

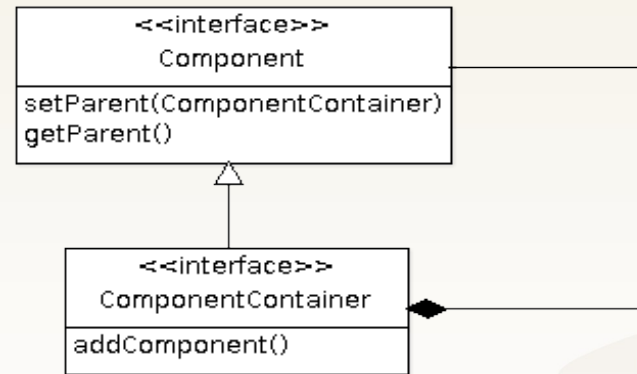
# Vaadin UI Bileşenleri

# UI Bileşenleri

- Vaadin arayüzleri geliştirirken **UI bileşenleri** bir araya getirilerek programlama yapılır
- UI bileşenleri temelde **ikiye ayrılır**:
- Kullanıcının etkileşimde bulunduğu **UI bileşenleri**
  - Label, Button, TextField, ComboBox,...
- Bileşenlerin yerleşimini düzenleyen **layout bileşenleri**
  - VerticalLayout, HorizontalLayout, FormLayout, ...

