



LAPORAN PRAKTIKUM

Halaman
1 of 30

A. IDENTITAS

NIM : 2403099
Nama Lengkap : AHMAD MAUALAN KUDUS
Kelas : D3 TI2B
Program Studi : D3 TEKNIK INFORMATIKA
Jurusan : TEKNIK INFORMATIKA

B. DESKRIPSI MATA KULIAH

Nama Matakuliah : Pemograman Berorientasi Objek
Dosen Pengampu : Fachrul Pralienka Bani Muhamad, M.Kom.
Pertemuan ke : ~~1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16~~
Materi : Dasar Pemrograman Java
Minggu : Sembilan
Ruang : Lab. Pemograman & R.501

C. HASIL PRAKTIKUM

A. Layered Architecture Pattern

Pola arsitektur yang mengatur komponen-komponen aplikasi menjadi beberapa lapisan (layer) terpisah secara logis dengan menerapkan prinsip pemisahan tanggung jawab (separation of concerns)

- Presentation Layer
Lapisan yang berinteraksi dengan pengguna.
 - Presentation Pattern
 - 1. Menampilkan data
 - 2. Menerima input
 - 3. Validasi input
- Business Logic / Application Layer
Lapisan (inti) logika bisnis yang mendefinisikan apa yang dilakukan aplikasi.
 - Domain Logic Pattern
 - 1. Menerapkan aturan bisnis spesifik
 - 2. Manajemen transaksi
 - 3. Memastikan konsistensi data
- Persistence / Data Access Layer



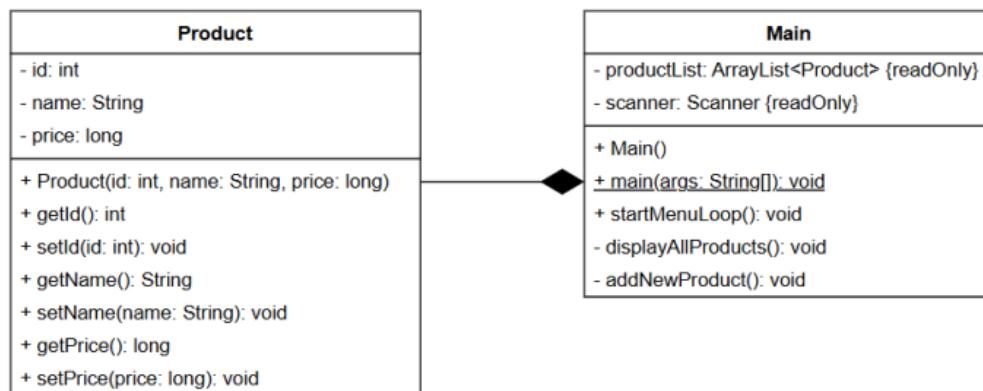
Lapisan yang menjembatani abstraksi bisnis dari detail penyimpanan data.

- Data Persistence Pattern
 - 1. Mengelola data (CRUD)
 - 2. Akses database (querying)
 - 3. Mengonversi data dari format database
- Database / Data Source Layer
Lapisan fisik penyimpanan data (database server)
 - Pattern
 - 1. Penyimpanan data
 - 2. Mengelola izin akses database
 - 3. Server DB fisik (MySQL, PostgreSQL, atau lainnya)

B. kode program TANPA PATTERN

1. Class Diagram

Class diagram



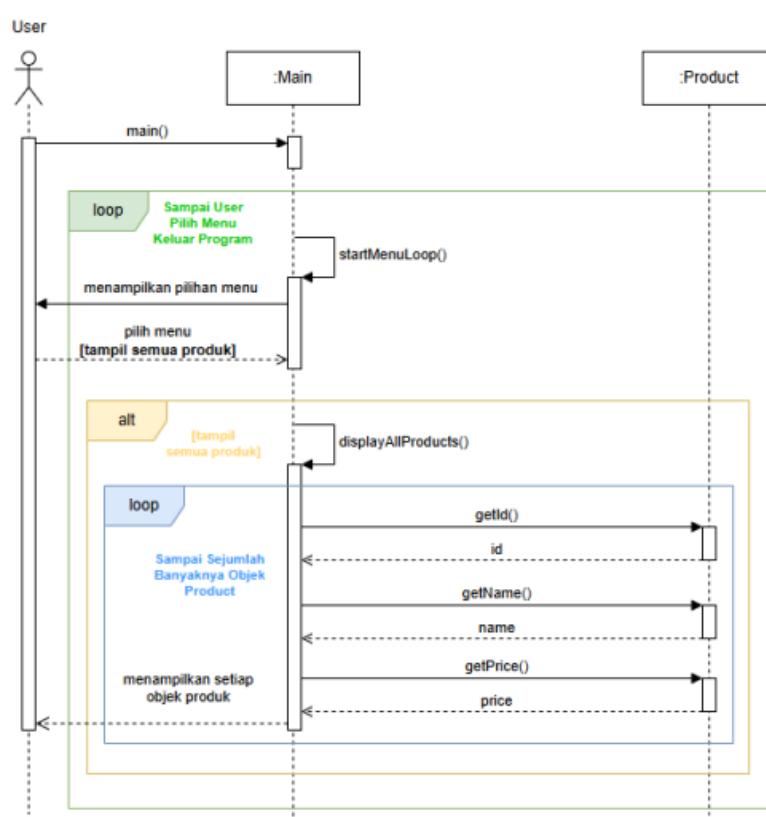
2. Sequence Diagram

- Menampilkan semua daftar produk

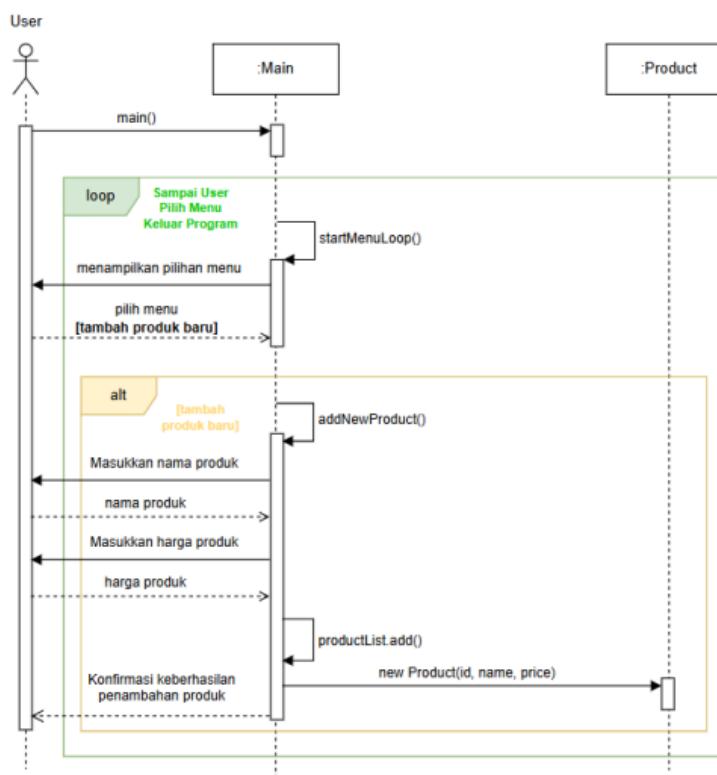


LAPORAN PRAKTIKUM

Halaman
3 of 30



- Menambah produk baru



3. Kode program

a. Product.java

Buat folder Bernama “pettern-nvc-good”, di dalam folder tersebut buat folder “model” di dalam foldernya buat file “Product.java” isi kode programnya dibawah ini :

```
EXPLORER ... :Controller.java Main.java Product.java ProductService.java ProductServiceDefault.java ProductConsoleView.java ...
PATTERN-NVC-GOOD
controller ProductController.java
model Product.java
service ProductService.java
view ProductConsoleView.java
Main.java

model > Product.java > Product > Product(int, String, long)
1 package model;
2
3 public class Product{
4     private int id;
5     private String name;
6     private long price;
7
8     public Product(int id, String name, long price) {
9         this.id = id;
10        this.name = name;
11        this.price = price;
12    }
13
14    public int getId() {
15        return id;
16    }
17
18    public void setId(int id) {
19        this.id = id;
20    }
21
22    public String getName() {
23        return name;
24    }
25
26    public void setName(String name) {
27        this.name = name;
28    }
}
```



```
29
30     public double getPrice() {
31         return price;
32     }
33
34     public void setPrice(long price) {
35         this.price = price;
36     }
37
38
39
40 }
```

