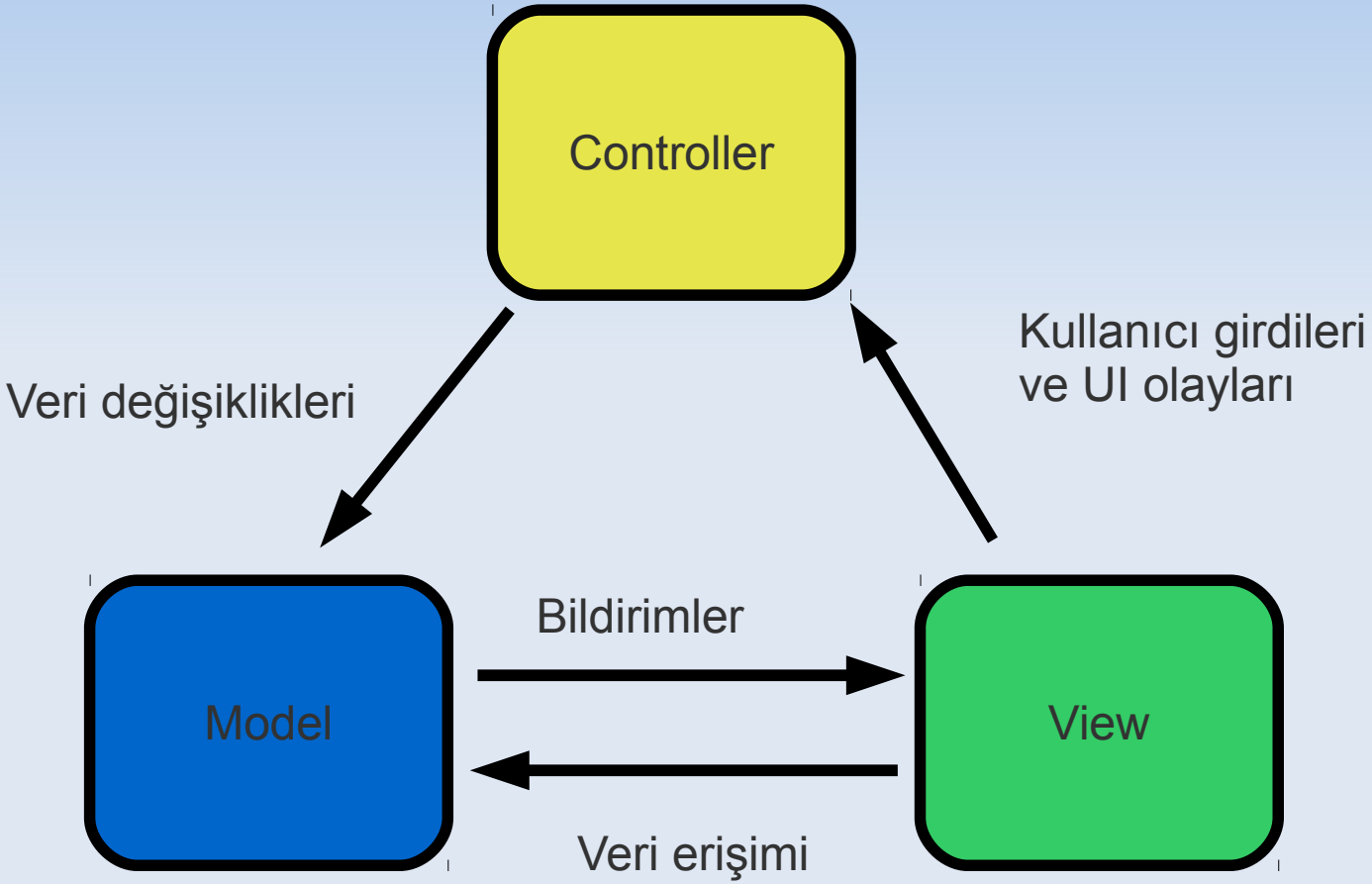
JAVA Eğitimleri

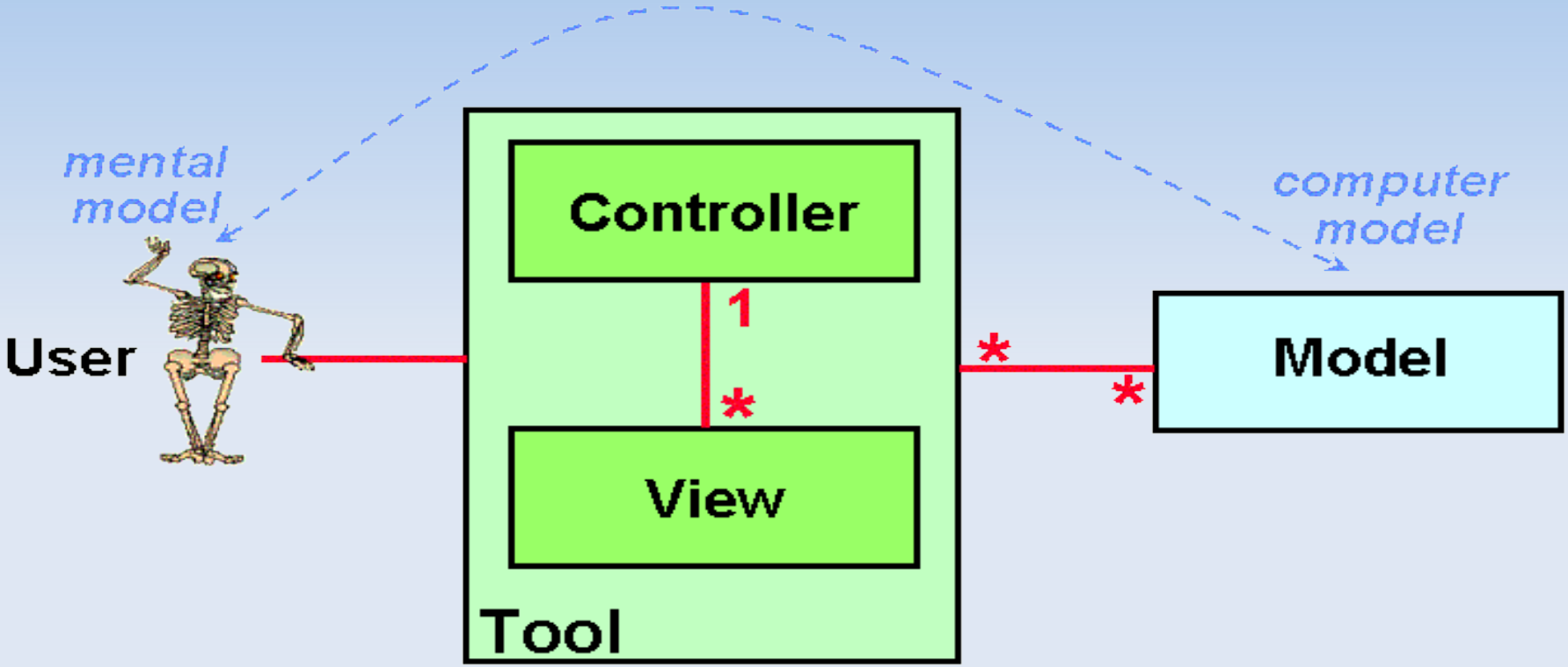
MVC, MVP ve Mediator ile TDD Tecrübeleri

Kenan SEVİNDİK

Mimarisel Bir Örüntü: MVC

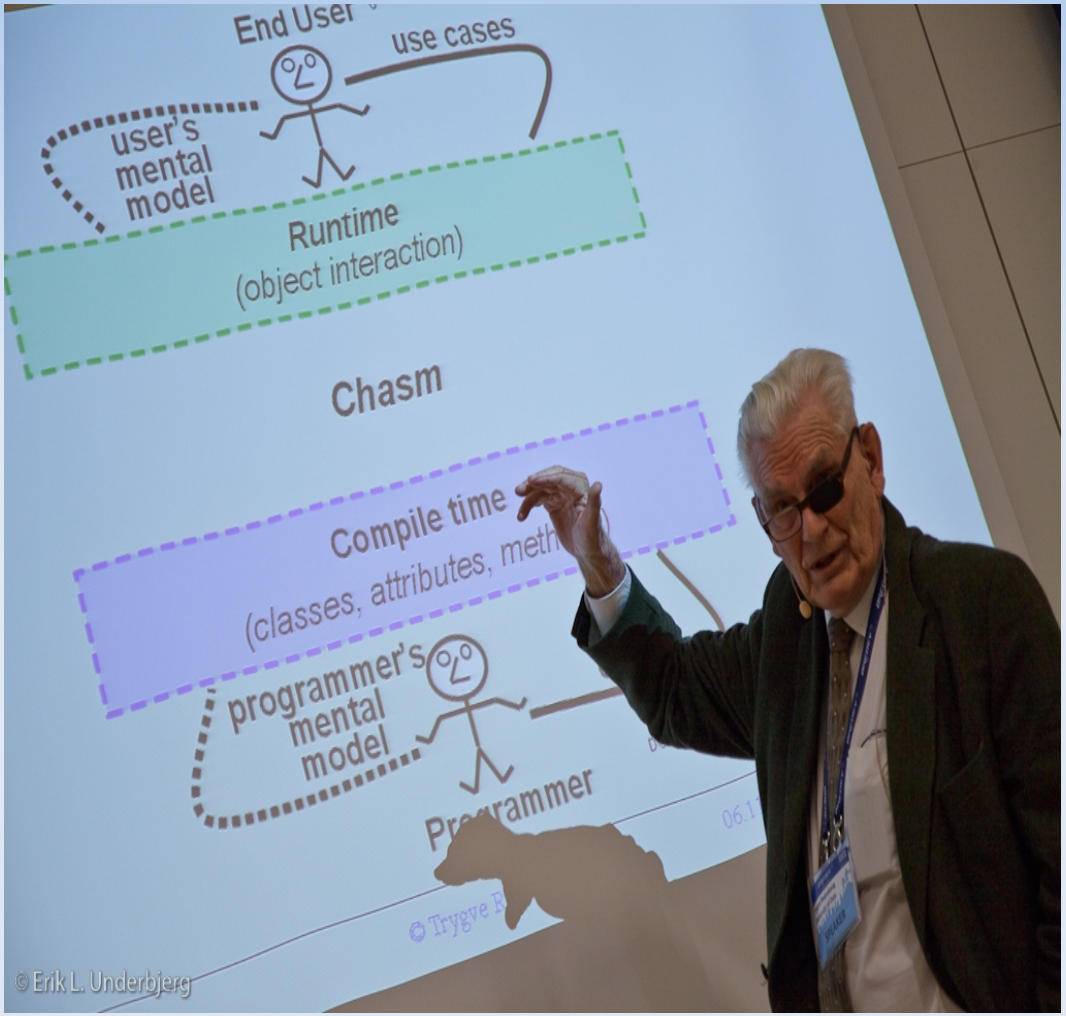


MVC'nin Amacı



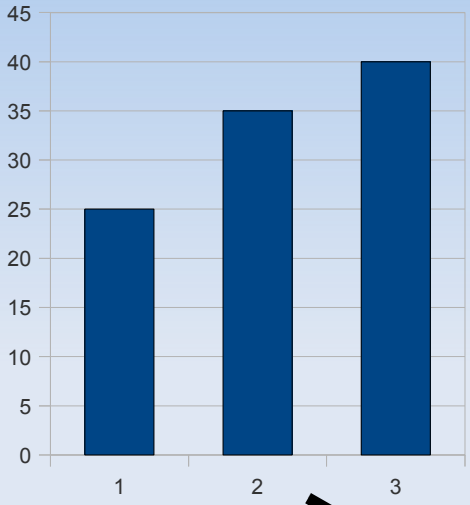
Trygve Reenskaug'un Hedefi Neydi...?

