



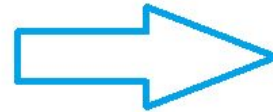
# MODEL

# VIEW

## События об изменении в модели:

```
event EventHandler<EmployeeEventArgs> EventEmployeeAddModel;  
event EventHandler<EmployeeEventArgs> EventEmployeeDeleteModel;
```

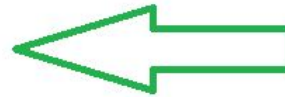
при возникновении событий в модели, view должна на них отреагировать и "перерисовать" изображение



```
public void AddEmployee(Employee employee)  
{  
    _listViewEmployee.Items.Add(employee);  
}  
public void DeleteEmployee(Employee employee)  
{  
    _listViewEmployee.Items.Remove(employee);  
}
```

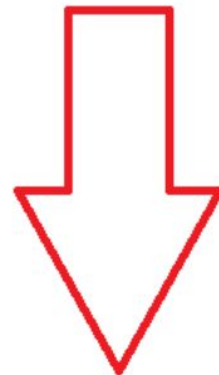
```
public void AddEmployee(Employee employee)  
{  
    Employees.Add(employee);  
}  
public void DeleteEmployee(Employee employee)  
{  
    Employees.Remove(employee);  
}
```

при возникновении событий на view, model должна отреагировать и изменить свои данные внутри модели




## События об изменении на view:

```
event EventHandler<EmployeeEventArgs> EventEmployeeAddView;  
event EventHandler<EmployeeEventArgs> EventEmployeeDelView;
```



# PRESENTER



Событие – это именованный делегат, при вызове которого, будут запущены все подписавшиеся на момент вызова события методы заданной сигнатуры, поэтому необходима подписка на события

#### PRESENTER

```
public class EmployeePresenter
{
    private IEmployeeMainView view;

    private IEmployeeersModel model;

    public EmployeePresenter(IEmployeeMainView employeeView, IEmployeeersModel
employeeModel)
    {
        view = employeeView;
        view.EventEmployeeAddView += view_EmpoyeeAdd;
        view.EventEmployeeDelView += view_EmployeeDelete;

        model = employeeModel;
        model.EventEmployeeAddModel += model_EmpoyeeAdd;
        model.EvenyEmployeeDeleteModel += model_EmployeeDel;
    }
}
```