b. Main.java

```
import java.util.ArrayList;
import java.util.Scanner;
import model.Product;

public class Main {

    private final ArrayList<Product> productList = new ArrayList<>();
    private final Scanner scanner = new Scanner(System.in);

    public Main() {
        // Data awal
        productList.add(new Product(id: 1, name: "Laptop ASUS", price: 9500000));
        productList.add(new Product(id: 2, name: "Monitor Dell", price: 2500000));
    }

    public void startMenuLoop() {
        boolean running = true;
        while (running) {
            System.out.println(x: "\n--- APLIKASI TANPA PATTERN ---");
            System.out.println(x: "1. Tampilkan Semua Produk");
            System.out.println(x: "2. Tambah Produk Baru");
            System.out.println(x: "3. Keluar");
            System.out.print(s: "Pilih opsi: ");
            try {
                int choice = Integer.parseInt(scanner.nextLine());
                switch (choice) {
                    case 1:
                        displayAllProducts();
                        break;
                }
            } catch (Exception e) {
                System.out.println(x: "Pilihan tidak valid");
            }
        }
    }
}
```



LAPORAN PRAKTIKUM

Halaman
6 of 30

```
        case 2:
            addNewProduct();
            break;
        case 3:
            running = false;
            System.out.println(x: "Terima kasih telah menggunakan aplikasi!");
            break;
        default:
            System.out.println(x: "Opsi tidak valid.");
    }
} catch (NumberFormatException e) {
    System.out.println(x: "Input tidak valid. Masukkan angka.");
}
scanner.close();
}

private void displayAllProducts() {
    System.out.println(x: "\n--- Daftar Produk ---");
    if (productList.isEmpty()) {
        System.out.println(x: "Tidak ada produk tersedia.");
    } else {
        for (Product product : productList) {
            System.out.println(product.getId() + " - " + product.getName() + " Rp. " + product.getPrice());
        }
    }
}
```

```
private void addNewProduct() {
    System.out.print(s: "Masukkan Nama Produk: ");
    String name = scanner.nextLine();

    if (name.trim().isEmpty()) {
        System.out.println(x: "Error: Nama produk tidak boleh kosong.");
        return;
    }

    System.out.print(s: "Masukkan Harga Produk: ");
    String priceString = scanner.nextLine();
    try {
        long price = Long.parseLong(priceString);
        if (price <= 0) {
            throw new IllegalArgumentException(s: "Harga harus angka positif di atas nol!");
        }
        int newId = productList.size() + 1;
        productList.add(new Product(newId, name, price));
        System.out.println(x: "Produk berhasil ditambahkan!");
    } catch (NumberFormatException e) {
        System.out.println(x: "Error: Harga tidak valid. Masukkan angka.");
    } catch (IllegalArgumentException e) {
        System.out.println("Error: " + e.getMessage());
    }
}

Run | Debug
public static void main(String[] args) {
    Main app = new Main();
```

```
        Main app = new Main();
        app.startMenuLoop();

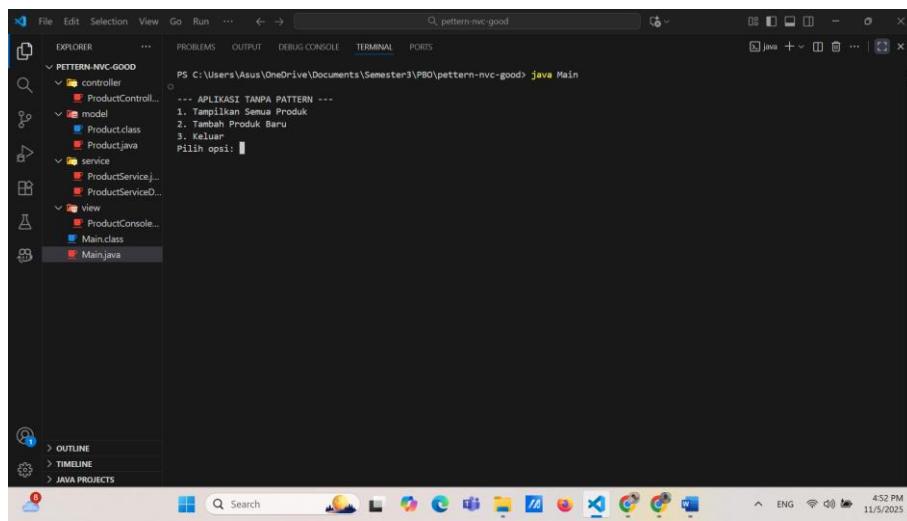
    }
```

4. Compile & Run



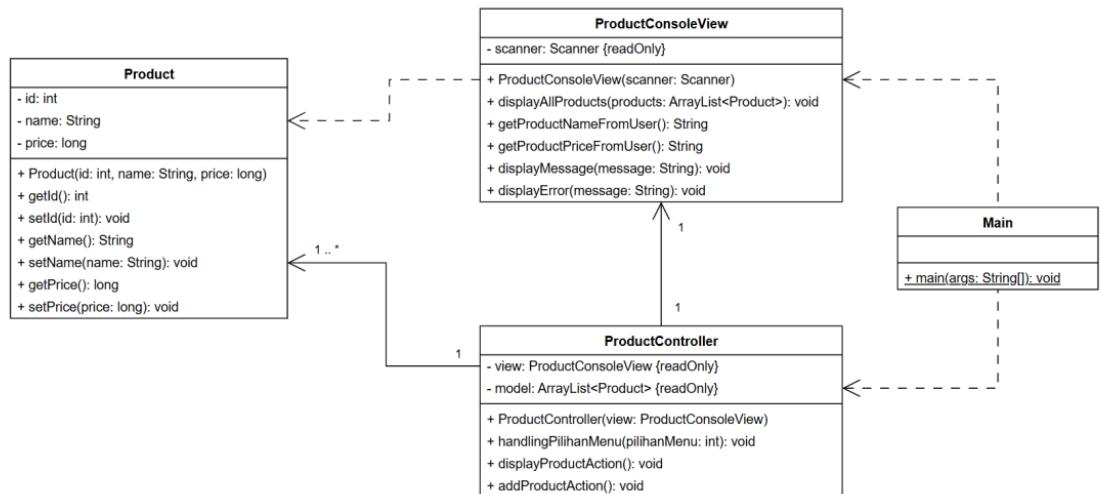
LAPORAN PRAKTIKUM

Halaman
7 of 30



C. kode program MVC Pattern

1. Class Diagram



2. Kode program

a. Product.java



LAPORAN PRAKTIKUM

Halaman
8 of 30

```
PATTERN-NVG-GOOD
controller
model
service
view
Main.java

model > Product.java > Product > Product(int, String, long)
1 package model;
2
3 public class Product{
4     private int id;
5     private String name;
6     private long price;
7
8     public Product(int id, String name, long price) {
9         this.id = id;
10        this.name = name;
11        this.price = price;
12    }
13
14    public int getId() {
15        return id;
16    }
17
18    public void setId(int id) {
19        this.id = id;
20    }
21
22    public String getName() {
23        return name;
24    }
25
26    public void setName(String name) {
27        this.name = name;
28    }
29
30    public double getPrice() {
31        return price;
32    }
33
34    public void setPrice(long price) {
35        this.price = price;
36    }
37
38
39
40 }
```

