Mata Kuliah	:	Pemrograman Berorientasi Objek
Tanggal	:	24 September 2024
Semester	:	3 (Gasal)
Topik	:	Studi Kasus Java OOP
Minggu/Pertemuan/Sesi	:	05/02/01-02
Aktivitas	:	Praktikum
Durasi	:	170 menit
Setoran	:	Dokumen Laporan PDF
Batas Penyerahan	:	-
Tempat Penyerahan	:	https://ecourse.del.ac.id/

Daftar Isi

A. Tujuan & Penilaian

1. Tujuan

- ✓ Mahasiswa mampu menggunakan Java OOP untuk menyelesaikan studi kasus tertentu.

2. Penilaian

No	Kriteria	Bobot Penilaian (%)
1	Latihan Java Standar Class	10
2	Studi Kasus Java OOP: Aplikasi Todo	30
3	Menambahkan Fitur Update pada Aplikasi Todo	60

3. Dokumen Laporan

Buat laporan praktikum menggunakan [Template Laporan Praktikum] [Contoh Laporan Praktikum]. Sebelum mengirimkan dokumen laporan pada E-Course silahkan ubah penamaan file dengan format:

{NIM}-pbo-laporan-praktikum-5.pdf

Sebagai contoh:

11S18005-pbo-laporan-praktikum-5.pdf

4. Membuat Workspace

Buat *workspace* baru pada VSCode dengan format penamaan "**{username}-pbo-praktikum-5**".

B. Aktivitas Praktikum

1. Latihan Java Standar Class

a) String Class

String Class di Java adalah kelas bawaan yang digunakan untuk mengelola dan memanipulasi data teks (kumpulan karakter).

| Method di String Class

Method	Keterangan
String toLowerCase()	Membuat string baru dengan format lower case
String toUpperCase()	Membuat string baru dengan format upper case
int length()	Mendapatkan panjang string
boolean startsWith(value)	Mengecek apakah diakhiri dengan string value
boolean endsWith(value)	Mengecek apakah diakhiri dengan string value
String[] split(value)	Memotong string dengan string value

Buat dan modifikasi isi file **"StringApp.java"**, seperti berikut:

StringApp.java

```
1  public class StringApp {
2    public static void main(String[] args) {
3
4      String name = "Abdullah Ubaid";
5      String nameLowerCase = name.toLowerCase();
6      String nameUppercase = name.toUpperCase();
7
8      System.out.println(name);
9      System.out.println(nameLowerCase);
10     System.out.println(nameUppercase);
11     System.out.println(name.length());
12     System.out.println(name.startsWith("Abd"));
13     System.out.println(name.endsWith("Ubaid"));
14
15     String[] names = name.split(" ");
16     for (var value : names) {
17       System.out.println(value);
18     }
19
20     System.out.println(" ".isBlank());
21     System.out.println(" ".isEmpty());
22     System.out.println("".isEmpty());
23     System.out.println(name.charAt(0));
24
25     char[] chars = name.toCharArray();
26     for (var value : chars) {
27       System.out.print(value + "|");
28     }
29   }
30 }
```

Analisis dan amati hasil kode program di atas pada terminal.

b) StringBuffer dan StringBuilder Class

String adalah tipe data immutable, artinya tidak bisa berubah isinya, saat kita mengubah string, sebenarnya yang dilakukan di Java adalah membuat String baru. Jika kita ingin memanipulasi String dalam jumlah banyak, sangat tidak disarankan menggunakan String, karena akan memakan memory yang cukup besar, untuk kasus seperti ini, disarankan menggunakan StringBuffer atau StringBuilder.

| StringBuffer vs StringBuilder

Kemampuan StringBuffer dan StringBuilder cukup sama, bisa digunakan untuk memanipulasi String yang membedakan adalah, StringBuffer itu thread safe, sedangkan StringBuilder tidak thread safe. Jika kita ingin memanipulasi String secara paralel bersamaan, disarankan menggunakan StringBufer, namun jika tidak butuh parallel, cukup gunakan StringBuilder karena StringBuffer dibuat agar thread safe, maka secara otomatis performanya lebih lambat dibandingkan StringBuilder.

Modifikasi isi file **"StringApp.java"**, seperti berikut:

StringApp.java

```
1  public class StringApp {
2    public static void main(String[] args) {
3      StringBuilder builder = new StringBuilder();
4      builder.append("Abdullah");
5      builder.append(" ");
6      builder.append("Ubaid");
7
8      String name = builder.toString();
9      System.out.println(name);
10
11     StringBuffer buffer = new StringBuffer("Abdullah Ubaid");
12     buffer.reverse();
13     System.out.println(buffer.toString());
14   }
15 }
```

Analisis dan amati hasil kode program di atas pada terminal.

c) StringJoiner Class

StringJoiner adalah class yang bisa digunakan untuk membuat String sequence yang dipisahkan dengan delimiter. StringJoiner juga mendukung prefix dan suffix jika kita ingin menambahkannya. Ini sangat bagus ketika ada kasus misal kita ingin mem-print Array dengan format yang kita mau misalnya.

Modifikasi isi file **"StringApp.java"**, seperti berikut:

StringApp.java

```
1  import java.util.StringJoiner;
2
3  public class StringApp {
4    public static void main(String[] args) {
5      StringJoiner joiner = new StringJoiner(", ", "{", "}");
6      joiner.add("Abdullah");
7      joiner.add("Ubaid");
8
9      String value = joiner.toString();
```

StringApp.java

```
      System.out.println(value);
    }
}
```

Analisis dan amati hasil kode program di atas pada terminal.

d) StringTokenizer Class

StringTokenizer class adalah class yang bisa digunakan untuk memotong String menjadi token atau string yang lebih kecil. Kita dapat memotong String dengan delimiter yang kita mau.

Modifikasi isi file **"StringApp.java"**, seperti berikut:

StringApp.java

```
import java.util.StringTokenizer;

public class StringApp {
  public static void main(String[] args) {
    String value = "Abdullah Ubaid";

    StringTokenizer tokenizer = new StringTokenizer(value, " ");
    while (tokenizer.hasMoreTokens()) {
      String result = tokenizer.nextToken();
      System.out.println(result);
    }
  }
}
```

Analisis dan amati hasil kode program di atas pada terminal.

e) Number Class

Semua number class yang bukan primitive memiliki parent class yang sama, yaitu class Number. Class number memiliki banyak method yang bisa digunakan untuk mengkonversi ke tipe number lain.

| Method di Number Class

Method	Keterangan
byte byteValue()	Mengubah menjadi tipe byte
double doubleValue()	Mengubah menjadi tipe double
float floatValue()	Mengubah menjadi tipe float
int intValue()	Mengubah menjadi int value
long longValue()	Mengubah menjadi long value
short shortValue()	Mengubah menjadi short value

| Konversi String ke Number

Long, Integer, Short dan Byte memiliki static method untuk melakukan konversi dari String ke Number. parseXxx(String) digunakan untuk mengkonversi dari string ke tipe data number primitif, sedangkan valueOf(String) digunakan untuk

mengkonversi dari string ke tipe data number non primitif. Method ini akan throw NumberFormatException jika ternyata gagal melakukan konversi String ke number.