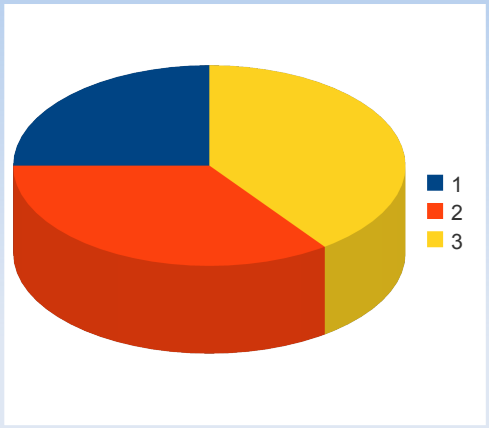
Trygve Reenskaug'un Hedefi

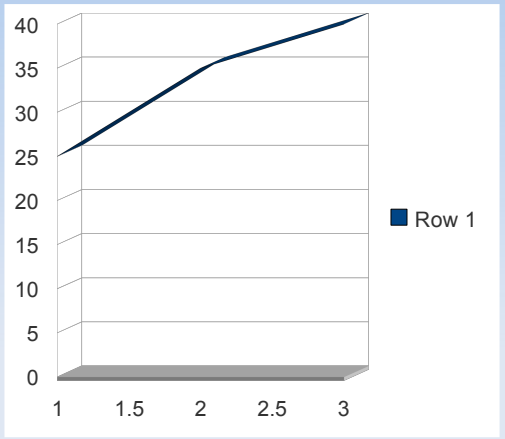


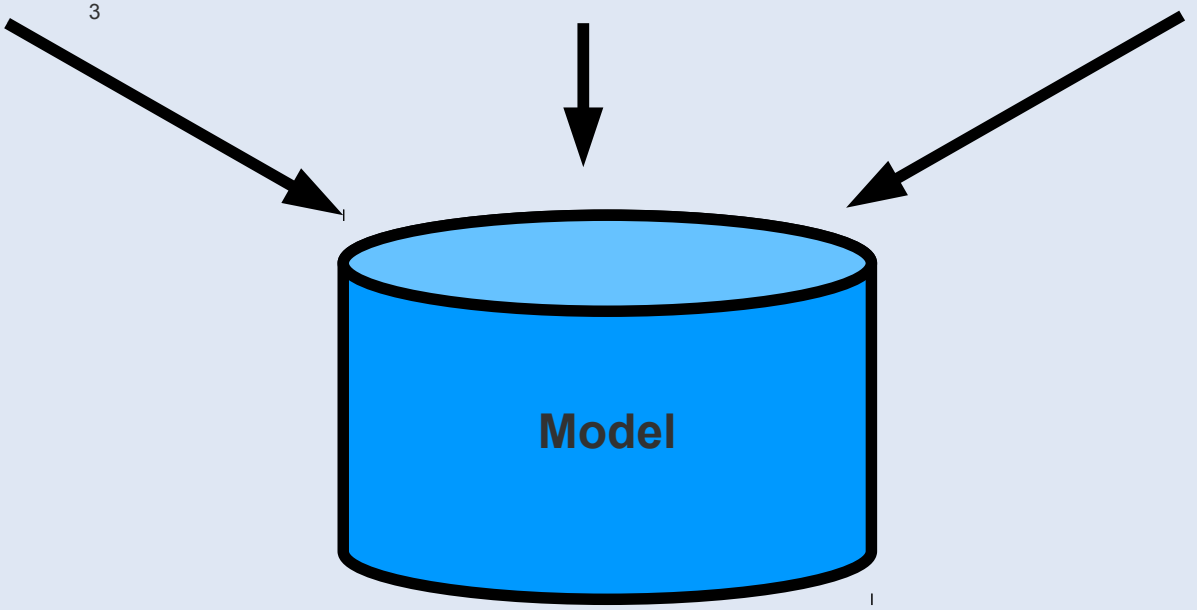
Kullanıcının zihnindeki "**mental model**" ile bilgisayar sistemindeki "**sayısal model**" arasındaki boşluğu doldurmaktır!

Trygve Reenskaug'un Hedefi







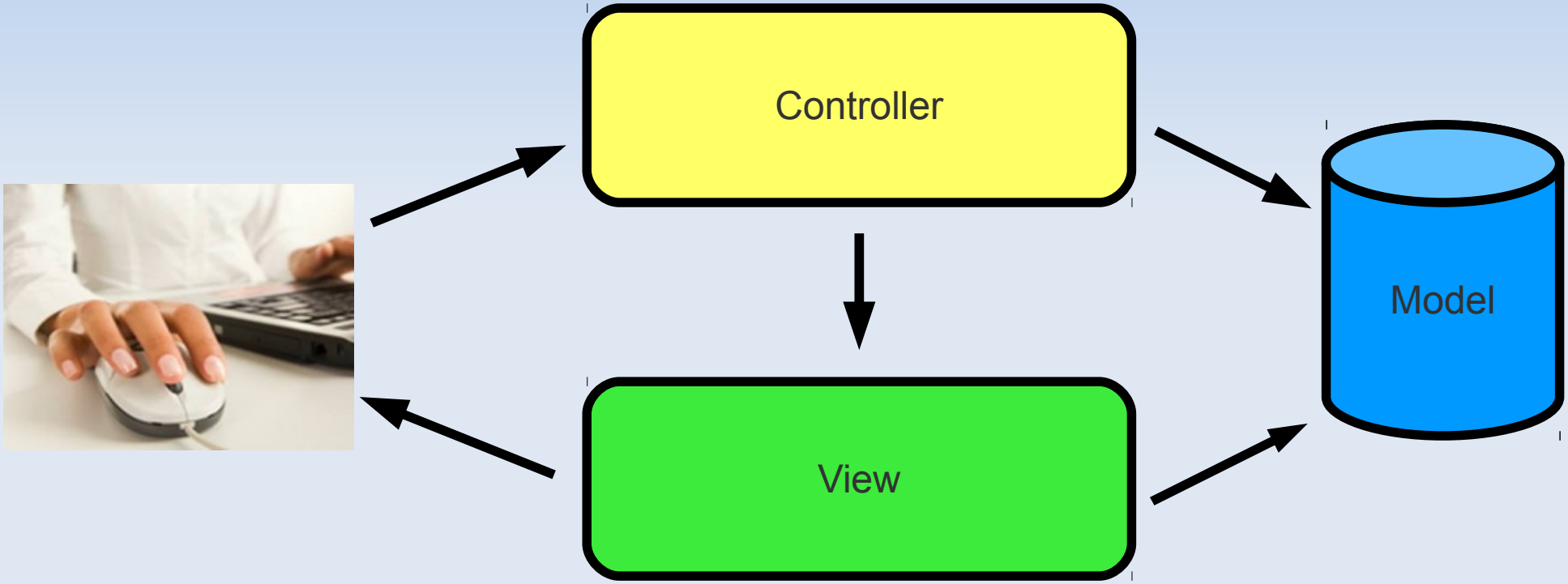


Günümüzdeki Kullanım Amacı:

“**Seperation of Concern**”

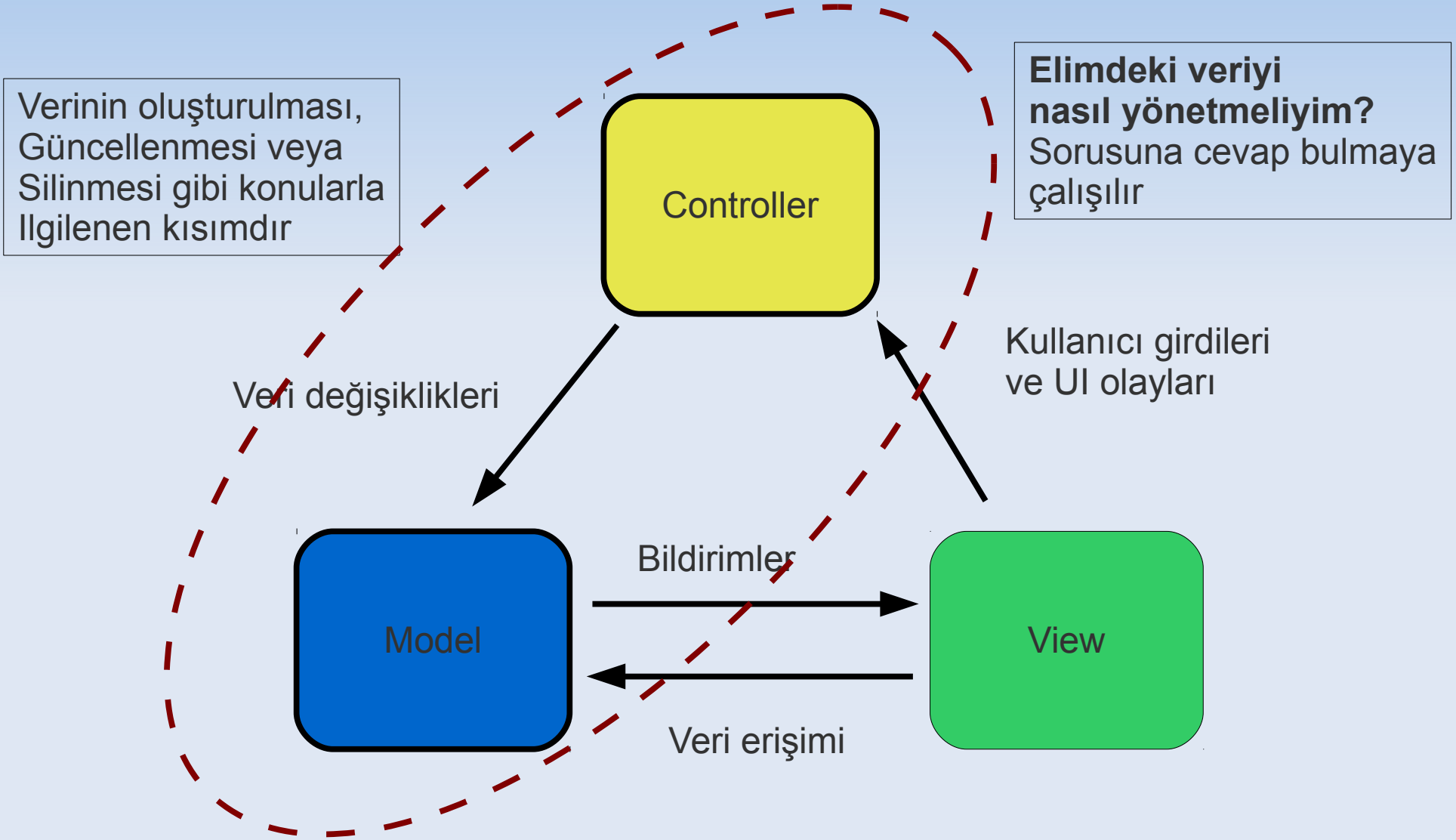
Reenskaug'un makalelerinde asıl amaç değildir, bir sonuçtur!

Günümüzdeki MVC Yorumlaması



MVP Sahneye...

MVC ve Veri Yönetimi



Elimdeki Veriyi Nasıl Yönetmeliyim?

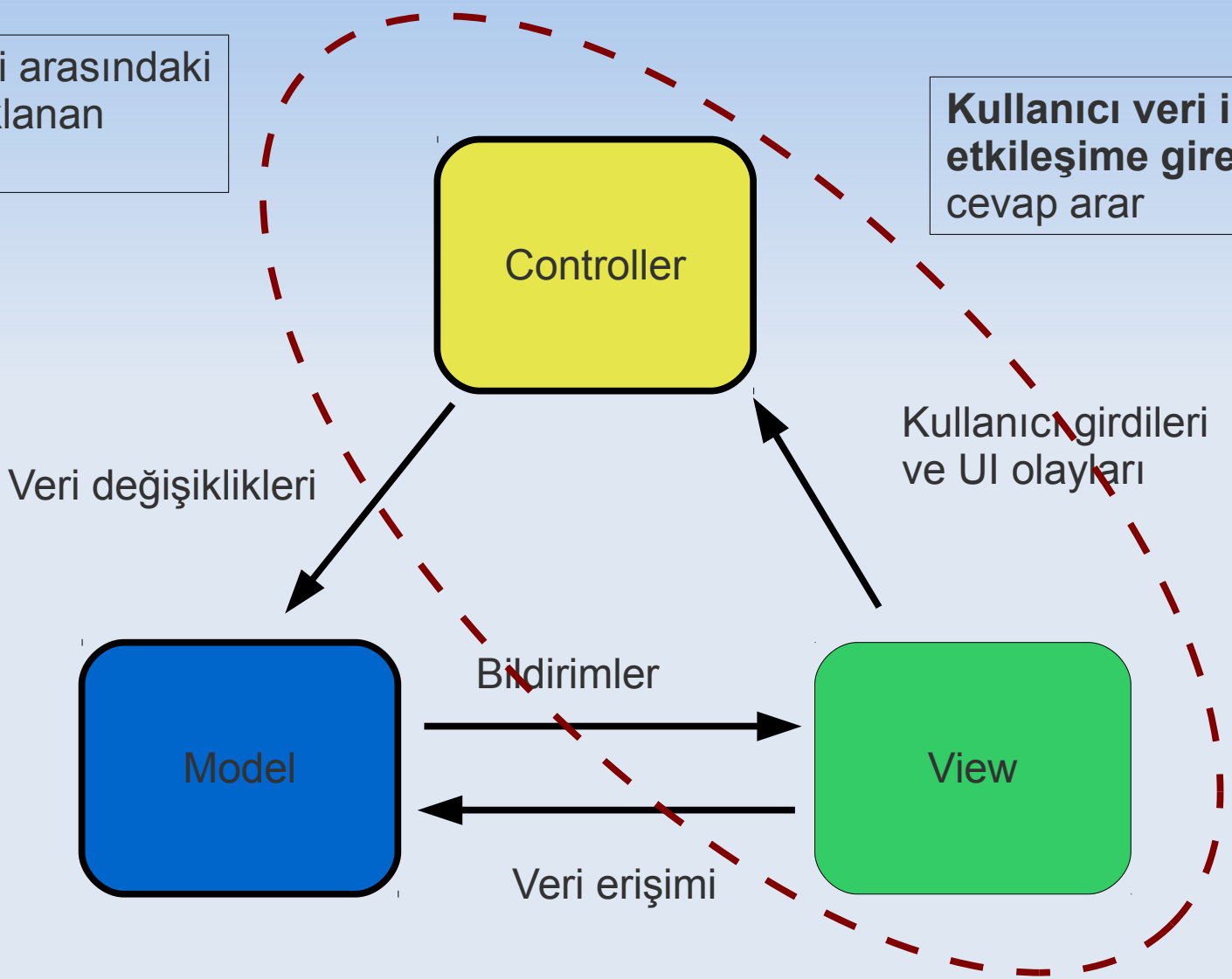
- **Elimdeki veri neden oluşur? Nedir?**
  - Encapsule edilmiş veri alanları, property'ler
- **Uygulama içerisinden veri nasıl seçilir?**
  - Metinsel seçimler
  - Satır ve sütun seçimleri
  - Bir grup elemanın veya bir bloğun seçilmesi
- **Veri nasıl güncellenmeli?**
  - CRUD işlemleri, taşıma, kopyalama, silme, yapıştırma...