b. ProductConsoleView

```
view > ProductConsoleView.java > ProductConsoleView
1 package view;
2
3 import java.util.ArrayList;
4 import java.util.Scanner;
5 import model.Product;
6
7 public class ProductConsoleView {
8     private final Scanner scanner;
9     public ProductConsoleView(Scanner scanner) {
10         this.scanner = scanner;
11     }
12
13     public void displayAllProducts(ArrayList<Product> products) {
14         System.out.println("----- Daftar Produk -----");
15         if (products.isEmpty()) {
16             System.out.println("Tidak ada produk tersedia.");
17         } else {
18             for (Product product : products) {
19                 System.out.println(product.getId() + " - " + product.getName() + " Rp. " + product.getPrice());
20             }
21         }
22     }
23
24     public String getProductNameFromUser () {
25         System.out.print("Masukkan Nama Produk: ");
26         return scanner.nextLine();
27     }
28     public String getProductPriceFromUser () {
29         System.out.print("Masukkan Harga Produk: ");
30         return scanner.nextLine();
```



```
    }
    public void displayMessage (String message){
        System.out.println("INFO: " + message);
    }
    public void displayError (String message){
        System.out.println("ERROR: " + message);
    }
}
```

c. ProductController.java

```
controller > ProductController.java > ProductController > ProductController(ProductConsoleView)
1 package controller;
2
3
4 import view.ProductConsoleView;
5 import java.util.ArrayList;
6 import model.Product;
7
8 public class ProductController{
9     private final ProductConsoleView view;
10    private final ArrayList<Product> model = new ArrayList<>();
11
12    public ProductController(ProductConsoleView view){
13        this.view = view;
14        model.add(new Product(id: 1, name: "Laptop ASUS", price: 950));
15    }
16
17    public void handleMenuChoice(int choice){
18        switch (choice) {
19            case 1:
20                displayProductsAction();
21                break;
22            case 2:
23                addProduct();
24                break;
25            case 3:
26                view.displayMessage(message: "keluar dari aplikasi");
27                break;
28            default:
29                view.displayError(message: "pilihan tidak valid");
30            }
31        }
32
33    private void displayProductsAction(){
34        view.displayAllProducts(model);
35    }
36
37    private void addProduct(){
38        String name = view.getProductFromUser();
39        String priceStr = view.getProductPriceFromUser();
40        try{
41            double price = Double.parseDouble(priceStr);
42            if (price <= 0 ) {
43                throw new IllegalArgumentException(s: "harga harus positif lebih dari 0 ");
44            }
45            int newId = model.size()+1;
46            model.add(new Product(newId, name, (long) price));
47            view.displayMessage(message: "Produk ditambahkan");
48        } catch(IllegalArgumentException e){
49            view.displayError("gagal menambah produk: "+ e.getMessage());
50        }
51    }
52 }
```