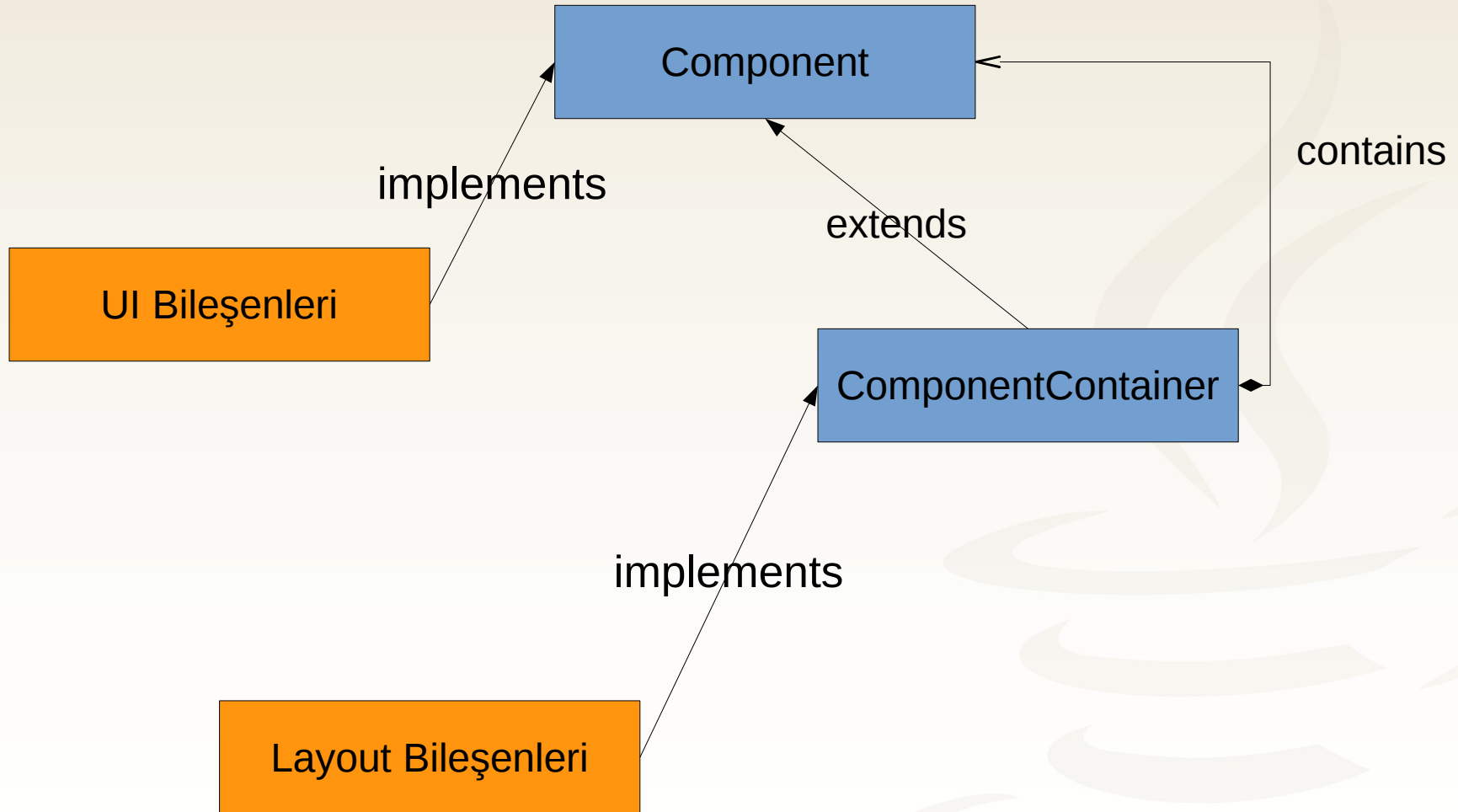
# UI Bileşenleri

- TextField
- TextArea
- ComboBox
- ListSelect
- OptionGroup
- TwinColSelect
- DateField
- CheckBox
- Button
- Table
- Tree



- HorizontalLayout
- VerticalLayout
- GridLayout
- Panel
- Window
- SplitPanel
- TabSheet
- Accordion

# UI Bileşenleri



# UI Bileşenleri

```
@Override
protected void init(VaadinRequest request) {

    VerticalLayout layout = new VerticalLayout();
```

```
    TextField firstName = new TextField("First Name");
    layout.addComponent(firstName);
```

```
    TextField lastName = new TextField("Last Name");
    layout.addComponent(lastName);
```

```
    Button button = new Button("Add");
    layout.addComponent(button);
```

```
    setContent(layout);
}
```

Bileşenler layout'a eklenir

Başlık



# UI Bileşenleri

- Component'ler ComponentContainer'lar altına **addComponent()** metodu ile eklenerek Component Tree oluşturulur
- Bir bileşenin bağlı olduğu parent bileşene **getParent()** metoduyla ulaşılabilir
- Component tree'nin **kökü** bir **ComponentContainer** instance'ıdır
- Kök bileşen current UI ile **UI.setContent()** metodu ile ilişkilendirilir

# UI Bileşenleri

```
@Override
protected void init(VaadinRequest request) {

    VerticalLayout layout = new VerticalLayout();
    setContent(layout);

    final TextArea address = new TextArea("Address");
    layout.addComponent(address);

    final ComboBox city = new ComboBox("City",
        petClinicService.getCities());
    layout.addComponent(city);

    final TextField telephone = new TextField("Telephone");
    layout.addComponent(telephone);

    final OptionGroup phoneType = new OptionGroup("Phone Type",
        Arrays.asList(PhoneType.values()));
    layout.addComponent(phoneType);

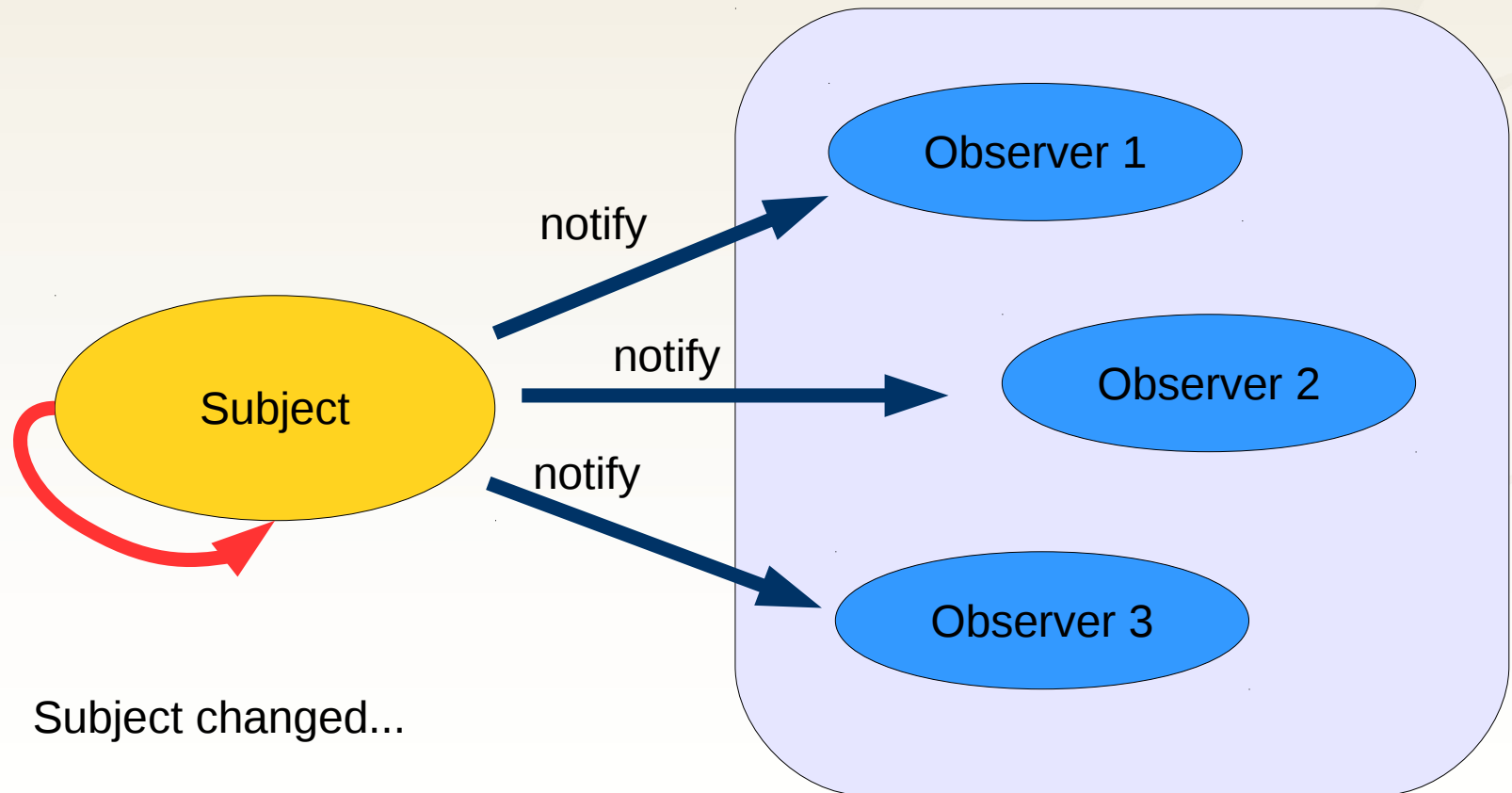
    Button button = new Button("Add");
    layout.addComponent(button);

    ...
}
```

