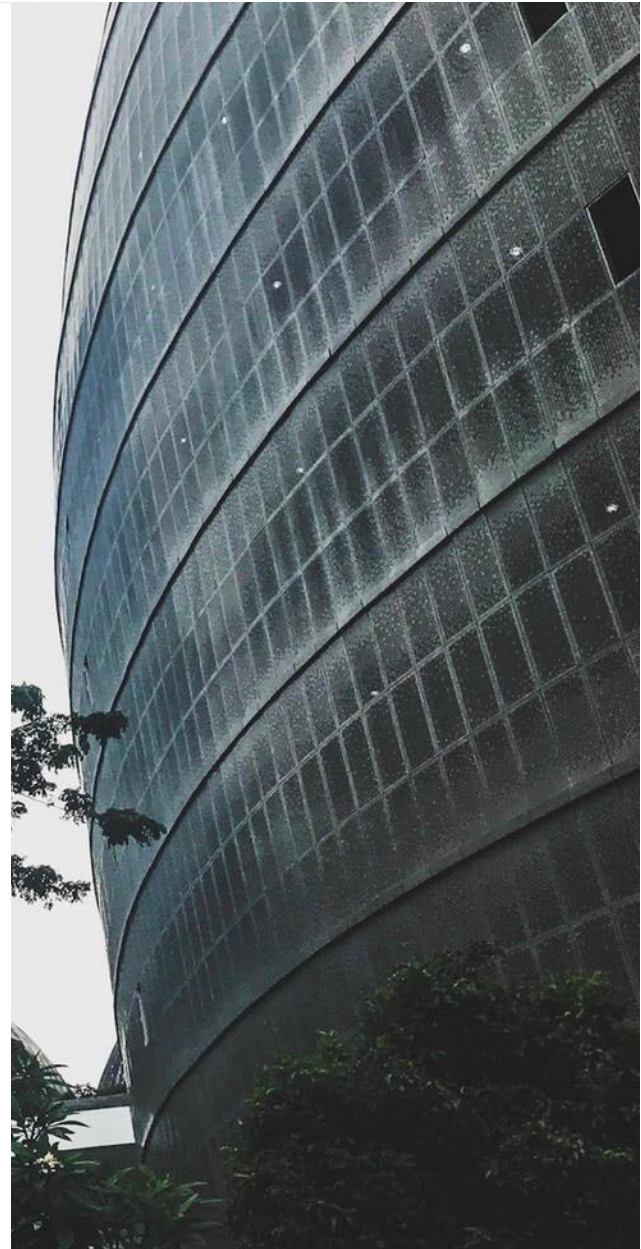


# MODUL PRAKTIKUM

IF402 – PEMROGRAMAN BERORIENTASI OBJEK  
PROGRAM SARJANA S1 INFORMATIKA  
FAKULTAS TEKNIK DAN INFORMATIKA

---



---

**PROGRAM STUDI INFORMATIKA  
FAKULTAS TEKNIK DAN INFORMATIKA  
UNIVERSITAS MULTIMEDIA NUSANTARA**

Gedung B Lantai 5, Kampus UMN  
Jl. Scientia Boulevard, Gading Serpong, Tangerang, Banten-15811 Indonesia  
Telp: +62-21.5422.0808 (ext. 1803), email: [ict.lab@umn.ac.id](mailto:ict.lab@umn.ac.id), web: [umn.ac.id](http://umn.ac.id)

## DAFTAR ISI

CAPAIAN PEMBELAJARAN PRAKTIKUM.....	3
MODUL 1 (OOP, OOP vs PEMROGRAMAN PROSEDURAL, KARAKTERISTIK OOP, SEJARAH JAVA, INTELLIJ IDEA) .....	4
MODUL 2 (KONSEP I/O, STRUKTUR KONTROL PEMILIHAN & PENGULANGAN).....	10
MODUL 3 (PRIMITIVE DATA TYPE, ARRAY, MULTIDIMENSIONAL ARRAY, STRING) ....	15
MODUL 4 (CLASS, OBJECT, METHOD, ENCAPSULATION, CONSTRUCTOR, DESTRUCTOR, STATIC MODIFIER).....	20
MODUL 5 (INHERITANCE, POLYMORPHISM, IMMUTABLE CLASS, ACCESS MODIFIERS) .....	30
MODUL 6 (CLASS HIERARCHY, CLASS DIAGRAM, ACTIVITY DIAGRAM, USE CASE DIAGRAM) .....	37
MODUL 7 (ABSTRACTION, INTERFACE, ABSTRACT CLASS) .....	45
MODUL 8 (CASTING PADA JAVA) .....	61
MODUL 9 (FUNCTION OVERRIDING & FUNCTION OVERLOADING) .....	69
MODUL 10 (EXCEPTION HANDLING) .....	73
MODUL 11 (KONSEP GUI, LAYOUT, CONTAINER, FRAME, PANEL, UI COMPONENTS LAINNYA).....	82
MODUL 12 (ACCORDION, TABS, TABLE) .....	100

1. Mahasiswa mampu menerangkan (C2) dan membangun (P2) perangkat lunak sederhana dengan Bahasa Pemrograman Java dengan menggunakan IntelliJ IDEA.
2. Mahasiswa mampu mencontohkan (C2) berbagai struktur kontrol dasar pada Bahasa Pemrograman Java dan menerapkan (C2) berbagai struktur kontrol pada Bahasa pemrograman Java.
3. Mahasiswa mampu membandingkan (C2) kegunaan dari setiap primitive datatype pada Bahasa Pemrograman Java dan memilih (A3) serta mengimplementasikan (C3) konsep dimensi pada array untuk setiap kasus yang sesuai.
4. Mahasiswa mampu mengabstraksikan (C2) properti-properti yang dimiliki sebuah objek menggunakan enkapsulasi pada Bahasa Pemrograman Java serta memilih (A3) dan menerapkan (P2) kontrol akses yang tepat pada setiap kasus.
5. Mahasiswa mampu menguraikan (C2) konsep Inheritance pada sebuah Concrete Class serta memodifikasi (C3) komponen-komponen pada sebuah kelas menggunakan konsep Polymorphism, dan membangun sebuah program menggunakan konsep Concrete Inheritance dan Polymorphism pada Bahasa Pemrograman Java.
6. Mahasiswa mampu mendiagramkan perancangan dasar (UML) untuk sebuah program berorientasi object serta mengembangkan (P4) sebuah program berdasarkan suatu diagram UML menggunakan Bahasa Pemrograman Java.
7. Mahasiswa dapat menerima (A2) dan menguraikan (C4) konsep-konsep abstraksi pada Pemrograman Berorientasi Objek serta membangun (P4) sebuah program berdasarkan konsep-konsep abstraksi yang telah dipelajari.
8. Mahasiswa mampu mematuhi (A2) dan menegaskan (C4) konsep casting pada Pemrograman Berorientasi Objek serta menentukan (P5) dan mensimulasikan (C3) proses-proses casting yang mungkin terjadi pada setiap datatype.
9. Mahasiswa mempelajari konsep function overloading dan function overriding.
10. Mahasiswa mampu memperjelas (C5) tata cara penanganan error menggunakan konsep exception handling dan menyempurnakan (A4) sebuah program dengan konsep exception handling.
11. Mahasiswa dapat membuat struktur (C4) dan membangun (P2) sebuah graphical user interface berdasarkan konsep layout, container, frame, ataupun component pada pemrograman berbasis objek.
12. Mahasiswa dapat membangun (C6, P5) sebuah aplikasi berdasarkan tata cara merepresentasikan data dengan menggunakan Built-in UI pada Bahasa Pemrograman Java.

## MODUL 1

# (OOP, OOP vs PEMROGRAMAN PROSEDURAL, KARAKTERISTIK OOP, SEJARAH JAVA, INTELLIJ IDEA)

### DESKRIPSI TEMA

Sejarah JAVA dan Pemrograman Berorientasi Objek vs Prosedural.

### CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

1. Mahasiswa mampu melakukan instalasi IDE untuk Pemrograman Berorientasi Objek.
2. Mahasiswa mampu membuat Aplikasi Sederhana dengan Bahasa Java.

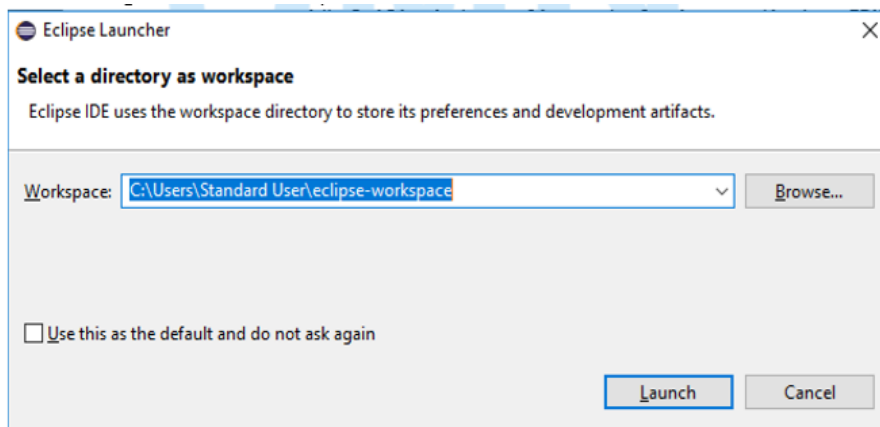
### PENUNJANG PRAKTIKUM

1. Eclipse.

### LANGKAH-LANGKAH PRAKTIKUM

#### Tutorial – Instalasi & Memulai Pemrograman Berorientasi Objek

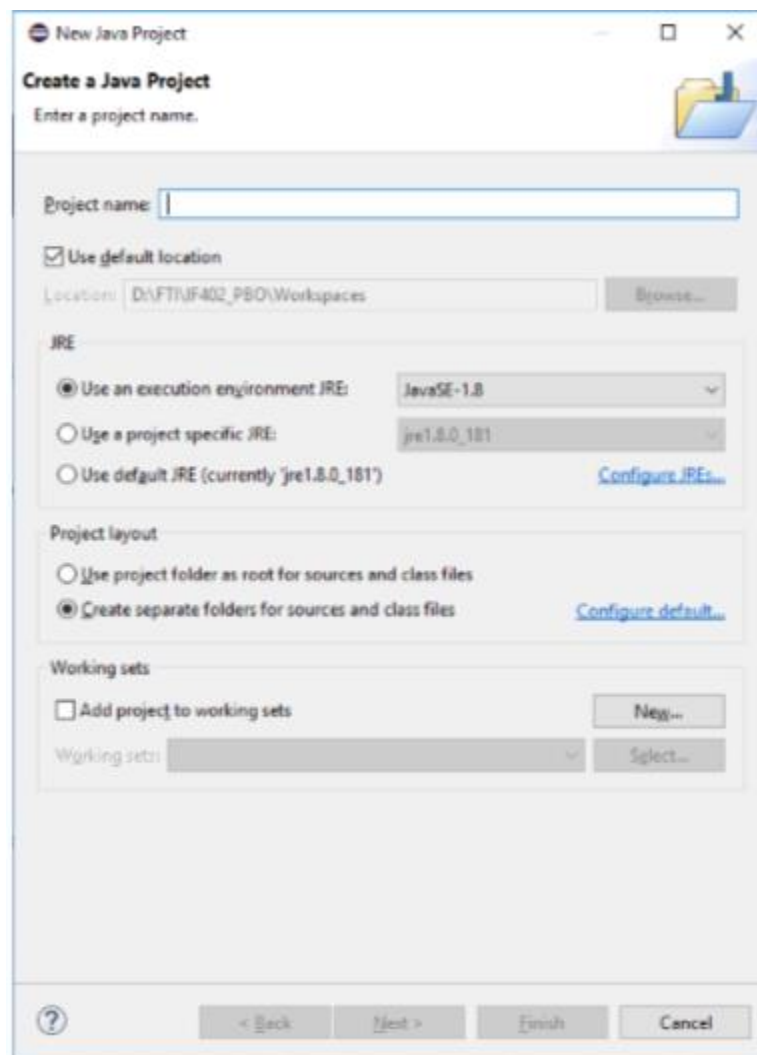
1. Unduh installer dan file yang ada pada tautan berikut:  
<https://drive.google.com/open?id=1jAjYQuj-yEp5Jj11ciDAiYPwMwZtrYDW>
  - eclipse-java-photon-R-win32-x86\_64.zip (selanjutnya disebut **eclipse**)
  - jdk-8u181-windows-x64.exe (selanjutnya disebut **JDK**)
2. Install JDK terlebih dahulu.
3. Extract Eclipse pada posisi yang mudah Anda ingat. (Pada PC Perkuliahan, posisi Eclipse terletak pada **D:\FTI\IF402\_PBO\**)
4. Jalankan **eclipse.exe** (atau **Eclipse** shortcut) dan Anda akan disuguhkan halaman seperti berikut.