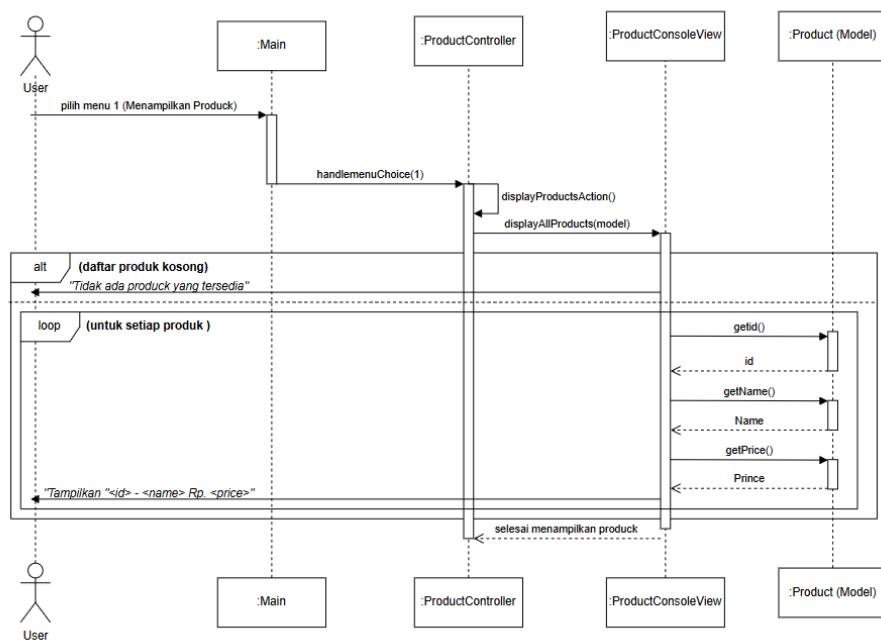
3. Sequence Diagram

- Menampilkan semua peroduct

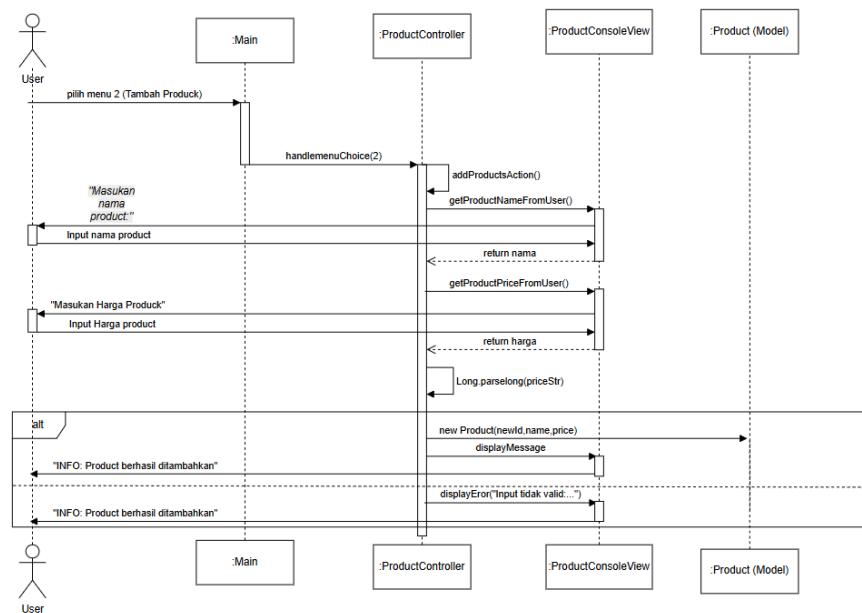


LAPORAN PRAKTIKUM

Halaman
10 of 30



b. Menambahkan Product

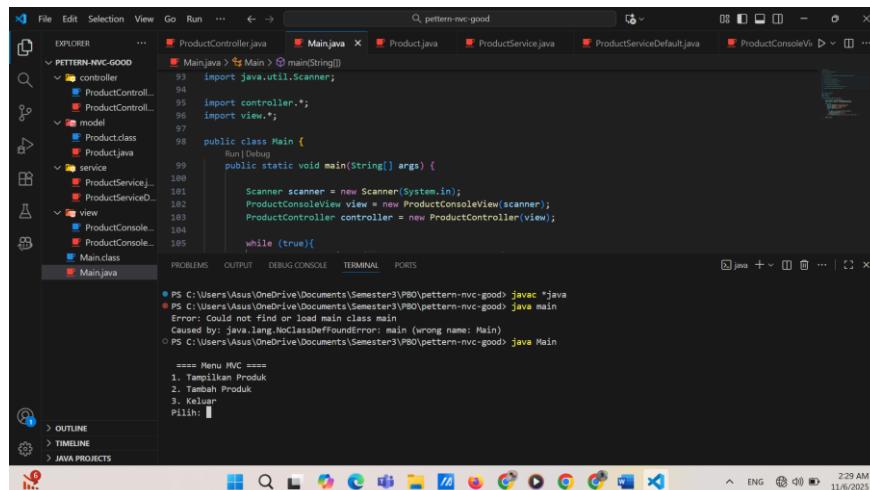


d. Main.java



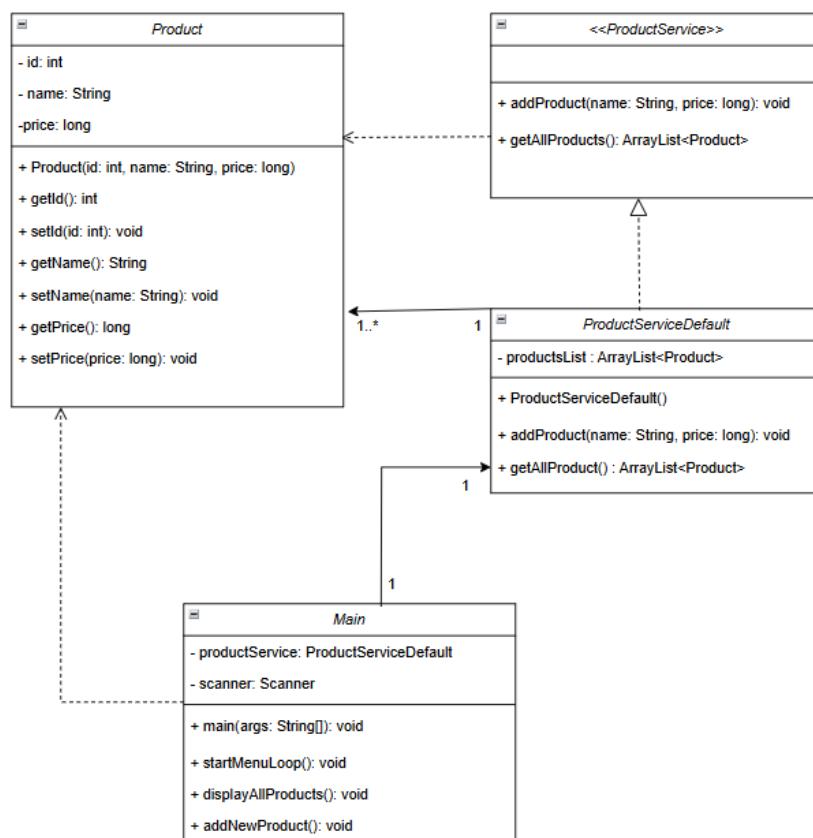
```
93 import java.util.Scanner;
94
95 import controller.*;
96 import view.*;
97
98 public class Main {
99     Run | Debug
100    public static void main(String[] args) {
101        Scanner scanner = new Scanner(System.in);
102        ProductConsoleView view = new ProductConsoleView(scanner);
103        ProductController controller = new ProductController(view);
104
105        while (true){
106            System.out.println(x: "\n === Menu MVC ===");
107            System.out.println(x: "1. Tampilkan Produk");
108            System.out.println(x: "2. Tambah Produk");
109            System.out.println(x: "3. Keluar");
110            System.out.print(s: "Pilih: ");
111
112            try {
113                int choice = Integer.parseInt(scanner.nextLine());
114                if (choice == 3 ) break;
115                controller.handleMenuChoice(choice);
116            } catch (Exception e) {
117                view.displayError(message: "input tidak valid. masukan angka. ");
118            }
119        }
120        scanner.close();
121    }
122}
```

4. Compile & Run



D. kode program Service Layer Pattern

1. Class Diagram



2. Kode program

a. Product.java

```
EXPLORER ... :Controller.java Main.java Product.java ProductService.java ProductServiceDefault.java ProductConsoleView.java ...
PATTERN-NVC-GOOD
controller ProductController.java
model Product.java
service ProductService.java
view ProductConsoleView.java
Main.java

Product.java
package model;
public class Product{
    private int id;
    private String name;
    private long price;

    public Product(int id, String name, long price) {
        this.id = id;
        this.name = name;
        this.price = price;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

The screenshot shows the Eclipse IDE interface with the Product.java file open in the editor. The code defines a `Product` class with attributes `id`, `name`, and `price`, and methods for setting and getting these values.



```
29
30     public double getPrice() {
31         return price;
32     }
33
34     public void setPrice(long price) {
35         this.price = price;
36     }
37
38
39
40 }
```

b. ProductService.java

```
service >  ProductService.java > ...
1  package service;
2
3  import java.util.ArrayList;
4  import model.Product;
5
6  public interface ProductService {
7
8      void addProduct(String name, long price);
9
10     ArrayList<Product> getAllProducts();
11
12 }
13
```

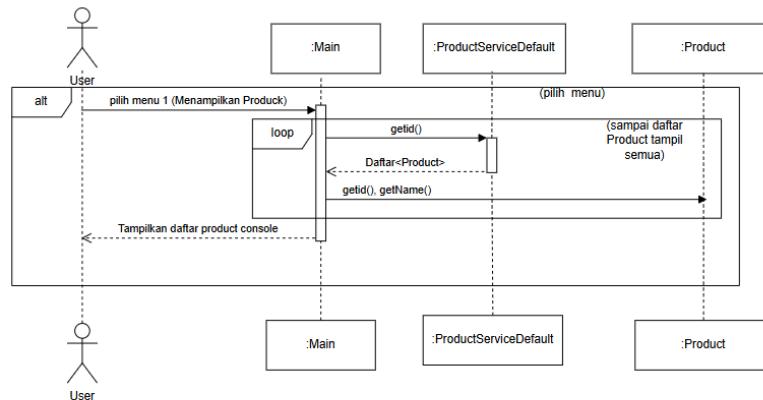
c. ProductServiceDefault.java



```
service > ProductServiceDefault.java > ...
1 package service;
2
3 import java.util.ArrayList;
4 import model.Product;
5
6 public class ProductServiceDefault implements ProductService {
7
8     private final ArrayList<Product> productList = new ArrayList<>();
9
10    public ProductServiceDefault() {
11        productList.add(new Product(id: 1, name: "Laptop ASUS", price: 9500000));
12        productList.add(new Product(id: 2, name: "Monitor Dell", price: 2500000));
13    }
14
15    @Override
16    public void addProduct(String name, long price) {
17        if (price <= 0) {
18            throw new IllegalArgumentException(
19                s: "Harga harus angka positif lebih dari 0");
20        }
21        int newId = productList.size() + 1;
22        productList.add(new Product(newId, name, price));
23    }
24
25    @Override
26    public ArrayList<Product> getAllProducts() {
27        return productList;
28    }
29 }
```

3. Sequence Diagram

a. Lihat Semua Product

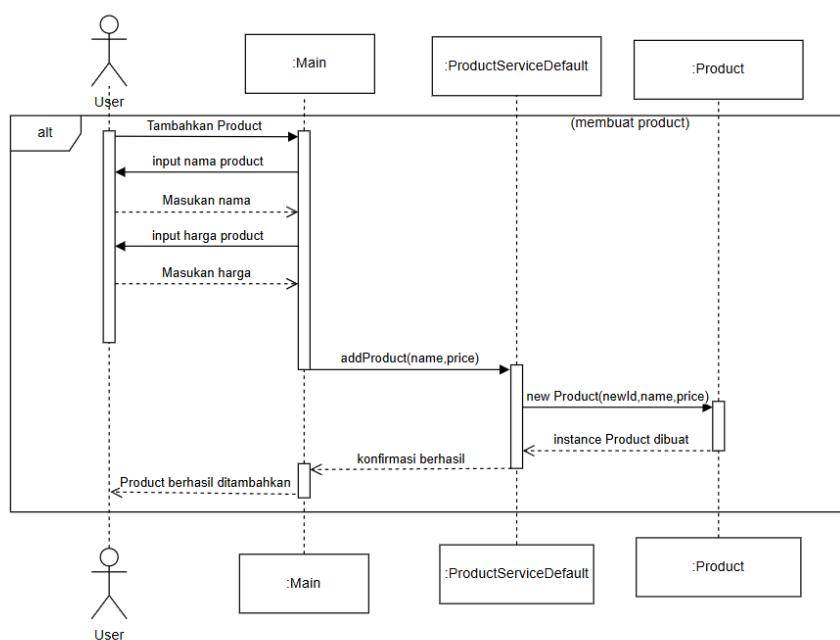


b. Tambahkan Product



LAPORAN PRAKTIKUM

Halaman
15 of 30



d. Main.java