MVC ve Kullanıcı Arayüzü Etkileşimi

JAVA Eğitimleri

Kullanıcı ile veri arasındaki etkileşime odaklanan kısımdır

Kullanıcı veri ile nasıl etkileşime girer? sorusuna cevap arar



Kullanıcı Veri İle Nasıl Etkileşime Girmeli?

- **Veri ekranda nasıl gösterilmeli?**
  - GUI oluşturma ve gösterme işlemi
- **Kullanıcı girdileri ve UI olaylarından veri güncelleme işlemlerine nasıl geçiş olmalı?**
  - Kullanıcı işlemleri, fare ve klayve girdileri

Model View Presenter

UI event'leri uygulamaya özel event'lere dönüştürülür

Presenter

UI üzerindeki değişiklikler Presenter tarafından yansıtılır

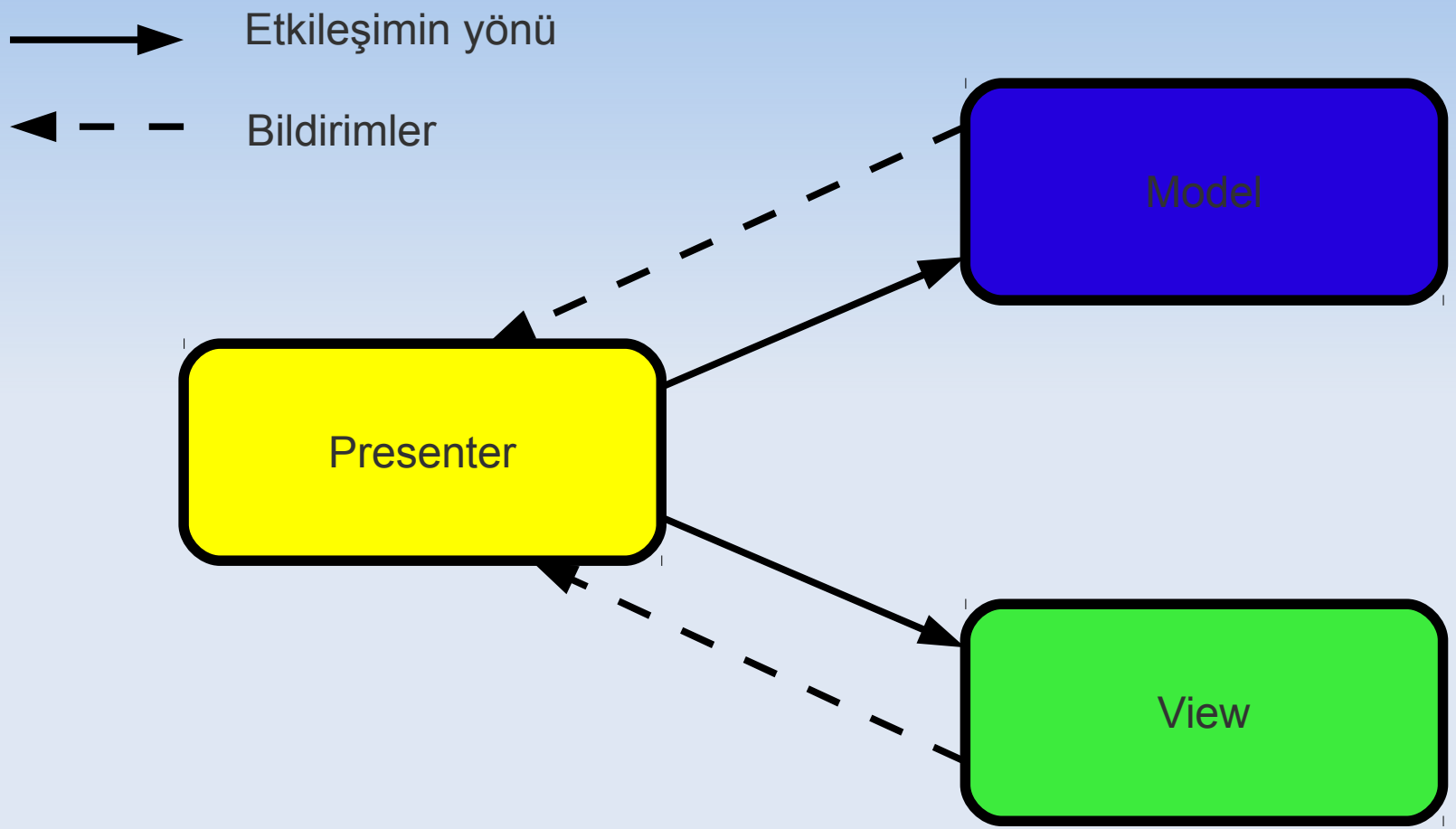
Presenter Model üzerinde Değişiklik Yapabilir Model verisine erişebilir

Model

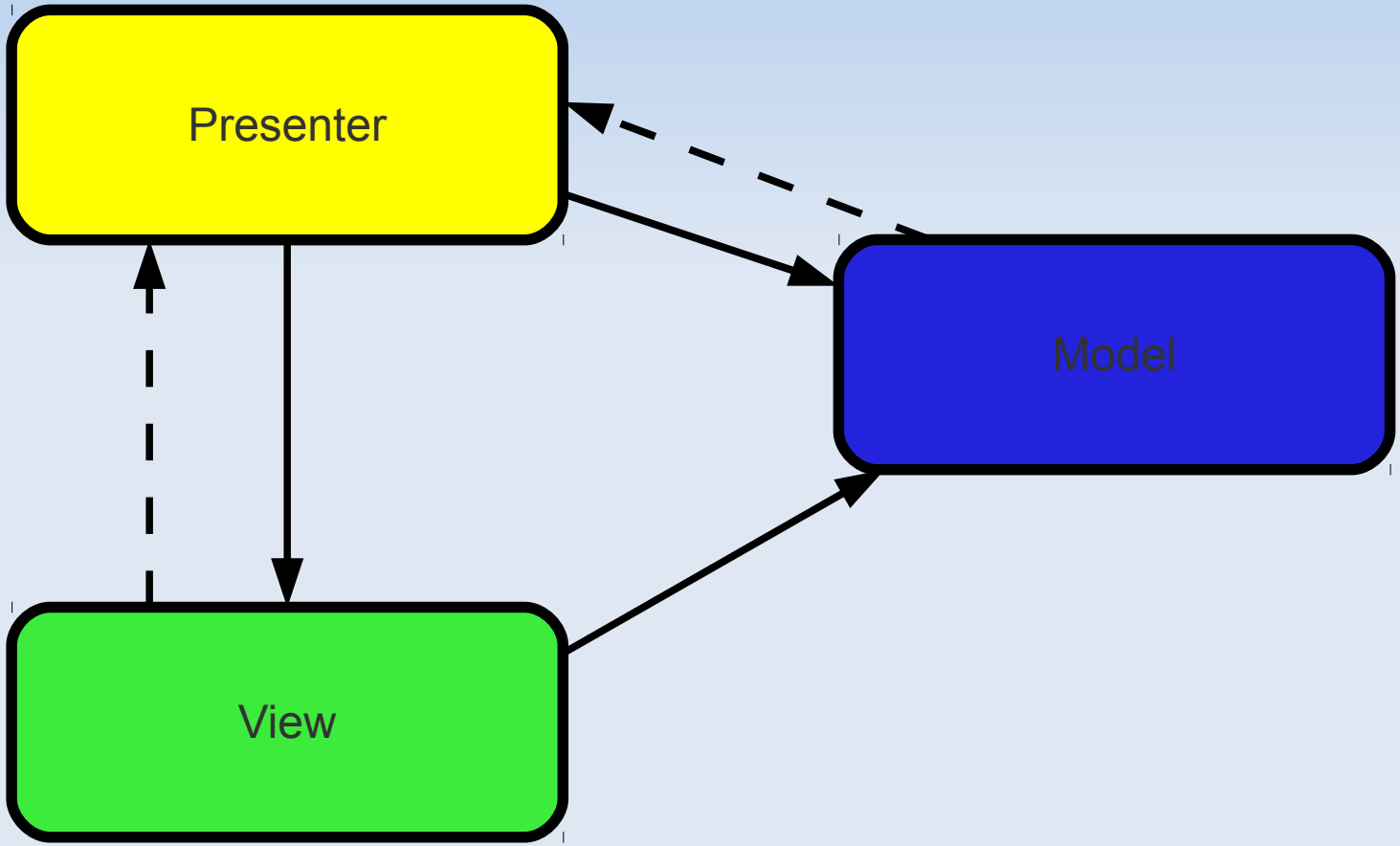
Model üzerindeki Değişiklikler Event'ler Ile Presenter'a iletilir

MVP Türevleri...

Passive View



Supervising Controller



Nereden Başlamalı?

Nasıl Kodlamalı?

Modelden?

Ekranlardan?

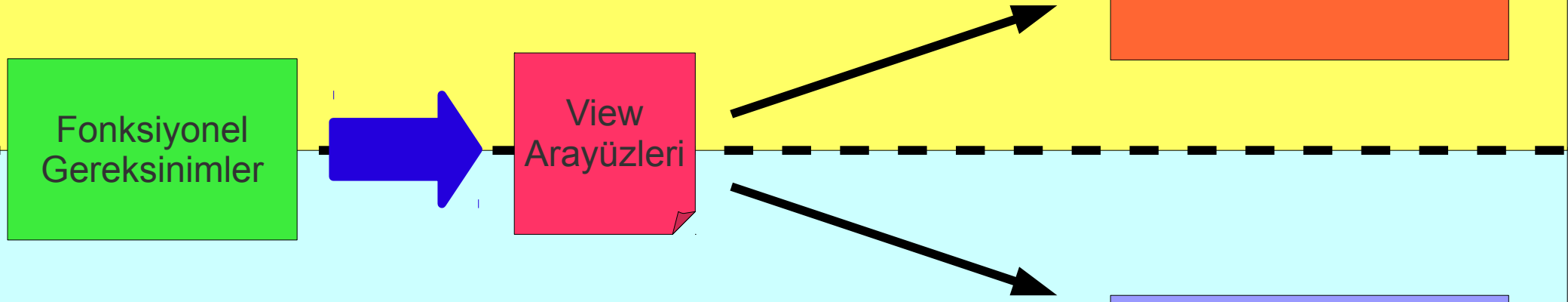
Önce Presenter

Çünkü...
kullanıcı senaryoları ve gereksinimler bire bir Presenter içindeki fonksiyonlara karşılık gelmektedir

Çünkü...

Geliştiriciler Presenter kısımlarını kodlamaya odaklanabilirler

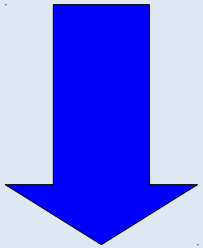
Kendi içlerinde de fonksiyonel gereksinimlere göre gruplara ayrılarak paralel çalışabilirler



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

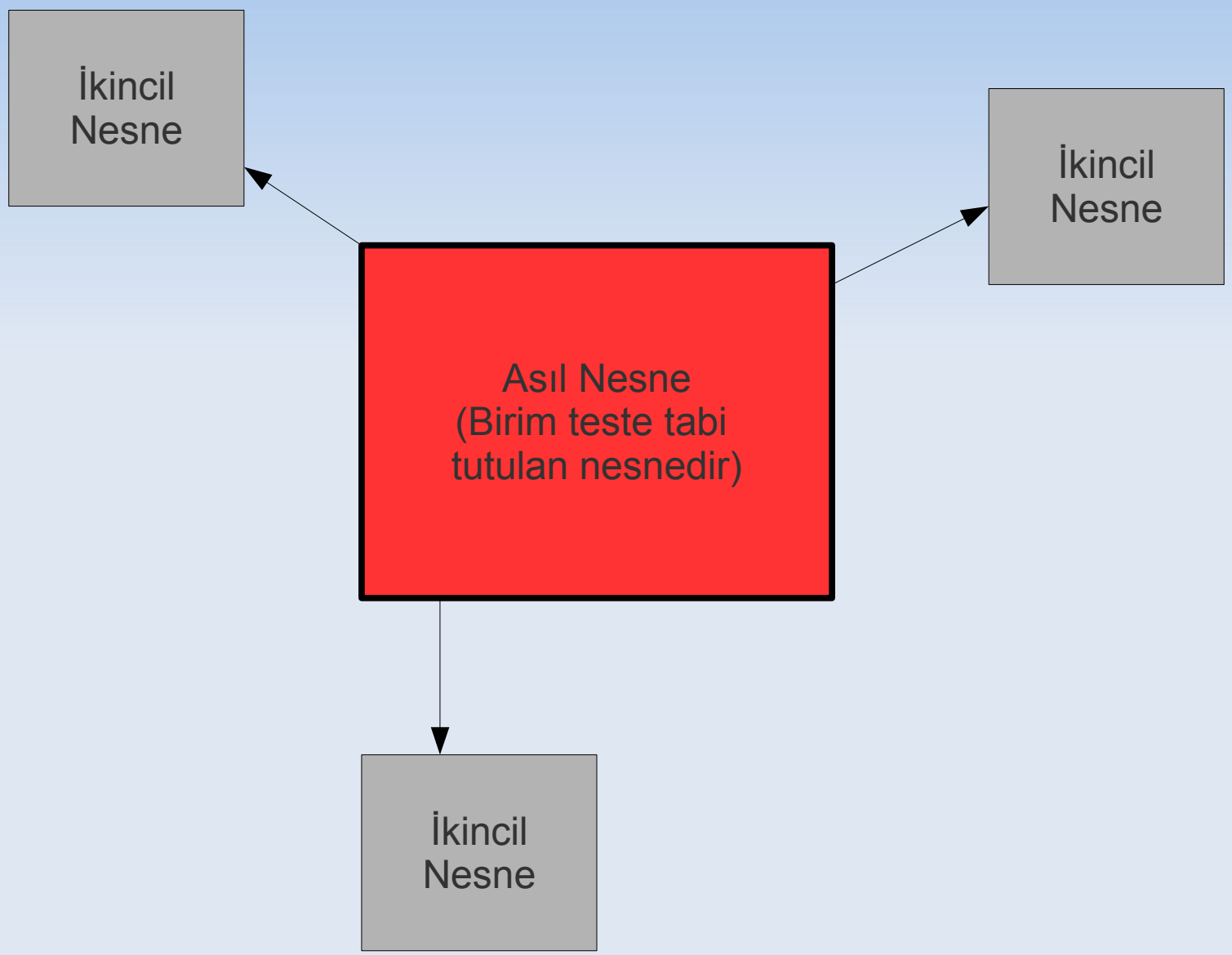
Çünkü...