5. Workspace merupakan folder dimana project Anda akan berada. Pilihlah folder yang sesuai dengan tempat Anda biasa meletakkan folder project. (Pada PC Perkuliahan, biasakan Workspace diletakkan pada posisi **D:\FTI\IF402\_PBO\Workspaces**, dan jangan mencentang "Use this as the default and do not ask again").
6. Selamat Anda sudah berhasil memasang dan membuka Program **Eclipse**.

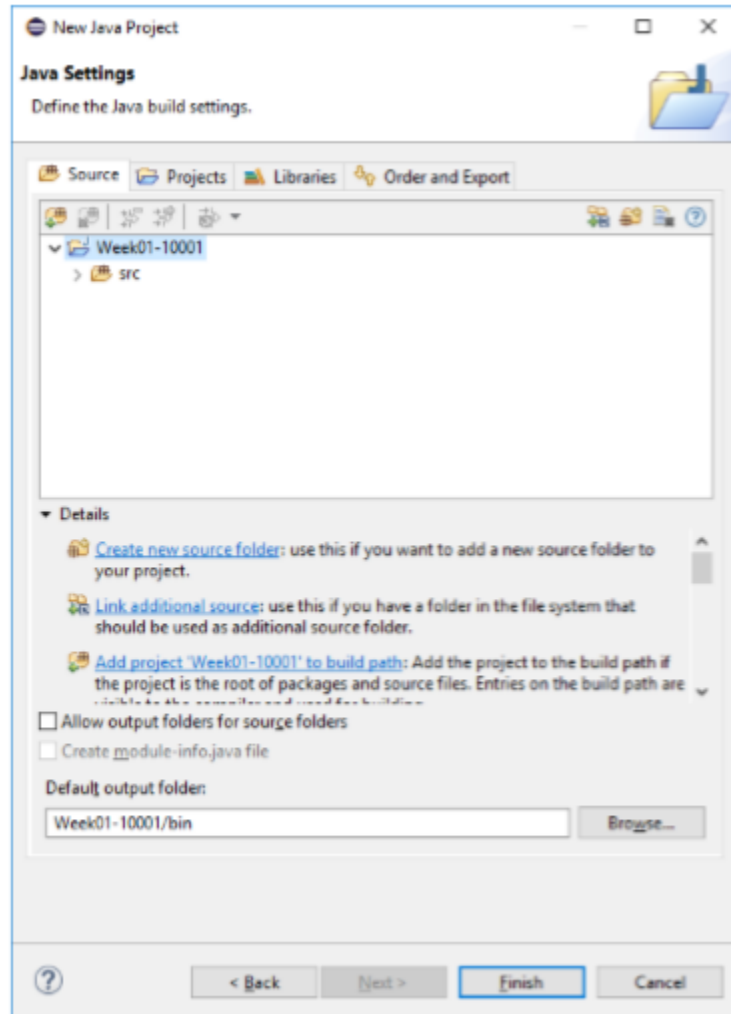
### Tutorial – Aplikasi Sederhana dengan Bahasa Java

1. Buka menu bar **File -> New -> Java Project** dan Anda akan disuguhkan tampilan **New Java Project – Create a Java Project** sebagai berikut

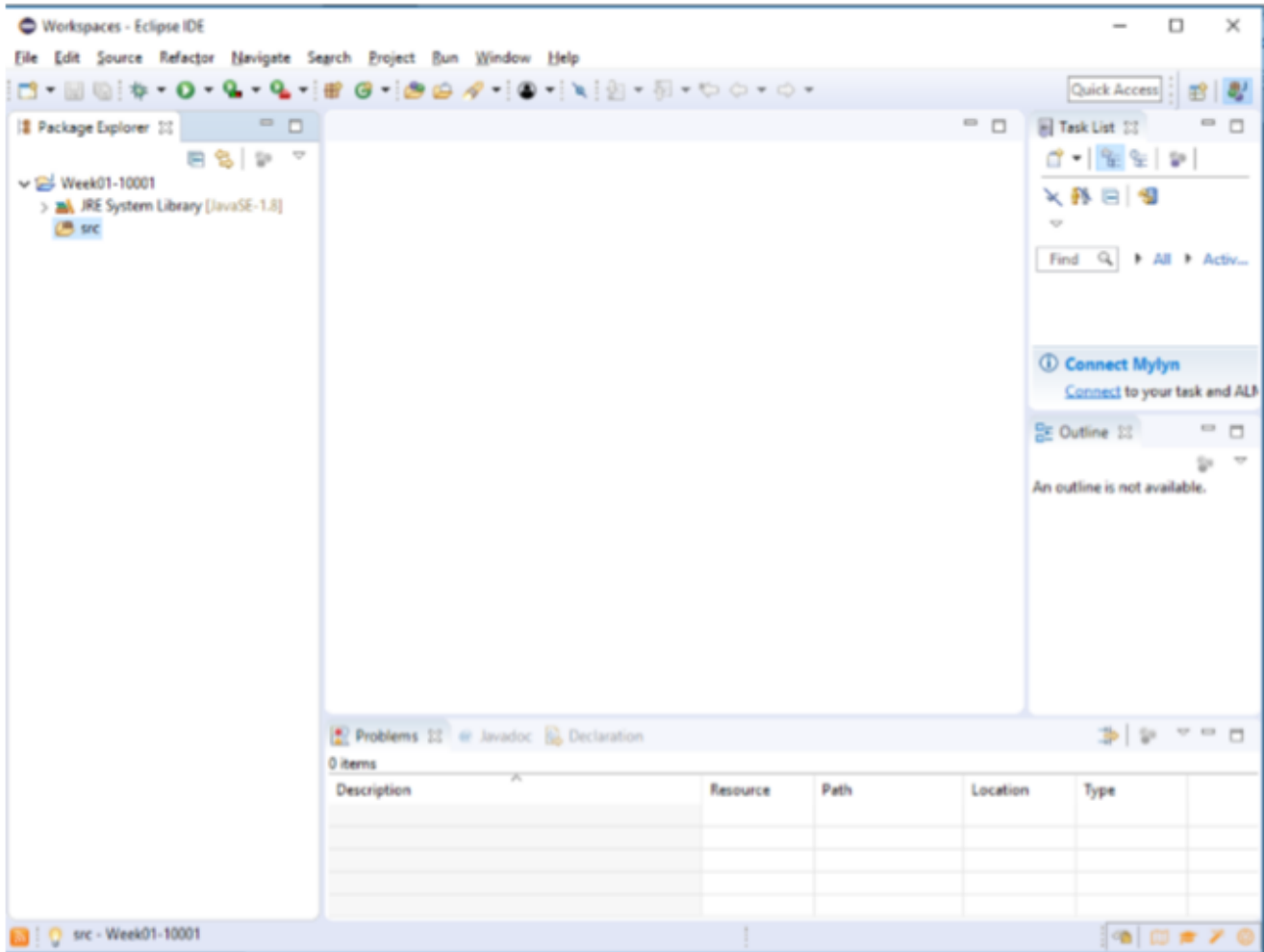


Pada perkuliahan, Anda diharapkan menggunakan naming convention untuk project berupa **Week[xx]-NIM**, e.g. **Week01-10001**.

2. Selanjutnya Anda akan disuguhkan tampilan **New Java Project – Java Setting** sebagai berikut, tekanlah tombol **Finish**.

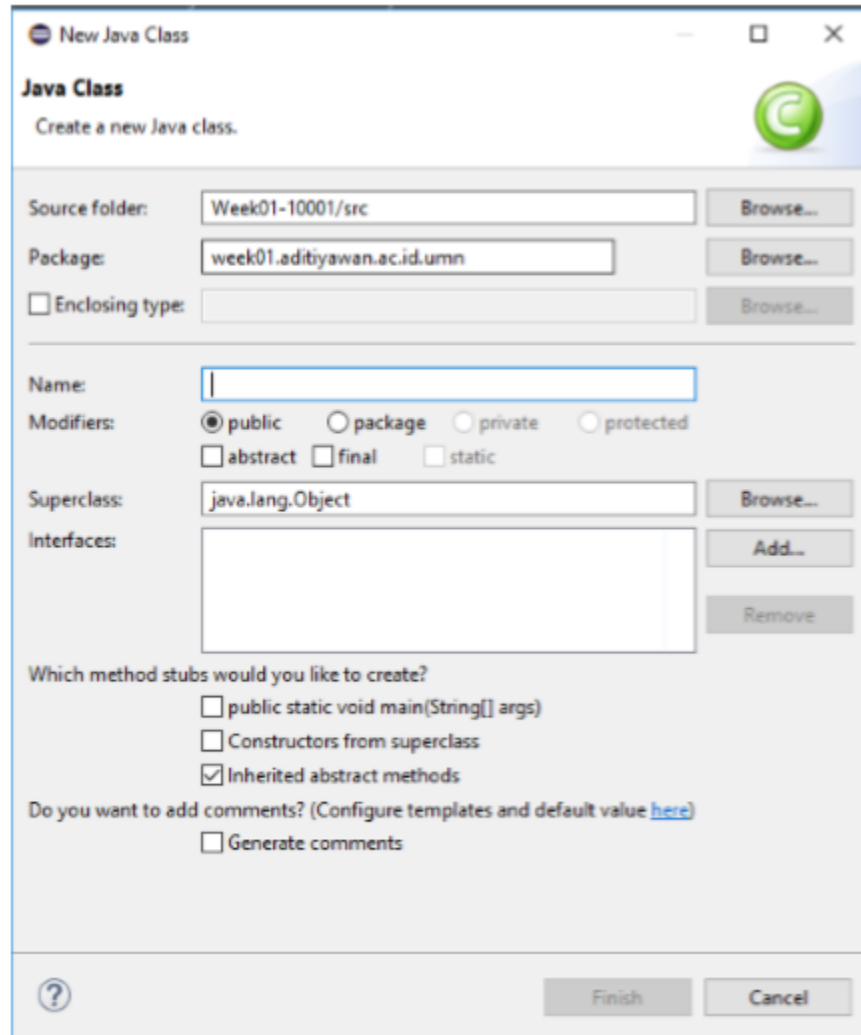


3. Selanjutnya Anda akan disuguhkan tampilan sebagai berikut:



4. Pada Tab sebelah kiri (**Package Explorer**) klik kanan pada folder **src**, kemudian pilih **New -> Package**. Package pada Java merupakan *namespace* yang berisi *Class* dan *Interface*. Secara fisik, direpresentasikan dalam bentuk folder. (Akan dibahas lebih dalam pada perkuliahan selanjutnya) Dalam perkuliahan ini, gunakan package dengan format **week[xx].[namadepan].id.ac.umn**, e.g.: **week01.aditiyawan.id.ac.umn**.

5. Selanjutnya, pada package yang sudah dibuat, klik kanan, kemudian pilih **New** -> **Class**, dan Anda akan diberikan tampilan sebagai berikut:



6. Input **name** dengan nama **Main** dan centang "public static void main(String[] args)", kemudian tekan tombol **Finish**

Pada Java, biasanya memberikan nama Class dengan format penamaan *UpperCamelCase* atau *PascalCase*.

Centang pada "public static void main (String[] args)" akan men-generate sebuah fungsi **main** yang berfungsi untuk menjalankan aplikasi yang akan dibuat.



7. Kemudian pada Main.java yang telah di-*generate* oleh IDE, pada bagian fungsi **main**, ketikkan kode berikut:

```
package week01.aditiyawan.ac.id.umn;  
  
public class Main {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        System.out.println("Selamat datang di aplikasi sederhana pertama !");  
    }  
}
```

Package pada line 01 yang akan di-*generate* akan berada tergantung dari nama package yang Anda buat.

8. Jalankan program yang sudah Anda buat dengan membuka menu bar **Run -> Run** (Shortcut: CTRL + F11) untuk menjalankan aplikasi yang sudah Anda buat. Output aplikasi dapat dilihat pada tab **Console** di bagian tengah bawah IDE.
9. Selamat, Anda sudah mempelajari modul Week01.

## REFERENSI

# MODUL 2

## (KONSEP I/O, STRUKTUR KONTROL PEMILIHAN & PENGULANGAN)



### DESKRIPSI TEMA

Struktur Kontrol Pemilihan dan Struktur Kontrol Pengulangan.

### CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

Mahasiswa memahami input dan control pada bahasa pemrograman Java.

### PENUNJANG PRAKTIKUM

1. Eclipse.

### LANGKAH-LANGKAH PRAKTIKUM

#### Scanner

Scanner adalah sebuah class dalam package java.util yang digunakan untuk mendapatkan input dari tipe primitive seperti int, double, string, dan lain-lain. Scanner merupakan cara termudah untuk membaca input dalam program Java, meskipun tidak terlalu efisien jika menginginkan metode input untuk scenario di mana waktu merupakan kendala seperti pemrograman kompetitif.

Berikut adalah contoh cara membuat program input sederhana pada Java.

```
//Import class Scanner
import java.util.Scanner;

public class PreludeScanner {
    public static void main(String[] args) {
        int umur;
        String nama;

        //Inisialisasi objek Scanner
        Scanner scanner = new Scanner(System.in);

        System.out.print("Nama: ");
        nama = scanner.next();

        System.out.print("Umur: ");
        umur = scanner.nextInt();

        System.out.println("-----");
        System.out.println("Nama : " + nama);
        System.out.println("Umur : " + umur);
    }
}
```

## Switch

Switch memungkinkan variable diuji kesetaraannya terhadap daftar kondisi. Setiap kondisi disebut case, dan variable yang sedang diaktifkan diperiksa untuk setiap case. Pada dasarnya, variable yang diuji berupa tipe data byte, short, char, int, primitive class lainnya, String, dan Wrapper.