```
127 import java.util.List;
128 import java.util.Scanner;
129
130 import model.Product;
131 import service.ProductServiceDefault;
132
133 public class Main {
134
135     private final ProductServiceDefault productService = new ProductServiceDefault();
136     private final Scanner scanner = new Scanner(System.in);
137
138     Run | Debug
139     public static void main(String[] args) {
140         Main app = new Main();
141         app.startMenuLoop();
142         app.scanner.close();
143     }
144
145     public void startMenuLoop() {
146         boolean running = true;
147         while (running) {
148             System.out.println(x: "\n--- APLIKASI SERVICE LAYER PATTERN ---");
149             System.out.println(x: "1. Tampilkan Semua Produk");
150             System.out.println(x: "2. Tambah Produk Baru");
151             System.out.println(x: "3. Keluar");
152             System.out.print(s: "Pilih opsi: ");
153
154             try {
155                 int choice = Integer.parseInt(scanner.nextLine());
156                 switch (choice) {
```



LAPORAN PRAKTIKUM

Halaman
16 of 30

```
156         case 1:
157             displayAllProducts();
158             break;
159         case 2:
160             addNewProduct();
161             break;
162         case 3:
163             running = false;
164             System.out.println("Terima kasih telah menggunakan aplikasi!");
165             break;
166         default:
167             System.out.println("Opsi tidak valid.");
168     }
169 } catch (NumberFormatException e) {
170     System.out.println("Input tidak valid. Masukkan angka.");
171 }
172 }
173 }
174
175 private void displayAllProducts() {
176     System.out.println("\n--- Daftar Produk ---");
177     List<Product> products = productService.getAllProducts();
178     for (Product product : products) {
179         System.out.println(product.getId() + " - " + product.getName()
180             + " - Rp. " + product.getPrice());
181     }
182 }
```

e. Compile & Run

```
File Edit Selection View Go Run ... ← → 🔍 pATTERN-NVC-GOOD Mainjava ProductController.java ProductService.java ProductServiceDefault.java ProductConsoleView.java
EXPLORER PETERN-NVC GOOD
controller
model
service
view
Main.java
Main.java > ...
133 public class Main {
134     ...
135     private void displayAllProducts() {
136         ...
137     }
138
139     private void addNewProduct() {
140         System.out.print("Masukkan Nama Produk: ");
141         String name = scanner.nextLine();
142         System.out.print("Masukkan Harga Produk: ");
143         String priceString = scanner.nextLine();
144
145         try {
146             long price = Long.parseLong(priceString);
147             productService.addProduct(name, price);
148         } catch (Exception e) {
149             System.out.println("Terjadi kesalahan saat menambahkan produk.");
150         }
151     }
152
153     private void removeProduct() {
154         ...
155     }
156
157     private void updateProduct() {
158         ...
159     }
160
161     private void exit() {
162         ...
163     }
164
165     public static void main(String[] args) {
166         Main.main(args);
167     }
168 }
169
170 PS Focus folder in explorer (ctrl + click) ts\Semester3\PBO\pattern-nvc-good
171 PS C:\Users\Asus\OneDrive\Documents\Semester3\PBO\pattern-nvc-good> java Main
172
173 --- APLIKASI SERVICE LAYER PATTERN ---
174 1. Tampilkan Semua Produk
175 2. Tambah Produk Baru
176 3. Keluar
177 Pilih opsi: 
```

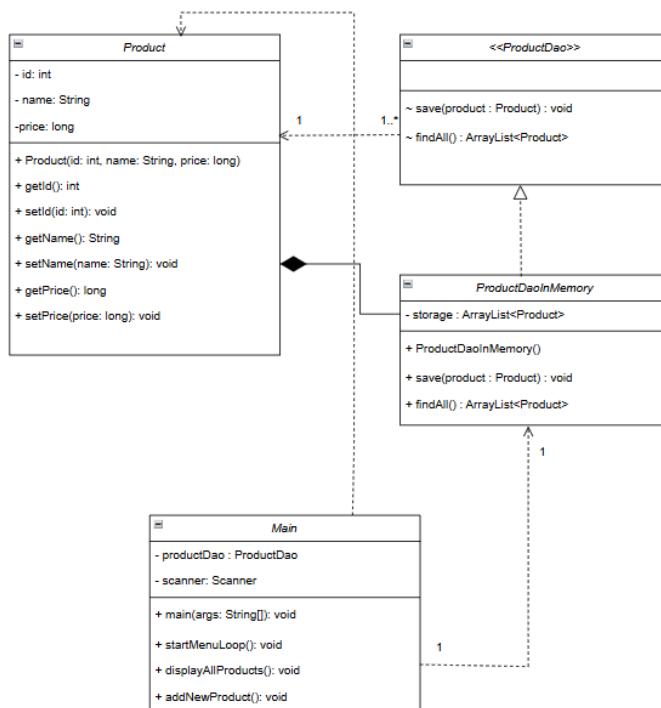
E. kode program DAO Pattern

1. Class Diagram



LAPORAN PRAKTIKUM

Halaman
17 of 30



2. Kode program

a. Product.java

```
EXPLORER ... :Controller.java Main.java Product.java X ProductService.java ProductServiceDefault.java ProductConsoleView.java ...
PATTERN-NVG-GOOD
controller ProductController.java
model Product.java
service ProductService.java
view ProductConsoleView.java
Main.java

model > Product.java > Product.java > Product(int, String, long)
1 package model;
2
3 public class Product{
4     private int id;
5     private String name;
6     private long price;
7
8     public Product(int id, String name, long price) {
9         this.id = id;
10        this.name = name;
11        this.price = price;
12    }
13
14    public int getId() {
15        return id;
16    }
17
18    public void setId(int id) {
19        this.id = id;
20    }
21
22    public String getName() {
23        return name;
24    }
25
26    public void setName(String name) {
27        this.name = name;
28    }
}
```

The screenshot shows the Eclipse IDE interface with the Product.java file open in the editor. The code defines a **Product** class with attributes for id, name, and price, and methods for Product constructor, getId, setId, getName, setName, and setPrice. The code is annotated with line numbers from 1 to 28.



