavaEgitimleri](https://twitter.com/JavaEgitimleri)