Berikut adalah contoh gambar program Java sederhana menggunakan Switch.

```
import java.util.Scanner;

public class PreludeSwitch {
    public static void main(String[] args) {
        String grade;
        Scanner scanner = new Scanner(System.in);

        System.out.print("Input your grade : ");
        grade = scanner.next();

        //variabel dalam kurung yang akan diuji
        switch(grade) {
            //Case jika variabel sama dengan case
            case "A":
                System.out.println("Perfect");
                break;
            case "B":
                System.out.println("Great");
                break;
            case "C":
                System.out.println("Cool");
                break;
            case "D":
                System.out.println("Bad");
                break;
            case "F":
                System.out.println("Fail");
                break;
            //Jika variabel tidak sama dengan semua case
            default:
                System.out.println("Invalid Grade");
        }
        System.out.println("Your grade is " + grade);
    }
}
```

## For Loop

For loop menyediakan cara ringkas penulisan struktur loop. Tidak seperti while loop, statement untuk mengkonsumsi inisialisasi, kondisi dan kenaikan / pengurangan dalam satu baris sehingga menyediakan struktur pengulangan yang lebih singkat dan mudah untuk debug.

Berikut adalah contoh program For loop pada Java.

```
public class PreludeForLoop {  
    public static void main(String[] args) {  
        for(int i=1; i<=10; i++) {  
            //i dimulai dengan nilai 1  
            //berjalan sampai i<=10  
            System.out.println("Angka ke " + i);  
        }  
    }  
}
```

## Enhanced For Loop

Enhanced for loop menyediakan cara yang lebih sederhana untuk melakukan iterasi melalui elemen koleksi atau array. Cara ini tidak fleksibel dan harus digunakan hanya ketika ada kebutuhan untuk iterasi melalui elemen secara berurutan tanpa mengetahui indeks elemen yang diproses.

Berikut adalah contoh program Enhanced for loop pada Java.

```
public class PreludeEnhancedForLoop {  
    public static void main(String[] args) {  
        String[] films = {"The Meg", "Slender Man", "Mile 22"};  
        System.out.println("Now Playing :");  
        //enhanced for loop  
        for(String film : films) {  
            //kondisi loop berjalan sampai i < films.length  
            System.out.println("- " + film);  
        }  
    }  
}
```

## Tutorial

1. Buatlah sebuah project dengan format Week[xx]-NIM, e.g. Week02-10001.
2. Buatlah package dengan format week[xx].[namadepan].id.ac.umn, e.g.: week02.aditiyawan.id.ac.umn.
3. Buat sebuah Class dengan nama Tutorial1 dan centang "public static void main(String[] args)"
4. Ketikkan kode berikut, pahamiilah, dan jalankan:

```
import java.util.Scanner;

public class Tutorial {
    public static void main(String[] args) {
        int input;
        Scanner scanner = new Scanner(System.in);

        String[] matkulIF = {
            "Matematika Diskrit",
            "Dasar-dasar Pemrograman",
            "Pemrograman Berorientasi Objek"
        };

        String[] matkulCE = {
            "Riset Operasi",
            "Jaringan Komputer",
            "Aljabar Linear"
        };

        String[] matkulIS = {
            "Sistem Database",
            "Administrasi Database"
        };

        System.out.println("Pilih kategori mata kuliah :");
        System.out.println("1. Informatics\n2. Computer Engineering\n3. Information System");
        System.out.print("Pilih : ");
        input = scanner.nextInt();

        switch(input) {
            case 1: System.out.println("Daftar mata kuliah IF : "); show(matkulIF); break;
            case 2: System.out.println("Daftar mata kuliah CE : "); show(matkulCE); break;
            case 3: System.out.println("Daftar mata kuliah IS : "); show(matkulIS); break;
            default: System.out.println("Pilihan tidak valid");
        }
    }

    public static void show(String[] matkul) {
        for(String eachMatkul : matkul) {
            System.out.println("- " + eachMatkul);
        }
    }
}
```

## Tugas

### Soal 1

Buatlah program sederhana untuk menentukan jumlah hari pada bulan yang dipilih. User akan diminta menginput nilai 1 sampai 12 untuk menentukan bulan. Hasil input user akan dicek menggunakan switch case dan menampilkan jumlah hari dan nama bulan. Berikut adalah contoh gambar program.

```
Masukan bulan(1-12) : 12
Bulan Desember memiliki 31 hari
```

```
Masukan bulan(1-12) : 2
Bulan Februari memiliki 28 hari
```

```
Masukan bulan(1-12) : 9
Bulan September memiliki 30 hari
```

### Soal 2

Buatlah program sederhana untuk menentukan angka prima atau bukan dari input user. Program akan meminta user menginput sebuah angka. Angka tersebut akan dicek untuk menentukan angka prima atau bukan. Berikut adalah contoh gambar Program.

```
Masukan angka : 53
Angka 53 adalah angka prima
```

```
Masukan angka : 8
Angka 8 adalah bukan angka prima
```

```
Masukan angka : 103
Angka 103 adalah angka prima
```

### Soal 3

Buatlah program sederhana untuk menghitung jumlah dari semua bilangan prima dengan batas yang ditentukan oleh user. User akan diminta menginput nilai terkecil dan nilai terbesar. Penjumlahan semua bilangan hanya diantara nilai terkecil dan nilai terbesar. Berikut adalah contoh gambar program.

```
Masukan nilai terkecil : 10
Masukan nilai terbesar : 20
Jumlah dari semua nilai prima dari 10 sampai 20 adalah 60
```

## REFERENSI

# MODUL 3

## (PRIMITIVE DATA TYPE, ARRAY, MULTIDIMENSIONAL ARRAY, STRING)

### DESKRIPSI TEMA

- Primitive Data Type
- Array & Multidimensional Array.

### CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

Mahasiswa memahami array, string, dan immutability pada bahasa pemrograman Java.

### PENUNJANG PRAKTIKUM

1. Eclipse

### LANGKAH-LANGKAH PRAKTIKUM

#### Array

Array menyimpan kumpulan berurutan yang berukuran tetap dari unsur-unsur dengan tipe data yang sama. Array digunakan untuk menyimpan kumpulan data, tetapi seringkali lebih berguna untuk memikirkan sebuah array sebagai kumpulan variabel dari jenis yang sama. Di Java semua array dialokasikan secara dinamis. Variabel dalam array diperintahkan dan masing-masing memiliki indeks mulai dari 0. Array Java juga dapat digunakan sebagai bidang statis, variabel lokal atau parameter metode.

```

public class PreludeArray {

    public static void main(String[] args) {
        //Deklarisasi
        //Maksimal kumpulan data yang dapat ditampung berjumlah 10
        String[] users = new String[10];

        //Mengisi nilai array
        users[0] = "Seng Hansun";
        users[1] = "Arya Wicaksana";
        //Jumlah maksimal sesuai dengan kumpulan data yang ada di Array
        int[] numbers = {1,2,3,4,5};

        printArray(numbers);

        numbers = setArray();

        printArray(numbers);
    }

    //Passing array ke method
    public static void printArray(int[] numbers) {
        //memproses array
        for(int i = 0; i < numbers.length; i++) {
            System.out.println(numbers[i]);
        }
    }

    //Returning array dari method
    public static int[] setArray() {
        int[] numbers = new int[5];
        for(int i = 0; i < 5; i++) {
            numbers[i] = i;
        }
        return numbers;
    }
}

```

## String

String adalah kumpulan urutan character. Dalam pemrograman bahasa Java, string diperlakukan sebagai objek. Objek string tidak dapat diubah yang berarti konstan dan tidak diubah setelah dibuat.

```

public class PreludeString {
    public static void main(String[] args) {
        //Membuat sebuah string
        String pbo1 = "Pemrograman Berorientasi Objek";
        String pbo2 = new String("Pemrograman Berorientasi Objek");

        //String Method
        pbo1.length(); // 30
        pbo1.charAt(12); // Berorientasi Objek
        pbo1.substring(24); // Objek
        pbo1.substring(12, 24); // Berorientasi

        String kode = "IF402 - ";
        String out = kode.concat(pbo1);
        // IF402 - Pemrograman Berorientasi Objek

    }
}

```



## Immutability

Objek immutable adalah objek yang keadaan internalnya tetap konstan setelah dibuat sepenuhnya. Ketergantungan maksimum pada objek yang tidak dapat diubah secara luas diterima sebagai strategi suara untuk membuat kode yang sederhana dan *reliable*. Objek immutable sangat berguna dalam aplikasi konkuren. Karena mereka tidak dapat mengubah keadaan, mereka tidak dapat rusak oleh gangguan *thread* atau diamati dalam keadaan yang tidak konsisten.

```
public class PreludeImmutability {  
  
    public static void main(String[] args) {  
        final String kode = "IF402";  
        kode = "IF400"; // Error  
    }  
}
```

## Tutorial

Tuliskan kode program pada gambar dibawah ini dan pahami setiap instruksi yang ada di program.

```
import java.util.Scanner;  
  
public class Tutorial1 {  
  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        String matkulIF[] = {  
            "Matematikka Distrik",  
            "Dasar-Dasar Pemrograman",  
            "Pemrograman Berorientasi Objek"};  
        String matkulCE[] = {  
            "Riset Operasi",  
            "Jaringan Komputer",  
            "Aljabar Linear"  
        };  
        String matkulIS[] = {  
            "Sistem Database",  
            "Administrasi Database"  
        };  
        System.out.println("Pilih kategori mata kuliah : ");  
        System.out.println("1.Informatics\n2.Computer Engineering\n3.Information System");  
        System.out.print("Pilih : ");  
        int input = scanner.nextInt();  
  
        switch(input) {  
            case 1 : System.out.println("Daftar mata kuliah IF : "); show(matkulIF); break;  
            case 2 : System.out.println("Daftar mata kuliah CE : "); show(matkulCE); break;  
            case 3 : System.out.println("Daftar mata kuliah IS : "); show(matkulIS); break;  
            default : System.out.println("Pilihan tidak valid");  
        }  
    }  
}
```

```
public static void show(String matkuls[]) {
    for(int i = 0; i < matkuls.length; i++) {
        System.out.println("- "+matkuls[i]);
    }
}
```

## Tugas

### Tugas 1

Jawablah soal-soal dibawah ini dalam sebuah file yang bernama Tugas1.txt

1. Jelaskan *multidimensional arrays* dan berikan contoh.
2. Sebutkan dan jelaskan 13 method-method string dalam Java
3. Sebutkan 2 keuntungan Immutability dalam Java.

### Tugas 2

Buatlah program sederhana yang berfungsi untuk menjalankan beberapa method string. Program akan menerima input berupa nama, setelah itu program akan meminta user memilih pilihan menu untuk menjalankan method string yang ada pada menu. Contoh hasil program akan seperti gambar dibawah ini.

```
Masukkan Nama Anda : James Gosling
-----
String anda : James Gosling
1. charAt          2. length
3. substring(n)    4. substring(m,n)
5. contains        6. concat
7. replace         8. split
9. lowerCase       10. upperCase
Pilih menu : |
```

```
Pilih menu : 1
-----charAt-----
Nama : James Gosling
Input : 10
Hasil : i
```

```
Pilih menu : 2
-----length-----
Nama : James Gosling
Input : 13
```

```
Pilih menu : 3
-----substring(n)-----
Nama : James Gosling
Input : 4
Hasil : s Gosling
```

```
Pilih menu : 4
-----substring(m,n)-----
Nama : James Gosling
Input mulai   : 5
Input akhir  : 13
Hasil : Gosling
```

```
Pilih menu : 5
-----contains-----
Nama : James Gosling
Input : ames
Hasil : true
```

```
Pilih menu : 6
-----concat-----
Nama : James Gosling
Input : - Java
Hasil : James Gosling - Java
```

```
Pilih menu : 7
-----replace-----
Nama : James Gosling
Input kata yang diganti : Gosling
Input kata pengganti   : Watt
Hasil : James Watt
```

```
Pilih menu : 8
-----split-----
Nama : James Gosling
Input : s
Hasil : Jame
Hasil : Go
Hasil : ling
```

```
Pilih menu : 9
-----lowerCase-----
Nama : James Gosling
Hasil : james gosling
```

```
Pilih menu : 10
-----upperCase-----
Nama : James Gosling
Hasil : JAMES GOSLING
```

## REFERENSI

[https://www.tutorialspoint.com/java/java\\_arrays.htm](https://www.tutorialspoint.com/java/java_arrays.htm)

<https://www.geeksforgeeks.org/string-class-in-java/>

<https://docs.oracle.com/javase/tutorial/essential/concurrency/immutable.html>

# MODUL 4

## (CLASS, OBJECT, METHOD, ENCAPSULATION, CONSTRUCTOR, DESTRUCTOR, STATIC MODIFIER)

### DESKRIPSI TEMA

- Class
- Object
- Method
- Encapsulation.

### CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

Mahasiswa memahami Java class dan object pada bahasa pemrograman Java.

### PENUNJANG PRAKTIKUM

1. Eclipse.

### LANGKAH-LANGKAH PRAKTIKUM

#### Class

Class adalah *blueprint* atau prototipe yang ditentukan pengguna dari mana objek dibuat. Class ini mewakili kumpulan atribut atau metode yang umum untuk semua objek dari satu jenis. Secara umum, deklarasi class dapat memasukkan komponen-komponen ini, secara berurutan :

- Modifiers : Class dapat bersifat publik atau memiliki akses default.
- Class name : Nama harus dimulai dengan huruf awal kapital.
- Superclass (Jika ada) : nama class parent (superclass), jika ada, didahului oleh kata kunci extends . Sebuah class (Subclass) hanya dapat extend satu parent.
- Interfaces (Jika ada) : daftar interface yang diterapkan oleh kelas dipisahkan dengan koma. Jika ada, Jika menerapkan interface didahului oleh kata kunci implements. Sebuah class dapat mengimplementasikan lebih dari satu interface.
- Body : Isi class dikelilingi oleh kurung, {}.

## **Object**

Object adalah unit dasar pemrograman berorientasi object dan mewakili entitas kehidupan nyata. Class menyediakan blueprint untuk object. Object dapat dibuat dari sebuah class. Object mempunyai status dan perilaku. Program Java yang khas menciptakan banyak objek, berinteraksi dengan menerapkan metode.

## **Constructor**

Constructor adalah method khusus yang akan dieksekusi pada saat pembuatan objek. Constructor digunakan untuk inisialisasi atau mempersiapkan data untuk objek. Sebuah class memiliki sebuah constructor. Jika constructor tidak tertulis secara eksplisit, java compiler membuat default constructor pada class tersebut.

## **Destructor**

Destructor adalah method khusus yang akan dieksekusi saat objek dihapus dari memori. Java sendiri tidak memiliki method destructor, karena Java menggunakan garbage collector untuk manajemen memorinya. Garbage Collector akan otomatis menghapus objek yang tidak terpakai.

```
//Deklarasi Class
public class Dog {
    //Deklarasi attribute
    String name;
    String breed;
    int age;
    String color;
    //Constructor
    public Dog(String name, String breed, int age, String color) {
        //set attribute name dengan passing variable
        //melewati parameter constructor
        this.name = name;
        this.breed = breed;
        this.age = age;
        this.color = color;
    }
    //Method untuk return attribute
    public String getName() {
        return name;
    }

    public String getBreed() {
        return breed;
    }

    public int getAge() {
        return age;
    }

    public String getColor() {
        return color;
    }

    public void bio() {
        System.out.println("Hi my name is "+this.getName()
            +".\nMy breed, age, and color are "+this.getBreed()
            +", "+this.getAge()+", "+this.getColor());
    }

    public static void main(String[] args) {
        //Membuat objek dog
        //Mengisi attribute objek dengan menggunakan constructor
        Dog tuffy = new Dog("Tuffy", "Papillon", 5, "white");
        tuffy.bio();
        //Membuat array of object dog
        Dog dogs[] = {
            new Dog("Bruno", "Golden", 3, "cream"),
            new Dog("Timmy", "Pomerania", 4, "orange"),
            new Dog("Rex", "Bulldog", 6, "black")
        };

        for(Dog dog : dogs) {
            dog.bio();
        }
    }
}
```

## Static Modifier

Static modifier berarti membagikan di semua contoh class tersebut dalam JVM. Static method atau variabel dikaitkan dengan class, bukan objek. Static mengatakan kepada compiler bahwa hanya ada satu salinan yang ada, terlepas dari beberapa kali kelas telah dipakai.

## Nested Class

Pada Java, memungkinkan untuk mendefinisikan class dalam class lain, class-class seperti itu dikenal sebagai class bertingkat. Nested class memungkinkan untuk secara logis mengelompokkan class-class yang hanya digunakan di satu tempat, sehingga meningkatkan penggunaan enkapsulasi, dan membuat kode yang lebih mudah dibaca dan dipelihara.

## Encapsulation

Enkapsulasi adalah salah satu dari empat konsep OOP. Tiga lainnya adalah inheritance, polymorphism, dan abstraksi. Enkapsulasi di Java adalah mekanisme membungkus data (variabel) dan kode yang bekerja pada data (metode) bersama sebagai satu kesatuan. Dalam enkapsulasi, variabel kelas akan disembunyikan dari class lain, dan hanya dapat diakses melalui metode class saat ini. Oleh karena itu disebut juga sebagai data hiding. Untuk mencapai enkapsulasi di Java, seperti poin dibawah berikut.

- Deklarasikan variabel class sebagai private.
- Menyediakan setter dan getter publik untuk memodifikasi dan melihat nilai-nilai variabel.

## Tutorial

Tuliskan kode program pada gambar dibawah ini dan pahami setiap instruksi yang ada di program. Buatlah sebuah class dengan nama MataKuliah.Java, kemudian ketiklah kode program pada gambar dibawah ini.

```
public class MataKuliah {  
  
    private String kode;  
    private String nama;  
    private int sks;  
  
    public MataKuliah() {}  
    public MataKuliah(String kode, String nama, int sks) {  
        this.kode = kode;  
        this.nama = nama;  
        this.sks = sks;  
    }  
}
```

```

public void setKode(String kode) {
    this.kode = kode;
}

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

public void setSks(int sks) {
    this.sks = sks;
}

public String getKode() {
    return kode;
}

public String getName() {
    return nama;
}

public int getSKS() {
    return sks;
}
}

```

Buatlah sebuah class dengan nama Main.Java, kemudian ketiklah kode program pada gambar dibawah ini.

```

import java.util.Scanner;

public class Main {
    //Membuat static array of object
    static MataKuliah[] matkuls = new MataKuliah[9];
    //Memberi nilai pada array of object
    public static void seedData() {
        matkuls[0] = new MataKuliah("IF402", "Pemrograman Berorientasi Objek", 3);
        matkuls[1] = new MataKuliah("IF100", "Dasar-dasar Pemrograman", 3);
        matkuls[2] = new MataKuliah("IF534", "Kecerdasan Buatan", 3);
        matkuls[3] = new MataKuliah("CE121", "Aljabar Linear", 3);
        matkuls[4] = new MataKuliah("CE441", "Jaringan Komputer", 3);
        matkuls[5] = new MataKuliah("CE232", "Sistem Digital", 3);
        matkuls[6] = new MataKuliah("UM162", "Pancasila", 2);
        matkuls[7] = new MataKuliah("UM152", "Agama", 2);
        matkuls[8] = new MataKuliah("UM142", "Bahasa Indonesia", 2);
    }

    public static void mainMenu() {
        System.out.println("-----Daftar Mata Kuliah-----");
        System.out.println("1. Lihat Semua Mata Kuliah");
        System.out.println("2. Lihat Mata Kuliah kode IF");
        System.out.println("3. Lihat Mata Kuliah kode CE");
        System.out.println("4. Lihat Mata Kuliah kode UM");
    }
}

```



```

public static void showData() {
    System.out.println("Daftar Mata Kuliah");
    for(MataKuliah matkul : matkuls) {
        System.out.println("-----");
        System.out.println("Kode      : "+matkul.getKode());
        System.out.println("Nama      : "+matkul.getNama());
        System.out.println("Jumlah SKS : "+matkul.getSKS());
    }
}

public static void filterData(String kode) {
    for(MataKuliah matkul : matkuls) {
        if(matkul.getKode().contains(kode)) {
            System.out.println("-----");
            System.out.println("Kode      : "+matkul.getKode());
            System.out.println("Nama      : "+matkul.getNama());
            System.out.println("Jumlah SKS : "+matkul.getSKS());
        }
    }
}

public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    seedData();
    int menu;
    for(;;) {
        mainMenu();
        System.out.print("Pilihan : ");
        menu = in.nextInt();
        switch(menu) {
            case 1 :
                showData();
                break;
            case 2 :
                filterData("IF");
                break;
            case 3 :
                filterData("CE");
                break;
            case 4 :
                filterData("UM");
                break;
            default :
                System.out.println("Input tidak valid");
        }
    }
}
}

```

## Tugas

### Tugas 1

Buatlah program sederhana yang berfungsi untuk memesan barang yang tersedia dalam sebuah toko peralatan tulis kantor. Program ini dapat melihat barang yang tersedia dalam toko, memesan barang dengan memasukkan jumlah dan biaya barang yang dipesan, dan melihat pemesanan barang. Program ini memiliki 2 menu yaitu Pesan Barang dan Lihat Pesanan. Menu Pesan Barang menampilkan daftar barang yang tersedia, kemudian pengguna akan diminta memasukkan id barang. Jika pengguna mengetik o dan pengguna memilih id barang yang tidak ada di daftar maka akan kembali ke menu utama. Sesudah pengguna memilih id barang maka akan diminta jumlah barang, jumlah barang harus diatas nol atau kurang dari stock. Pengguna akan diminta memasukkan harga yang sesuai dengan biaya pesanan.

Berikut adalah contoh gambar program.

#### **Class Barang**

Attribute:

Id, stock, harga (private int)

nama (private String)

Method:

getId(), getStock(), getHarga() (public int),

getNama() (public String)

minusStock(qty (int)) (public void)

#### **Class Order**

Attribute:

id, jumlah (private int)

barang (private Barang)

total (public static int)

Method:

Order() (public)

Order(id (int), barang (Barang), jumlah (int))

getId(), getJumlah() (public int)

getBarang() (public Barang)

## Menu Awal Program

```
|-----Menu Toko Multiguna-----  
1. Pesan Barang  
2. Lihat Pesanan  
Pilihan :
```

## Menu Pesan Barang

```
Daftar Barang Toko Multiguna
```

```
ID      : 1  
Nama    : Pulpen Easy Gel 0.5mm  
Stock   : 120  
Harga   : 2000
```

```
-----  
ID      : 2  
Nama    : Penggaris 30cm  
Stock   : 30  
Harga   : 5000
```

```
-----  
ID      : 3  
Nama    : Tipe-x Roller  
Stock   : 30  
Harga   : 7000
```

```
-----  
ID      : 4  
Nama    : Pensil Mekanik  
Stock   : 50  
Harga   : 5000
```

```
-----  
ID      : 5  
Nama    : Buku Tulis  
Stock   : 100  
Harga   : 6000
```

```
-----  
Ketik 0 untuk batal  
Pesan Barang :
```

## Pesanan Barang

Daftar Barang Toko Multiguna

ID : 1  
 Nama : Pulpen Easy Gel 0.5mm  
 Stock : 120  
 Harga : 2000

-----  
 ID : 2  
 Nama : Penggaris 30cm  
 Stock : 20  
 Harga : 5000

-----  
 ID : 3  
 Nama : Tipe-x Roller  
 Stock : 30  
 Harga : 7000

-----  
 ID : 4  
 Nama : Pensil Mekanik  
 Stock : 50  
 Harga : 5000

-----  
 ID : 5  
 Nama : Buku Tulis  
 Stock : 100  
 Harga : 6000

-----  
 Ketik 0 untuk batal  
 Pesan Barang (ID) : 2  
 Masukkan Jumlah : 10  
 10 @ Penggaris 30cm dengan total harga 50000  
 Masukkan jumlah uang : 50000  
 berhasil dipesan

ID Barang tidak sesuai dengan pilihan

Ketik 0 untuk batal  
 Pesan Barang (ID) : 7  
 -----Menu Toko Multiguna-----  
 1. Pesan Barang  
 2. Lihat Pesanan  
 Pilihan :

Ketik 0 untuk batal  
 Pesan Barang (ID) : 0  
 |-----Menu Toko Multiguna-----  
 1. Pesan Barang  
 2. Lihat Pesanan  
 Pilihan :

Jumlah Barang tidak sesuai dengan stock

```
Ketik 0 untuk batal  
Pesan Barang (ID) : 2  
Masukkan Jumlah : 30  
Masukkan Jumlah : 0  
Masukkan Jumlah :
```

---

Jumlah uang tidak mencukupi

```
Ketik 0 untuk batal  
Pesan Barang (ID) : 2  
Masukkan Jumlah : 10  
10 @ Penggaris 30cm dengan total harga 50000  
Masukkan jumlah uang : 10  
Jumlah uang tidak mencukupi
```

### Menu Lihat Pesanan

```
-----Menu Toko Multiguna-----  
1. Pesan Barang  
2. Lihat Pesanan  
Pilihan : 2  
Daftar Pesanan Toko Multiguna  
ID      : 1  
Nama    : Penggaris 30cm  
Jumlah  : 10  
Total   : 50000  
-----
```

## REFERENSI

<https://www.geeksforgeeks.org/classes-objects-java/>  
[https://www.tutorialspoint.com/java/java\\_object\\_classes.htm](https://www.tutorialspoint.com/java/java_object_classes.htm)  
[https://www.tutorialspoint.com/java/java\\_encapsulation.htm](https://www.tutorialspoint.com/java/java_encapsulation.htm)  
<https://www.petanikode.com/java-oop-constructor/>  
[https://www.tutorialspoint.com/java/java\\_encapsulation.htm](https://www.tutorialspoint.com/java/java_encapsulation.htm)  
<https://www.geeksforgeeks.org/nested-classes-java/>

# MODUL 5

## (INHERITANCE, POLYMORPHISM, IMMUTABLE CLASS, ACCESS MODIFIERS)

### DESKRIPSI TEMA

Inheritance & Polymorphism.

### CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

Mahasiswa memahami konsep inheritance dan polymorphism.

### PENUNJANG PRAKTIKUM

1. Eclipse.

### LANGKAH-LANGKAH PRAKTIKUM

#### Inheritance

Pengaturan class dalam hierarki untuk menghindari duplikasi dan mengurangi redundansi. Class-class dalam hirarki yang lebih rendah mewarisi semua attribute dan method dari hirarki yang lebih tinggi. Kelas dalam hirarki bawah disebut subclass (atau turunan, turunan, kelas diperpanjang). Kelas dalam hirarki atas disebut superclass (atau basis, kelas induk). Dengan menarik semua variabel dan metode umum ke dalam superclasses, dan meninggalkan variabel dan metode khusus dalam subclass, redundansi dapat sangat dikurangi atau dihilangkan karena variabel dan metode umum ini tidak perlu diulang dalam semua subclass.

```
public class Animal {  
    public Animal() {}  
    public void eat() {  
        System.out.println("eating...");  
    }  
}  
  
public class Dog extends Animal{  
    public Dog() {}  
    public void bark() {  
        System.out.println("barking...");  
    }  
}
```

```
public class Main {  
  
    public static void main(String[] args) {  
        Dog d = new Dog();  
        d.bark(); //barking...  
        d.eat(); //eating...  
    }  
  
}
```

## Polymorphism

Polimorfisme adalah kemampuan suatu objek untuk mengambil banyak bentuk. Penggunaan polimorfisme yang paling umum di OOP terjadi ketika referensi kelas induk digunakan untuk merujuk ke objek kelas turunan.

```
public class Circle {  
  
    private double radius;  
  
    public Circle(double radius) {  
        this.radius = radius;  
    }  
  
    public double getRadius() {  
        return radius;  
    }  
  
    public double getArea() {  
        return radius * radius * Math.PI;  
    }  
  
    public String toString() {  
        return "Circle[radius="+ radius+"]";  
    }  
  
}
```

```

public class Cylinder extends Circle{

    private double height;

    public Cylinder(double height, double radius) {
        super(radius);
        this.height = height;
    }

    public double getHeight() {
        return this.height;
    }

    public double getVolumne() {
        return super.getArea() * height;
    }
    // Menimpa method turunan
    @Override
    public double getArea() {
        return 2.0 * Math.PI * getRadius() * height;
    }
    // Menimpa method turunan
    @Override
    public String toString() {
        return "Cylinder[height=" + height + "," + super.toString() + "];"
    }
}

```

## Immutable Object

Suatu objek disebut immutable jika variabel-nya tidak dapat dimodifikasi oleh siapa pun dengan cara apa pun setelah constructornya. Immutable objek tidak memberikan perilaku apa pun untuk mengubah keadaanya.

```

//Deklarasi class sebagai final, class lain tidak dapat extend
public class ImmutableEmployee {
    //buat semua attribute private dan final
    private final int id;
    private final String name;
    //hanya constructor yang dapat mengisi nilai
    public ImmutableEmployee(int id, String name) {
        this.id = id;
        this.name = name;
    }
    //jangan buat method set atau method yang mengubah nilai
    public int getId() {return id;}
    public String getName() {return name;}

    @Override
    public String toString() {
        return "ImmutableEmployee{" +
            "id="+id+
            ", name='"+name+"'";
    }
}

```



## Access Modifier

Access Modifier membantu untuk membatasi ruang lingkup class, constructor, variabel, method atau anggota data.

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

## Tutorial

Tuliskan kode program pada gambar dibawah ini dan pahami setiap instruksi yang ada di program.

Buatlah class dengan nama Shape.java dan ketiklah isi class tersebut seperti gambar dibawah ini.

```
public class Shape {
    private String color;

    public Shape() {}
    public Shape(String color) {
        this.color = color;
    }

    public void setColor(String color) { this.color = color; }

    public String getColor() { return color; }

    public double getArea() { return 0; }
    public double getPerimeter() { return 0; }
}
```

Buatlah class dengan nama Circle.java dan ketiklah isi class tersebut seperti gambar dibawah ini.

```
public class Circle extends Shape{
    private double radius;

    public Circle() {}
    public Circle(double radius, String color) {
        super(color);
        this.radius = radius;
    }

    public double getRadius() { return radius; }
    public double getArea() { return Math.PI * radius * radius; }
    public double getPerimeter() {return Math.PI * 2 * radius; }

}
```

Buatlah class dengan nama Main.java dan ketiklah isi class tersebut seperti gambar dibawah ini.

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);

        System.out.print("Masukkan jari-jari lingkaran : ");
        double radius = s.nextDouble();
        System.out.print("Masukkan warna : ");
        String color = s.next();

        Circle circle = new Circle(radius,color);
        System.out.println("-----Lingkaran-----");
        System.out.println("Warna : "+circle.getColor());
        System.out.println("Jari-jari : "+circle.getRadius());
        System.out.println("Keliling Lingkaran : "+circle.getPerimeter());
        System.out.println("Luas Lingkaran : "+circle.getArea());

    }

}
```

## Tugas

### Tugas 1

Buatlah program sederhana dengan melanjutkan tutorial. Program ini berfungsi untuk menghitung luas dan keliling bangun ruang. Buatlah class sesuai dengan ruang bangun yang ada di program. Contoh program seperti gambar dibawah ini.

```
-----Program Menghitung Bangun Ruang-----
```

1. Lingkaran
2. Persegi
3. Persegi Panjang
4. Segitiga
5. Keluar

Pilihan : 1

```
-----Lingkaran-----
```

Masukkan jari-jari : 7

Warna : Biru

Jari-jari : 7.0

Keliling Lingkaran : 43.982297150257104

Luas Lingkaran : 153.93804002589985

```
-----Program Menghitung Bangun Ruang-----
```

1. Lingkaran
2. Persegi
3. Persegi Panjang
4. Segitiga
5. Keluar

Pilihan : 2

```
-----Persegi-----
```

Masukkan panjang sisi : 6

Warna : Merah

Panjang Sisi : 6

Keliling Persegi : 24.0

Luas Persegi : 36.0

```
-----Program Menghitung Bangun Ruang-----
```

1. Lingkaran
2. Persegi
3. Persegi Panjang
4. Segitiga
5. Keluar

Pilihan : 3

```
-----Persegi Panjang-----
```

Masukkan panjang : 4

Masukkan lebar : 6

Warna : Ungu

Panjang & Lebar : 4 & 6

Keliling Persegi Panjang : 20.0

Luas Persegi Panjang : 24.0

```
-----Program Menghitung Bangun Ruang-----
1. Lingkaran
2. Persegi
3. Persegi Panjang
4. Segitiga
5. Keluar
Pilihan : 4
-----Segitiga Siku-Siku-----
Masukkan alas : 6
Masukkan tinggi : 8
Warna          : Hitam
Alas & Tinggi  : 6 & 8
Keliling Segitiga : 24.0
Luas Segitiga   : 24.0

-----Program Menghitung Bangun Ruang-----
1. Lingkaran
2. Persegi
3. Persegi Panjang
4. Segitiga
5. Keluar
Pilihan : 5
Keluar program...

-----Program Menghitung Bangun Ruang-----
1. Lingkaran
2. Persegi
3. Persegi Panjang
4. Segitiga
5. Keluar
Pilihan : 9
Input tidak valid
```

## REFERENSI

[http://www.ntu.edu.sg/home/ehchua/programming/java/j3b\\_oopinheritancepolymorphism.html](http://www.ntu.edu.sg/home/ehchua/programming/java/j3b_oopinheritancepolymorphism.html)  
<https://www.javatpoint.com/inheritance-in-java>  
<https://programmingmitra.blogspot.com/2018/02/how-to-create-immutable-class-in-java.html>  
<https://www.javatpoint.com/access-modifiers>

# MODUL 6

## (CLASS HIERARCHY, CLASS DIAGRAM, ACTIVITY DIAGRAM, USE CASE DIAGRAM)

### DESKRIPSI TEMA

- Class Hierarchy
- Class Diagram
- Activity Diagram
- Use Case Diagram.

### CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

Mahasiswa mampu membuat program dengan menggunakan rancangan UML.

### PENUNJANG PRAKTIKUM

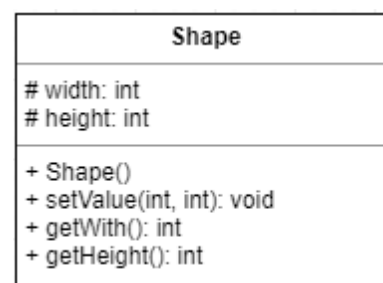
1. Eclipse.

### LANGKAH-LANGKAH PRAKTIKUM

#### Class Diagram

Class diagram merupakan diagram UML yang digunakan untuk merepresentasikan class berdasarkan atribut, method dan struktur relationship antar class tersebut. Class diagram dibuat agar programmer dapat menentukan member access control untuk atribut dari sebuah class serta memberikan daftar method yang harus dibuat untuk setiap class. Dengan class diagram developer juga akan lebih mudah dalam proses menentukan relationship antar class yang ada. Berikut code sederhana beserta representasinya dalam bentuk class diagram.