Buat dan modifikasi isi file **"NumberApp.java"**, seperti berikut:

NumberApp.java

```
1  public class NumberApp {
2    public static void main(String[] args) {
3      Integer intValue = 10;
4      Long longValue = intValue.longValue();
5      Double doubleValue = longValue.doubleValue();
6      Short shortValue = doubleValue.shortValue();
7      System.out.println(shortValue);
8
9      String strValue = "99.99";
10
11     double strToPrimitive = Double.parseDouble(strValue);
12     System.out.println(strToPrimitive);
13
14     Double strToDouble = Double.valueOf(strValue);
15     System.out.println(strToDouble);
16   }
17 }
```

Analisis dan amati hasil kode program di atas pada terminal.

f) Math Class

Class Math merupakan class utilities yang berisikan banyak sekali static method untuk operasi numeric, seperti trigonometric, logarithm, akar pangkat, dan lain-lain.

| Method di Math Class

Method	Keterangan
double cos(double)	Menghitung cos di trigonometri
double sin(double)	Menghitung sin di trigonometri
double tan(double)	Menghitung tan di trigonometri
min(number1, number2)	Mengambil nilai terkecil
max(number1, number2)	Mengambil nilai terbesar
dan lainnya...	

Buat dan modifikasi isi file **"MathApp.java"**, seperti berikut:

MathApp.java

```
1  public class MathApp {
2    public static void main(String[] args) {
3      var min = Math.min(1000, 2000);
4      System.out.println(min);
5
6      var max = Math.max(1000, 2000);
7      System.out.println(max);
```

MathApp.java

```
8
9           System.out.println(Math.PI);
10      }
11  }
```

Analisis dan amati hasil kode program di atas pada terminal.

g) BigNumber Class

Jika kita ada kebutuhan untuk menggunakan angka yang besar sehingga melebihi kapasitas Long dan Double, di Java sudah disediakan class untuk handle data besar tersebut. BigInteger adalah class untuk handle tipe data Integer, dan BigDecimal adalah class untuk handle tipe data floating-point.

| Method di BigInteger & BigDecimal

Method	Keterangan
add	Penjumlahan
substract	Pengurangan
multiply	Perkalian
divide	Pembagian
mod	Sisa bagi
dan lainnya...	

Buat dan modifikasi isi file **"BigNumberApp.java"**, seperti berikut:

BigNumberApp.java

```
1   import java.math.BigInteger;
2
3   public class BigNumberApp {
4     public static void main(String[] args) {
5       BigInteger a = new
    BigInteger("1_000_000_000_000_000_000_000_000_000_000_000_000_000"
    .replace("_", ""));
6       BigInteger b = new
    BigInteger("1_000_000_000_000_000_000_000_000_000_000_000_000_000"
    .replace("_", ""));
7
8       BigInteger c = a.add(b);
9       System.out.println(c);
10    }
11  }
```

Analisis dan amati hasil kode program di atas pada terminal.

h) Scanner Class

Class Scanner hadir sejak Java 5. Class Scanner adalah class yang bisa digunakan untuk membaca input, entah dari file, console, dan lain-lain. Class Scanner cocok digunakan sebagai object untuk membaca input user saat membuat program Java menggunakan console / terminal.

| Method di Scanner

Method	Keterangan
nextLine()	Membaca string satu baris.
nextInt()	Membaca int
nextLong()	Membaca long
nextBoolean()	Membaca boolean
dan lainnya...	

Buat dan modifikasi isi file **"ScannerApp.java"**, seperti berikut:

ScannerApp.java

```
1  import java.util.Scanner;
2
3  public class ScannerApp {
4    public static void main(String[] args) {
5      Scanner scanner = new Scanner(System.in);
6
7      System.out.print("Nama: ");
8      String nama = scanner.nextLine();
9
10     System.out.print("Umur : ");
11     Integer age = scanner.nextInt();
12
13     System.out.println("Hello " + nama + " umur Anda adalah " + age);
14     scanner.close();
15   }
16 }
```

Analisis dan amati hasil kode program di atas pada terminal.

i) Date dan Calendar Class

Tiap bahasa pemrograman biasanya memiliki representasi tanggal, di Java juga sama, ada class Date & Calendar yang bisa digunakan sebagai representasi tanggal.

| Hubungan Date dan Calendar

Class Date adalah class representasi tanggal sampai presisi millisecond. Namun di class Date sudah banyak method-method yang di deprecated, sehingga untuk memanipulasi date tanggal, kita sekarang harus melakukan kombinasi antara class Date dan Calendar. Sederhananya Date untuk representasi tanggal, dan Calendar untuk memanipulasi tanggal.

Buat dan modifikasi isi file **"DateApp.java"**, seperti berikut:

DateApp.java

```
1  import java.util.Calendar;
2  import java.util.Date;
3
4  public class DateApp {
5    public static void main(String[] args) {
6      Date tanggal = new Date(1234567890L);
7      System.out.println(tanggal);
8
9      Calendar calendar = Calendar.getInstance();
10     calendar.set(Calendar.YEAR, 2023);
11     calendar.set(Calendar.MONTH, Calendar.SEPTEMBER);
12     calendar.set(Calendar.DATE, 6);
```

DateApp.java

```
13        calendar.set(Calendar.HOUR_OF_DAY, 8);
14        calendar.set(Calendar.MINUTE, 30);
15        calendar.set(Calendar.SECOND, 0);
16
17        Date result = calendar.getTime();
18        System.out.println(result);
19      }
20    }
```

Analisis dan amati hasil kode program di atas pada terminal.

j) System Class

Class System adalah class yang berisikan banyak utility static method di Java, contohnya kita sering menggunakan methode println milik field out di class System.

| Method di System Class

Method	Keterangan
String getenv(key)	Mendapatkan environment variabel sistem operasi
void exit(status)	Menghentikan program Java
long currentTimeMilis()	Mendapatkan waktu saat ini dalam milisecond
long nanoTime()	Mendapatkan waktu saat ini dalam nanosecond
void gc()	Menjalankan Java garbage collection
dan lainnya...	