```
29
30     public double getPrice() {
31         return price;
32     }
33
34     public void setPrice(long price) {
35         this.price = price;
36     }
37
38
39
40 }
```

b. ProductDao.java

```
dao >  ProductDao.java > ...
1 package dao;
2
3 import java.util.ArrayList;
4 import model.Product;
5
6 public interface ProductDao {
7     void save(Product product);
8     ArrayList<Product> findAll();
9 }
10
```

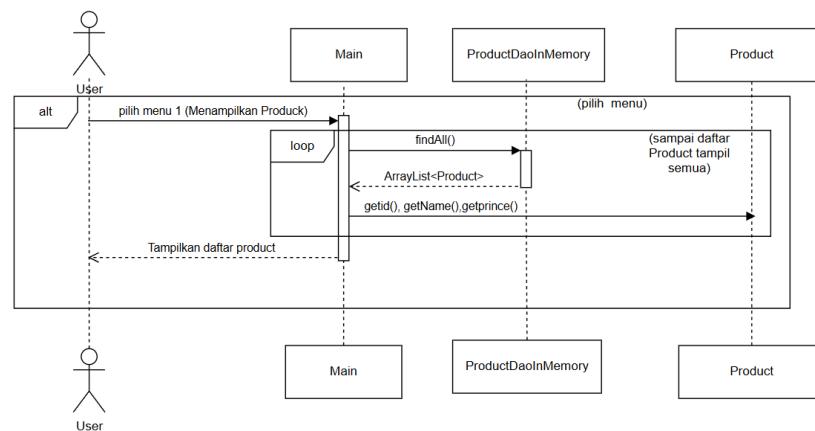
c. ProductDaoMemoy.java



```
dao > memory > ProductDaoInMemory.java > ...
1  package dao.memory;
2
3  import java.util.ArrayList;
4
5  import dao.ProductDao;
6  import model.Product;
7
8  public class ProductDaoInMemory implements ProductDao {
9
10     private final ArrayList<Product> storage = new ArrayList<>();
11
12     public ProductDaoInMemory() {
13         storage.add(new Product(id: 1, name: "Laptop ASUS", price: 9500000));
14         storage.add(new Product(id: 2, name: "Monitor Dell", price: 2500000));
15     }
16
17     @Override
18     public void save(Product product) {
19         storage.add(product);
20     }
21
22     @Override
23     public ArrayList<Product> findAll() {
24         return storage;
25     }
26
27 }
```

3. Sequence Diagram

a. Melihat semua product

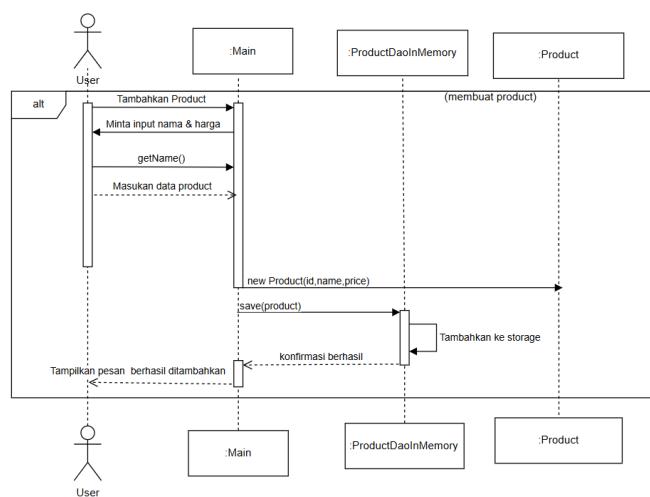


b. Menambahkan Product



LAPORAN PRAKTIKUM

Halaman
20 of 30



d. Main.java

```
206 import java.util.ArrayList;
207 import java.util.Scanner;
208
209 import dao.ProductDao;
210 import dao.memory.ProductDaoInMemory;
211 import model.Product;
212
213 public class Main {
214     private final ProductDao productDao = new ProductDaoInMemory();
215     private final Scanner scanner = new Scanner(System.in);
216
217     Run | Debug
218     public static void main(String[] args) {
219         Main app = new Main();
220         app.startMenuLoop();
221         app.scanner.close();
222     }
223
224     public void startMenuLoop() {
225         boolean running = true;
226         while (running) {
227             System.out.println(x: "\n--- APLIKASI DAO PATTERN ---");
228             System.out.println(x: "1. Tampilkan Semua Produk");
229             System.out.println(x: "2. Tambah Produk Baru");
230             System.out.println(x: "3. Keluar");
231             System.out.print(s: "Pilih opsi: ");
232             try {
233                 int choice = Integer.parseInt(scanner.nextLine());
234                 switch (choice) {
```



```
234
235         case 1:
236             displayProducts();
237             break;
238         case 2:
239             addNewProduct();
240             break;
241         case 3:
242             running = false;
243             break;
244         default:
245             System.out.println(x: "Opsi tidak valid.");
246     }
247 } catch (Exception e) {
248     System.out.println("Input error: " + e.getMessage());
249 }
250 }
251
252 private void displayProducts() {
253     System.out.println(x: "\n--- Daftar Produk ---");
254     ArrayList<Product> products = productDao.findAll();
255     for (Product product : products) {
256         System.out.println(product);
257     }
258 }
259
260 private void addNewProduct() {
261     System.out.print(s: "Nama Produk: ");
262     String name = scanner.nextLine();
263     System.out.print(s: "Harga Produk: ");
264     long price = Long.parseLong(scanner.nextLine());
265     if (price <= 0) {
266         System.out.println(x: "Error: Harga harus angka positif lebih dari 0.");
267         return;
268     }
269     int newId = productDao.findAll().size() + 1;
270     Product新产品 = new Product(newId, name, price);
271     productDao.save(新产品);
272     System.out.println(x: "Produk berhasil ditambahkan!");
273 }
274 }
```

e. Compile & Run

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure under "PATTERN-NVC-GOOD". The "src" folder contains "Main.java", "ProductDAOInMemory.java", "ProductDAO.java", "Product.java", "ProductService.java", and "ProductServiceDefault.java". The "controller" and "view" folders are expanded, showing their respective sub-components.
- Code Editor (Center):** Displays the content of "Main.java". The code implements a DAO pattern for products, using a scanner for input and System.out.println for output. It includes methods for displaying all products and adding new products, with validation for price.
- Terminal (Bottom):** Shows two command-line sessions:
 - PS C:\Users\Asus\OneDrive\Documents\Semester3\PBO\pattern-nvc-good: **java** *java
 - PS C:\Users\Asus\OneDrive\Documents\Semester3\PBO\pattern-nvc-good: **java** Main
- Output (Bottom):** Displays the application's response:

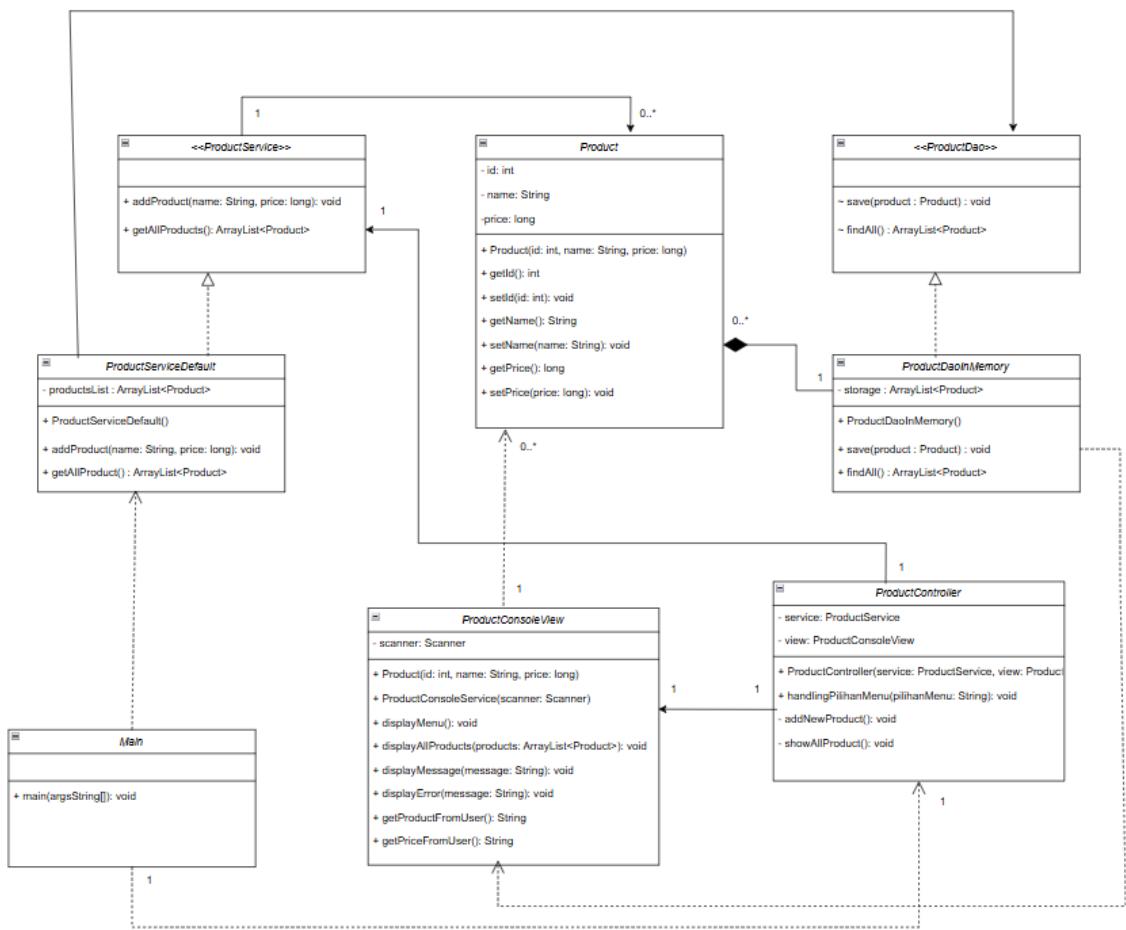
--- APLIKASI DAO PATTERN ---
1. Tampilkan Semua Produk
2. Tambah Produk Baru
3. Keluar

Pilih opsi: []



F. kode program MVC + Service Layer + DAO Pattern

1. Class Diagram



2. Kode program

a. Product.java



LAPORAN PRAKTIKUM

Halaman
23 of 30