Select veya Option bileşenlerinin değerleri constructor parametresi olarak verilebilir. Parametreler herhangi türde bir Java nesnesi olabilir.

# UI Bileşenleri ve Event Yönetimi

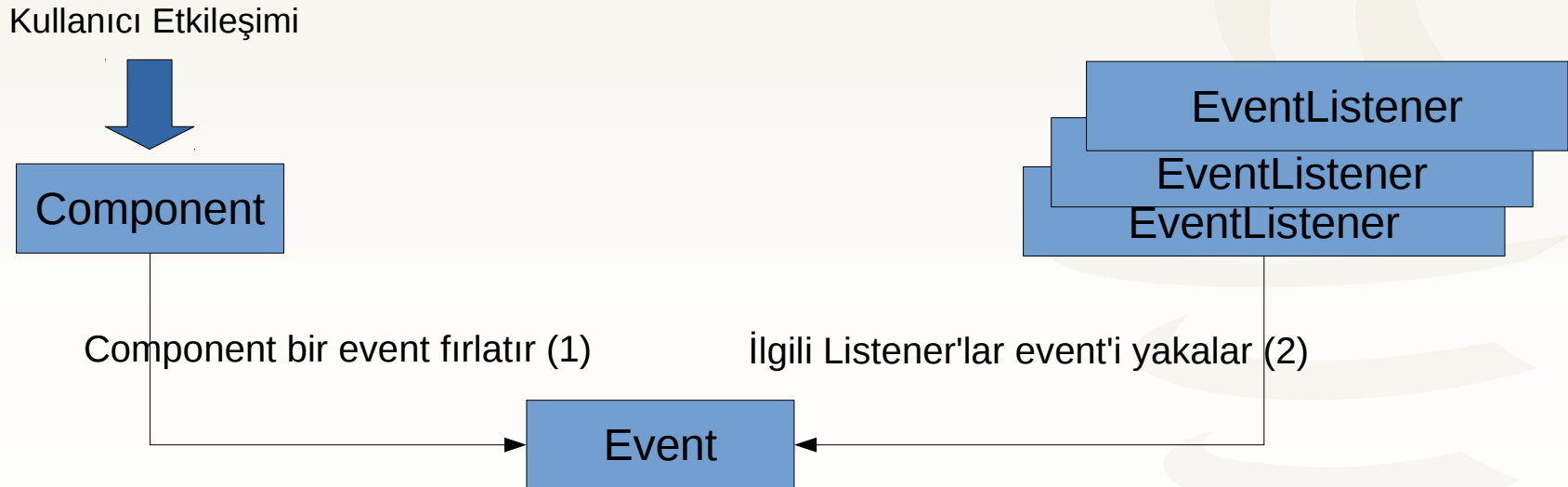
Vaadin, kullanıcılar ile bileşenler arasındaki etkileşimleri yönetmek için **Event-Driven bir programlama alt yapısı** sunar