Buat dan modifikasi isi file **"SystemApp.java"**, seperti berikut:

SystemApp.java

```
1   public class SystemApp {
2     public static void main(String[] args) {
3       System.out.println(System.currentTimeMillis());
4       System.out.println(System.nanoTime());
5
6       System.out.println(System.getenv("APP"));
7       System.out.println(System.getenv("OS"));
8
9       System.gc();
10      System.exit(1);
11
12      System.out.println("Tidak tampil");
13    }
14  }
```

Analisis dan amati hasil kode program di atas pada terminal.

k) Runtime Class

Ketika aplikasi Java kita berjalan, kita bisa melihat informasi environment tempat aplikasi Java berjalan. Informasi itu terdapat di class Runtime. Class Runtime tidak bisa dibuat, secara otomatis Java akan membuat single object.

Kita bisa mengakses object tersebut menggunakan static method getRuntime() milik class Runtime.

| Method di Runtime Class

Method	Keterangan
int availableProcessors()	Mendapatkan jumlah core cpu
long freeMemory()	Mendapatkan jumlah memory bebas di JVM
long totalMemory()	Mendapatkan jumlah total memory di JVM
long maxMemory()	Mendapatkan jumlah maksimum memory di JVM
void gc()	Menjalankan garbage collector untuk menghilangkan data di memory yang sudah tidak terpakai

Buat dan modifikasi isi file **"RuntimeApp.java"**, seperti berikut:

RuntimeApp.java

```
1  public class RuntimeApp {
2    public static void main(String[] args) {
3      Runtime runtime = Runtime.getRuntime();
4
5      System.out.println(runtime.availableProcessors());
6      System.out.println(runtime.freeMemory());
7      System.out.println(runtime.totalMemory());
8      System.out.println(runtime.maxMemory());
9    }
10 }
```

Analisis dan amati hasil kode program di atas pada terminal.

1) UUID Class

Saat membuat aplikasi, kadang kita ada kasus ingin membuat data unique, misal untuk kebutuhan data primary key misalnya. Java menyediakan sebuah class UUID atau singkatan dari Universally Unique Identifier. UUID adalah format standard untuk membuat unique value yang telah terjamin.

Buat dan modifikasi isi file **"UUIDApp.java"**, seperti berikut:

UUIDApp.java

```
1  import java.util.UUID;
2
3  public class UUIDApp {
4    public static void main(String[] args) {
5      for (var i = 0; i < 100; i++) {
6        UUID uuid = UUID.randomUUID();
7        String key = uuid.toString();
8        System.out.println(key);
9      }
10   }
11 }
```

Analisis dan amati hasil kode program di atas pada terminal.

m) Base64 Class

Sejak Java 8, Java sudah menyediakan class untuk melakukan encoding base64. Buat programmer web pasti tahu tentang base64, yaitu encoding yang bisa digunakan untuk mengubah binary data ke text yang aman. Aman disini bukan dari sisi security, tapi dari kesalahan parsing.

Buat dan modifikasi isi file **"Base64App.java"**, seperti berikut:

Base64App.java

```
1  import java.util.Base64;
2
3  public class Base64App {
4    public static void main(String[] args) {
5      String original = "Abdullah Ubaid";
6      String encoded =
   Base64.getEncoder().encodeToString(original.getBytes());
7      System.out.println(encoded);
8
9      byte[] bytes = Base64.getDecoder().decode(encoded);
10     String result = new String(bytes);
11     System.out.println(result);
12   }
13 }
```

Analisis dan amati hasil kode program di atas pada terminal.

n) Objects Class

Awas jangan tertukar, ini class Objects bukan Object. Objects adalah class utility yang berisikan banyak static method yang bisa digunakan untuk operasi object atau melakukan pengecekan sebelum suatu operasi dilakukan.

Modifikasi isi file **"ObjectApp.java"**, seperti berikut:

ObjectApp.java

```
1  import java.util.Objects;
2
3  public class ObjectApp {
4
5    public static class Data {
6      private String data;
7
8      public Data(String data) {
9        this.data = data;
10     }
11
12     public String getData() {
13       return data;
14     }
15
16     public void setData(String data) {
17       this.data = data;
18     }
19
20     @Override
21     public int hashCode() {
22       final int prime = 31;
23       int result = 1;
```

ObjectApp.java

```
24          result = prime * result + ((data == null) ? 0 : data.hashCode());
25          return result;
26        }
27
28        @Override
29        public String toString() {
30          return "Data [data=" + data + "]";
31        }
32
33        @Override
34        public boolean equals(Object obj) {
35          if (this == obj)
36            return true;
37          if (obj == null)
38            return false;
39          if (getClass() != obj.getClass())
40            return false;
41          Data other = (Data) obj;
42          if (data == null) {
43            if (other.data != null)
44              return false;
45          } else if (!data.equals(other.data))
46            return false;
47          return true;
48        }
49      }
50
51      public static void main(String[] args) {
52        execute(null);
53        execute(new Data("Abdullah"));
54      }
55
56      public static void execute(Data data) {
57        System.out.println(Objects.toString(data));
58        System.out.println(Objects.hash(data));
59      }
60    }
```

Analisis dan amati hasil kode program di atas pada terminal.

o) Random Class

Random class adalah class yang bisa kita gunakan untuk melakukan generate random number.

Modifikasi isi file **"RandomApp.java"**, seperti berikut:

RandomApp.java

```
1   import java.util.Random;
2
3   public class RandomApp {
4     public static void main(String[] args) {
5       Random random = new Random();
6       for (int i = 0; i < 10; i++) {
7         int value = random.nextInt(1000);
8         System.out.println(value);
9       }
10    }
12  }
```

Analisis dan amati hasil kode program di atas pada terminal.

p) Properties Class

Kebanyakan aplikasi Java akan menyimpan konfigurasi file dalam bentuk properties file. Properties file adalah file yang berisi key value yang dipisahkan dengan tanda sama dengan (=). Properties file bisa kita gunakan untuk menyimpan konfigurasi aplikasi kita.