```
public class Shape {
    protected int width, height;
    public Shape() {
        System.out.println("Shape Created");
    }
    public void setValue(int width, int height) {
        this.width = width;
        this.height = height;
    }
    public int getWidth() { return width; }
    public int getHeight() { return height; }
}
```

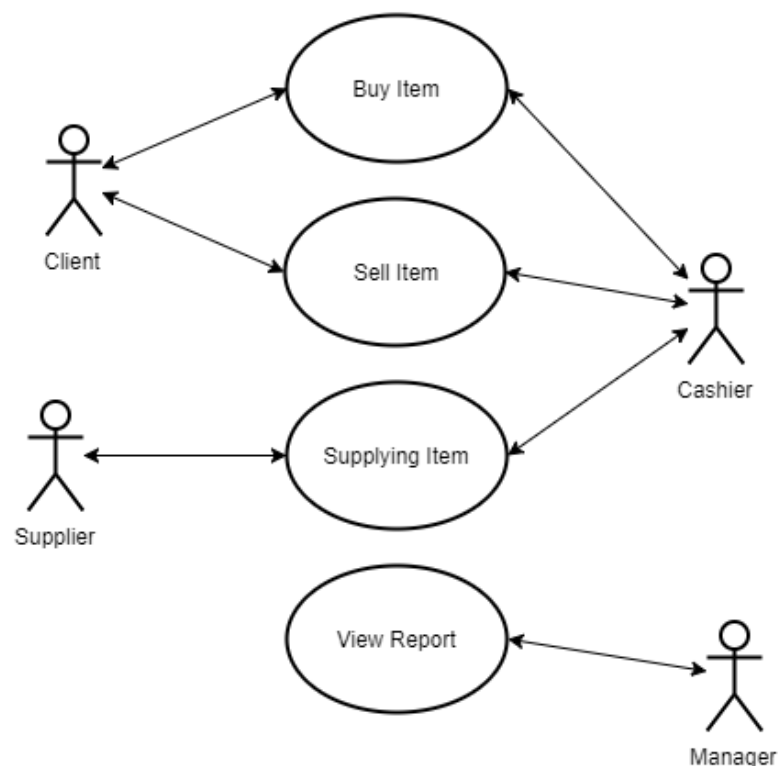


Simbol	Arti
--------	------

#	Protected
-	Private
+	Public
~	Package
/	Derived

## Use Case Diagram

Use Case diagram digunakan untuk mendeskripsikan entitas pada sebuah sistem dan kapabilitas entitas tersebut pada sistem. Use case diagram hanya berfungsi untuk memberi gambaran mengenai relasi antara entitas dan sistem. Sebagai contoh anda dapat melihat use case diagram dibawah ini. Use case ini merepresentasikan proses transaksi pada sebuah toko.



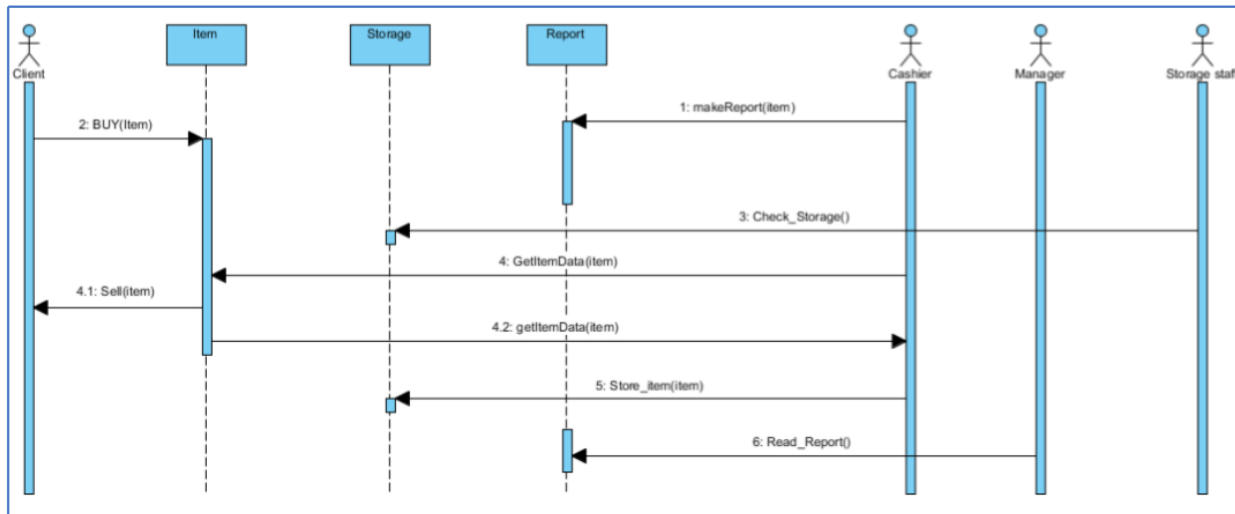
Pada use case diatas dapat dilihat bahwa entity buyer memiliki kemampuan untuk membeli dan menjual item. Begitu juga dengan cashier yang dapat membeli dari client atau menjual kepada client. Kita dapat merepresentasikan buy item dan sell item sebagai sebuah method.

Sedangkan untuk client dan cashier dapat kita definisikan sebagai objek. Supplier dapat melakukan supply item dan kasir juga bisa menambahkan item dari transaksinya dengan client.

Manager hanya memiliki kemampuan untuk melihat report. Berdasarkan hal ini buatlah sebuah program yang mengacu terhadap use case diatas.

## Sequence Diagram

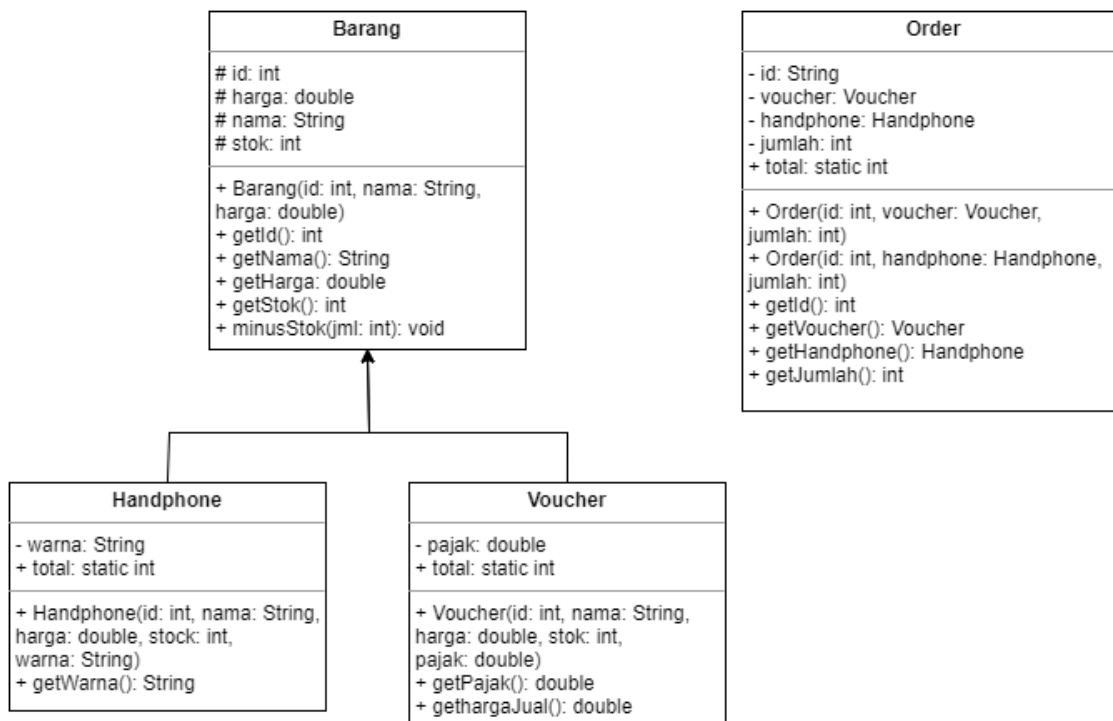
Sequence diagram digunakan untuk mendefinisikan interaksi antar objek. Sequence diagram digunakan untuk menunjukkan alur data yang terjadi di pada entitas yang berada pada sistem.



Berdasarkan use case dan sequence diagram diatas buatlah sebuah aplikasi mini market sesuai dengan program contoh yang diberikan.

## Tugas

### Tugas 1



Buatlah program sederhana berdasarkan gambar class diagram diatas.

Program ini berfungsi untuk memesan handphone dan voucher. Program ini dapat melihat handphone dan voucher yang tersedia dalam toko, memesan handphone dengan memasukkan jumlah dan biaya barang yang dipesan, memesan voucher dengan memasukkan biaya yang sudah ditambah dengan pajak, melihat pemesanan barang dan memasukkan handphone atau voucher baru ke dalam toko.

Program ini memiliki 3 menu yaitu Pesan Barang, Lihat Pesanan dan Barang Baru.

Menu Pesan Barang menampilkan daftar barang yang tersedia yaitu handphone dan voucher. Jika barang tidak ada maka akan menampilkan "barang tidak tersedia". Setelah pemilihan barang pengguna akan diminta memasukkan id barang. Jika pengguna mengetik "o" dan pengguna memilih id barang yang tidak ada di daftar maka akan kembali ke menu utama. Sesudah pengguna memilih id barang, jika handphone maka akan diminta jumlah barang, jumlah barang harus diatas nol atau kurang dari stok. Pengguna akan diminta memasukkan harga yang sesuai dengan biaya pesanan, jika voucher akan dikenakan tambahan pajak.

Menu Lihat Pesanan berisi daftar pesanan.

Menu Barang Baru berfungsi untuk memasukkan handphone atau voucher baru dengan memasukkan data sesuai dengan atribut class. Berikut adalah contoh gambar program.

### Menu utama

```
|-----Menu Toko Voucher & HP-----  
1. Pesan Barang  
2. Lihat Pesanan  
3. Barang Baru  
Pilihan :
```

### Menu Barang Baru

```
-----Menu Toko Voucher & HP-----  
1. Pesan Barang  
2. Lihat Pesanan  
3. Barang Baru  
Pilihan : 3  
Voucher / Handphone (V/H): v  
Nama : Google Play  
Harga : 20000  
Stok : 20  
PPN : 0.1  
|Voucher telah berhasil diinput
```



```
-----Menu Toko Voucher & HP-----
1. Pesan Barang
2. Lihat Pesanan
3. Barang Baru
Pilihan : 3
Voucher / Handphone (V/H): h
Nama : Oppo F9
Harga : 3999000
Stok : 10
Warna : Hitam
Handphone telah berhasil diinput
```

### Menu Pesan Barang

```
-----Menu Toko Voucher & HP-----
1. Pesan Barang
2. Lihat Pesanan
3. Barang Baru
Pilihan : 1
Daftar Barang Toko Voucher & HP
1. Handphone
2. Voucher
Pilihan : 1
ID : 1
Nama : Samsung S9+ Hitam
Stock : 10
Harga : 14499000
-----
ID : 2
Nama : iPhone X Hitam
Stock : 10
Harga : 17999000
-----
Ketik 0 untuk batal
Pesan Barang (ID) : |
```

### Pemesanan barang

```
Ketik 0 untuk batal
Pesan Barang (ID) : 1
Masukkan Jumlah : 1
1 @ Samsung S9+ dengan total harga 14499000
Masukkan jumlah uang : 15000000
Berhasil dipesan
```

-----Menu Toko Voucher & HP-----

```
1. Pesan Barang
2. Lihat Pesanan
3. Barang Baru
Pilihan : 1
Daftar Barang Toko Voucher & HP
1. Handphone
2. Voucher
Pilihan : 2
ID      : 1
Nama    : Google Play
Nominal : 20000
Stok    : 100
Harga   : 22000
```

```
-----
Ketik 0 untuk batal
Pesan Barang (ID) : 1
Masukkan Jumlah : 10
10 @ Google Play 20000 dengan harga 220000
Masukkan jumlah uang : 220000
Berhasil dipesan
```

#### Stok tidak mencukupi jumlah pemesanan

```
ID      : 1
Nama    : Samsung S9+ Hitam
Stok    : 9
Harga   : 14499000
```

```
-----
Ketik 0 untuk batal
Pesan Barang (ID) : 1
Masukkan Jumlah : 10
Stok tidak mencukupi
```

#### Pemilihan Barang yang tidak sesuai

```
Pilihan : 1
ID      : 1
Nama    : Samsung S9+ Hitam
Stok    : 9
Harga   : 14499000
```

```
-----
Ketik 0 untuk batal
Pesan Barang (ID) : 9
```

### Jumlah uang tidak mencukupi

ID : 1  
Nama : Google Play  
Nominal : 20000  
Stok : 90  
Harga : 22000

-----

Ketik 0 untuk batal  
Pesan Barang (ID) : 1  
Masukkan Jumlah : 1  
1 @ Google Play 20000 dengan harga 22000  
Masukkan jumlah uang : 10000  
Jumlah uang tidak mencukupi

### Menu Lihat Pesanan

-----Menu Toko Voucher & HP-----  
1. Pesan Barang  
2. Lihat Pesanan  
3. Barang Baru  
Pilihan : 2  
Daftar Pesanan Toko Multiguna  
ID : OH0  
Nama : Samsung S9+ Hitam  
Jumlah : 1  
Total : 14499000

-----

ID : OV1  
Nama : Google Play 20000  
Jumlah : 10  
Total : 220000

-----

### Menu Barang Baru

-----Menu Toko Voucher & HP-----  
1. Pesan Barang  
2. Lihat Pesanan  
3. Barang Baru  
Pilihan : 3  
Voucher / Handphone (V/H): v  
Nama : Google Play  
Harga : 20000  
Stok : 100  
PPN : 0.1  
Voucher telah berhasil diinput

-----Menu Toko Voucher & HP-----  
1. Pesan Barang  
2. Lihat Pesanan  
3. Barang Baru  
Pilihan : 3  
Voucher / Handphone (V/H): h  
Nama : Samsung S9+  
Harga : 14499000  
Stok : 10  
Warna : Hitam  
Handphone telah berhasil diinput

## REFERENSI

<https://www.lucidchart.com/pages/uml-class-diagram>

# MODUL 7

## (ABSTRACTION, INTERFACE, ABSTRACT CLASS)



### DESKRIPSI TEMA

- Abstraction
- Interface
- Abstract Class.

### CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

Mahasiswa mempelajari konsep *interfaces* dan *multiple inheritance*.

### PENUNJANG PRAKTIKUM

1. Eclipse.

### LANGKAH-LANGKAH PRAKTIKUM

#### Interface

Interface adalah tipe referensi di Java. Ini mirip dengan class. Ini adalah kumpulan abstract method. Sebuah class yang mengimplementasikan interface mewarisi abstract method interface. Bersamaan abstract method, interface juga dapat berisi default method, static method, nested types. Membuat interface mirip dengan membuat class. Tetapi suatu class menggambarkan atribut dan perilaku suatu objek. Dan interface berisi perilaku yang diimplementasikan oleh class. Kecuali class yang mengimplementasikan antarmuka abstrak, semua method interface perlu didefinisikan di class.

Perbedaan interface dengan class :

- Tidak bisa membuat objek / instansi
- Interface tidak punya sebuah constructor
- Semua method dalam interface bersifat abstrak
- Interface tidak bisa berisi instance field
- Interface tidak bisa di-extend class.
- Sebuah interface bisa di-extend banyak interface

Contoh program mengimplementasikan interface

```
public interface Animal {
    public void eat();
    public void travel();
}

public class MammalInt implements Animal{

    public MammalInt() {}

    @Override
    public void eat() {
        System.out.println("Mammal eats");
    }

    @Override
    public void travel() {
        System.out.println("Mammal travel");
    }

    public static void main(String[] args) {
        MammalInt m = new MammalInt();
        m.eat();
        m.travel();
    }
}
```

Berikut adalah hasil output dari contoh program interface.

```
Mammal eats
Mammal travel
```

### Abstract Class

Sebuah class yang dideklarasikan sebagai abstrak dikenal sebagai abstract class. Abstract class bisa memiliki abstract method dan non-abstract method. Abstract class perlu di-extend dan diimplementasi method-nya. Abstract class tidak bisa diinstansi.

Contoh program sederhana mengimplementasikan abstract class

```
public abstract class Bike {

    public Bike() {
        System.out.println("Bike is created");
    }

    abstract void run();

    void changeGear(){
        System.out.println("gear changed");
    }

}
```

```

public class Honda extends Bike{

    public Honda() {}

    void run(){
        System.out.println("running safely..");
    }

}

public class Main {

    public static void main(String[] args) {
        Bike obj = new Honda();
        obj.run();
        obj.changeGear();
    }

}

```

Berikut adalah hasil output dari contoh program abstract class.