Geliştirme sürecinde **TDD** **yaklaşımına uygun** çalışmaya olanak sağlamaktadır



View, Model ve ihtiyaç duyulan servis bileşenleri mock'lanarak Presenter'a verilir

Presenter ve TDD



İkincil Nesneler ve Mock İşlemi

- Gerçek implementasyonları hazır olmayabilir
- Hazır olsa bile test ortamında yaratılması çalıştırılması zor olabilir
- Ya da çok yavaş çalışabilir, network veya dosya sistemi ile ilişkisi olabilir
- GUI bağlantısı söz konusu olabilir
- Bu ve benzeri nedenlerle ikincil nesnelerin **asılları yerine sahteleri** kullanılır
- Bunlara “**mock**” nesneler adı verilir

Durum / Etkileşim Tabanlı Test Yaklaşımları

Etkileşim Tabanlı Yaklaşım

- **Mock** nesneler üzerinde, **test edilen davranışla ilgili metotlar**ın uygun sayıda ve şekilde asıl nesne tarafından çağırıldığını kontrol eden birim test yaklaşımıdır

Durum Tabanlı Yaklaşım

- Birincil ve ikincil nesnelerin ilgili davranış sonrasında doğru **state değerleri**ni yansıtıp yansıtmadıklarını kontrol eden birim test yaklaşımıdır
- İkincil nesneler olarak **asılları** kullanılır

Örnek 1

- Örneğin kullanıcımızın, text alana girilen bir metin içindeki Türkçe karakterleri anında İngilizce'ye dönüştüren bir program talep ettiğini farz edelim.

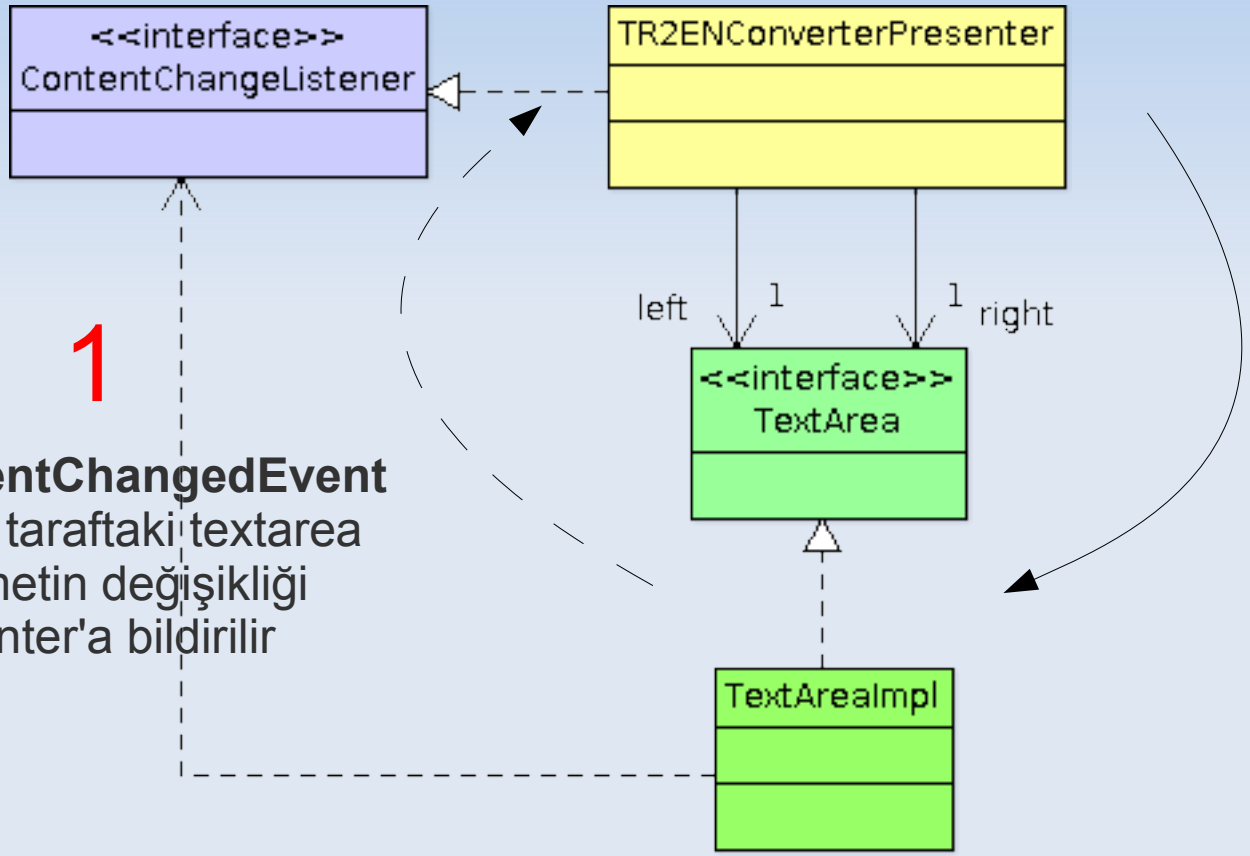
UI Mockup

Kullanıcı ile diyalogların sonucunda kullanıcı arayüzü taslak olarak ortaya çıkar. Kullanıcının ihtiyaç duyduğu, olmasını istediği fonksiyonaliteler, bu fonksiyonları UI üzerinden nasıl tetikleyeceği gibi konular kullanım durumu senaryolarının oluşturulması esnasında tespit edilir.

Sınıf ve Arayüzler

2

Metin içindeki karakterlerin dönüşümü Presenter tarafından yönetilir



1

ContentChangedEvent ile sol taraftaki textarea daki metin değişikliği Presenter'a bildirilir

3

TextArea.setContent(...) metodu ile sağ taraftaki textarea'nın içeriği güncellenir

Birim Testleri


```
@Mock
private TextArea left;

@Mock
private TextArea right;

@Mock
private ContentChangedEvent event;
```

Action kısmı

Event kısmı

```
@Test
public void contentShouldBeConvertedWhenTextChanged() {
   Mockito.when(event.getText()).thenReturn("çÇöÖşŞıİğĞüÜ");

   TR2ENConverterPresenter presenter = new
            TR2ENConverterPresenter(left, right);

   presenter.contentChanged(event);

   Mockito.verify(right).setContent("cCoOsSiIgGuU");
}
```

Presenter Implementasyonu


```
private TextArea left, right;

public TR2ENConverterPresenter(TextArea left, TextArea right) {
   this.left = left;
   this.right = right;

   left.addContentChangeListener(this);
}

@Override
public void contentChanged(ContentChangedEvent event) {
   String content = event.getText();

   content = content.replace('ç', 'c').replace('Ç',
         'C').replace('ı', 'i')
         .replace('İ', 'I').replace('ö', 'o').replace('Ö', 'O')
         .replace('ş', 's').replace('Ş', 'S').replace('ğ', 'g')
         .replace('Ğ', 'G').replace('ü', 'u').replace('Ü', 'U');

   right.setContent(content);
}
```

View Implementasyonu


```
public void addContentChangeListener(ContentChangeListener
listener) {
   listeners.add(listener);
}
```

Presenter event listener olarak kendini register etmek için bu metodu kullanır

```
public String getContent() {
   return (String) textArea.getValue();
}

public void setContent(String content) {
   textArea.setValue(content);
}
```

UI state'e erişime ve değiştirmeye imkan sağlayan metotlar

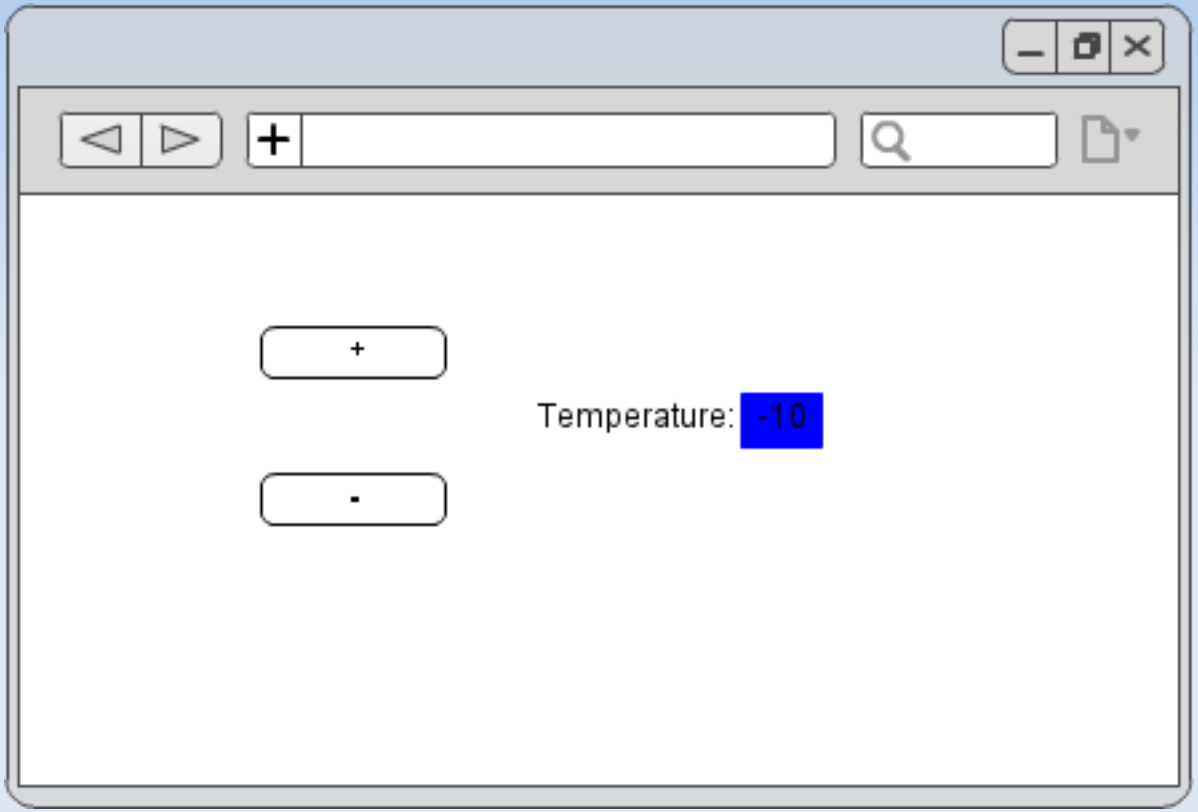
UI event'in uygulama event'ine dönüşümü

```
public void textChange(TextChangeEvent event) {
   ContentChangedEvent contentChangedEvent = new
                  ContentChangedEvent(event.getText());
   for(ContentChangeListener listener:listeners) {
      listener.contentChanged(contentChangedEvent);
   }
}
```