Buat dan modifikasi isi file **"app.properties"**, seperti berikut:

app.properties

```
# Konfigurasi Sample
# 05/08/2023 - 09:59 WIB
version=1.0
name.first=ABDULLAH
name.last=Ubaid
```

Buat dan modifikasi isi file **"PropertiesApp.java"**, seperti berikut:

PropertiesApp.java

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Properties;

public class PropertiesApp {
  public static void main(String[] args) {
    try {
      Properties properties = new Properties();
      properties.load(new FileInputStream("app.properties"));

      String version = properties.getProperty("version");
      String firstName = properties.getProperty("name.first");
      String lastName = properties.getProperty("name.last");

      System.out.println(version);
      System.out.println(firstName);
      System.out.println(lastName);

      properties.put("hobby", "Coding");
      properties.store(new FileOutputStream("app.properties"),
"Komentar");

    } catch (FileNotFoundException e) {
      System.out.println("File tidak ditemukan");
    } catch (IOException ex) {
      System.out.println("Gagal memuat data dari file");
    }
  }
}
```

Analisis dan amati hasil kode program di atas pada terminal, selanjutnya perhatikan isi file app.properties

q) Arrays Class

Array class adalah class yang berisikan static method yang bisa digunakan untuk memanipulasi data array, seperti pencarian dan pengurutan.

| Method di Arrays Class

Method	Keterangan
binarySearch(array, value)	Mencari value dari array
copyOf(...)	Menyalin data array
equals(array1, array2)	Membandingkan array1 dan array2
sort(array)	Mengurutkan array
toString(array)	Mengembalikan representasi string
dan lainnya...	

Buat dan modifikasi isi file **"ArraysApp.java"**, seperti berikut:

ArraysApp.java

```
1  import java.util.Arrays;
2
3  public class ArraysApp {
4    public static void main(String[] args) {
5      int[] numbers = {
6          1, 4, 20, 19, 5, 6, 99, 3, 7, 11, 2
7      };
8
9      Arrays.sort(numbers);
10     System.out.println(Arrays.toString(numbers));
11
12     System.out.println(Arrays.binarySearch(numbers, 2));
13     System.out.println(Arrays.binarySearch(numbers, 6));
14     System.out.println(Arrays.binarySearch(numbers, 100));
15
16     int[] result = Arrays.copyOf(numbers, 5);
17     System.out.println(Arrays.toString(result));
18
19     int[] result2 = Arrays.copyOfRange(numbers, 5, 10);
20     System.out.println(Arrays.toString(result2));
21   }
22 }
```

Analisis dan amati hasil kode program di atas pada terminal.

r) Regular Expression

Regular Expression atau disingkat regex adalah cara untuk melakukan pola pencarian. Biasanya dilakukan untuk pencarian dalam data string. Regex adalah pencarian yang lebih advanced dibandingkan pencarian text biasanya, missal kita ingin mencari semua kata yang mengandung diawali huruf a dan diakhiri huruf a, dan lain-lain.

| Regex Package

Java sudah menyediakan package java.util.regex yang berisikan utilitas untuk melakukan proses regular expression. Secara garis besar terdapat 2 class yang dapat digunakan, yaitu Pattern class dan Matcher class. Pattern class adalah representasi hasil kompilasi dari pola regular expression yang kita buat. Matcher class adalah engine untuk melakukan pencarian dari pattern yang sudah dibuat.

| Aturan Regular Expression

Aturan regular expression sangat banyak, kita bisa lihat detail aturan-aturannya di halaman javadoc class Pattern melalui link berikut:
https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html

Buat dan modifikasi isi file **"RegexApp.java"**, seperti berikut:

RegexApp.java

```
1  import java.util.regex.Matcher;
2  import java.util.regex.Pattern;
3
4  public class RegexApp {
5    public static void main(String[] args) {
6      String name = "Abdullah Ubaid senang belajar PBO";
7      Pattern pattern = Pattern.compile("[a-zA-Z]*[a][a-zA-Z]*");
8      Matcher matcher = pattern.matcher(name);
9
10     while (matcher.find()) {
11       String result = matcher.group();
12       System.out.println(result);
13     }
14   }
15 }
```

Analisis kegunaan kode program pada line 7 dan amati hasil kode program di atas pada terminal.

Modifikasi isi file **"App.java"**, seperti berikut:

App.java

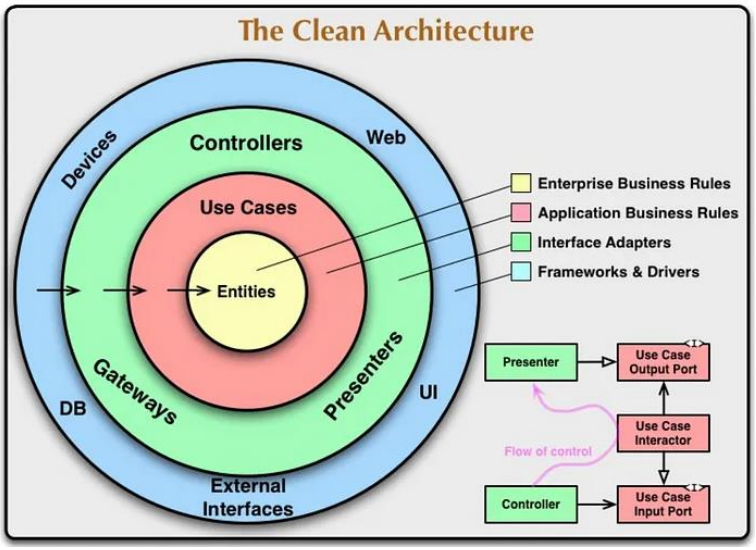
```
1  public class App {
2    public static void main(String[] args) {
3
4      var person1 = new Person();
5      person1.name = "Abdullah";
6      person1.address = "Sitoluama";
7      // person1.country = "Singapura";
8
9      System.out.println(person1.name);
10     System.out.println(person1.address);
11     System.out.println(person1.country);
12
13   }
14 }
```

Analisis kegunaan kode program pada line 5, 6, 9, 10, dan 11. Mengapa kode program pada line 7, jika dihilangkan komen-nya akan mengalami error. Amati hasilnya pada terminal.

2. Studi Kasus Java OOP: Aplikasi Todo

Kita akan mengimplementasikan Clean Architecture karena telah belajar Java OOP agar membuat struktur aplikasi yang baik.

| Clean Architecture



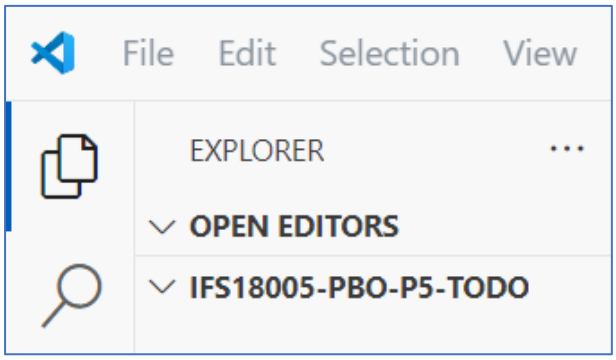
Gambar 1. The Clean Architecture oleh Robert C. Martin

The Clean Architecture merupakan sebuah buku yang dibuat oleh Robert C. Martin (Uncle Bob) disana menjelaskan Clean Architectur untuk membuat struktur kode proyek yang baik sehingga mudah untuk dikembangkan (*scalable*). Clean Architecture untuk pengembangan aplikasi akan membagi layer menjadi 4 bagian. Untuk layer paling luar yang bewarna biru merupakan Framework & Drivers, ini bukan hal yang kita handle karena bergantung dari teknologi yang kita gunakan, sebagai contoh kita menggunakan Driver Database dan Spring Framework. Layer yang akan kita handle adalah dari warna hijau, merah dan kuning. Layer berwarna hijau merupakan view dari aplikasi yang dibuat, dalam kasus kita nantinya tampilan aplikasi akan berupa teks pada console atau terminal. Pada layer berwarna merah merupakan buisness logic dan layer terakhir yang berwarna kuning merupakan presentasi dari datanya. Dengan menerapkan Clean Architecture kita dapat mencegah ketergantungan antar komponen dan layer dalam aplikasi.

Aturan Ketergantungan menyatakan bahwa ketergantungan kode sumber hanya dapat mengarah ke dalam. Ini berarti tidak ada apa pun di lingkaran dalam yang dapat mengetahui apa pun tentang sesuatu di lingkaran luar. yaitu lingkaran dalam seharusnya tidak bergantung pada apa pun di lingkaran luar. Panah Hitam yang diwakili dalam diagram menunjukkan aturan ketergantungan. Ini adalah aturan penting yang membuat arsitektur ini berfungsi.

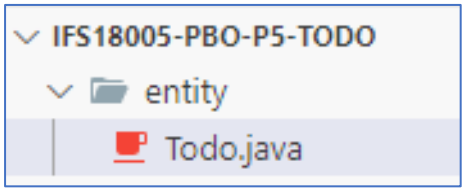
a) Setup Proyek

Buat workspace baru dengan format penamaan "**{username}-pbo-p5-todo**".



b) Membuat Entity

Silahkan membuat package baru dengan nama "**entity**", di dalam package tersebut silahkan tambahkan class baru dengan nama "**Todo.java**".



Modifikasi isi file "**entity/Todo.java**", seperti berikut:

Todo.java

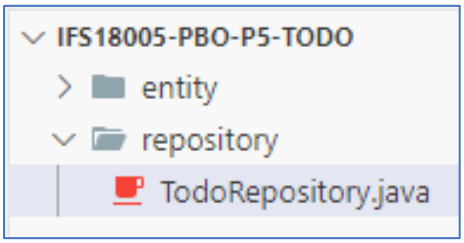
```
1  package entity;
2
3  public class Todo {
4    public static int total = 0;
5
6    private int id;
7    private String title;
8    private boolean finished;
9
10   private void setId() {
11     total++;
12     id = total;
13   }
14
15   public Todo() {
16     setId();
17   }
18
19   public Todo(String title) {
20     setId();
21     this.title = title;
22   }
23
24   public Todo(String title, boolean finished) {
25     setId();
26     this.title = title;
27     this.finished = finished;
28   }
29
30   public int getId() {
31     return id;
```

Todo.java

