



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
Ruangan : 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



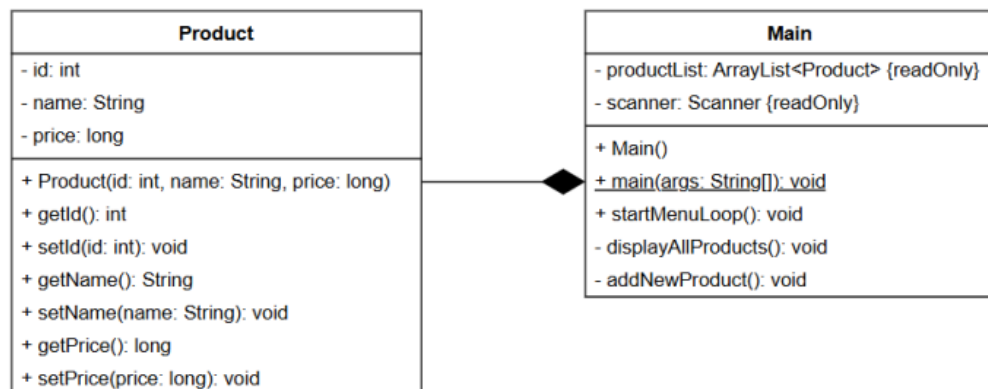
Lapisan yang menjembatani abstraksi bisnis dari detail penyimpanan data.

- 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

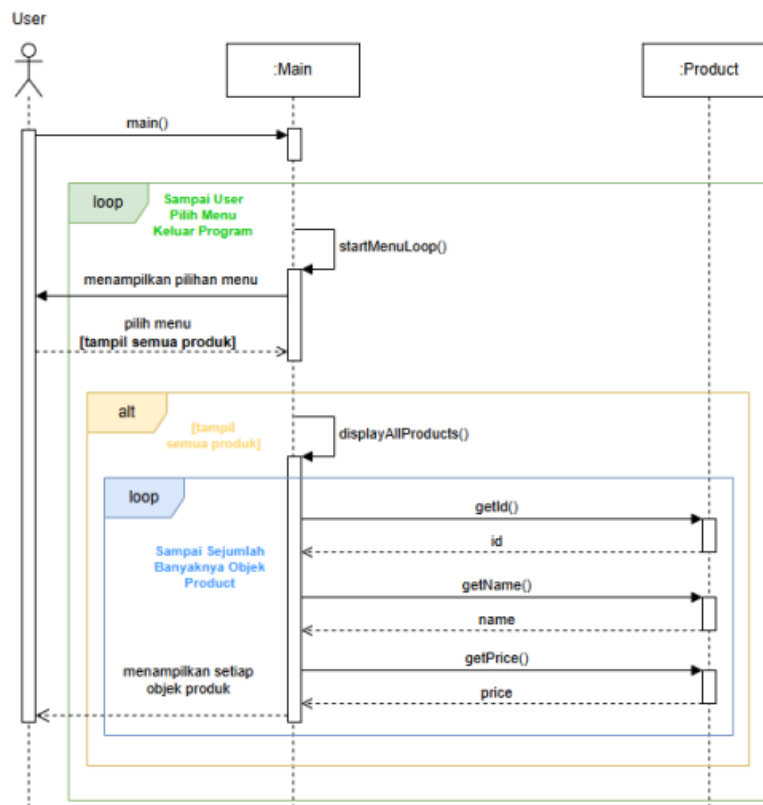
1. Class Diagram

Class diagram

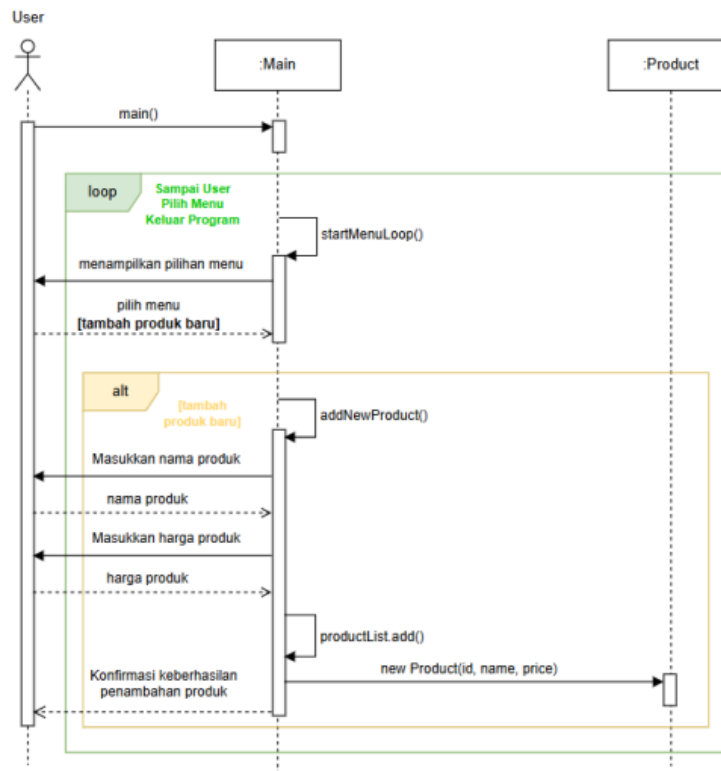


2. Sequence Diagram

- Menampilkan semua daftar produk



- 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 X  ProductService.java  ProductServiceDefault.java  ProductConsoleView.java  ...
PATTERN-NVC-GOOD
├── controller
│   └── ProductControl...
├── model
│   └── Product.java
├── service
│   ├── ProductService...
│   └── ProductServiceD...
├── view
│   ├── ProductConsole...
│   └── Main.java
└── OUTLINE