# UI Bileşenleri ve Event Yönetimi

Bunun için öncelikle event listener'lar bileşenlere register olurlar  
Kullanıcı bir işlem yaptığında, örneğin bir butona tıkladığında ilgili UI bileşeni tarafından **bir event fırlatılır**  
Bu event, **event listener(lar)** tarafından yakalanır ve gerekli işlem gerçekleştirilir



# UI Bileşenleri ve Event Yönetimi

```
final TextField firstName = new TextField("First Name");  
layout.addComponent(firstName);
```

```
final TextField lastName = new TextField("Last Name");  
layout.addComponent(lastName);
```

```
Button button = new Button("Add");  
layout.addComponent(button);
```

Listenerler bileşenlere  
register olur

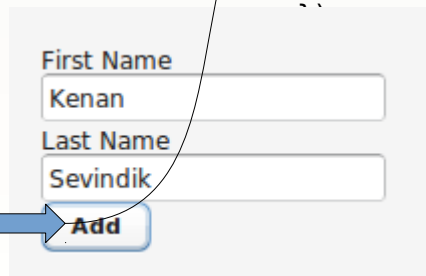
```
button.addClickListener(new Button.ClickListener() {
```

```
    public void buttonClick(ClickEvent event) {
```

```
        Owner owner = new Owner();  
        owner.setFirstName(firstName.getValue());  
        owner.setLastName(lastName.getValue());
```

```
        petClinicService.createOwner(owner);
```

```
    }
```



UI Bileşeni listener'ı  
kullanıcı etkileşimi hakkında  
event oluşturarak notify  
eder


# UI Bileşenleri ve Required Alanlar

```
TextField firstName = new TextField("First Name");  
firstName.setRequired(true);  
firstName.setRequiredError("First name is required!");
```

```
TextField lastName = new TextField("Last Name");  
lastName.setRequired(true);  
lastName.setRequiredError("Last name is required!");
```

```
TextArea address = new TextArea("Address");  
address.setRequired(true);  
address.setRequiredError("Address is required!");
```

Input bileşenin üzerine gelindiğinde  
required mesajı tooltip olarak  
gösterilecektir

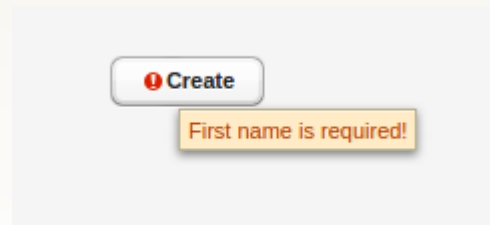


The screenshot shows a form with three input fields: "First Name", "Last Name", and "Address". Each field has a red asterisk next to its label. A tooltip is visible over the "Last Name" field, displaying the message "Last name is required!". Arrows from the code snippets point to these fields: the first code block points to the "First Name" field, the second code block points to the "Last Name" field, and the third code block points to the "Address" field.

# UI Bileşenleri ve Required Alanlar

```
@Override  
public void buttonClick(ClickEvent event) {  
    firstName.validate();  
    lastName.validate();  
    address.validate();  
    ...  
}
```

Validate metodu çağrıldığı vakit bileşen değeri boş ise bir exception fırlatılır. Vaadin'in DefaultErrorHandler'ı bu exception'ı yakalayıp hatanın meydana geldiği andaki bileşenin üzerinde bir ünlem ikonu ve required alanın mesajının görüntülenmesini sağlar.



```
com.vaadin.data.Validator$EmptyValueException: First name is required!  
at com.vaadin.ui.AbstractField.validate(AbstractField.java:954)  
at com.javaegitimleri.petclinic.view.VetDetailView.buttonClick(VetDetailView.java:58)
```

# UI Bileşenlerinin Ortak Özellikleri

- **Caption:**
  - Constructor üzerinden ya da **setCaption()** ile set edilebilir
- **Description (tooltip):**
  - **setDescription()** metodu ile set edilir. XHTML tagleri kullanılarak zenginleştirilebilir
- **Icon:**
  - **setIcon()** ile set edilir