```
32    }
33
34    public String getTitle() {
35      return title;
36    }
37
38    public void setTitle(String title) {
39      this.title = title;
40    }
41
42    public boolean isFinished() {
43      return finished;
44    }
45
46    public void setFinished(boolean finished) {
47      this.finished = finished;
48    }
49
50    @Override
51    public String toString() {
52      String status = (finished) ? "Selesai" : "Belum Selesai";
53      return String.format("%d | %s | %s", id, title, status);
54    }
55
56  }
```

c) Membuat Repository

Silahkan membuat package baru dengan nama "**repository**", di dalam package tersebut silahkan tambahkan class baru dengan nama "**TodoRepository.java**".



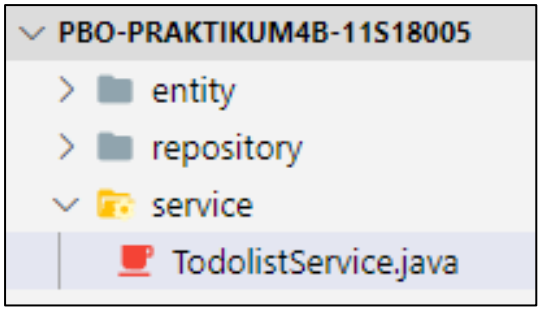
Modifikasi isi file "**repository/TodoRepository.java**", seperti berikut:

TodoRepository.java

```
1   package repository;
2
3   import entity.Todo;
4
5   public interface TodoRepository {
6     Todo[] repoGetAllTodo();
7
8     void repoAddTodo(Todo newTodo);
9
10    boolean repoRemoveTodo(Integer idTodo);
11  }
```

d) Membuat Service

Silahkan membuat package baru dengan nama "**service**", di dalam package tersebut silahkan tambahkan class baru dengan nama "**TodolistService.java**".



Modifikasi isi file **"service/TodoService.java"**, seperti berikut:

TodoService.java

```
1  package service;
2
3  public interface TodoService {
4    void serviceShowTodo();
5
6    void serviceAddTodo(String title);
7
8    void serviceRemoveTodo(Integer id);
9  }
```

e) Membuat View

Silahkan membuat package baru dengan nama **"view"**, di dalam package tersebut silahkan tambahkan class baru dengan nama **"TodoView.java"**.

Modifikasi isi file **"view/TodoView.java"**, seperti berikut:

TodoView.java

```
1  package view;
2
3  public class TodoView {
4
5    /**
6     * Menampilkan view todo
7     */
8    public void viewShowTodo() {
9
10   }
11
12   /**
13    * Menampilkan view add todo
14    */
15   public void viewAddTodo() {
16
17   }
```

TodoView.java

```
18
19    /**
20     * Menampilkan view remove todo
21     */
22    public void viewRemoveTodo() {
23
24    }
25
26    /**
27     * Menampilkan view update todo
28     */
29    public void viewUpdateTodo() {
30
31    }
32
33  }
```

f) Implementasi Repository

(!) impl singkatan dari implementasi.

Buat dan modifikasi isi file **"repository/TodoRepositoryImpl.java"**, seperti berikut:

TodoRepositoryImpl.java

```
1   package repository;
2
3   import entity.Todo;
4
5   public class TodoRepositoryImpl implements TodoRepository {
6
7     public Todo[] data = new Todo[10];
8
9     @Override
10    public Todo[] repoGetAllTodo() {
11      return data;
12    }
13
14    @Override
15    public void repoAddTodo(Todo newTodo) {
16      resizeIfFull();
17      // tambahkan ke posisi yang data array-nya NULL
18      for (int i = 0; i < data.length; i++) {
19        if (data[i] == null) {
20          data[i] = newTodo;
21          break;
22        }
23      }
24    }
25
26    @Override
27    public boolean repoRemoveTodo(Integer idTodo) {
28      int position = 0;
29      for (Todo todo : data) {
30        if (todo == null)
31          break;
32
33        position++;
34        if (todo.getId() == idTodo)
35          break;
36      }
37
38      if (position <= 0)
39        return false;
40
```

TodoRepositoryImpl.java