```
EXPLORER ... :Controller.java Main.java Product.java ProductService.java ProductServiceDefault.java ProductConsoleView.java D v ...
PATTERN-NVG-GOOD model > Product.java > Product > Product(int, String, long)
controller ProductController.java
model Product.java
service ProductService.java
view ProductConsoleView.java
Main.java

public class Product{
    private int id;
    private String name;
    private long price;

    public Product(int id, String name, long price) {
        this.id = id;
        this.name = name;
        this.price = price;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(long price) {
        this.price = price;
    }
}
```

b. ProductDao.java

```
dao > ProductDao.java > ...
1 package dao;
2
3 import java.util.ArrayList;
4 import model.Product;
5
6 public interface ProductDao {
7     void save(Product product);
8     ArrayList<Product> findAll();
9 }
10
```



c. ProductDaoMemory.java

```
dao > memory > ProductDaoInMemory.java > ...
1  package dao.memory;
2
3  import java.util.ArrayList;
4
5  import dao.ProductDao;
6  import model.Product;
7
8  public class ProductDaoInMemory implements ProductDao {
9
10     private final ArrayList<Product> storage = new ArrayList<>();
11
12     public ProductDaoInMemory() {
13         storage.add(new Product(id: 1, name: "Laptop ASUS", price: 9500000));
14         storage.add(new Product(id: 2, name: "Monitor Dell", price: 2500000));
15     }
16
17     @Override
18     public void save(Product product) {
19         storage.add(product);
20     }
21
22     @Override
23     public ArrayList<Product> findAll() {
24         return storage;
25     }
26 }
27 }
```

d. ProductService.java

```
service > ProductService.java > ...
1  package service;
2
3  import java.util.ArrayList;
4  import model.Product;
5
6  public interface ProductService {
7
8      void addProduct(String name, long price);
9
10     ArrayList<Product> getAllProducts();
11
12 }
13 }
```

e. ProductServicedefault.java



LAPORAN PRAKTIKUM

Halaman
25 of 30

```
service > ProductServiceDefault.java > ...
1 package service;
2
3 import java.util.ArrayList;
4 import model.Product;
5
6 public class ProductServiceDefault implements ProductService {
7
8     private final ArrayList<Product> productList = new ArrayList<>();
9
10    public ProductServiceDefault() {
11        productList.add(new Product(id: 1, name: "Laptop ASUS", price: 9500000));
12        productList.add(new Product(id: 2, name: "Monitor Dell", price: 2500000));
13    }
14
15    @Override
16    public void addProduct(String name, long price) {
17        if (price <= 0) {
18            throw new IllegalArgumentException(
19                s: "Harga harus angka positif lebih dari 0");
20        }
21        int newId = productList.size() + 1;
22        productList.add(new Product(newId, name, price));
23    }
24
25    @Override
26    public ArrayList<Product> getAllProducts() {
27        return productList;
28    }
29 }
```

f. ProductServiceDefault.java

```
service > ProductServiceDefault.java > ...
1 package service;
2
3 import java.util.ArrayList;
4 import model.Product;
5
6 public class ProductServiceDefault implements ProductService {
7
8     private final ArrayList<Product> productList = new ArrayList<>();
9
10    public ProductServiceDefault() {
11        productList.add(new Product(id: 1, name: "Laptop ASUS", price: 9500000));
12        productList.add(new Product(id: 2, name: "Monitor Dell", price: 2500000));
13    }
14
15    @Override
16    public void addProduct(String name, long price) {
17        if (price <= 0) {
18            throw new IllegalArgumentException(
19                s: "Harga harus angka positif lebih dari 0");
20        }
21        int newId = productList.size() + 1;
22        productList.add(new Product(newId, name, price));
23    }
24
25    @Override
26    public ArrayList<Product> getAllProducts() {
27        return productList;
28    }
29 }
```

g. ProductConsoleView.java



LAPORAN PRAKTIKUM

Halaman
26 of 30