Örnek 2

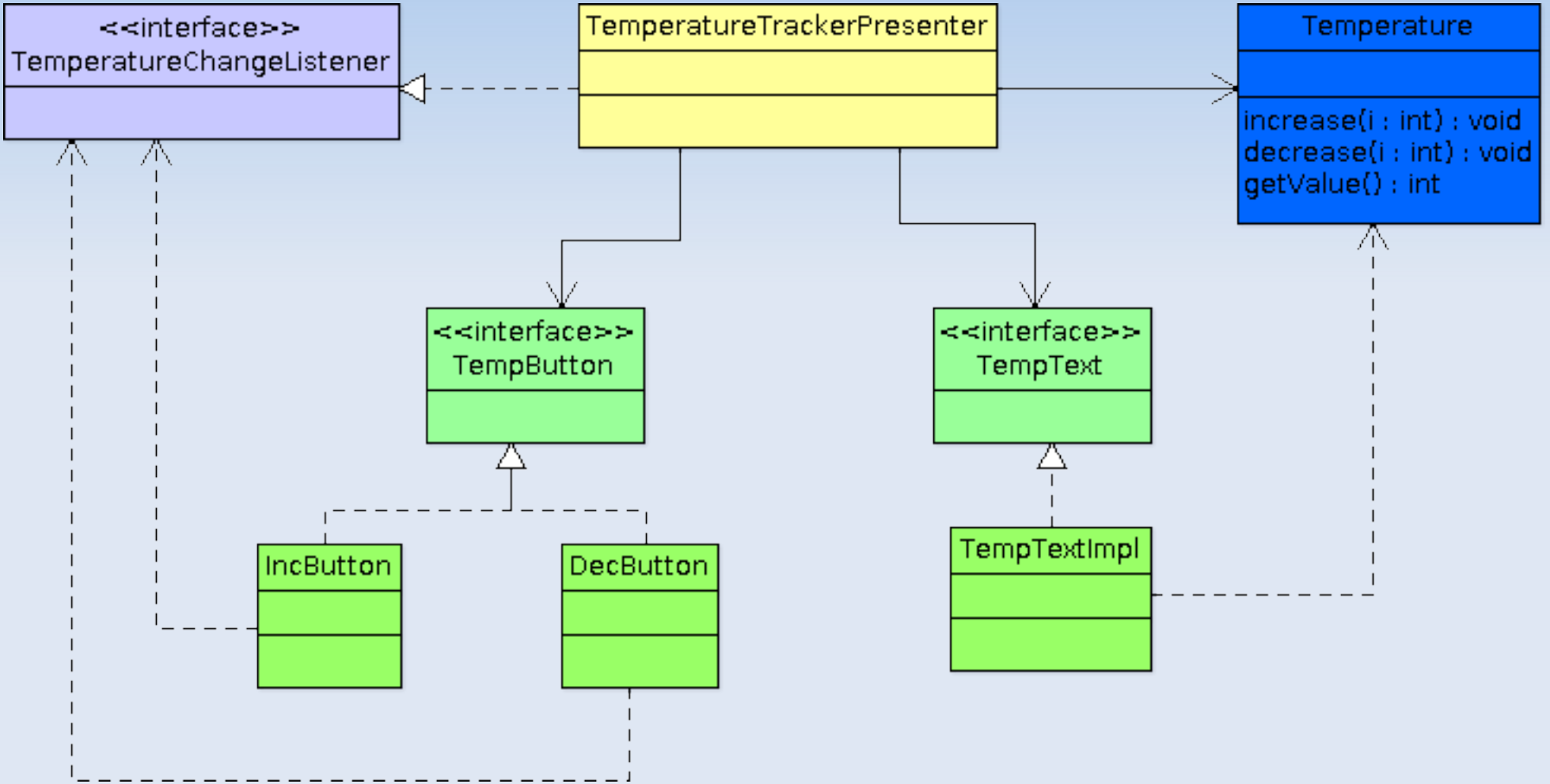
- Bu örnekte de kullanıcımız sıcaklık değerini birer birer artırıp düşürdüğü bir gösterge istiyor. Sıcaklığa göre göstergedeki değerin arka planı dinamik olarak renk değiştirmeli.

UI Mockup



Kullanıcı + ve – butonlarına tıklayarak sıcaklık değerini birer birer artırıp düşürebilecek. Sıcaklık değeri 0'ın altında iken arka plan mavi, sıfırın üstünde yeşil, 20 derece'nin üstünde sarı, 40 derecenin üstünde de kırmızı olmalı.

Sınıf ve Arayüzler



Birim Testleri


```
@Mock
private TempButton incButton;

@Mock
private TempButton decButton;

@Mock
private TempText tempText;

@Mock
private TemperatureChangeEvent event;

private Temperature temperature;

private TemperatureTrackerPresenter presenter;
```

Birim Testleri

JAVA Eğitimleri

```
@Test
public void textShouldBeUpdatedWhenTemperatureChanges() {
    Mockito.when(event.getChange()).thenReturn(1);

    Mockito.when(event.getType()).thenReturn(Change.INCREASE);

    Assert.assertEquals(0, temperature.getValue());

    presenter.temperatureChanged(event);


    Assert.assertEquals(1, temperature.getValue());

    Mockito.verify(tempText).refresh();
}
```

Birim Testleri

JAVA Eğitimleri

```
@Test
public void colorShoulBeRedWhenTemperatureAboveForty() {
    Mockito.when(event.getChange()).thenReturn(41);

    Mockito.when(event.getType()).thenReturn(Change.INCREASE);

    Assert.assertEquals(0, temperature.getValue());

    presenter.temperatureChanged(event);

    Assert.assertEquals(41, temperature.getValue());

    Mockito.verify(tempText).red();

    Mockito.verify(tempText).refresh();
}
```

Presenter Impl.

JAVA Eğitimleri

```
public void temperatureChanged(TemperatureChangeEvent event) {
    int value = event.getChange();

    if(event.getType() == Change.INCREASE) {
        temperature.increase(value);
    } else {
        temperature.decrease(value);
    }

    value = temperature.getValue();

    if(value < 0) {
        tempText.blue();
    } else if (value > 0 && value < 24) {
        tempText.green();
    } else if (value > 24 && value < 40) {
        tempText.yellow();
    } else if (value > 40) {
        tempText.red();
    }

    tempText.refresh();
}
```

IncButton View Impl.


```
public void addTemperatureChangeEventListener(
    TemperatureChangeListener changeListener) {
    listeners.add(changeListener);
}

public IncButton() {
    Button btn = new Button("+");
    btn.addListener(this);
    setCompositionRoot(btn);
}

@Override
public void buttonClick(ClickEvent event) {
    TemperatureChangeEvent temperatureChangeEvent = new
                TemperatureChangeEvent(1, Change.INCREASE);

    for(TemperatureChangeListener listener:listeners) {
        listener.temperatureChanged(temperatureChangeEvent);
    }
}
```

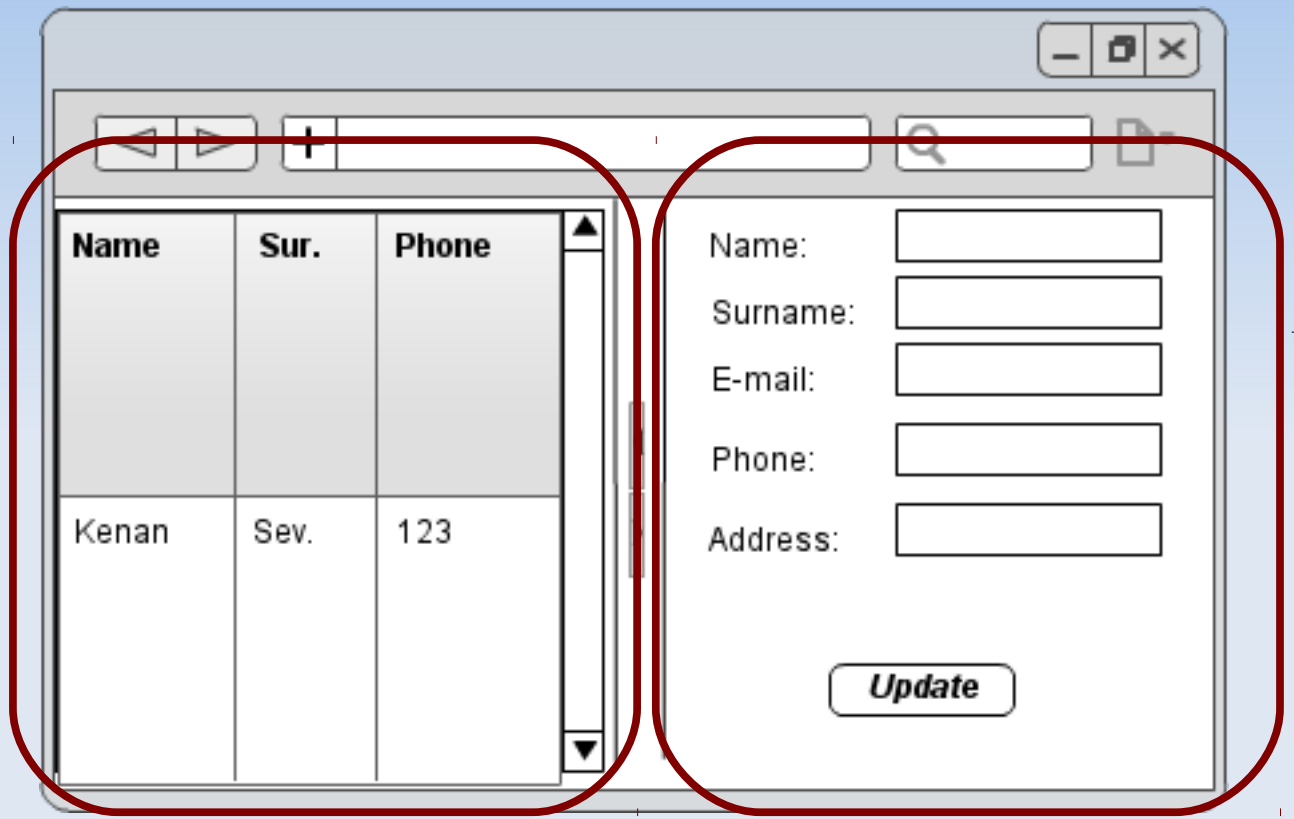
TempText View Impl.


```
private Label label;

public TempTextImpl(Temperature temperature) {
   HorizontalLayout ho = new HorizontalLayout();

   ho.setSpacing(true);
   ho.addComponentAsFirst(new Label("Temperature :"));
   label = new Label(new MethodProperty(temperature,
"value"));
   ho.addComponent(label);

   setCompositionRoot(ho);
}

@Override
public void blue() {
   label.setStyleName("bluelabel");
}
```

Örnek 3

- Üçüncü örneğimizde kullanıcımız kişisel bağlantı bilgilerini yönettiği bir uygulama istemektedir. Uygulama ekranında sahip olduğu bağlantılar listelenecek, listelenen kayıtlardan herhangi biri seçildiğinde detay ekranında görüntülenip güncellenebilecektir.

