**PRAKTIKUM-8 PDPL**

***ACHITECTURAL PATTERN – MODEL VIEW CONTROLLER***

**Mata Kuliah : Pola Desain Perangkat Lunak**

**Semester : 6 (Enam)**

**Dosen : Tifanny Nabarian, S.Kom., M.T.I.**

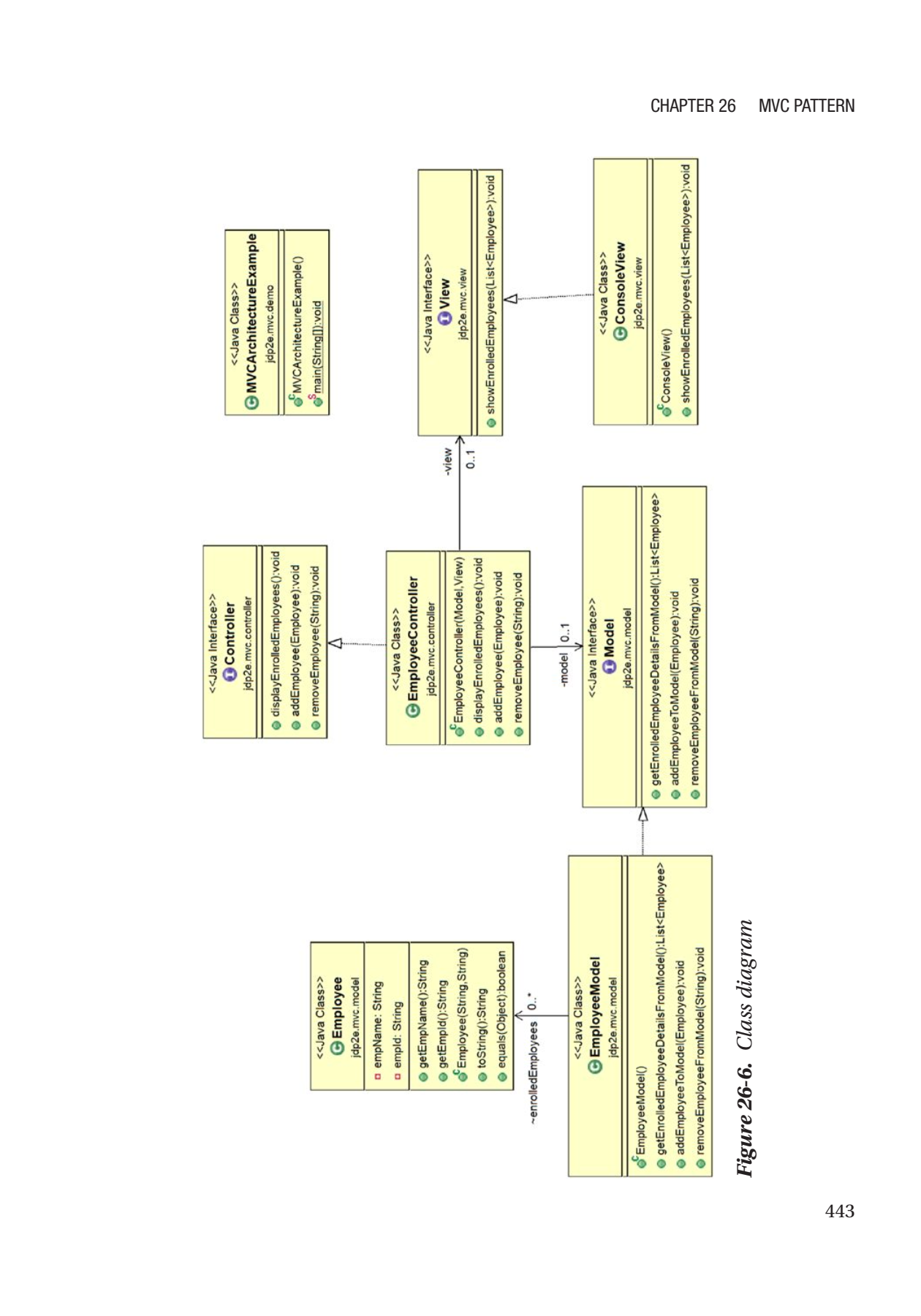
**Nama Mahasiswa : Muhammad Azhar Rasyad**