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 }
```



```
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;
                }
            }
        }
    }
}
```



```
        case 2:
            addNewProduct();
            break;
        case 3:
            running = false;
            System.out.println(x: "Terima kasih telah menggunakan aplikasi!");
            break;
        default:
            System.out.println(x: "Ops! 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();

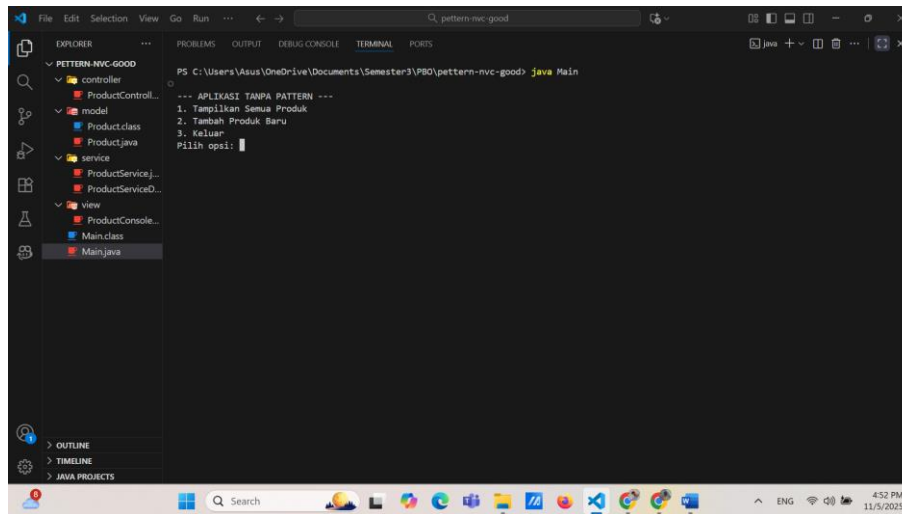
    if (name.trim().isEmpty()) {
        System.out.println(x: "Error: Nama produk tidak boleh kosong.");
        return;
    }

    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();
```

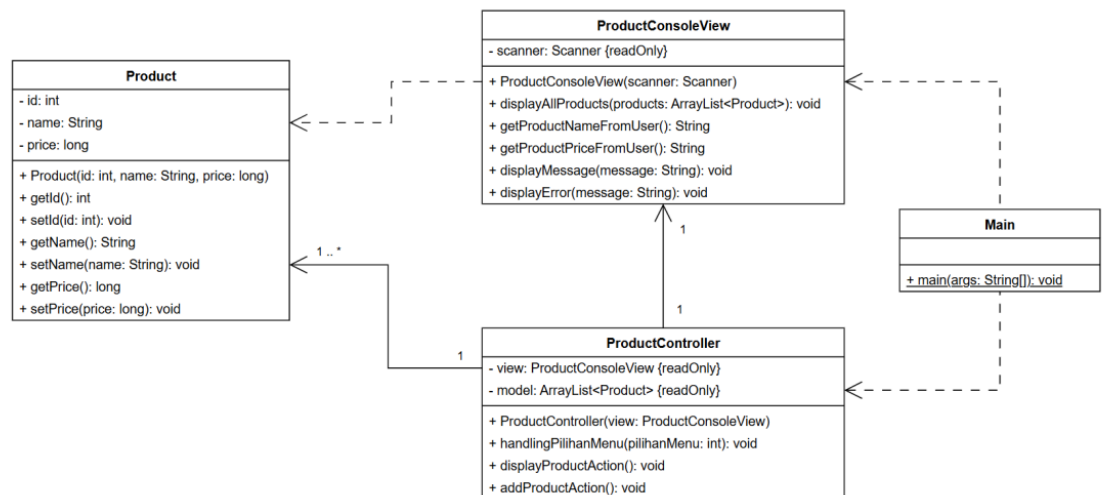
```
Run | Debug
public static void main(String[] args) {
    Main app = new Main();
    app.startMenuLoop();
}
}
```

4. Compile & Run



C. kode program MVC Pattern

1. Class Diagram



2. Kode program

a. Product.java