UI Mockup

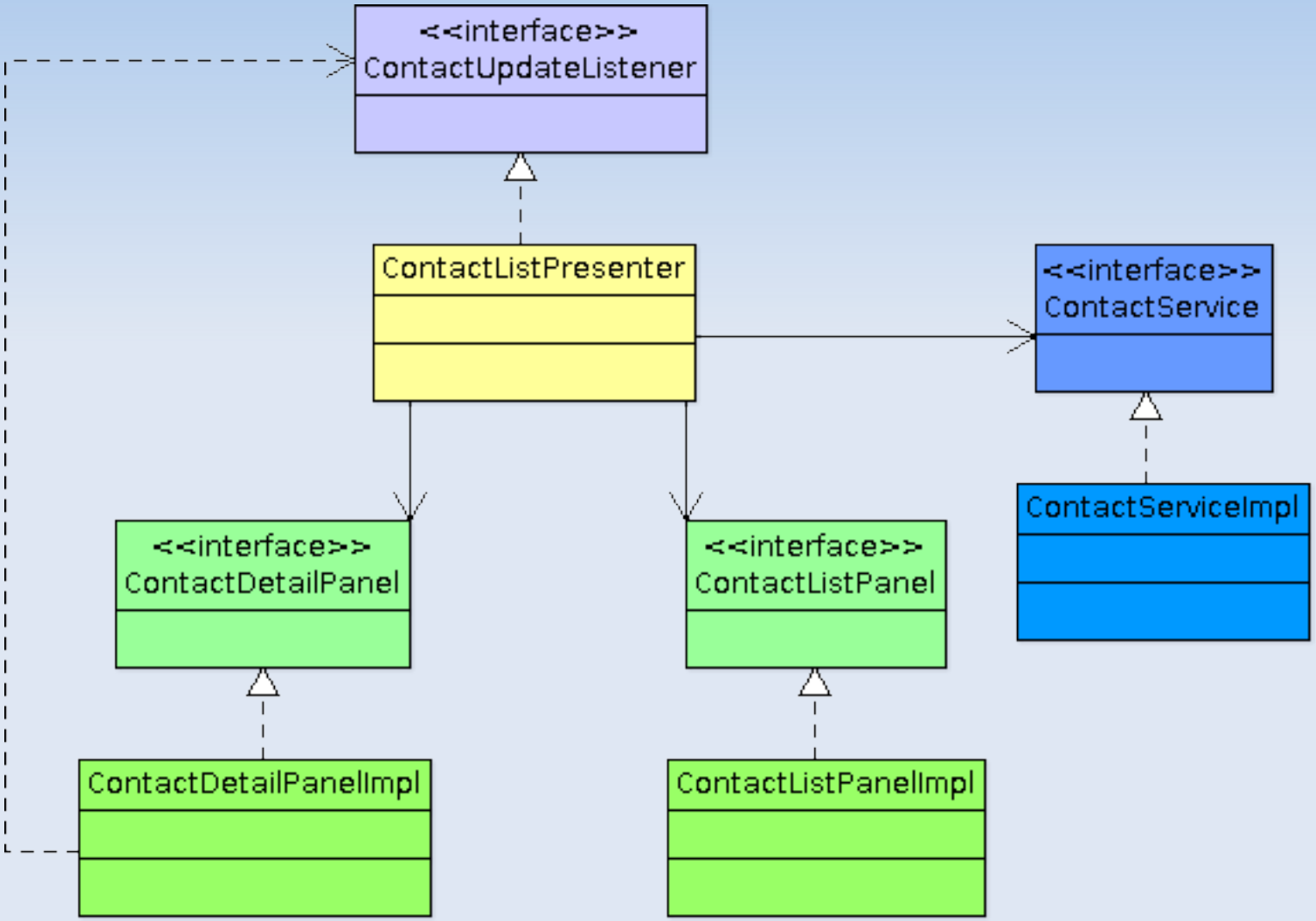


ContactList Panel

Contact DetailPanel

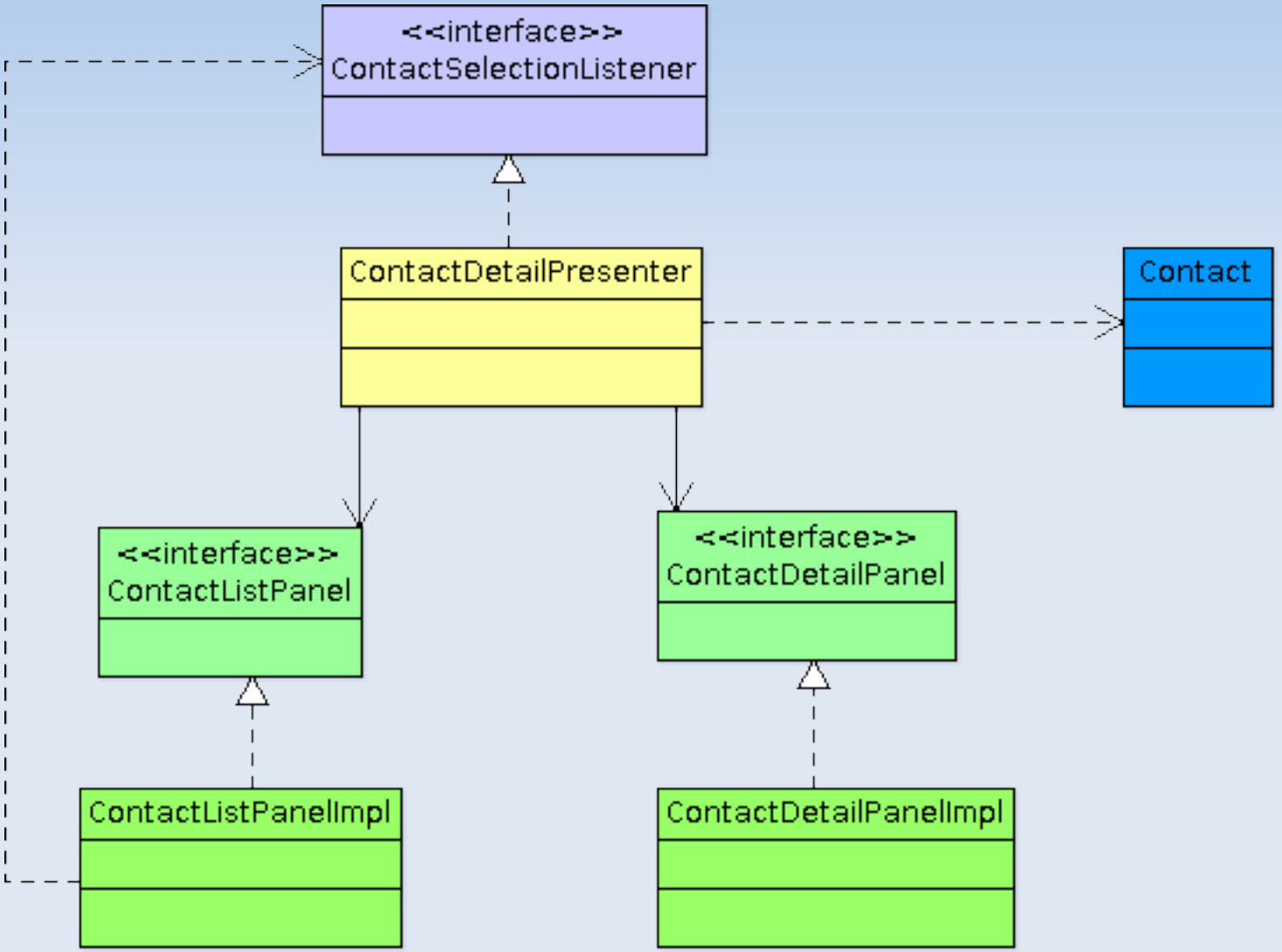
GUI, ContactListPanel ve ContactDetailPanel kısımlarından oluşmaktadır. ListPanel'de mevcut contact nesneleri listelenmektedir. Listeden seçilen herhangi bir Contact'a ait bilgiler sağ taraftaki DetailPanel'de görüntülenir. Bilgiler değiştirildikten sonra Update butonuna basıldığında değişiklikler kaydedilir ve aynı zamanda ListPanel'e de yansıtılır

Sınıf ve Arayüzler



Sınıf ve Arayüzler

JAVA Eğitimleri



Contact List Birim Testleri


```
private ContactListPresenter presenter;

@Mock
private ContactListPanel listPanel;

@Mock
private ContactDetailPanel detailPanel;

@Mock
private ContactService service;

private Collection<Contact> contacts;

@Mock
private ContactUpdatedEvent event;

@Mock
private Contact contact;
```

Contact List Birim Testleri


```
@Test
public void contactsShouldBeShownWhenPageIsLoaded() {
   Mockito.verify(service).getContacts();

   Mockito.verify(listPanel).loadContacts(contacts);
}

@Test
public void contactShouldBeReloadedWhenUpdated() {
   Mockito.verify(detailPanel)
         .addContactUpdateListener(presenter);

   presenter.contactUpdated(event);

   Mockito.verify(event).getContact();

   Mockito.verify(listPanel).reloadContact(contact);
}
```

Contact List Presenter Impl.


```
public ContactListPresenter(ContactListPanel listPanel,
      ContactDetailPanel detailPanel, ContactService service)
{
   this.listPanel = listPanel;
   this.service = service;

   Collection<Contact> contacts = service.getContacts();

   listPanel.loadContacts(contacts);

   detailPanel.addContactUpdateListener(this);
}

@Override
public void contactUpdated(ContactUpdatedEvent event) {
   Contact contact = event.getContact();

   listPanel.reloadContact(contact);
}
```

Contact List View Impl.


```
public void loadContacts(Collection<Contact> contacts) {
   BeanItemContainer<Contact> dataSource = new
         BeanItemContainer<Contact>(Contact.class, contacts);

   table.setContainerDataSource(dataSource);
   table.setVisibleColumns(new Object[]
               {"name","surname","email","phone"});
   table.setColumnHeaders(new String[]{"Name","Surname","E-
Mail","Phone"});
}

public void valueChange(ValueChangeEvent event) {
   Contact contact = (Contact) table.getValue();

   ContactSelectedEvent selectedEvent = new
         ContactSelectedEvent(contact);

   for(ContactSelectionListener listener:listeners) {
      listener.contactSelected(selectedEvent);
   }
}
```

Contact List View Impl.


```
@Override
public void reloadContact(Contact contact) {

    BeanItemContainer container = (BeanItemContainer)

    table.getContainerDataSource();

    container.removeItem(contact);

    container.addBean(contact);

    table.requestRepaintAll();
}
```

Contact Detail Birim Testleri


```
@Mock
private ContactDetailPanel detailPanel;

@Mock
private ContactListPanel listPanel;

private ContactDetailPresenter presenter;

@Mock
private Contact contact;

@Mock
private ContactSelectedEvent event;

@Test
public void contactShouldBeDisplayedInDetailPanelWhenSelected() {
presenter.contactSelected(event);

Mockito.verify(detailPanel).displayContact(contact);
Mockito.verify(event).getSelectedContact();

Mockito.verify(listPanel).addContactSelectionListener(presenter);
}
```

Contact Detail Presenter Impl.


```
private ContactDetailPanel detailPanel;

public ContactDetailPresenter(ContactDetailPanel detailPanel,
      ContactListPanel listPanel) {

   this.detailPanel = detailPanel;

   listPanel.addContactSelectionListener(this);
}

@Override
public void contactSelected(ContactSelectedEvent event) {

   Contact contact = event.getSelectedContact();

   detailPanel.displayContact(contact);
}
```

Contact Detail View Impl.

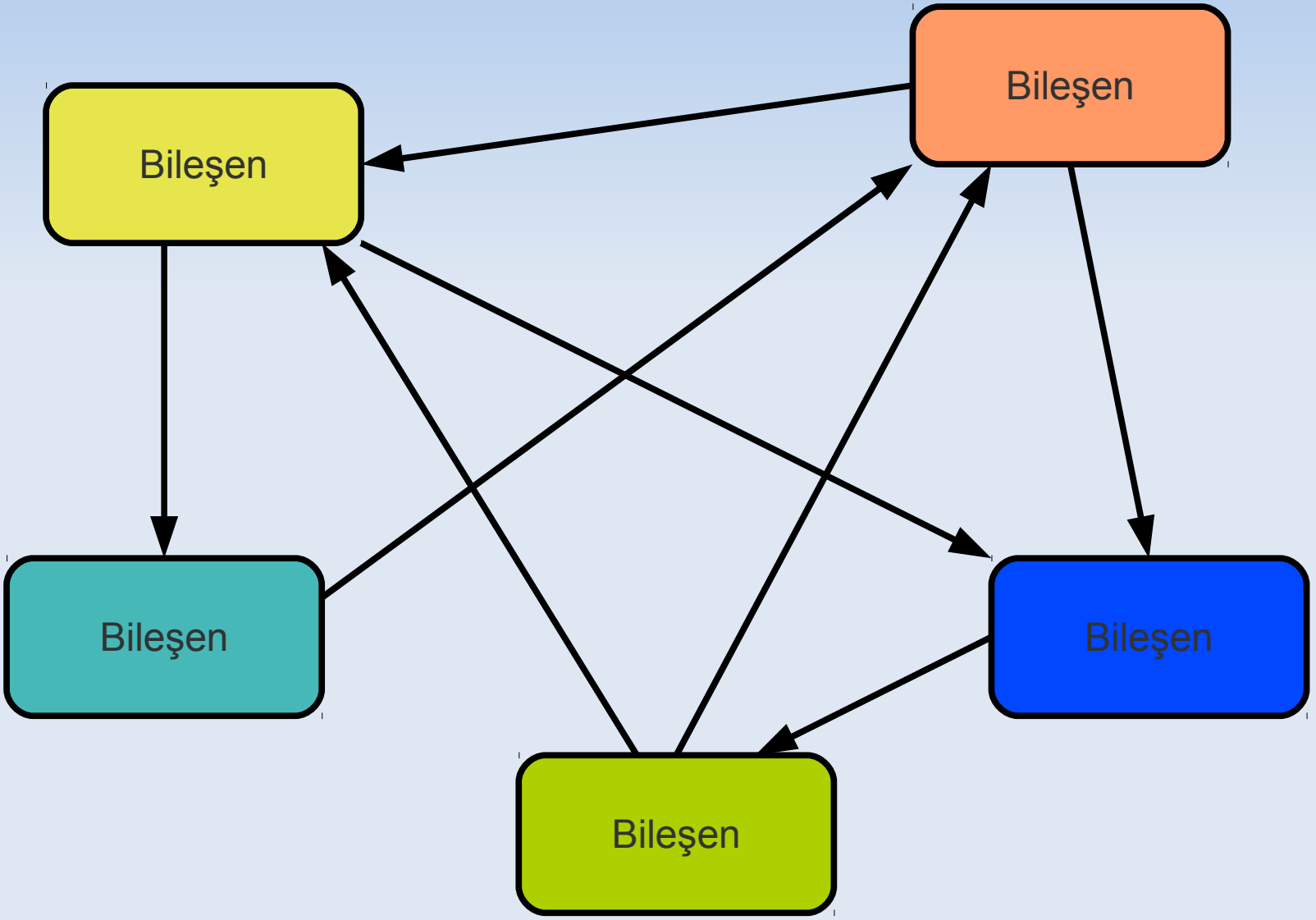
JAVA Eğitimleri

```
@Override
public void displayContact(Contact contact) {

    BeanItem item = new BeanItem(contact);
    form.setItemDataSource(item);
    form.setVisibleItemProperties(new Object[]
          {"name","surname","email","phone","address"});
}

@Override
public void buttonClick(ClickEvent event) {
    Contact contact = (Contact) ((BeanItem)
    form.getItemDataSource()).getBean();
    ContactUpdatedEvent updateEvent = new
                    ContactUpdatedEvent(contact);

    for(ContactUpdateListener listener:listeners) {
       listener.contactUpdated(updateEvent);
    }
}
```

ContactListPanel ve ContactDetailPanel ile etkileşime girecek bir bileşen daha eklenirse...

Örneğin, update butonunun bulunduğu kısım bir ToolBarPanel bileşenine dönüştürülür ve update butonunun sadece Contact seçildiği zaman görünür olması istenirse nasıl bir **problem** ortaya çıkabilir?