**NIM : 0110217029**

**Instruksi**

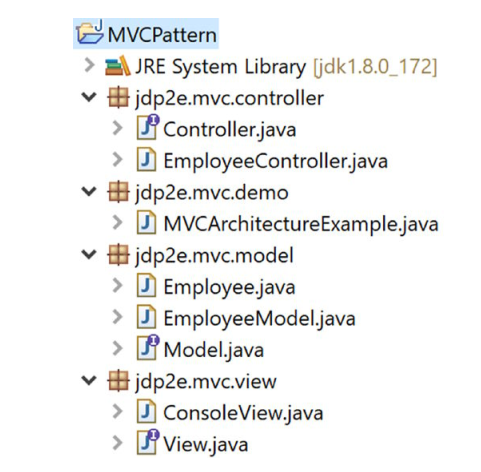
* Kerjakan tugas di bawah ini secara **individu.**
* Kerjakan secara **berurutan.**
* Modifikasi *source code*, dengan cara menambahkan **NIM** Anda pada setiap *caption* yang ditampilkan di *result* yang di*screenshot*.

1. **Perhatikan ilustrasi berikut ini:**



Gambar 1. Ilustrasi Relasi

1. **Pahamilah struktur package yang akan dibuat.**



Gambar 2. Package Explorer

1. **Buatlah package:** **jdp2e.mvc.model, kemudian tambahkan 3 kelas dengan sourcecode sebagai berikut:**

**Employee.java**

|  |
| --- |
| **package** jdp2e.mvc.model;  **public** **class** Employee {  **private** String empName;  **private** String empId;    **public** String getEmpName() {  **return** empName;  }    **public** String getEmpId() {  **return** empId;  }    **public** Employee(String empName, String empId)  {  **this**.empName=empName;  **this**.empId=empId;  }    @Override  **public** String toString()  {  **return** empName + "'s employee id is: "+ empId ;  }    @Override  //To check uniqueness.  **public** **boolean** equals(Object o) {  **if** (**this** == o) **return** **true**;  **if** (!(o **instanceof** Employee))  **return** **false**;  Employee empObject = (Employee) o;  **if** (!empName.equals(empObject.empName))  **return** **false**;  //cannot use the following for an int  **if** (!empId.equals(empObject.empId))  **return** **false**;  **return** **true**;  }  } |

**Model.java (Interface)**

|  |
| --- |
| **package** jdp2e.mvc.model;  **import** java.util.List;  //Model interface  **public** **interface** Model  {  List<Employee> getEnrolledEmployeeDetailsFromModel();  **void** addEmployeeToModel(Employee employeee);  **void** removeEmployeeFromModel(String employeeId);  } |

**EmployeeModel.java**

|  |
| --- |
| **package** jdp2e.mvc.model;  **import** java.util.ArrayList;  **import** java.util.List;  **import** java.util.ListIterator;    //EmployeeModel class  **public** **class** EmployeeModel **implements** Model  {  List<Employee> enrolledEmployees;    **public** EmployeeModel()  {  //Adding 3 employees at the beginning.  enrolledEmployees = **new** ArrayList<Employee>();  enrolledEmployees.add(**new** Employee("Amit","E1"));  enrolledEmployees.add(**new** Employee("John","E2"));  enrolledEmployees.add(**new** Employee("Sam","E3"));  }  **public** List<Employee> getEnrolledEmployeeDetailsFromModel()  {  **return** enrolledEmployees;  }  //Adding an employee to the model(student list)  @Override  **public** **void** addEmployeeToModel(Employee employee)  {  System.***out***.println("\nTrying to add an employee to the registered list.");    **if**( !enrolledEmployees.contains(employee))  {  enrolledEmployees.add(employee);  System.***out***.println(employee+" [added recently.]");  }  **else**  {  System.***out***.println(employee+" is already added in the system.");  }  }  //Removing an employee from model(student list)  @Override  **public** **void** removeEmployeeFromModel(String employeeId)  {  **boolean** flag=**false**;  ListIterator<Employee> employeeIterator=enrolledEmployees.listIterator();  System.***out***.println("\nTrying to remove an employee from the registered list.");  **while**(employeeIterator.hasNext())  {  Employee removableEmployee=((Employee)employeeIterator.next());  **if**(removableEmployee.getEmpId().equals(employeeId))  {  //To avoid ConcurrentModificationException,try to  //remember to invoke remove() on the iterator but not on  //the list.  employeeIterator.remove();  System.***out***.println("Employee " + removableEmployee.getEmpName()+ " with id "+ employeeId+" is removed now.");  flag=**true**;  }  }  **if**(flag==**false**)  {  System.***out***.println("###Employee Id " + employeeId +" Not found.###");  }  }  } |