```

Bike is created
running safely..
gear changed

```

### Abstract class vs Interface

Abstract class	Interface
Dapat memiliki abstract dan non-abstract method	Hanya dapat memiliki method abstract
Abstract class dapat berisi variabel-variabel tidak final	Variabel yang dideklarasikan di interface secara default final
Abstract class dapat memiliki variabel final, non-final, static dan non-static	Interface hanya memiliki variabel static dan final
Abstract class dapat mengimplementasikan interface	Interface tidak bisa mengimplementasikan abstract class
Abstract class di-extend dengan kata kunci "extends"	Interface diimplementasikan dengan kata kunci "implements"
Abstract class dapat meng-extend class java lain dan mengimplementasikan beberapa interface	Interface dapat mengimplementasikan interface lainnya
Abstract class dapat memiliki anggota class seperti private, protected, public, dll	Anggota interface bersifat public secara default.

## Observer Pattern

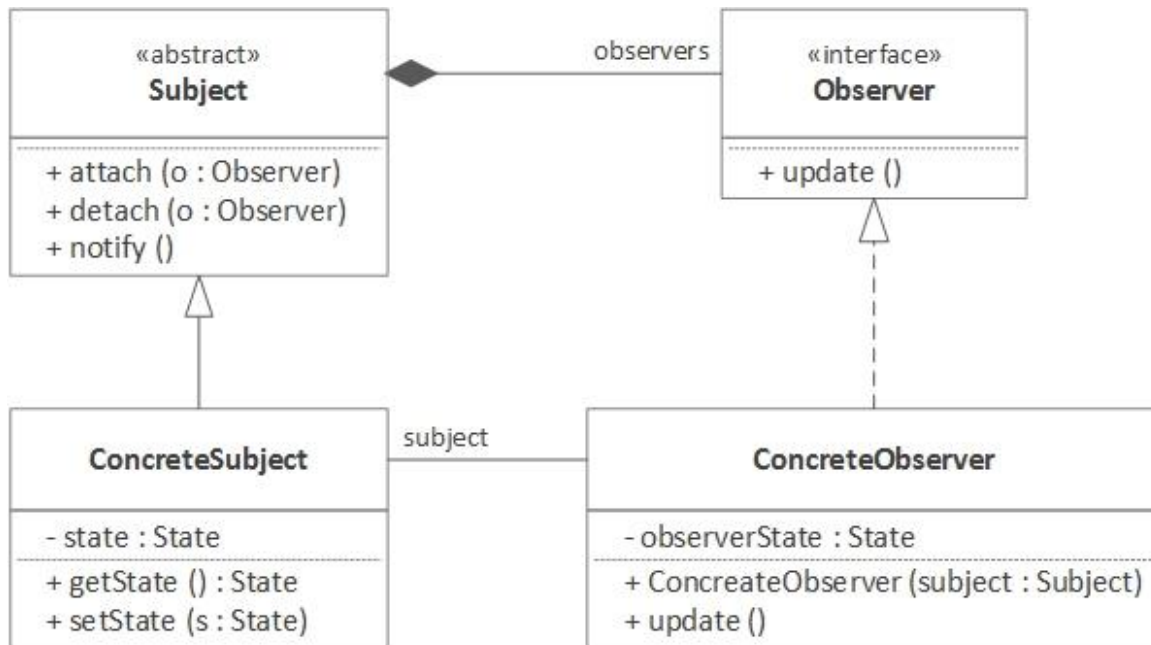
Observer adalah penentu one-to-many ketergantungan antar objek sehingga ketika satu objek berubah status, semua tanggungannya diberitahu dan diperbarui secara otomatis.

Dalam banyak aplikasi software, status satu objek bergantung pada yang lain. Sebagai contoh aplikasi berfokus pada pemrosesan data numerik, data ini dapat ditampilkan pada Graphical User Interface (GUI) menggunakan spreadsheet atau grafik, atau keduanya: Ketika data numerik yang mendasari diperbarui, komponen GUI yang sesuai harus diperbarui demikian. Inti masalah ini adalah bagaimana memperbolehkan komponen GUI untuk diperbarui ketika perubahan dibuat ke data numerik yang mendasari tanpa menggabungkan komponen GUI langsung ke data numerik yang mendasarinya.

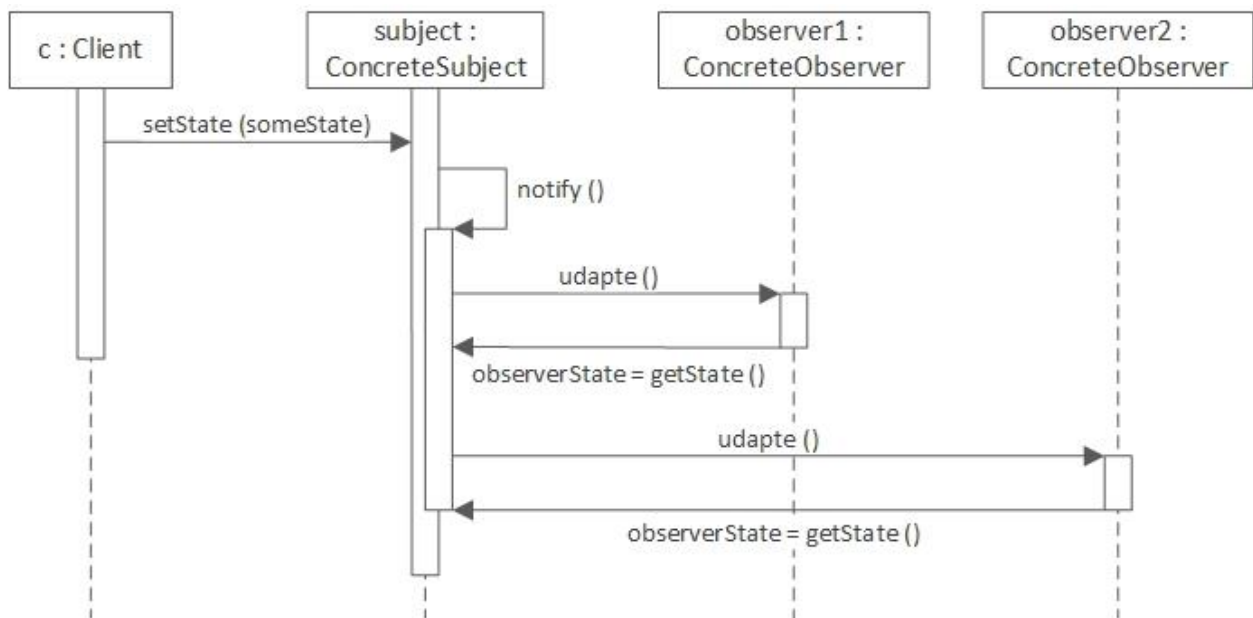
Solusi yang naif dan tidak dapat ditiru adalah menyediakan objek yang mengelola data numerik yang mendasari dengan referensi ke spreadsheet dan grafik komponen GUI, memungkinkan objek untuk memberitahukan komponen GUI ketika data numerik berubah. Solusi sederhana ini dengan cepat membuktikan ketidakmampuannya untuk mengatasi aplikasi yang lebih kompleks ketika lebih banyak komponen GUI ditambahkan. Sebagai contoh, jika ada 20 komponen GUI yang bergantung pada data numerik, objek yang mengelola data numerik ini akan diperlukan untuk mempertahankan referensi ke 20 komponen GUI ini. Ketika jumlah objek yang bergantung pada data minat tumbuh, kopling antara pengelola data dan objek yang bergantung pada data ini menjadi lebih tidak beraturan.

Solusi yang lebih baik akan memungkinkan objek untuk mendaftar (register) untuk pembaruan ke data yang menarik, memungkinkan manajer data ini untuk memberitahukan (notify) objek-objek yang terdaftar ketika data minat berubah. Secara tidak resmi, objek yang tertarik pada data memberi tahu manajer, "Beritahu saya ketika terjadi perubahan pada data." Selanjutnya, setelah objek telah terdaftar untuk pembaruan data yang menarik, itu bisa tidak terdaftar (unregistered), memastikan bahwa manajer tidak lagi memberitahukan objek perubahan yang terjadi pada data yang menarik. Dalam konteks definisi asli GoF, objek yang terdaftar untuk pembaruan disebut pengamat (observer), manajer data yang diinginkan disebut subjek (subject), data yang diinginkan disebut keadaan (state) subjek, proses mendaftarkan observer disebut melampirkan (attaching), dan proses unregistering observer disebut detaching. Seperti yang dinyatakan sebelumnya, pola ini juga disebut pola Publish-Subscribe, karena klien berlangganan observer untuk subjek dan ketika pembaruan dibuat untuk keadaan subjek, subjek mempublikasikan pembaruan ini ke pelanggan (pola desain ini diperluas ke arsitektur umum, yang disebut Arsitektur Publish-Subscribe). Menarik konsep-konsep ini bersama menjadi satu gambar, kita memperoleh diagram kelas berikut:





Untuk menerima pembaruan pada perubahan state, sebuah *ConcreteObserver* dibuat, dan referensi ke *ConcreteSubject* yang sedang diamati diteruskan ke konstruktornya. Ini menyediakan *ConcreteObserver* dengan referensi ke *ConcreteSubject*, dari mana state akan diperoleh ketika pengamat diberitahu tentang perubahan ke state. Sederhananya, *ConcreteObserver* akan diberitahu tentang pembaruan pada subjek, pada saat itu, *ConcreteObserver* akan menggunakan referensi ke *ConcreteSubject* yang diperoleh melalui konstruktor untuk mendapatkan state dari *ConcreteSubject*, pada akhirnya menyimpan objek state yang di-retrieved di bawah atribut *observerState* dari the *ConcreteObserver*. Proses ini divisualisasikan dalam diagram urutan di bawah ini:



## Tutorial

Buatlah sebuah class bernama 'Employee', potongann kode dari class dapat dilihat pada gambar di bawah ini.

```
public class Employee {  
    private String name;  
    private String position;  
  
    public Employee(String name, String position) {  
        this.name = name;  
        this.position = position;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public String getPosition() {  
        return position;  
    }  
  
    public void setPosition(String position) {  
        this.position = position;  
    }  
}
```

Buatlah sebuah interface bernama 'EmployeeAddedListener', potongann kode dari interface dapat dilihat pada gambar di bawah ini.

```
public interface EmployeeAddedListener {  
    public void onEmployeeAdded(Employee employee);  
}
```

Buatlah sebuah class bernama 'Office', potongann kode dari class dapat dilihat pada gambar di bawah ini.

```
public class Office {

    private List<Employee> employees = new ArrayList<>();
    private List<EmployeeAddedListener> listeners = new ArrayList<>();

    public void addEmployee (Employee employee) {
        // Tambah karyawan ke daftar karyawan
        this.employees.add(employee);
        // beritahu daftar listener
        this.notifyEmployeeAddedListeners(employee);
    }

    public void registerEmployeeAddedListener (EmployeeAddedListener listener) {
        // tambah listener ke daftar listener
        this.listeners.add(listener);
    }

    public void unregisterEmployeeAddedListener (EmployeeAddedListener listener) {
        // menghilangkan listener dari daftar listener
        this.listeners.remove(listener);
    }

    protected void notifyEmployeeAddedListeners (Employee employee) {
        // beritahu setiap listener dalam daftar listener
        this.listeners.forEach(listener -> listener.onEmployeeAdded(employee));
    }

}
```

Buatlah sebuah class bernama 'PrintNameEmployeeAddedListener' yang mengimplemen interface 'EmployeeAddedListener', potongann kode dari class dapat dilihat pada gambar di bawah ini.

```
public class PrintEmployeeDataAddedListener implements EmployeeAddedListener{

    @Override
    public void onEmployeeAdded(Employee employee) {
        // Print nama karyawan yang baru ditambahkan
        System.out.println("Ditambahkan karyawan baru dengan posisi '"+employee.getPosition()+"' bernama '"+employee.getName()+"'");
    }

}
```

Buatlah sebuah class bernama 'Main', potongann kode dari class dapat dilihat pada gambar di bawah ini.

```
public class Main {  
  
    public static void main(String[] args) {  
        Office office = new Office();  
  
        office.registerEmployeeAddedListener(new PrintEmployeeDataAddedListener());  
  
        office.addEmployee(new Employee("Andre", "IT Manager"));  
    }  
}
```

Buatlah sebuah class bernama 'CountingEmployeeAddedListener' yang mengimplemen interface 'EmployeeAddedListener', potongann kode dari class dapat dilihat pada gambar di bawah ini.

```
public class CountingEmployeeAddedListener implements EmployeeAddedListener{  
  
    private static int employeesAddedCount = 0;  
  
    @Override  
    public void onEmployeeAdded (Employee employee) {  
        // Inkremen jumlah karyawan  
        employeesAddedCount++;  
  
        // Print jumlah karyawan  
        System.out.println("Total karyawan : " + employeesAddedCount);  
    }  
}
```

Tambahkan potongan kode pada class 'Main' seperti pada gambar di bawah ini.