```
EXPLORER  ...  Controller.java  Main.java  Product.java X  ProductService.java  ProductServiceDefault.java  ProductConsoleView.java  ...  
PATTERN-NVC-GOOD  
  controller  
    ProductController.java  
  model  
    Product.java  
  service  
    ProductService.java  
    ProductServiceDefault.java  
  view  
    ProductConsoleView.java  
  Main.java  
OUTLINE  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
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;  
    }  
  
    public double getPrice() {  
        return price;  
    }  
  
    public void setPrice(long price) {  
        this.price = price;  
    }  
}
```

b. ProductConsoleView

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

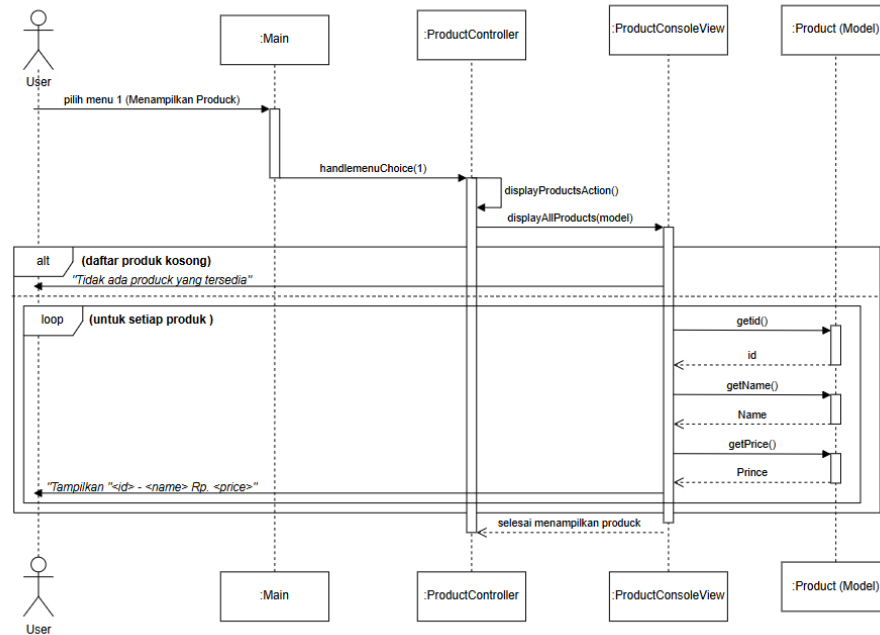



```
}  
  
public void displayMessage (String message){  
    System.out.println("INFO: " + message);  
}  
  
public void displayError (String message){  
    System.out.println("ERROR: " + message);  
}  
  
}
```

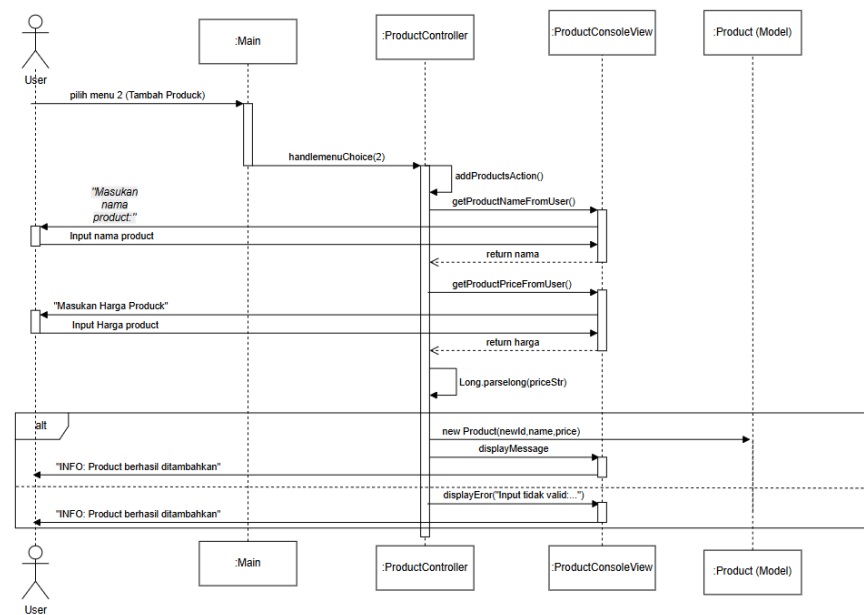
c. ProductController.java

```
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.getProductNameFromUser();  
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
 - a. Menampilkan semua peroduct