```
41      if ((position - 1) >= data.length)
42        return false;
43
44      if (data[position - 1] == null)
45        return false;
46
47      for (int i = (idTodo - 1); i < data.length; i++) {
48        if (i == (data.length - 1)) {
49          data[i] = null;
50        } else {
51          data[i] = data[i + 1];
52        }
53      }
54      return true;
55    }
56
57    private boolean isFull() {
58      var isFull = true;
59      for (int i = 0; i < data.length; i++) {
60        if (data[i] == null) {
61          // model masih ada yang kosong
62          isFull = false;
63          break;
64        }
65      }
66      return isFull;
67    }
68
69    private void resizeIfFull() {
70      // Jika penuh, resize ukuran array 2x lipat
71      if (isFull()) {
72        var temp = data;
73        data = new Todo[data.length * 2];
74
75        for (int i = 0; i < temp.length; i++) {
76          data[i] = temp[i];
77        }
78      }
79    }
80
81  }
```

g) Implementasi Service

Buat dan modifikasi isi file **"service/TodoServiceImpl.java"**, seperti berikut:

TodoServiceImpl.java

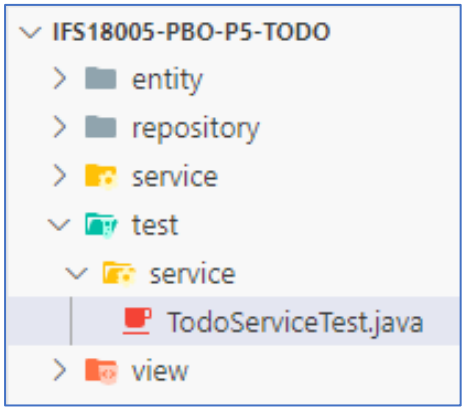
```
1   package service;
2
3   import entity.Todo;
4   import repository.TodoRepository;
5
6   public class TodoServiceImpl implements TodoService {
7     private TodoRepository todoRepository;
8
9     public TodoServiceImpl(TodoRepository todoRepository) {
10      this.todoRepository = todoRepository;
11    }
12
13    @Override
14    public void serviceShowTodo() {
15      Todo[] todos = todoRepository.repoGetAllTodo();
16      System.out.println("Daftar Todo:");
17      int counter = 0;
```

TodoServiceImpl.java

```
18      for (Todo todo : todos) {
19        if (todo != null) {
20          counter++;
21          System.out.println(todo);
22        } else {
23          break;
24        }
25      }
26
27      if (counter <= 0) {
28        System.out.println("- Data todo belum tersedia!");
29      }
30    }
31
32    @Override
33    public void serviceAddTodo(String todo) {
34      Todo newTodo = new Todo(todo);
35      todoRepository.repoAddTodo(newTodo);
36    }
37
38    @Override
39    public void serviceRemoveTodo(Integer id) {
40      boolean success = todoRepository.repoRemoveTodo(id);
41      if (!success) {
42        System.out.printf("[!] Gagal menghapus todo dengan ID: %d.\n", id);
43        return;
44      }
45    }
46  }
```

h) Test Repository & Service

Silahkan membuat package baru dengan nama "**test**", di dalam package tersebut tambahkan package baru dengan nama "**service**", di dalam package tersebut silahkan tambahkan class baru dengan nama "**TodoServiceTest.java**".



Buat dan modifikasi isi file "**test/service/TodoServiceTest.java**", seperti berikut:

TodoServiceTest.java

```
1   package test.service;
2
3   import entity.Todo;
4   import repository.TodoRepository;
5   import repository.TodoRepositoryImpl;
6   import service.TodoService;
7   import service.TodoServiceImpl;
8
```

TodoServiceTest.java

```
9  public class TodoServiceTest {
10
11   public static void main(String[] args) {
12     testShowTodo();
13     testAddTodo();
14     testRemoveTodo();
15   }
16
17   public static void testShowTodo() {
18     System.out.println("Test Show Todo:");
19     TodoRepositoryImpl todoRepositoryImpl = new TodoRepositoryImpl();
20     todoRepositoryImpl.data[0] = new Todo("Belajar PBO");
21     todoRepositoryImpl.data[1] = new Todo("Belajar BASDAT");
22     todoRepositoryImpl.data[2] = new Todo("Belajar AOK");
23     todoRepositoryImpl.data[3] = new Todo("Belajar SISDIG");
24
25     TodoService todoService = new TodoServiceImpl(todoRepositoryImpl);
26     todoService.serviceShowTodo();
27     System.out.println();
28   }
29
30   public static void testAddTodo() {
31     System.out.println("Test Add Todo:");
32     TodoRepository todoRepository = new TodoRepositoryImpl();
33     TodoService todoService = new TodoServiceImpl(todoRepository);
34
35     todoService.serviceAddTodo("Belajar PBO");
36     todoService.serviceAddTodo("Belajar BASDAT");
37     todoService.serviceAddTodo("Belajar AOK");
38     todoService.serviceAddTodo("Belajar SISDIG");
39
40     todoService.serviceShowTodo();
41     System.out.println();
42   }
43
44   public static void testRemoveTodo() {
45     System.out.println("Test Remove Todo:");
46     TodoRepository todoRepository = new TodoRepositoryImpl();
47     TodoService todoService = new TodoServiceImpl(todoRepository);
48
49     todoService.serviceAddTodo("Belajar PBO");
50     todoService.serviceAddTodo("Belajar BASDAT");
51     todoService.serviceAddTodo("Belajar AOK");
52     todoService.serviceAddTodo("Belajar SISDIG");
53
54     todoService.serviceShowTodo();
55
56     // Menghapus data yang tidak tersedia
57     todoService.serviceRemoveTodo(5);
58     // Menghapus data yang tersedia
59     todoService.serviceRemoveTodo(3);
60     todoService.serviceShowTodo();
61
62     todoService.serviceRemoveTodo(1);
63     todoService.serviceRemoveTodo(2);
64     todoService.serviceShowTodo();
65
66     todoService.serviceRemoveTodo(4);
67     todoService.serviceShowTodo();
68     System.out.println();
69   }
70
71 }
```

Jalankan aplikasi pada terminal dengan menuliskan perintah:

- Compile

```
javac -d output test/service/TodoServiceTest.java
```

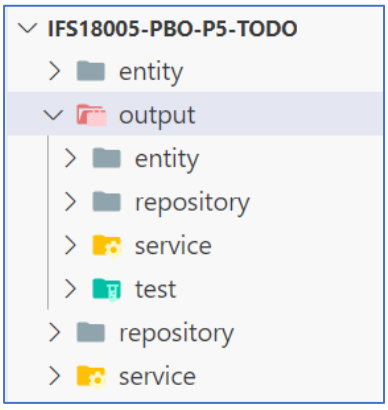
- Run