1. **Lanjutkan dengan membuat package baru :** **jdp2e.mvc.view, dan tambahkan 2 kelas dengan sourcode sbb.,**

**View.java (Interface)**

|  |
| --- |
| **package** jdp2e.mvc.view;  **import** java.util.List;  **import** jdp2e.mvc.model.Employee;  **public** **interface** View  {  **void** showEnrolledEmployees(List<Employee> enrolledEmployees);  } |

**ConsoleView.java**

|  |
| --- |
| **package** jdp2e.mvc.view;  **import** java.util.List;  **import** jdp2e.mvc.model.Employee;  //ConsoleView class  **public** **class** ConsoleView **implements** View  {  @Override  **public** **void** showEnrolledEmployees(List<Employee> enrolledEmployees)  {  System.***out***.println("\n \*\*\*This is a console view of currently enrolled employees.\*\*\* ");  **for**( Employee employee : enrolledEmployees)  {  System.***out***.println(employee);  }  System.***out***.println("---------------------");  }  } |

1. **Lanjutkan dengan membuat package baru :** **jdp2e.mvc.controller, dan tambahkan 2 kelas dengan sourcode sbb.,**

**Controller.java (Interface)**

|  |
| --- |
| **package** jdp2e.mvc.controller;  **import** jdp2e.mvc.model.Employee;  //Controller  **public** **interface** Controller  {  **void** displayEnrolledEmployees();  **void** addEmployee(Employee employee);  **void** removeEmployee(String employeeId);  } |

**EmployeeController.java**

|  |
| --- |
| **package** jdp2e.mvc.controller;  **import** java.util.List;  **import** jdp2e.mvc.model.\*;  **import** jdp2e.mvc.view.\*;  **public** **class** EmployeeController **implements** Controller  {  **private** Model model;  **private** View view;    **public** EmployeeController(Model model, View view)  {  **this**.model = model;  **this**.view = view;  }  @Override  **public** **void** displayEnrolledEmployees()  {  //Get data from Model  List<Employee> enrolledEmployees = model.getEnrolledEmployeeDetailsFromModel();  //Connect to View  view.showEnrolledEmployees(enrolledEmployees);  }    //Sending a request to model to add an employee to the list.  @Override  **public** **void** addEmployee(Employee employee)  {  model.addEmployeeToModel(employee);  }  //Sending a request to model to remove an employee from the list.  @Override  **public** **void** removeEmployee(String employeeId)  {  model.removeEmployeeFromModel(employeeId);  }  } |

1. **Lanjutkan dengan membuat package baru:** **jdp2e.mvc.demo, dan menambahkan kelas client -> MVCArchitectureExample, dengan source sbb.:**

**MVCArchitectureExample.java**

|  |
| --- |
| **package** jdp2e.mvc.demo;  **import** jdp2e.mvc.model.\*;  **import** jdp2e.mvc.view.\*;  **import** jdp2e.mvc.controller.\*;  **public** **class** MVCArchitectureExample {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub    System.***out***.println("\*\*\*MVC architecture Demo\*\*\*\n");    //Model  Model model = **new** EmployeeModel();    //View  View view = **new** ConsoleView();    //Controller  Controller controller = **new** EmployeeController(model, view);  controller.displayEnrolledEmployees();    //Add an employee  controller.addEmployee(**new** Employee("Kevin","E4"));  controller.displayEnrolledEmployees();    //Remove an existing employee using the employee id.  controller.removeEmployee("E2");  controller.displayEnrolledEmployees();    //Cannot remove an employee who does not belong to the list.  controller.removeEmployee("E5");  controller.displayEnrolledEmployees();    //Avoiding duplicate entry  controller.addEmployee(**new** Employee("Kevin","E4"));  }  } |