b. Menambahkan Product

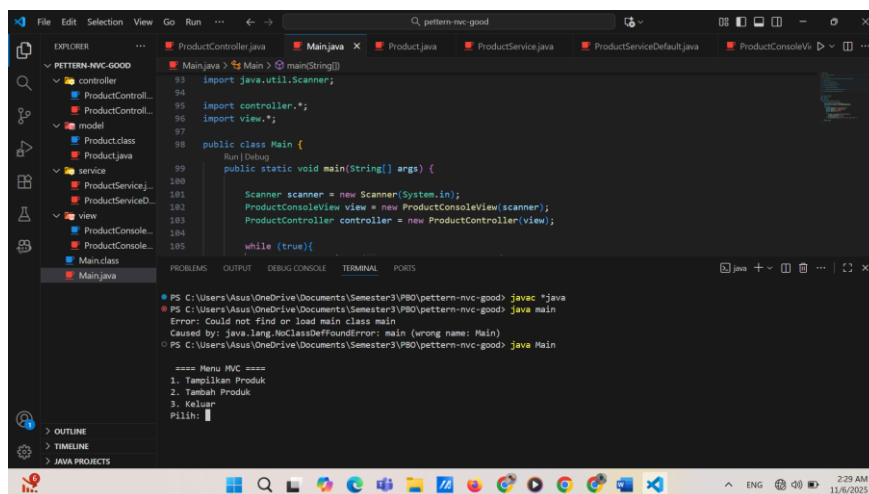


d. Main.java



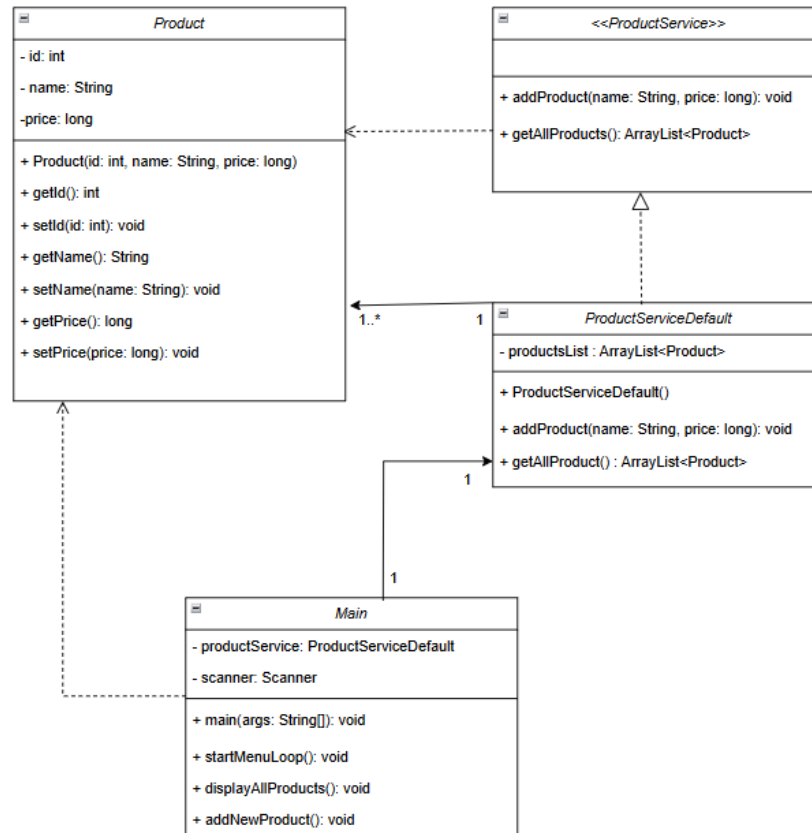
```
93 import java.util.Scanner;
94
95 import controller.*;
96 import view.*;
97
98 public class Main {
99     Run | Debug
100     public static void main(String[] args) {
101
102         Scanner scanner = new Scanner(System.in);
103         ProductConsoleView view = new ProductConsoleView(scanner);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

4. Compile & Run



D. code program Service Layer Pattern

1. Class Diagram



2. Kode program


a. Product.java

```
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 }
```



```
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
```

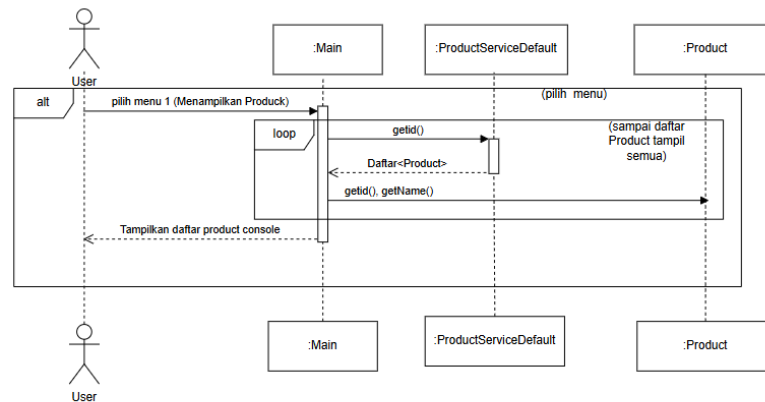
c. ProductServiceDefault.java



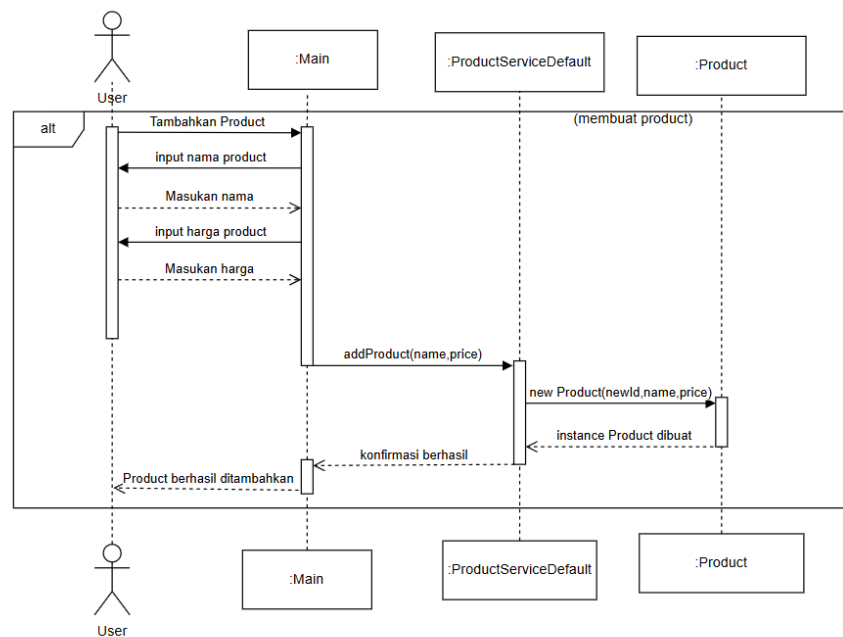
```
service > ProductServiceDefault.java > ...
1 package service;
2
3 import java.util.ArrayList;
4 import model.Product;
5
6 public class ProductServiceDefault implements ProductService {
7
8     private final ArrayList<Product> productList = new ArrayList<>();
9
10    public ProductServiceDefault() {
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