# UI Bileşenlerinin Ortak Özellikleri

- **Locale:**
  - Bileşenin ülke ve dil ayarlarını belirtir
  - **setLocale()** metodu ile değiştirilebilir
  - Default değeri eğer **parent bileşen mevcut ise** bunun Locale değeridir, aksi takdirde **VaadinSession'ın Locale değeridir**
  - VaadinSession ise Locale bilgisi set edilmemiş ise, sistemin default Locale'ini **Locale.getDefault()** ile döner

# UI Bileşenlerinin Ortak Özellikleri

- **Enabled:**
  - Bileşenin aktif olup olmadığını belirtir
  - **setEnabled()** metoduyla set edilir
  - Disabled moddaki bileşenler **readonly** duruma alınır
  - Bu bileşenlerden sunucuya gelen etkileşimler göz ardı edilir
  - Bir bileşen disable yapıldığı vakit altındaki bileşenlerde **recursive disable** edilir

# UI Bileşenlerinin Ortak Özellikleri

- **ReadOnly:**
  - Bileşeni readonly duruma geçirmek için **setReadOnly()** metodu çağrılır
  - Disabled durumunda olduğu gibi kullanıcı etkileşimleri kabul edilmez
  - Aynı zamanda **setValue()** ile programatik olarak bile değiştirilmeye izin verilmez
  - Readonly değeri **sadece söz konusu bileşeni** etkiler



# UI Bileşenlerinin Ortak Özellikleri

- **Visible:**
  - Bileşenleri ekranda göstermemek (gizlemek) için **setVisible()** metodu kullanılır
  - Bu bileşenler, component tree'de olmalarına karşın ekranda görüntülenmezler
  - Bir bileşen ancak onun bütün parent bileşenleri visible ise ekranda görünebilir

# UI Bileşenlerinin Ortak Özellikleri

- **Focus (Tab Index):**
  - Tarayıcıda görüntülenen bileşenler arasında TAB tuşu ile gezinerek sırayla focus olmaları sağlanabilir
  - Bileşenlerin hangi sırada focus olacağı **setTabIndex()** ile belirlenebilir
- **Immediate:**
  - Ekran bileşenlerine girilen değer, bileşen unfocus olduğu anda sunucuya iletilmesini sağlamak için **setImmediate()** metodu kullanılır

# UI Bileşenlerinin Ortak Özellikleri

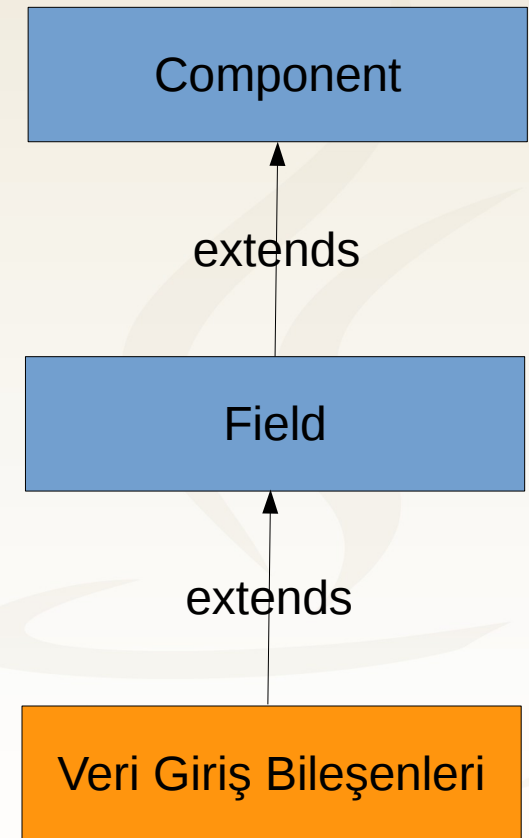
- **Style:**
  - Bileşene CSS üzerinden görsellik katmak için style eklenebilir
  - Bunun için **addStyleName()** kullanılır
  - Programatik olarak mevcut bir style çıkarılmak istenirse de **removeStyleName()** kullanılabilir