1. **Update source pada langkah 6, dengan menampilkan NIM Anda. Kemudian jalankan *main* program tersebut.**

|  |
| --- |
| *Screenshot Hasil Running Main Program* |

1. Berikan penjelasan terkait proses implementasi *MVC Pattern* pada source 3,4,5, dan 6.

|  |
| --- |
| **Bismillah,** penjelasan dimulai dari kelas client MVCArchitectureExample. Pada client tersebut memanggil kelas Model, Controller, dan View, hal pertama yang dilakukan adalah menginisiasi Model. Model pada kasus ini ada 3 kelas yaitu Model (sebagai interface), Employee (sebagai model utama dalam menyimpan data employee), dan EmployeeModel (sebagai penghubung antara interface Model dengan data Employee). Selanjutnya menginisiasi View dan bukannya Controller, karena Controller dalam hal ini membutuhkan 2 komponen yaitu model dan view yang dapat dilihat pada kode berikut :    Kode di atas dapat diketahui bahwa Controller membutuhkan adanya model dan view terlebih dahulu sebelum dibuat sebuah Controller. Kembali ke View sebelumnya, pada kasus ini terdapat 2 kelas yaitu View (sebagai interface) dan ConsoleView (sebagai tampilan yang dilihat user) sementara dalam Controller juga terdapat 2 kelas yaitu Controller (sebagai interface) dan EmployeeController (sebagai penghubung antara EmployeeModel dengan ConsoleView).  Langkah selanjutnya setelah inisiasi Model yaitu menggunakan function yang sudah dibuat di Controller untuk dapat diimplementasikan ke dalam View, diantaranya function yang digunakan pada client yaitu :   * displayEnrolledEmployees() berfungsi untuk menampilkan employee yang telah terdaftar. * addEmployee() berfungsi menambahkan employee. * removeEmployee() berfungsi mengeluarkan employee.   Berikut implementasi kode program dari ketiga function di atas    Implementasi dimulai dengan menampilkan daftar employee yang disimpan pada kelas EmployeeModel berikut    dan setelah itu akan ditampilkan pada ConsoleView, namun ada kendala dimana ConsoleView tidak dapat terhubung langsung dengan EmployeeModel karena tidak ada penghubungnya dalam kasus ini. Sehingga dibutuhkannya EmployeeController dalam mengatasi hal tersebut dengan ilustrasi berikut ini  EmployeeModel -> EmployeeController -> ConsoleView  Alhasil menampilkan hasil seperti ini    Setelah ditampilkan ada implementasi lagi yaitu addEmployee, function ini sama halnya seperti sebelumnya yaitu jika ingin menambahkan employee maka data akan disimpan ke dalam EmployeeModel dengan kode program berikut    Karena function addEmployee() terletak pada EmployeeController sehingga dapat langsung terhubung ke EmployeeModel dengan kode berikut    Dapat dikatakan alur kerjanya yaitu addEmployee pada EmployeeController merequest addEmployeeToModel yang ada pada EmployeeModel kemudian addEmployeeToModel mengembalikan kembali ke addEmployee, dengan ilustrasi berikut  EmployeeController -> EmployeeModel -> EmployeeController  atau  EmployeeController <--> EmployeeModel  Tampilan yang dihasilkan seperti berikut jika berhasil dilakukan penambahan employee    dan berikut tampilan jika tidak berhasil menambahkan employee karena sudah terdaftar    Implementasi terakhir yaitu removeEmployee dimana sama seperti sebelumnya addEmployee bekerja, dengan menggunakan data pada EmployeeModel dan dipanggil melalui EmployeeController, berikut kedua kode programnya      Berikut tampilannya jika berhasil dikeluarkan    dan jika tidak berhasil dikeluarkan    Demikian penjelasan yang dapat saya uraikan, apabila ada kekeliruan mohon maaf, sekian terima kasih. |

1. Berdasarkan sourcecode pada praktikum ini, variasi pattern MVC nomor berapakah yang diimplementasikan? (cek variasi implementasi di slide materi). Berikan penjelasan Anda!

|  |
| --- |
| Menurut saya termasuk ke dalam variasi implementasi ke 2    Karena sebelumnya terlihat bahwa Model tidak dapat berinteraksi langsung dengan View melainkan harus melalui Controller begitupun sebaliknya, dan jika dilihat pada variasi ke 1 dan ke 3 dapat dikatakan Model dapat langsung berinteraksi dengan View tanpa adanya perantara. |

**\*\*\***