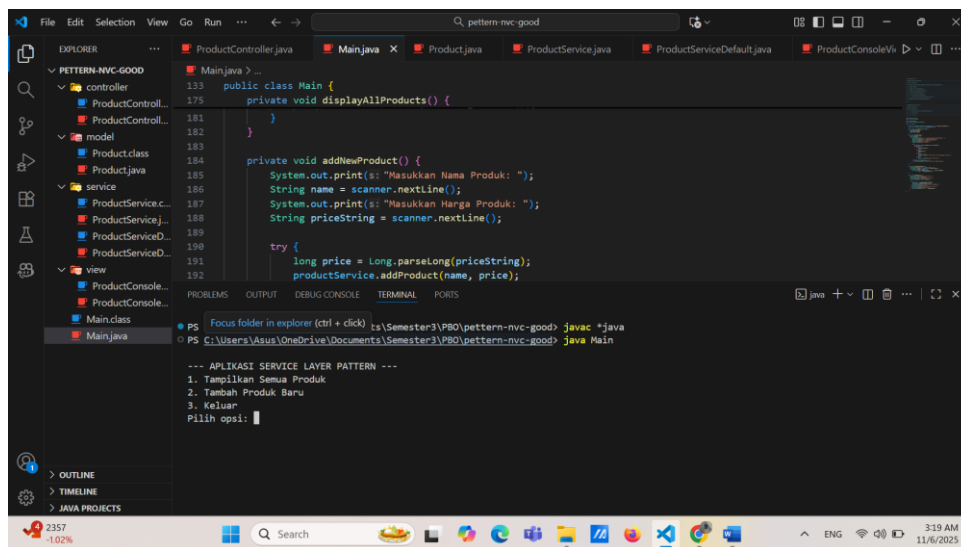
d. Main.java

```
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) {
```



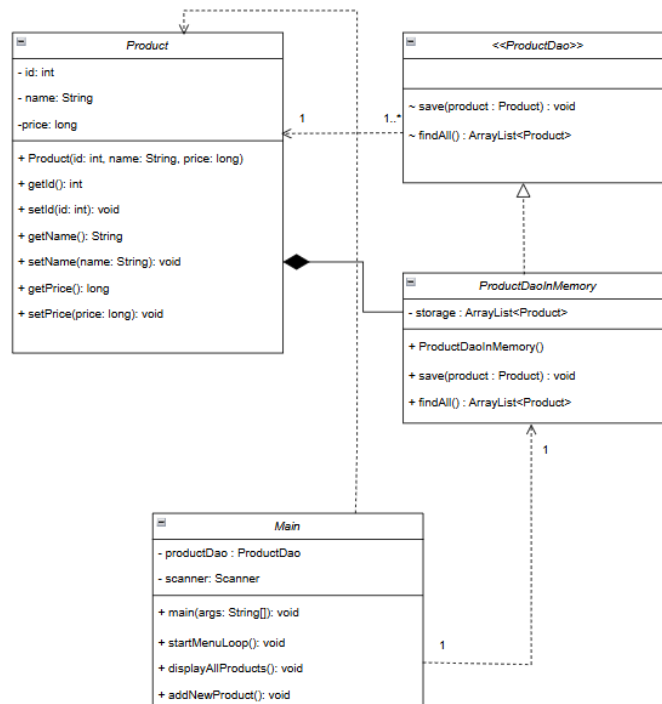
```
156         case 1:
157             displayAllProducts();
158             break;
159         case 2:
160             addNewProduct();
161             break;
162         case 3:
163             running = false;
164             System.out.println(x: "Terima kasih telah menggunakan aplikasi!");
165             break;
166         default:
167             System.out.println(x: "Opsi tidak valid.");
168     }
169 } catch (NumberFormatException e) {
170     System.out.println(x: "Input tidak valid. Masukkan angka.");
171 }
172 }
173 }
174
175 private void displayAllProducts() {
176     System.out.println(x: "\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



E. kode program DAO Pattern

1. Class Diagram



2. Kode program

a. Product.java

```
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 }
```

The screenshot shows the `Product.java` file in an IDE. The code defines a `Product` class with private attributes `id`, `name`, and `price`. It includes a constructor and three pairs of getter and setter methods. The IDE's Explorer panel on the left shows the project structure, and the Outline panel at the bottom left is visible.