# UI Bileşenlerinin Ortak Özellikleri

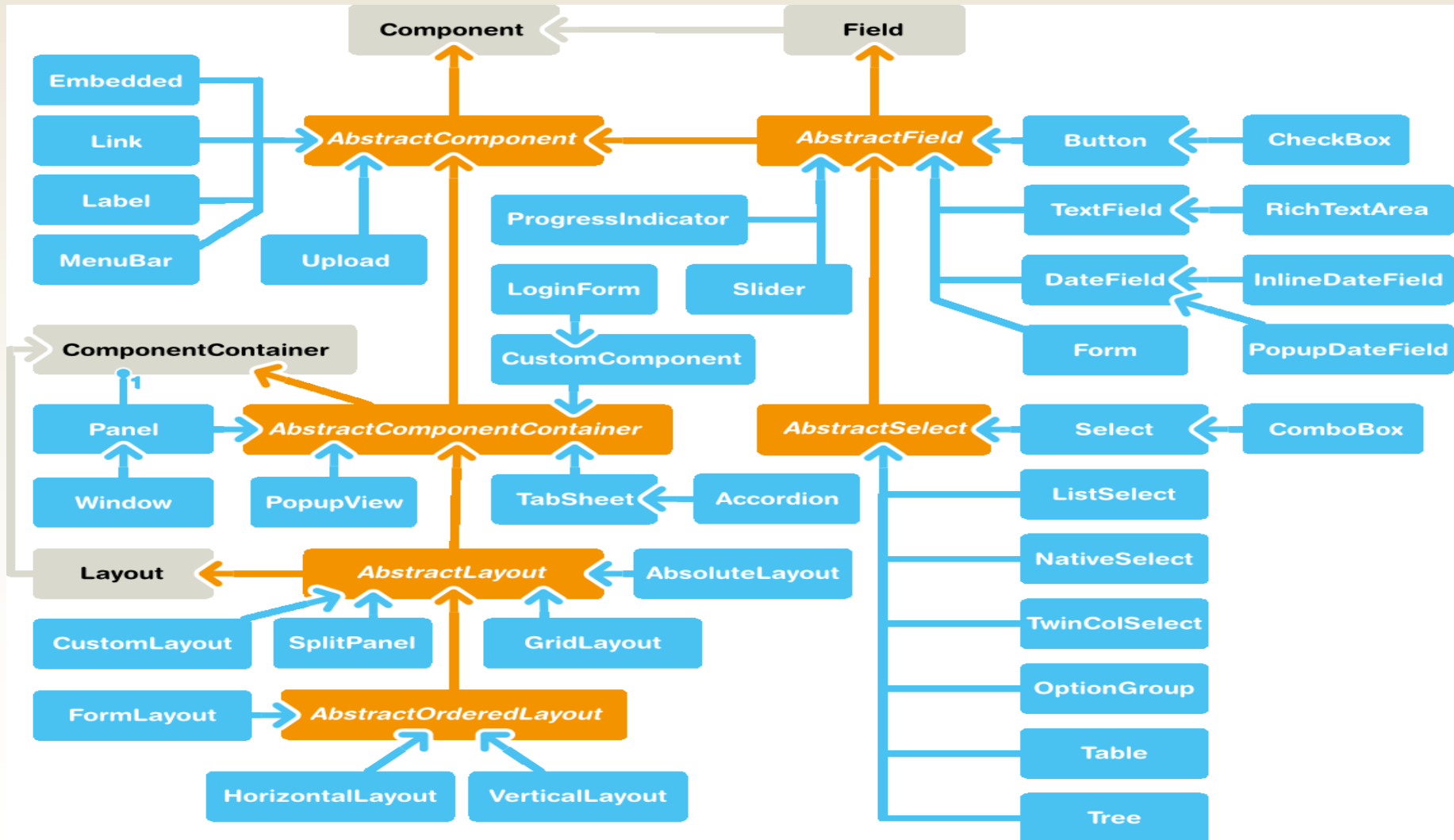
- **Size:**
  - Bileşenlerin ekranda kapladıkları alan **setWidth()** ve **setHeight()** metodları ile belirlenebilir
  - **setSizeFull()** metodu kısaca yükseklik ve genişliği **%100**'e çeker
  - **setSizeUndefined()** metodu ise bir genişlik veya yükseklik belirtmeden bileşenin sığabileceği en **optimum yeri** kaplamasını sağlar

# Field Bileşenleri

- Bazı Component'ler veri girişi amacıyla kullanılır
  - TextField, TextArea, ...
- Field içerisinde tutulan değer **getValue()** metoduyla elde edilir
- Field içerisine bir değer set etmek için ise **setValue()** kullanılır



# UI Bileşenleri: Genel Resim



- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- <http://www.java-egitimleri.com>
- [info@java-egitimleri.com](mailto:info@java-egitimleri.com)