```
java -cp output test.service.TodoServiceTest
```

- Tampilan pada terminal

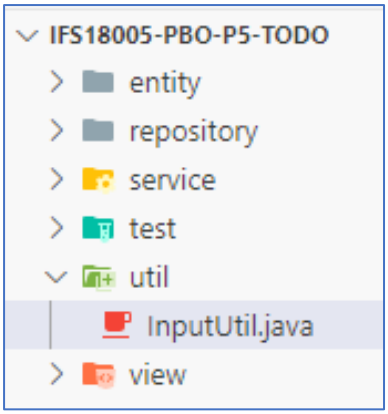
```
ifs18005-pbo-p5-todo> javac -d output test/service/TodoServiceTest.java
ifs18005-pbo-p5-todo> java -cp output test.service.TodoServiceTest
Test Show Todo:
Daftar Todo:
1 | Belajar PBO | Belum Selesai
2 | Belajar BASDAT | Belum Selesai
3 | Belajar AOK | Belum Selesai
4 | Belajar SISDIG | Belum Selesai

Test Add Todo:
Daftar Todo:
1 | Belajar PBO | Belum Selesai
2 | Belajar BASDAT | Belum Selesai
3 | Belajar AOK | Belum Selesai
4 | Belajar SISDIG | Belum Selesai

Test Remove Todo:
Daftar Todo:
1 | Belajar PBO | Belum Selesai
2 | Belajar BASDAT | Belum Selesai
3 | Belajar AOK | Belum Selesai
4 | Belajar SISDIG | Belum Selesai
[!] Gagal menghapus todo dengan ID: 5.
Daftar Todo:
1 | Belajar PBO | Belum Selesai
2 | Belajar BASDAT | Belum Selesai
4 | Belajar SISDIG | Belum Selesai
Daftar Todo:
4 | Belajar SISDIG | Belum Selesai
Daftar Todo:
- Data todo belum tersedia!
```

- Hasil *compile* akan disimpan pada folder output



i) Util

Silahkan membuat package baru dengan nama "**util**", di dalam package tersebut silahkan tambahkan class baru dengan nama "**InputUtil.java**".



Modifikasi isi file **"util/InputUtil.java"**, seperti berikut:

InputUtil.java

```
1  package util;
2
3  import java.util.Scanner;
4
5  public class InputUtil {
6    private static Scanner scanner = new Scanner(System.in);
7
8    public static String input(String info) {
9      System.out.print(info + " : ");
10     String data = scanner.nextLine();
11     return data;
12   }
13
14 }
```

j) Implementasi View

Modifikasi isi file **"view/TodoView.java"**, seperti berikut:

TodoView.java

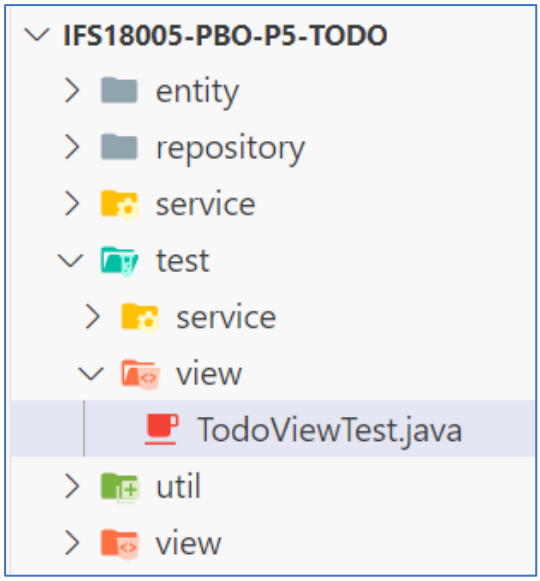
```
1  package view;
2
3  import service.TodoService;
4  import util.InputUtil;
5
6  public class TodoView {
7
8    private TodoService todoService;
9
10   public TodoView(TodoService todoService) {
11     this.todoService = todoService;
12   }
13
14   /**
15    * Menampilkan view todo
16    */
17   public void viewShowTodo() {
18     while (true) {
19       todoService.serviceShowTodo();
20
21       System.out.println("Menu:");
22       System.out.println("1. Tambah");
23       System.out.println("2. Ubah");
24       System.out.println("3. Hapus");
25       System.out.println("x. Keluar");
26
27       var input = InputUtil.input("Pilih");
28       var stop = false;
```

TodoView.java

```
29          switch (input) {
30            case "1" -> viewAddTodo();
31            case "2" -> viewUpdateTodo();
32            case "3" -> viewRemoveTodo();
33            case "x" -> stop = true;
34            default -> System.out.println("[!] Pilihan tidak dimengerti.");
35          }
36
37          if (stop) {
38            break;
39          }
40
41          System.out.println();
42        }
43      }
44
45      /**
46       * Menampilkan view add todo
47       */
48      public void viewAddTodo() {
49        System.out.println("[Menambah Todo]");
50        var title = InputUtil.input("Judul (x Jika Batal)");
51
52        if (!title.equals("x")) {
53          todoService.serviceAddTodo(title);
54        }
55      }
56
57      /**
58       * Menampilkan view remove todo
59       */
60      public void viewRemoveTodo() {
61        System.out.println("[Menghapus Todo]");
62
63        var strIdTodo = InputUtil.input("[ID Todo] yang dihapus (x Jika
    Batal)");
64
65        if (!strIdTodo.equals("x")) {
66          int idTodo = Integer.valueOf(strIdTodo);
67          todoService.serviceRemoveTodo(idTodo);
68        }
69      }
70
71      /**
72       * Menampilkan view update todo
73       */
74      public void viewUpdateTodo() {
75
76      }
77
78  }
```

k) Test View

Silahkan membuat package baru dengan nama "**view**" di dalam package "**test**", di dalam package tersebut silahkan tambahkan class baru dengan nama "**TodoViewTest.java**".



Modifikasi isi file **"test/view/TodoViewTest.java"**, seperti berikut:

TodoViewTest.java