```
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
```

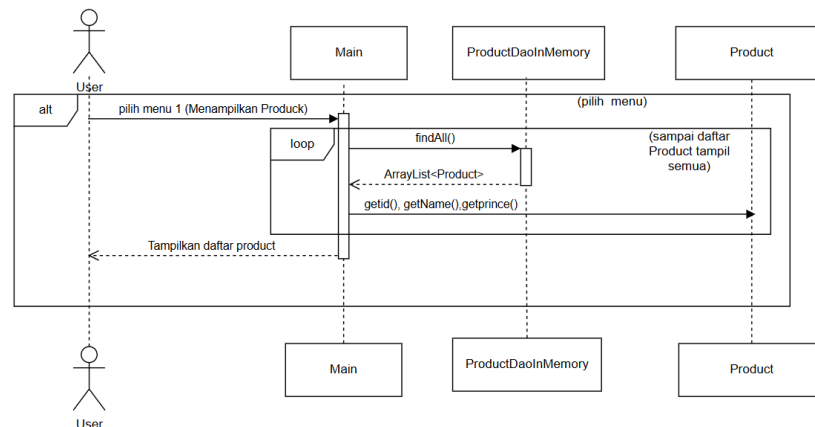
c. ProductDaoMemoy.java



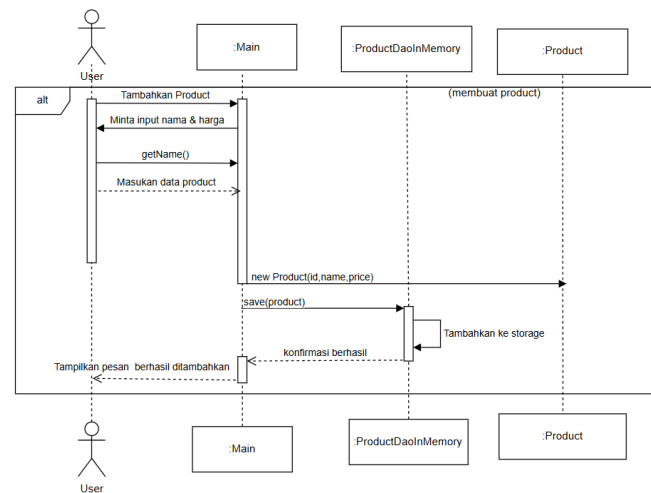
```
dao > memory > ProductDaoInMemory.java > ...
1
2 package dao.memory;
3
4 import java.util.ArrayList;
5
6 import dao.ProductDao;
7 import model.Product;
8
9 public class ProductDaoInMemory implements ProductDao {
10
11     private final ArrayList<Product> storage = new ArrayList<>();
12
13     public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

3. Sequence Diagram

a. Melihat semua product



b. Menambahkan Product



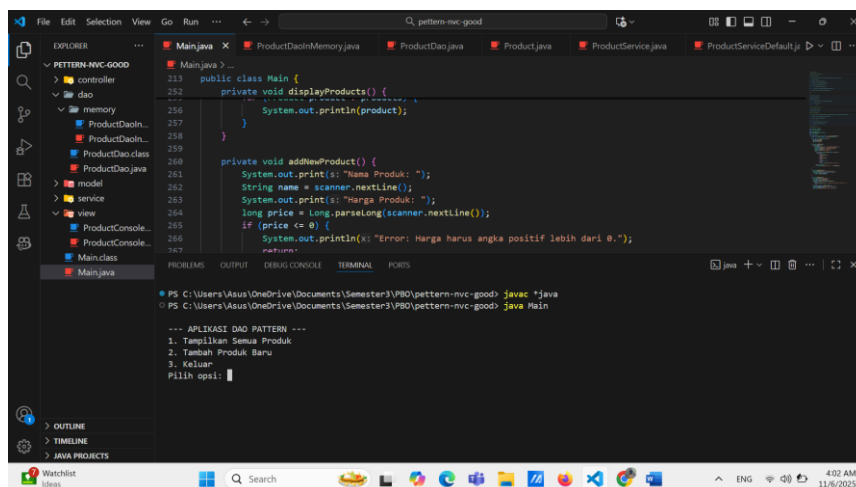
d. Main.java

```
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         case 1:
235             displayProducts();
236             break;
237         case 2:
238             addNewProduct();
239             break;
240         case 3:
241             running = false;
242             break;
243         default:
244             System.out.println(x: "Opsi tidak valid.");
245     }
246     } catch (Exception e) {
247         System.out.println("Input error: " + e.getMessage());
248     }
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 newProduct = new Product(newId, name, price);
271     productDao.save(newProduct);
272     System.out.println(x: "Produk berhasil ditambahkan!");
273 }
274 }
275
```