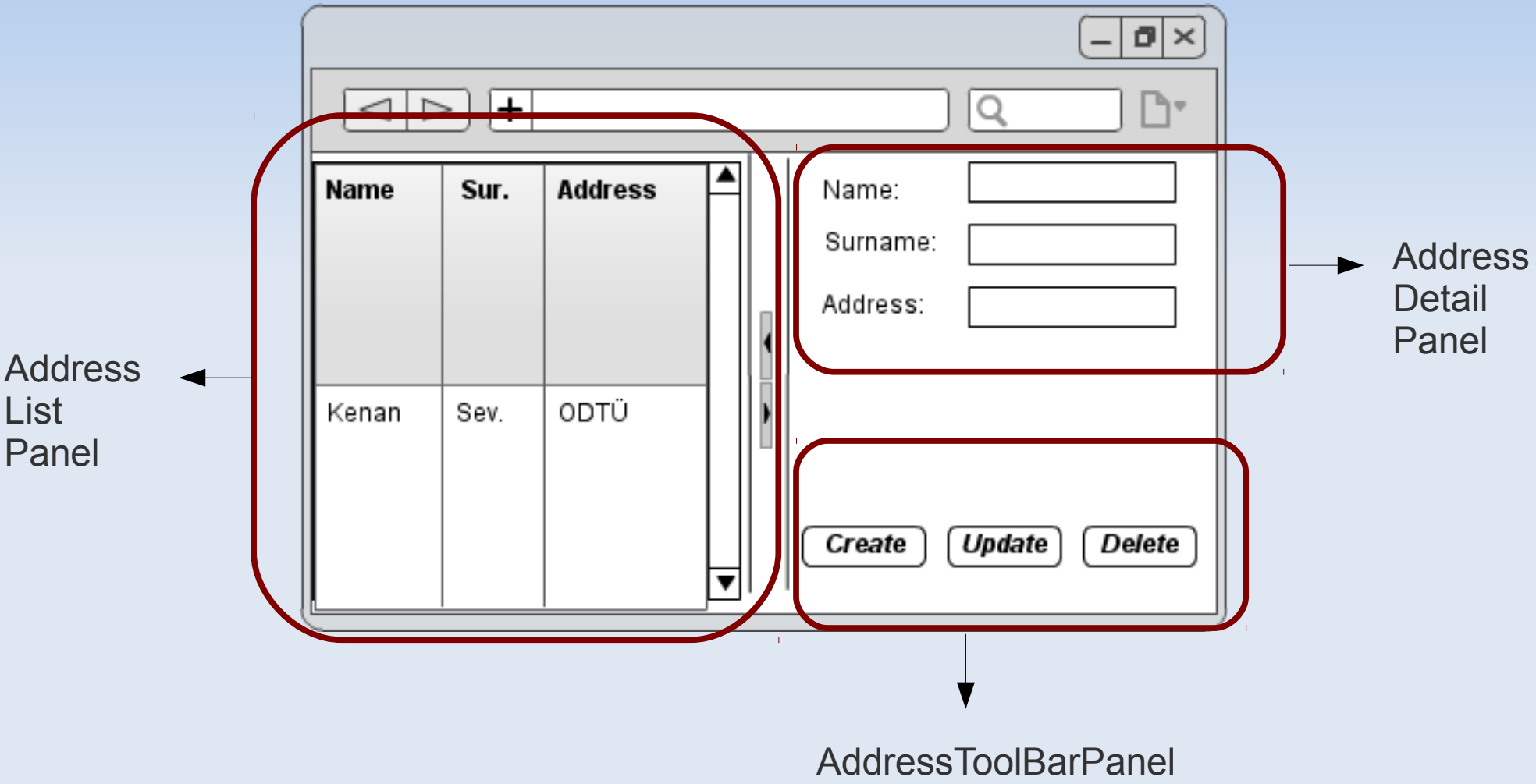
Mediator Öncesi



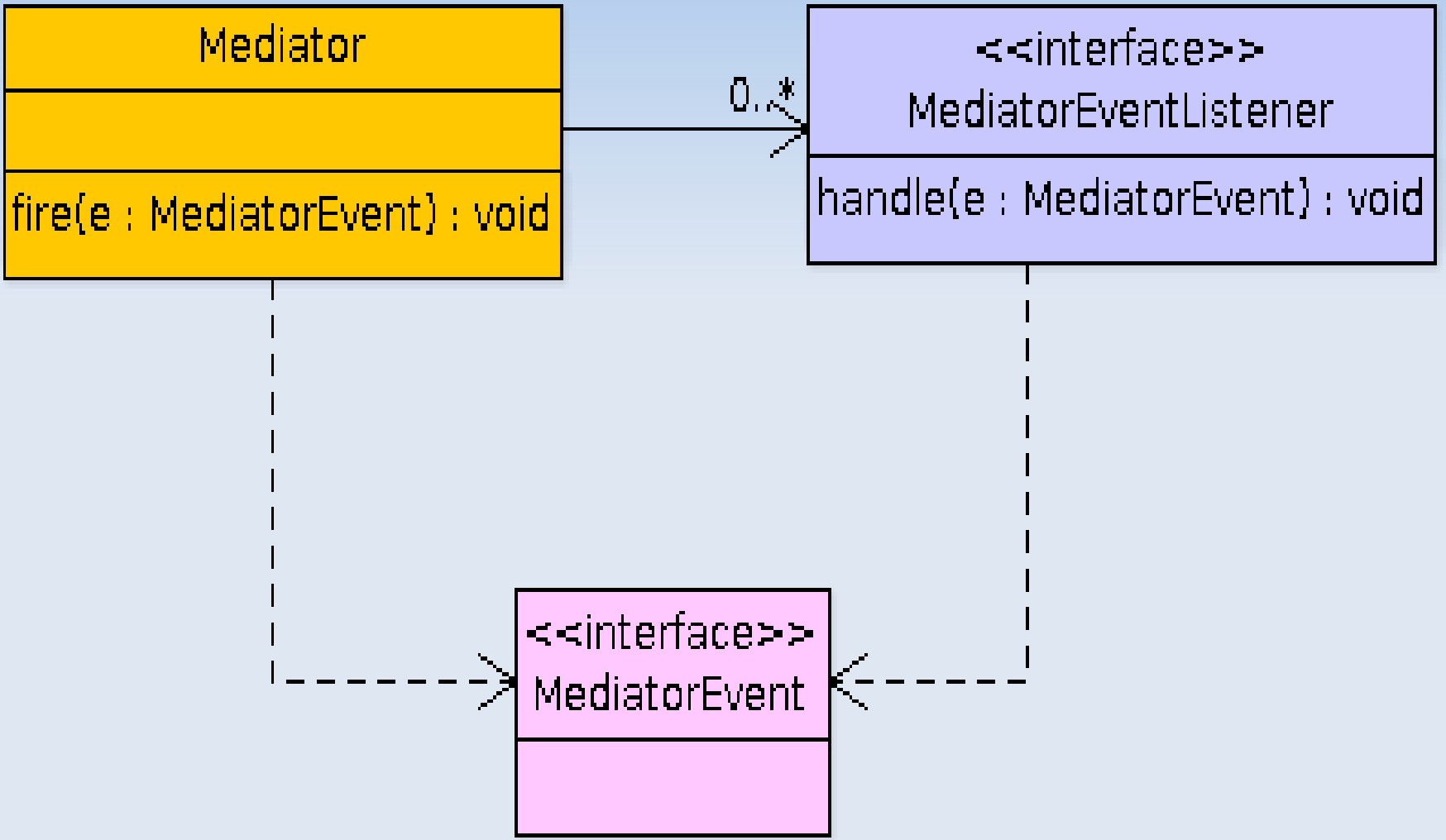
Örnek 4

- Kullanıcımız bağlantılarını yönettiği uygulamasına benzer şekilde adres bilgilerini yönettiği bir uygulama istemektedir.
- Ancak uygulamanın bileşenlerinin mümkün olduğunca birbirlerinden bağımsız ve farklı bölümlerde yeniden kullanılabilir olmasına dikkat edilmesi istenmektedir.

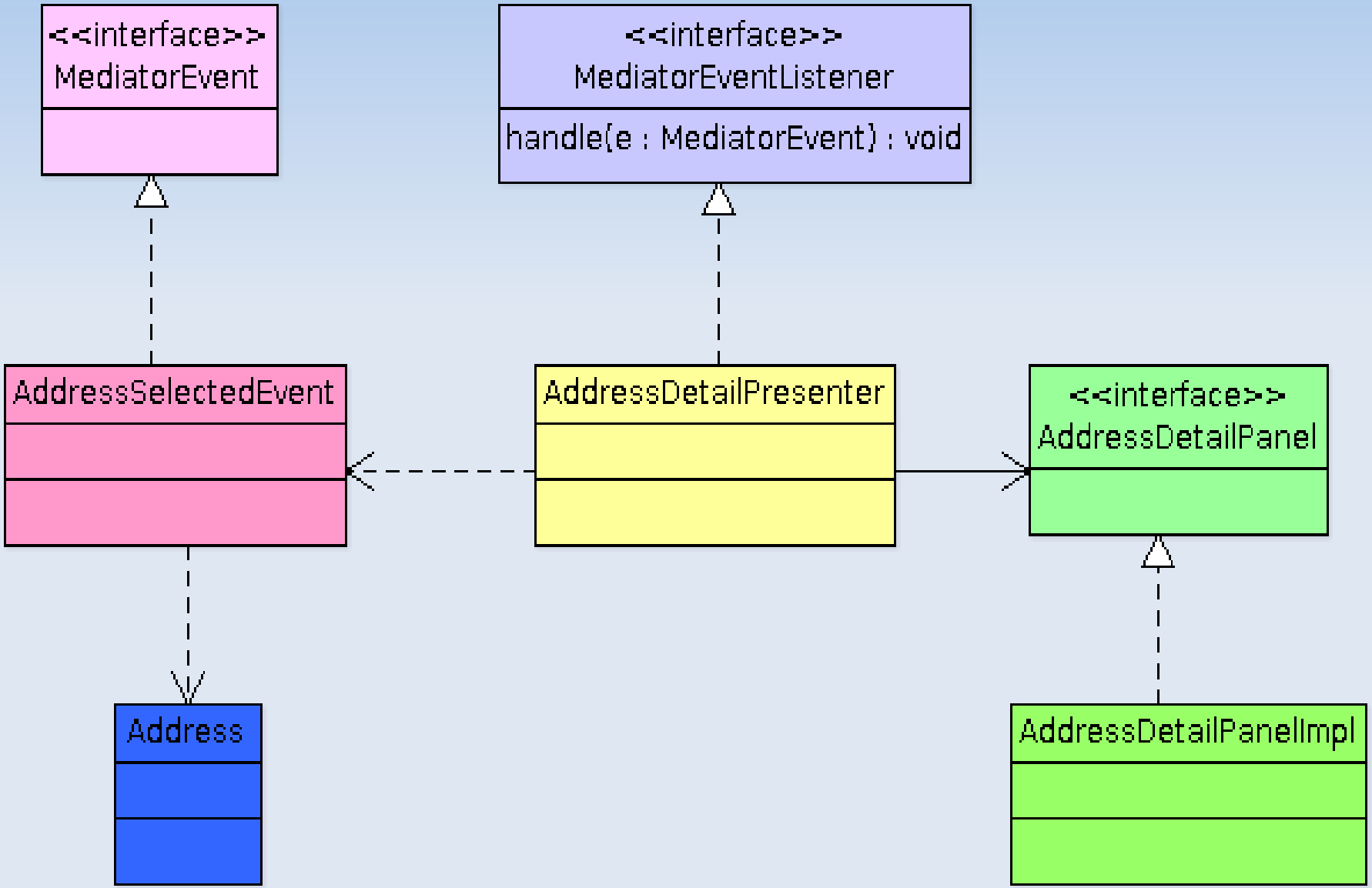
UI Mockup


Name
Sur.
Address
Kenan
Sev.
ODTÜ
Name:
Surname:
Address:
Create
Update
Delete
Address List Panel
Address Detail Panel
AddressToolBarPanel