```
1  package test.view;
2
3  import entity.Todo;
4  import repository.TodoRepository;
5  import repository.TodoRepositoryImpl;
6  import service.TodoService;
7  import service.TodoServiceImpl;
8  import view.TodoView;
9
10 public class TodoViewTest {
11
12   public static void main(String[] args) {
13     testViewShowTodo();
14     testViewAddTodo();
15     testViewRemoveTodo();
16   }
17
18   public static void testViewShowTodo() {
19     System.out.println("Test View Show Todo:");
20     Todo.total = 0;
21     TodoRepository todoRepository = new TodoRepositoryImpl();
22     TodoService todoService = new TodoServiceImpl(todoRepository);
23     TodoView todoView = new TodoView(todoService);
24
25     todoService.serviceAddTodo("Belajar PBO");
26     todoService.serviceAddTodo("Belajar BASDAT");
27     todoService.serviceAddTodo("Belajar AOK");
28     todoService.serviceAddTodo("Belajar SISDIG");
29
30     todoView.viewShowTodo();
31     System.out.println();
32   }
33
34   public static void testViewAddTodo() {
35     System.out.println("Test View Add Todo:");
36     Todo.total = 0;
37     TodoRepository todoRepository = new TodoRepositoryImpl();
38     TodoService todoService = new TodoServiceImpl(todoRepository);
39     TodoView todoView = new TodoView(todoService);
40
41     todoView.viewAddTodo();
```

TodoViewTest.java

```
42        todoService.serviceShowTodo();
43
44        todoView.viewAddTodo();
45        todoService.serviceShowTodo();
46        System.out.println();
47      }
48
49      public static void testViewRemoveTodo() {
50        System.out.println("Test View Remove Todo:");
51        Todo.total = 0;
52        TodoRepository todoRepository = new TodoRepositoryImpl();
53        TodoService todoService = new TodoServiceImpl(todoRepository);
54        TodoView todoView = new TodoView(todoService);
55
56        todoService.serviceAddTodo("Belajar PBO");
57        todoService.serviceAddTodo("Belajar BASDAT");
58        todoService.serviceAddTodo("Belajar AOK");
59        todoService.serviceAddTodo("Belajar SISDIG");
60
61        todoService.serviceShowTodo();
62        todoView.viewRemoveTodo();
63
64        todoService.serviceShowTodo();
65        todoView.viewRemoveTodo();
66
67        todoService.serviceShowTodo();
68        System.out.println();
69      }
70    }
```

Jalankan aplikasi pada terminal dengan menuliskan perintah:

- Compile

```
javac -d output test/view/TodoViewTest.java
```

- Run

```
java -cp output test.view.TodoViewTest
```

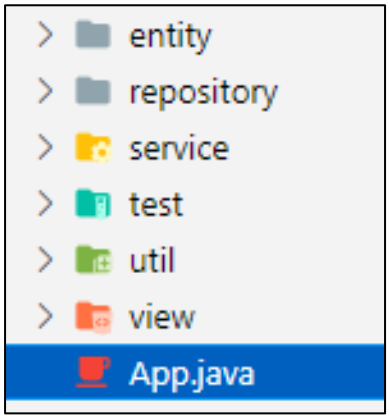
- Tampilan pada terminal

```
ifs18005-pbo-p5-todo> javac -d output test/view/TodoViewTest.java
ifs18005-pbo-p5-todo> java -cp output test.view.TodoViewTest
Test View Show Todo:
Daftar Todo:
1 | Belajar PBO | Belum Selesai
2 | Belajar BASDAT | Belum Selesai
3 | Belajar AOK | Belum Selesai
4 | Belajar SISDIG | Belum Selesai
Menu:
1. Tambah
2. Ubah
3. Hapus
x. Keluar
Pilih : x

Test View Add Todo:
[Menambah Todo]
Judul (x Jika Batal) : Belajar OOP
Daftar Todo:
1 | Belajar OOP | Belum Selesai
[Menambah Todo]
Judul (x Jika Batal) : x
Daftar Todo:
1 | Belajar OOP | Belum Selesai
```

```
Test View Remove Todo:
Daftar Todo:
1 | Belajar PBO | Belum Selesai
2 | Belajar BASDAT | Belum Selesai
3 | Belajar AOK | Belum Selesai
4 | Belajar SISDIG | Belum Selesai
[Menghapus Todo]
[ID Todo] yang dihapus (x Jika Batal) : 1
Daftar Todo:
2 | Belajar BASDAT | Belum Selesai
3 | Belajar AOK | Belum Selesai
4 | Belajar SISDIG | Belum Selesai
[Menghapus Todo]
[ID Todo] yang dihapus (x Jika Batal) : x
Daftar Todo:
2 | Belajar BASDAT | Belum Selesai
3 | Belajar AOK | Belum Selesai
4 | Belajar SISDIG | Belum Selesai
```

1) Menjalankan Aplikasi

Silahkan membuat class baru pada root workspace dengan nama "**App.java**"



Buat dan modifikasi isi file "**App.java**", seperti berikut:

App.java

```
1  import repository.TodoRepository;
2  import repository.TodoRepositoryImpl;
3  import service.TodoService;
4  import service.TodoServiceImpl;
5  import view.TodoView;
6
7  public class App {
8    public static void main(String[] args) {
9      TodoRepository todoRepository = new TodoRepositoryImpl();
10     TodoService todoService = new TodoServiceImpl(todoRepository);
11     TodoView todoView = new TodoView(todoService);
12
13     todoView.viewShowTodo();
14   }
15 }
```

Jalankan aplikasi pada terminal dengan menuliskan perintah:

- Compile

```
javac -d output App.java
```

- Run

```
java -cp output App
```

3. Menambahkan Fitur Update pada Aplikasi Todo

Silahkan menambahkan fitur baru untuk dapat mengubah data todo yang telah ditambahkan ke dalam model. Selanjutnya ubah penamaan file App menjadi **Todo_{NIM}.java**

Berikut merupakan contoh program saat dijalankan:

```
Daftar Todo:
- Data todo belum tersedia!
Menu:
1. Tambah
2. Ubah
3. Hapus
x. Keluar
Pilih : 1
[Menambah Todo]
Judul (x Jika Batal) : Belajar OOP

Daftar Todo:
1 | Belajar OOP | Belum Selesai
Menu:
1. Tambah
2. Ubah
3. Hapus
x. Keluar
Pilih : 1
[Menambah Todo]
Judul (x Jika Batal) : Belajar HTML-CSS-JS

Daftar Todo:
1 | Belajar OOP | Belum Selesai
2 | Belajar HTML-CSS-JS | Belum Selesai
Menu:
1. Tambah
2. Ubah
3. Hapus
x. Keluar
Pilih : 2
[Mengubah Todo]
[ID Todo] yang diubah (x Jika Batal) : 1
[!] Tekan enter untuk menggunakan judul lama.
Judul Baru [Judul Lama: Belajar OOP] : Belajar Java OOP
Status Todo [0: Belum Selesai, 1: Selesai] : 1

Daftar Todo:
1 | Belajar Java OOP | Selesai
2 | Belajar HTML-CSS-JS | Belum Selesai
Menu:
1. Tambah
2. Ubah
3. Hapus
x. Keluar
Pilih : x
```

(!) Latar belakang oranye merupakan inputan.