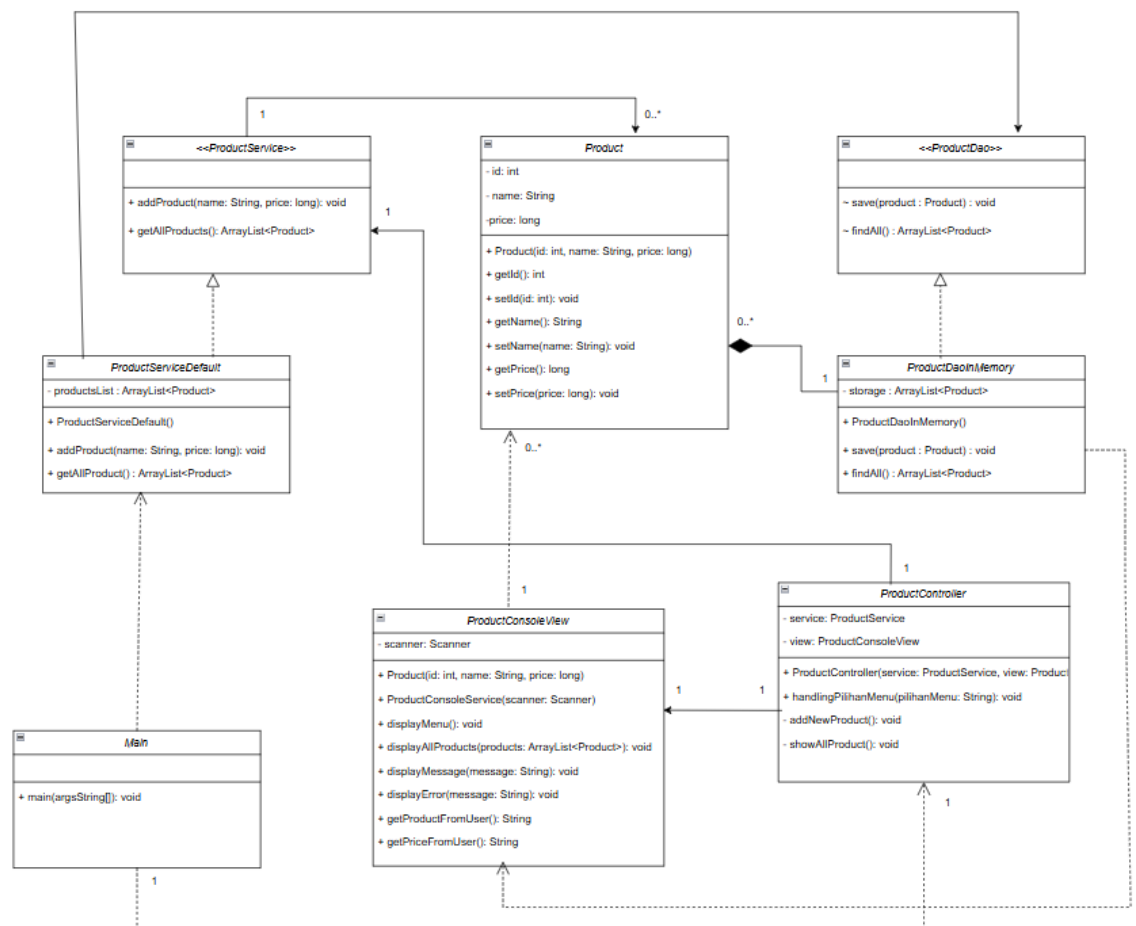
e. Compile & Run





F. kode program MVC + Service Layer + DAO Pattern

1. Class Diagram



2. Kode program

a. Product.java



```
EXPLORER  ...  Controller.java  Main.java  Product.java X  ProductService.java  ProductServiceDefault.java  ProductConsoleView.java  ...  
PETERNA-MVC-GOOD  
  controller  
    ProductController.java  
  model  
    Product.java  
  service  
    ProductService.java  
    ProductServiceDefault.java  
  view  
    ProductConsoleView.java  
  Main.java  
OUTLINE  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
model > 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     }  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
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
2 package dao.memory;
3
4 import java.util.ArrayList;
5
6 import dao.ProductDao;
7 import model.Product;
8
9 public class ProductDaoInMemory implements ProductDao {
10
11     private final ArrayList<Product> storage = new ArrayList<>();
12
13     public ProductDaoInMemory() {
14         storage.add(new Product(id: 1, name: "Laptop ASUS", price: 9500000));
15         storage.add(new Product(id: 2, name: "Monitor Dell", price: 2500000));
16     }
17
18     @Override
19     public void save(Product product) {
20         storage.add(product);
21     }
22
23     @Override
24     public ArrayList<Product> findAll() {
25         return storage;
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