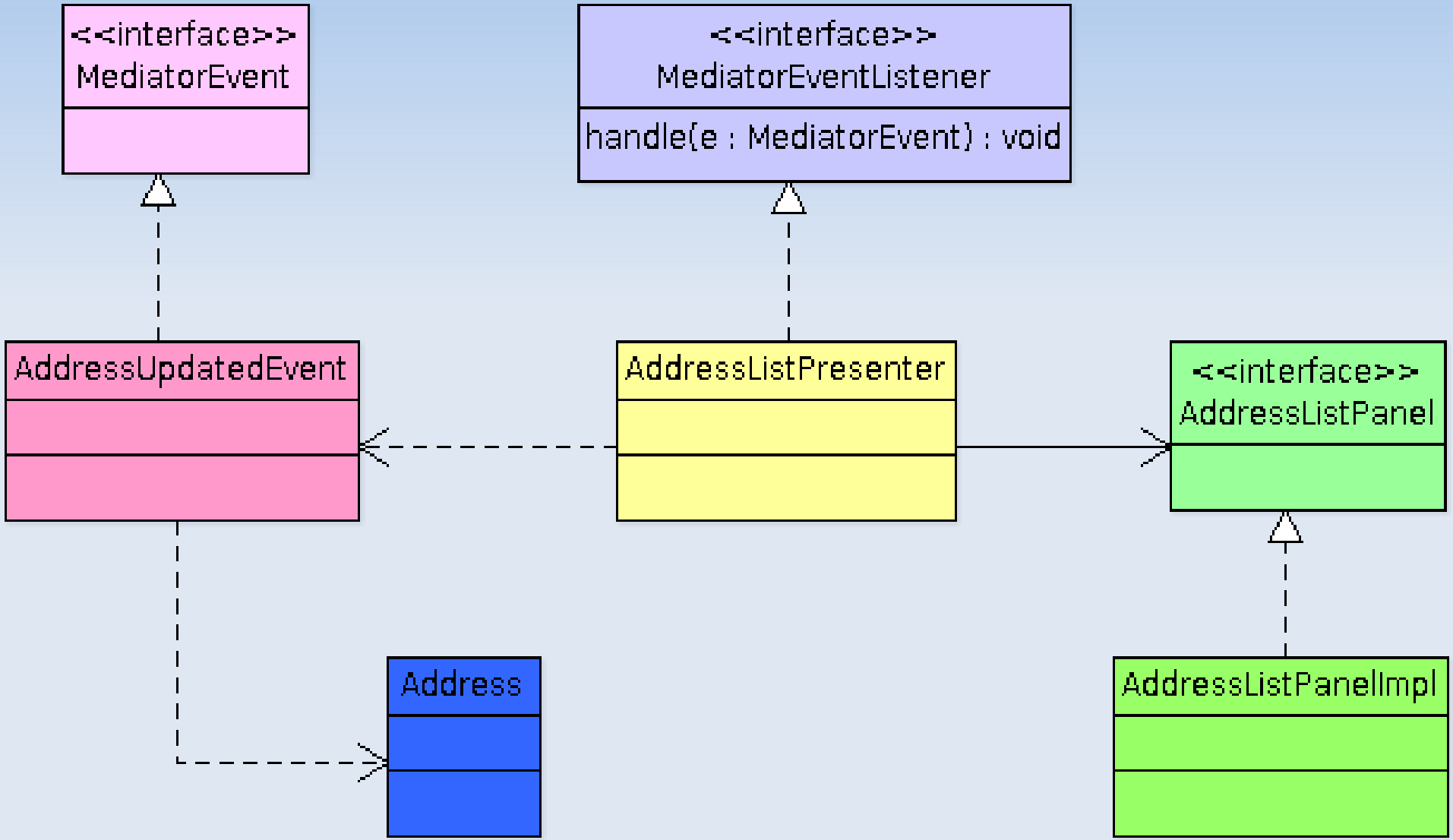
Sınıf ve Arayüzler



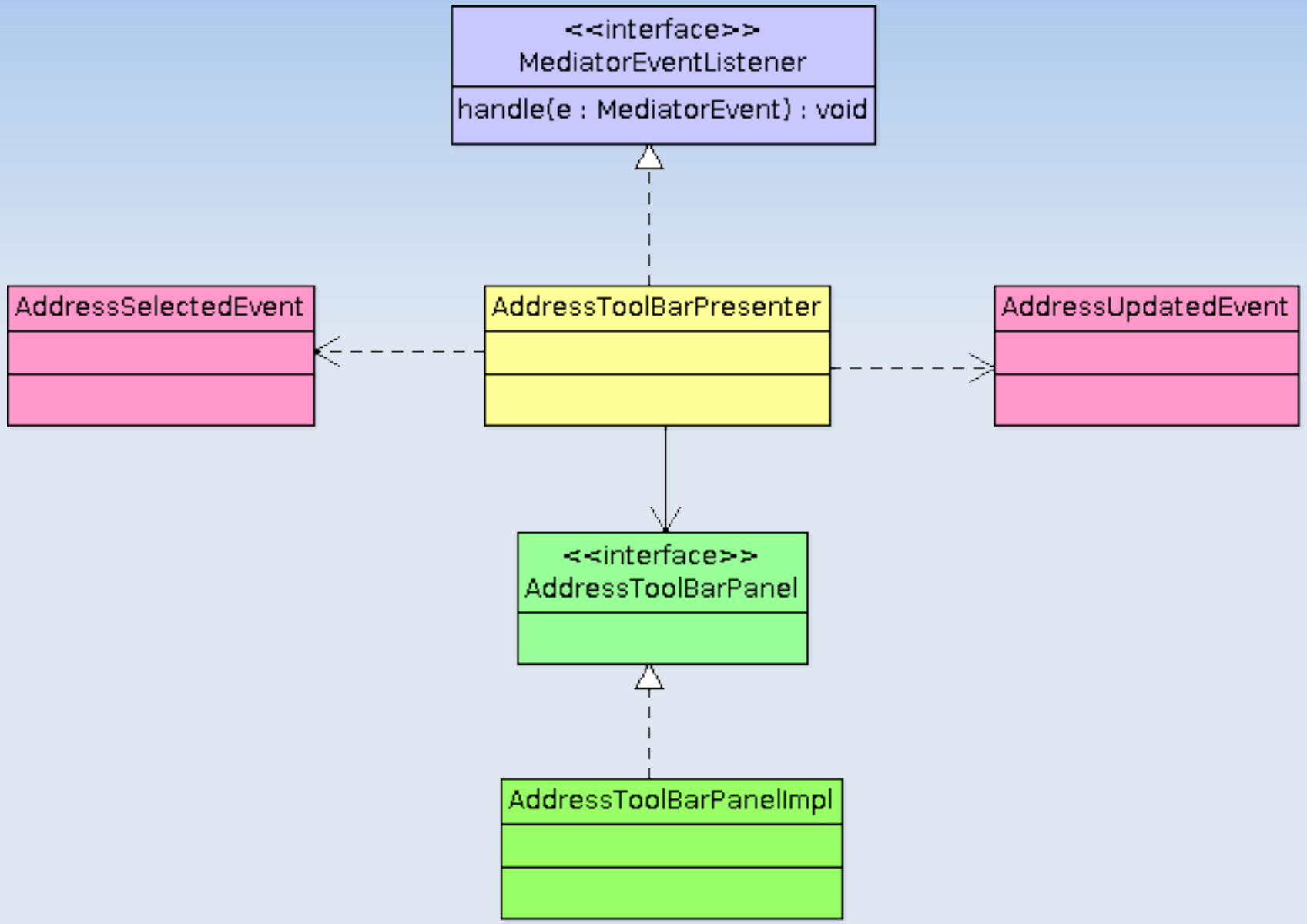
Sınıf ve Arayüzler



Sınıf ve Arayüzler



Sınıf ve Arayüzler



Mediator Impl.


```
private Collection<MediatorEventListener> listeners = new
         ArrayList<MediatorEventListener>();

public void addListener(MediatorEventListener listener) {
   listeners.add(listener);
}

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

public void fire(MediatorEvent event) {

   for(MediatorEventListener listener:listeners) {

      listener.handle(event);

   }

}
```

Address List Presenter Impl.


```
private AddressListPanel listPanel;

public AddressListPresenter(AddressListPanel listPanel,
      AddressService addressService) {

   this.listPanel = listPanel;
   listPanel.loadAddresses(addressService.getAddresses());
}

@Override
public void handle(MediatorEvent event) {

   if(event instanceof AddressUpdatedEvent) {
      AddressUpdatedEvent updateEvent =
(AddressUpdatedEvent)event;
      listPanel.reloadAddress(updateEvent.getAddress());
   }

}
```

Address Detail Presenter Impl.


```
private AddressDetailPanel detailPanel;

public AddressDetailPresenter(AddressDetailPanel detailPanel)
{
   this.detailPanel = detailPanel;
}

@Override
public void handle(MediatorEvent event) {

   if(event instanceof AddressSelectedEvent) {

      AddressSelectedEvent selectedEvent =
                     (AddressSelectedEvent)event;

      detailPanel.displayAddress(
            selectedEvent.getSelectedAddress());

   }
}
```

Address ToolBar Presenter Impl.


```
private AddressToolBarPanel addressToolBarPanel;

public AddressToolBarPresenter(AddressToolBarPanel
addressToolBarPanel) {
   this.addressToolBarPanel = addressToolBarPanel;
   addressToolBarPanel.swithToSelectionMode();
}

@Override
public void handle(MediatorEvent event) {
   if(event instanceof AddressSelectedEvent) {
      addressToolBarPanel.swithToUpdateMode();

      addressToolBarPanel.setAddress(
         ((AddressSelectedEvent)event).getSelectedAddress());
   } else if(event instanceof AddressUpdatedEvent) {

      addressToolBarPanel.swithToSelectionMode();
   }
}
```

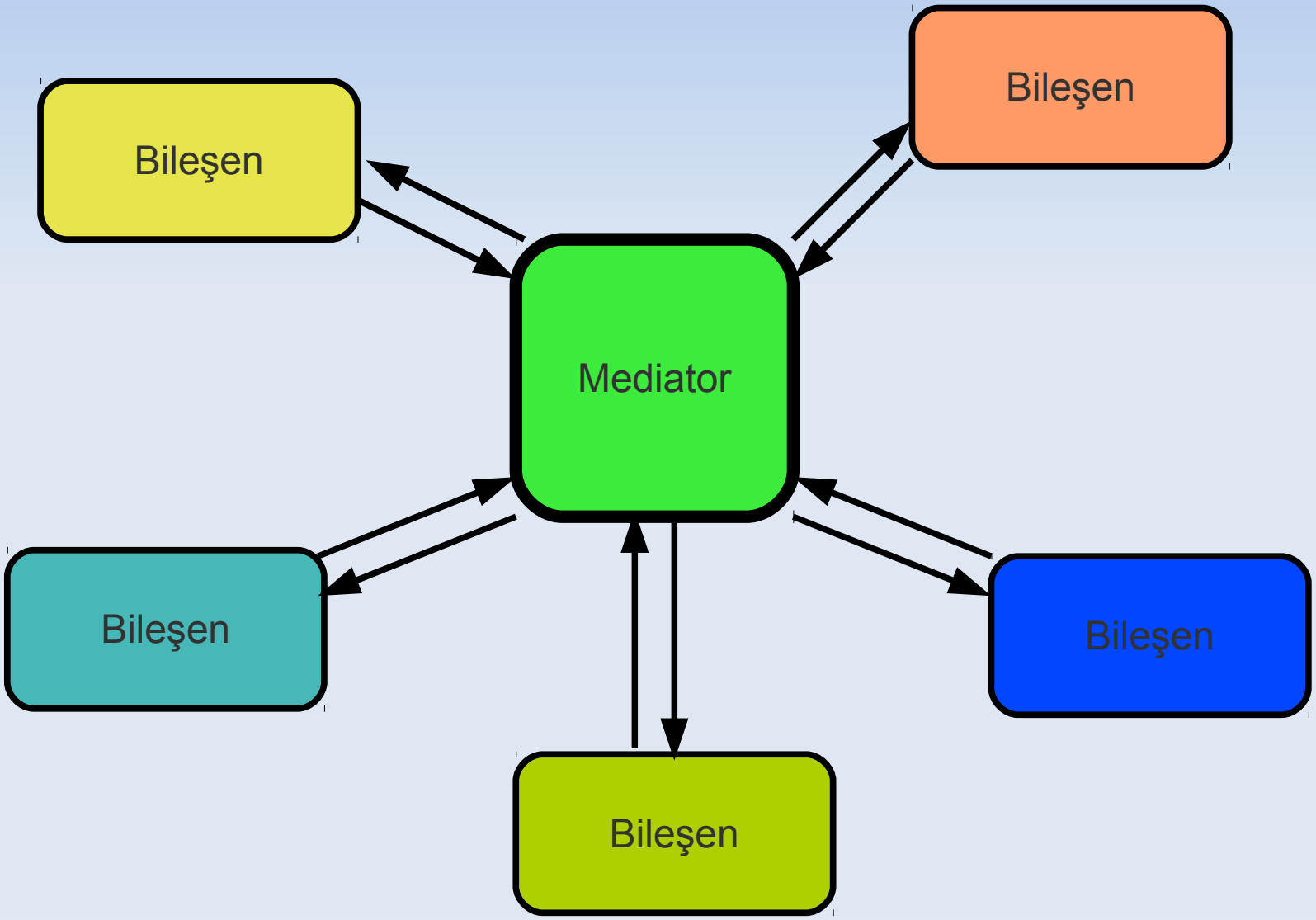
Address List View Impl.


```
public AddressListPanelImpl(Mediator mediator) {
   this.mediator = mediator;

...
}

@Override
public void loadAddresses(Collection<Address> addresses) {
...
}

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

   AddressSelectedEvent selectedEvent = new
                     AddressSelectedEvent(address);
   mediator.fire(selectedEvent);
}
```

Address ToolBar View Impl.


```
public AddressToolBarPanelImpl(Mediator mediator) {
   this.mediator = mediator;
...
}

@Override
public void setAddress(Address address) {
   this.address = address;
}

@Override
public void buttonClick(ClickEvent event) {

   if(event.getButton() == updateButton) {
      mediator.fire(new AddressUpdatedEvent(address));
   }

}
```

Mediator Sonrası



Soru & Cevap

İletişim

- Harezmi Bilişim Çözümleri Ltd.
- Kurumsal Java Eğitimleri
- http://www.harezmi.com.tr
- http://www.java-egitimleri.com
- info@java-egitimleri.com