```
42 package view;
43
44 import java.util.ArrayList;
45 import java.util.Scanner;
46 import model.Product;
47
48 public class ProductConsoleView {
49     private final Scanner scanner;
50
51     public ProductConsoleView(Scanner scanner) {
52         this.scanner = scanner;
53     }
54
55     public void displayMenu() {
56         System.out.println(" --- APLIKASI + SERVICE LAYER + DAO PATTERN ---");
57         System.out.println("1. Tampilkan semua produk");
58         System.out.println("2. Tambah produk baru");
59         System.out.println("3. Keluar");
60         System.out.print("Pilih opsi: ");
61     }
62
63     public void displayAllProducts(ArrayList<Product> products) {
64         System.out.println("n--- Daftar Produk ---");
65         if (products.isEmpty()) {
66             System.out.println("Tidak ada produk tersedia.");
67         } else {
68             for (Product product : products) {
69                 System.out.println(product.getId() + " - " + product.getName()
70                     + " - Rp " + product.getPrice());
71             }
72         }
73     }
74
75     public void displayMessage(String message) {
76         System.out.println("INFO: " + message);
77     }
78
79     public void displayError(String message) {
80         System.out.println("ERROR: " + message);
81     }
82
83     public String getProductNameFromUser() {
84         System.out.print("Masukkan Nama Produk: ");
85         return scanner.nextLine();
86     }
87
88     public long getProductPriceFromUser() {
89         System.out.print("Masukkan Harga Produk: ");
90         return Long.parseLong(scanner.nextLine());
91     }
92
93     public int getMenuChoiceFromUser() {
94         return Integer.parseInt(scanner.nextLine());
95     }
96 }
```

h. ProductController.java



LAPORAN PRAKTIKUM

Halaman
27 of 30

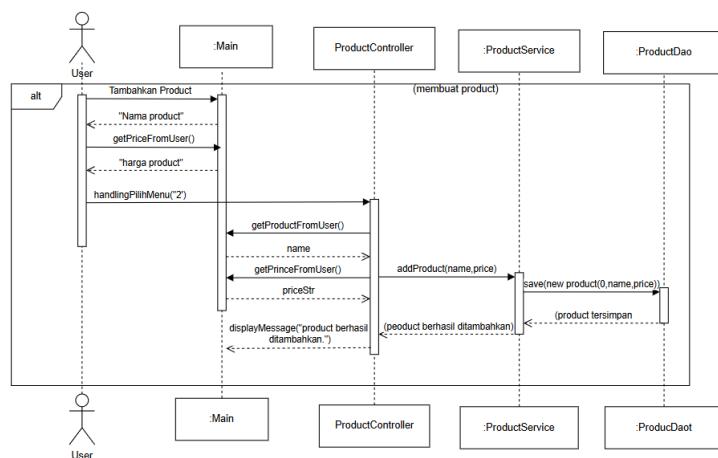
```
55 package controller;
56
57 import service.ProductService;
58 import view.ProductConsoleView;
59
60 public class ProductController {
61     private final ProductService service;
62     private final ProductConsoleView view;
63
64     public ProductController(ProductService service, ProductConsoleView view) {
65         this.service = service;
66         this.view = view;
67     }
68
69     public void handleMenuChoice(String menuChoice) {
70         try {
71             int menu = Integer.parseInt(menuChoice);
72             switch (menu) {
73                 case 1:
74                     showAllProducts();
75                     break;
76                 case 2:
77                     addNewProduct();
78                     break;
79                 case 3:
80                     view.displayMessage(message: "Aplikasi ditutup.");
81                     System.exit(status: 0);
82                     break;
83                 default:
84                     view.displayError(message: "Opsi tidak valid.");
85                     break;
86             }
87         } catch (NumberFormatException e) {
88             view.displayError(message: "Input tidak valid. Masukkan angka.");
89         }
90     }
91
92     private void addNewProduct() {
93         String name = view.getProductFromUser();
94         long price = view.getProductPriceFromUser();
95         try {
96             service.addProduct(name, price);
97             view.displayMessage(message: "Produk berhasil ditambahkan!");
98         } catch (Exception e) {
99             view.displayError("Gagal menambah produk: " + e.getMessage());
100        }
101    }
102
103    private void showAllProducts() {
104        view.displayAllProducts(service.getAllProducts());
105    }
106 }
```

3. Sequence Diagram
- a. Tambahkan Product

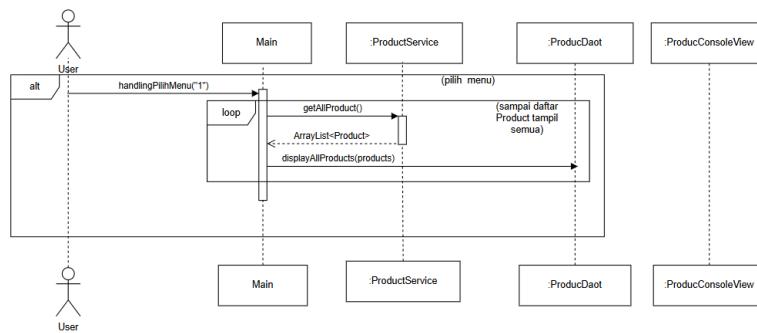


LAPORAN PRAKTIKUM

Halaman
28 of 30



b. Lihat semua Product



i. Main.java



LAPORAN PRAKTIKUM

Halaman
29 of 30

```
206 import java.util.ArrayList;
207 import java.util.Scanner;
208
209 import dao.ProductDao;
210 import dao.memory.ProductDaoInMemory;
211 import model.Product;
212
213 public class Main {
214     private final ProductDao productDao = new ProductDaoInMemory();
215     private final Scanner scanner = new Scanner(System.in);
216
217     Run|Debug
218     public static void main(String[] args) {
219         Main app = new Main();
220         app.startMenuLoop();
221         app.scanner.close();
222     }
223
224     public void startMenuLoop() {
225         boolean running = true;
226         while (running) {
227             System.out.println(x: "\n--- APLIKASI DAO PATTERN ---");
228             System.out.println(x: "1. Tampilkan Semua Produk");
229             System.out.println(x: "2. Tambah Produk Baru");
230             System.out.println(x: "3. Keluar");
231             System.out.print(s: "Pilih opsi: ");
232             try {
233                 int choice = Integer.parseInt(scanner.nextLine());
234                 switch (choice) {
235                     case 1:
236                         displayProducts();
237                         break;
238                     case 2:
239                         addNewProduct();
240                         break;
241                     case 3:
242                         running = false;
243                         break;
244                     default:
245                         System.out.println(x: "Opsi tidak valid.");
246                 }
247             } catch (Exception e) {
248                 System.out.println("Input error: " + e.getMessage());
249             }
250         }
251     }
252
253     private void displayProducts() {
254         System.out.println(x: "\n--- Daftar Produk ---");
255         ArrayList<Product> products = productDao.findAll();
256         for (Product product : products) {
257             System.out.println(product);
258         }
259     }
260
261     private void addNewProduct() {
262         System.out.print(s: "Nama Produk: ");
263         String name = scanner.nextLine();
264         System.out.print(s: "Harga Produk: ");
265         long price = Long.parseLong(scanner.nextLine());
266         if (price <= 0) {
267             System.out.println(x: "Error: Harga harus angka positif lebih dari 0.");
268             return;
269         }
270         int newId = productDao.findAll().size() + 1;
271         Product newProduct = new Product(newId, name, price);
272         productDao.save(newProduct);
273         System.out.println(x: "Produk berhasil ditambahkan!");
274     }
275 }
```

j. Compile & Run



The screenshot shows the Eclipse IDE interface with the following details:

- File Explorer (Left):** Shows the project structure with packages like PETTE, MainJava, and MainMemory.
- Code Editor (Top):** Displays the `MainJava` class with its main method and imports.
- Terminal (Bottom):** Shows the command-line output of the application running in Java, including class loading and execution details.

```
File Edit Selection View Go Run ... ← → Search pattern-nvc-good
EXPLORER ...
PETTE ...
MainJava > Main
282 import dao.ProductDao;
283 import dao.memory.ProductDaoInMemory;
284 import service.ProductService;
285 import service.ProductServiceDefault;
286 import view.ProductConsoleView;
287
288 public class Main {
289     public static void main(String[] args) {
290         ProductDao productDao = new ProductDaoInMemory();
291         ProductService productService = new ProductServiceDefault(productDao);
292
293         Scanner scanner = new Scanner(System.in);
294         ProductConsoleView productView = new ProductConsoleView(scanner);
295     }
296 }
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Ahsan\OneDrive\Documents\Semester3\VBO\pattern-nvc-good> java -jar pattern-nvc-good.jar
Error: Could not find or load main class Main
Caused by: java.lang.ClassNotFoundException: Main
PS C:\Users\Ahsan\OneDrive\Documents\Semester3\VBO\pattern-nvc-good> del *.class
PS C:\Users\Ahsan\OneDrive\Documents\Semester3\VBO\pattern-nvc-good> del pattern-nvc-good.jar
PS C:\Users\Ahsan\OneDrive\Documents\Semester3\VBO\pattern-nvc-good> javac controller/*.java dao/*.java model/*.java service/*.java
PS C:\Users\Ahsan\OneDrive\Documents\Semester3\VBO\pattern-nvc-good> jar cfe pattern-nvc-good.jar Main Main.class controller/*.class dao/*.class
model/*.class service/*.class view/*.class
PS C:\Users\Ahsan\OneDrive\Documents\Semester3\VBO\pattern-nvc-good> java -jar pattern-nvc-good.jar
>>>
>>> APLIKASI : SERVICE LAYER + DAO PATTERN ...
1. Tambahkan produk
2. Tambah produk baru
3. Keluar
Pilih opsi: 1

```

Repository

<https://github.com/OhLanns/patternTugas2403099.git>