```
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



```
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(x: "--- APLIKASI + SERVICE LAYER + DAO PATTERN ---");
57         System.out.println(x: "1. Tampilkan semua produk");
58         System.out.println(x: "2. Tambah produk baru");
59         System.out.println(x: "3. Keluar");
60         System.out.print(s: "Pilih opsi: ");
61     }
62
63     public void displayAllProducts(ArrayList<Product> products) {
64         System.out.println(x: "\n--- Daftar Produk ---");
65         if (products.isEmpty()) {
66             System.out.println(x: "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(s: "Masukkan Nama Produk: ");
85         return scanner.nextLine();
86     }
87
88     public long getProductPriceFromUser() {
89         System.out.print(s: "Masukkan Harga Produk: ");
90         return Long.parseLong(scanner.nextLine());
91     }
92
93     public int getMenuChoiceFromUser() {
94         return Integer.parseInt(scanner.nextLine());
95     }
96 }
```

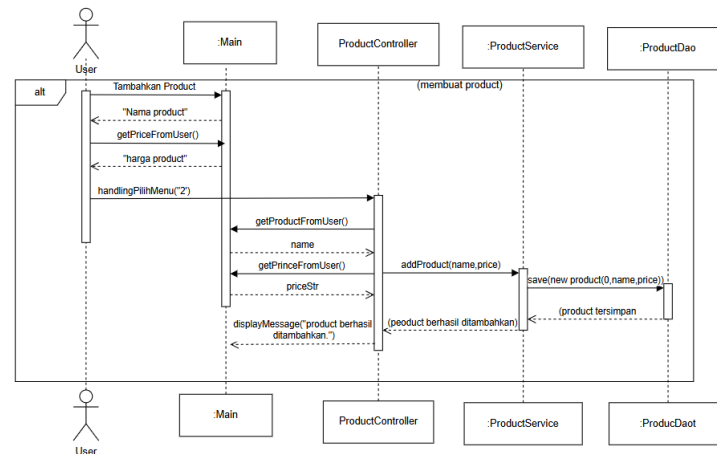
h. ProductController.java



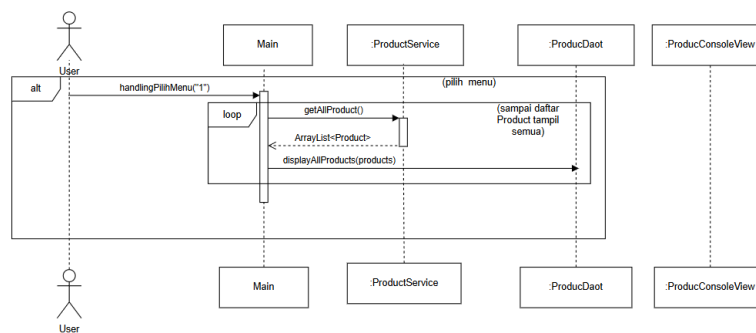
```
55 package controller;
56
57 import service.ProductService;
58 import view.ProductConsoleView;
59
60 public class ProductController {
61     private final ProductService service;
62     private final ProductConsoleView view;
63
64     public ProductController(ProductService service, ProductConsoleView 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.getProductnameFromUser();
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



b. Lihat semua Product



i. 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) {
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: "Ops! 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



LAPORAN PRAKTIKUM

Halaman
30 of 30

The screenshot shows an IDE with a project named 'pettern-nvc-good'. The Explorer panel on the left shows the project structure with packages: controller, dao, memory, model, service, and view. The Main.java file is open in the editor, showing the following code:

```
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         Scanner scanner = new Scanner(System.in);
293         ProductConsoleView productView = new ProductConsoleView(scanner);
294     }
295 }
```

The console window at the bottom shows the following output:

```
PS C:\Users\Asus\OneDrive\Documents\Semester3\PSD\pettern-nvc-good> java -jar pettern-nvc-good.jar
Error: Could not find or load main class view.Main
Caused by: java.lang.ClassNotFoundException: view.Main
PS C:\Users\Asus\OneDrive\Documents\Semester3\PSD\pettern-nvc-good> del *.class
PS C:\Users\Asus\OneDrive\Documents\Semester3\PSD\pettern-nvc-good> javac controller/*.java dao/*.java memory/*.java model/*.java service
e/*.java view/*.java Main.java
PS C:\Users\Asus\OneDrive\Documents\Semester3\PSD\pettern-nvc-good> jar cfe pettern-nvc-good.jar Main Main.class controller/*.class dao/*.cl
ass dao/memory/*.class model/*.class service/*.class view/*.class
PS C:\Users\Asus\OneDrive\Documents\Semester3\PSD\pettern-nvc-good> java -jar pettern-nvc-good.jar
>>
--- APLIKASI + SERVICE LAYER + DAO PATTERN ---
1. Tampilkan semua produk
2. Tambah produk baru
3. Keluar
Pilih opsi: [ ]
```

Repository

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