```
public class Main {  
  
    public static void main(String[] args) {  
        Office office = new Office();  
  
        office.registerEmployeeAddedListener(new PrintEmployeeDataAddedListener());  
        office.registerEmployeeAddedListener(new CountingEmployeeAddedListener());  
  
        office.addEmployee(new Employee("Andre", "IT Manager"));  
        office.addEmployee(new Employee("Vincent", "Senior Programmer"));  
        office.addEmployee(new Employee("Jonathan", "Internship Programmer"));  
    }  
}
```

Hasil output dari program

```
Ditambahkan karyawan baru dengan posisi 'IT Manager' bernama 'Andre'  
Total karyawan : 1  
Ditambahkan karyawan baru dengan posisi 'Senior Programmer' bernama 'Vincent'  
Total karyawan : 2  
Ditambahkan karyawan baru dengan posisi 'Internship Programmer' bernama 'Jonathan'  
Total karyawan : 3
```

## Tugas

### Java Interface

Buatlah sebuah interface bernama 'ClassInfo', potongann kode dari interface dapat dilihat pada gambar di bawah ini.

```
public interface ClassInfo {  
    public String getClassName();  
}
```

## Java Abstract Class

Buatlah sebuah abstract class bernama 'Payment'. Isii dari class dapat dilihat seperti pada gambar di bawah ini.

```
public abstract class Payment implements ClassInfo{
    protected boolean isPaidOff;
    protected Item item;

    public abstract int pay();

    public Payment() {
        this.isPaidOff = false;
        this.item = null;
    }
    public Payment(Item item) {
        this.isPaidOff = false;
        this.item = item;
    }

    public Item getItem() {
        return item;
    }

    public String getItemName() {
        return item.getName();
    }

    public String getStatus() {
        if(isPaidOff) {
            return "FINISHED";
        }
        return "ON PROGRESS";
    }

    public int getRemainingAmount() {
        if(isPaidOff) {
            return 0;
        }
        return item.getPrice();
    }

    public boolean getPaidOff() {
        return isPaidOff;
    }
}
```

Buatlah sebuah class bernama 'Cash'. Isi dari class ini dapat dilihat seperti pada gambar berikut.

```
public class Cash extends Payment{

    public Cash(Item item) {
        super(item);
    }

    public int pay() {
        if(isPaidOff) {
            return 0;
        }
        isPaidOff = true;
        return this.item.getPrice();
    }

    public String getClassName() {
        return "CASH";
    }
}
```

Buatlah sebuah class bernama 'Main'. Isi dari class ini dapat dilihat seperti pada gambar berikut.

```
public class Main {

    static ArrayList<Item> listOfItems = new ArrayList<Item>();
    static ArrayList<Payment> listOfPayments = new ArrayList<Payment>();
    static Scanner s = new Scanner(System.in);

    public static void seedData() {
        listOfItems.add(new Item("Kulkas", "Electronic", 4800000));
        listOfItems.add(new Item("TV", "Electronic", 1280000));
        listOfItems.add(new Item("Laptop", "Computer", 6000000));
        listOfItems.add(new Item("PC", "Computer", 12000000));
    }

    public static void printItem(Item item) {
        System.out.println("Nama : "+item.getName());
        System.out.println("Tipe : "+item.getType());
        System.out.println("Harga : "+item.getPrice());
    }
}
```

```

public static void main(String[] args) {

    int opt = 0, id = 0;
    seedData();
    do {
        System.out.println("---Program Toko Elektronik---");
        System.out.println("1. Pesan Barang");
        System.out.println("2. Lihat Pesanan");
        System.out.println("0. Keluar");
        System.out.print("Pilihan : ");
        opt = s.nextInt();
        if(opt == 1) {
            System.out.println("----Daftar Barang----");
            for(int i = 0; i < listOfItems.size(); i++) {
                System.out.println("No      : "+(i+1));
                printItem(listOfItems.get(i));
                System.out.println("-----");
            }
            System.out.print("Pilih : ");

        }else if(opt == 2) {

        }else {
            System.out.println("Salah Input");
        }
    }while(opt != 0);
}
}

```

Buatlah sebuah class bernama 'Credit' yang juga merupakan turunan dari class 'Payment'. Tambahkan property installment:Integer dan maxInstallmentAmount:Integer pada class Credit. Tambahkan juga constructor yang menerima Item, dan maxInstallmentAmount untuk melakukan proses pengaturan Item pada parent dan maxInstallmentAmount. Pada constructor installmentAmount juga di set menjadi 0 (pembayaran belum dilakukan).

Pada class 'Credit' lakukan override pada method pay(). Method ini akan mengembalikan jumlah yang akan dibayar menggunakan rumus: harga\_barang / maxInstallmentAmount. Method ini juga mengecek apakah cicilan sudah terpenuhi dan mengubah nilai boolean isPaidOff menjadi 'true' jika cicilan sudah terpenuhi.

Lakukan juga proses override pada method getRemainingAmount(). Jika pembayaran sudah lunas (isPaidOff), method ini akan mengembalikan angka nol. Jika tidak method akan mengembalikan jumlah cicilan yang harus dibayar menggunakan perhitungan installmentAmount, maxInstallmentAmount, dan harga barang.



## Implementasi Class Info Interface

Implementasikan method `getClassName` pada class `Cash` dan `Credit` dimana method ini akan mengembalikan nama dari class masing-masing.

## Tugas

Buatlah program pembelian elektronik sederhana. Program ini dapat melihat daftar barang yang ingin dipesan dan program ini juga dapat melihat daftar transaksi pembayaran barang. Pembayaran pada program ini terdapat dua cara yaitu `cash` dan `credit`.

Berikut adalah contoh program pembelian elektronik.

### Menu Utama

```
---Program Toko Elektronik---  
1. Pesan Barang  
2. Lihat Pesanan  
Pilihan : |
```

### Menu Pesan Barang

```
---Program Toko Elektronik---  
1. Pesan Barang  
2. Lihat Pesanan  
Pilihan : 1  
----Daftar Barang----  
No      : 1  
Nama    : Kulkas  
Tipe    : Electronic  
Harga   : 4800000  
-----  
No      : 2  
Nama    : TV  
Tipe    : Electronic  
Harga   : 1280000  
-----  
No      : 3  
Nama    : Laptop  
Tipe    : Computer  
Harga   : 6000000  
-----  
No      : 4  
Nama    : PC  
Tipe    : Computer  
Harga   : 12000000  
-----  
Pilih   : 1  
.
```

```
Pilih : 1
Nama : Kulkas
Tipe : Electronic
Harga : 4800000
----Tipe pembayaran----
1. Cash
2. Credit
Pilih : 1
Bayar (Y/N): Y
Harga Pembayaran : 4800000
Bayar : 4800000
|Transaksi telah dibayar lunas
```

```
Pilih : 2
Nama : TV
Tipe : Electronic
Harga : 1280000
----Tipe pembayaran----
1. Cash
2. Credit
Pilih : 1
Bayar (Y/N): N
|Transaksi telah disimpan
```

```
Pilih : 3
Nama : Laptop
Tipe : Computer
Harga : 6000000
----Tipe pembayaran----
1. Cash
2. Credit
Pilih : 2
Lama Cicilan (3/6/9/12): 0
Lama Cicilan (3/6/9/12): 13
Lama Cicilan (3/6/9/12): 3
Harga Pembayaran : 2000000
Bayar : 2000000
|Transaksi telah dibayar
```

## Menu Lihat Pesanan

---Program Toko Elektronik---

1. Pesan Barang
2. Lihat Pesanan

Pilihan : 2

No : 1  
 Nama : Kulkas  
 Tipe : Electronic  
 Status : FINISHED  
 Sisa Pembayaran : 0

-----  
 No : 2  
 Nama : TV  
 Tipe : Electronic  
 Status : ON PROGRESS  
 Sisa Pembayaran : 1280000

-----  
 No : 3  
 Nama : Laptop  
 Tipe : Computer  
 Status : ON PROGRESS  
 Sisa Pembayaran : 4000000

-----  
 Pilih No Transaksi :

Pilih No Transaksi : 1

Nama : Kulkas  
 Tipe : Electronic  
 Status : FINISHED  
 Sisa Pembayaran : 0  
 Harga Pembayaran : 0  
 Transaksi telah lunas

Pilih No Transaksi : 2

Nama : TV  
 Tipe : Electronic  
 Status : ON PROGRESS  
 Sisa Pembayaran : 1280000  
 Harga Pembayaran : 1280000  
 Bayar : 500000  
 Bayar : 1280000  
 Transaksi telah dibayar lunas

Pilih No Transaksi : 3

Nama : Laptop  
 Tipe : Computer  
 Status : ON PROGRESS  
 Sisa Pembayaran : 4000000  
 Harga Pembayaran : 2000000  
 Bayar : 2000000  
 Transaksi telah dibayar

```
Transaksi telah dibayar
---Program Toko Elektronik---
1. Pesan Barang
2. Lihat Pesanan
Pilihan : 2
No           : 1
Nama         : Kulkas
Tipe         : Electronic
Status       : FINISHED
Sisa Pembayaran : 0
-----
No           : 2
Nama         : TV
Tipe         : Electronic
Status       : FINISHED
Sisa Pembayaran : 0
-----
No           : 3
Nama         : Laptop
Tipe         : Computer
Status       : ON PROGRESS
Sisa Pembayaran : 2000000
-----
```

## REFERENSI

<https://www.javatpoint.com/abstract-class-in-java>

[https://www.tutorialspoint.com/java/java\\_interfaces.htm](https://www.tutorialspoint.com/java/java_interfaces.htm)

<https://dzone.com/articles/the-observer-pattern-using-modern-java>

# MODUL 8

## (CASTING PADA JAVA)



### DESKRIPSI TEMA

Casting pada Java.

### CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

Mahasiswa mempelajari konsep *upcasting*, *downcasting* dan *package*.

### PENUNJANG PRAKTIKUM

1. Eclipse.

### LANGKAH-LANGKAH PRAKTIKUM

#### Upcasting

Upcasting adalah mengubah sebuah reference dari kelas turunan menjadi kelas dasarnya. Dengan upcasting kita bisa menyimpan kelas turunan ke dalam kelas dasar.

Berikut adalah contoh upcasting

```
public class Pekerja {  
    public Pekerja() {}  
  
    public void tanyaIdentitas() {  
        System.out.println("Saya pekerja biasa");  
    }  
}  
  
public class CEO extends Pekerja{  
    public CEO() {}  
  
    public void tanyaIdentitas() {  
        System.out.println("Saya seorang CEO");  
    }  
}  
  
public class Karyawan extends Pekerja{  
    public Karyawan() {}  
  
    public void tanyaIdentitas() {  
        System.out.println("Saya seorang Karyawan");  
    }  
}
```

```
public class Main {  
  
    public static void main(String[] args) {  
        Pekerja pekerja = new Pekerja();  
        pekerja.tanyaIdentitas();  
        //contoh upcasting  
        pekerja = new CEO();  
        pekerja.tanyaIdentitas();  
        //contoh upcasting  
        Karyawan karyawan = new Karyawan();  
        pekerja = (Pekerja)karyawan;  
        pekerja.tanyaIdentitas();  
    }  
}
```

Berikut adalah hasil output dari contoh program upcasting.

```
Saya pekerja biasa  
Saya seorang CEO  
Saya seorang Karyawan
```

### Downcasting

Downcasting merupakan kebalikan dari upcasting, mengubah sebuah kelas dasar menjadi kelas turunan, method / atribut pada kelas turunan yang tidak terdapat di kelas dasar dapat diakses.

Berikut adalah contoh downcasting.

```
public class CEO extends Pekerja{  
  
    public CEO() {}  
  
    public void tanyaIdentitas() {  
        System.out.println("Saya seorang CEO");  
    }  
  
    public void tanyaPendapatan() {  
        System.out.println("Pendapatan saya 100 juta per bulan");  
    }  
}
```

```
public class Main {  
  
    public static void main(String[] args) {  
  
        //contoh downcasting  
        CEO c = new CEO();  
        Pekerja p = new CEO();  
        c = (CEO)p;  
        c.tanyaPendapatan();  
  
    }  
  
}
```

## Package

Package dalam Java adalah sebuah mekanisme untuk merangkum sebuah grup class, sub packages dan interfaces. Kegunaan package adalah

- Mencegah konflik penamaan
- Membuat pencarian / penempatan dan penggunaan class, interfaces, enumerations dan annotations lebih mudah.
- Menyediakan akses terkendali: dilindungi dan default memiliki kontrol akses tingkat paket. Anggota yang protected dapat diakses oleh class dalam paket yang sama dan subclassesnya. Anggota default dapat diakses oleh class yang hanya dalam paket yang sama.
- Paket dapat dianggap sebagai enkapsulasi data.

Terdapat 2 jenis packages dalam Java yaitu Built-in packages dan User defined packages.

- Built-in Packages

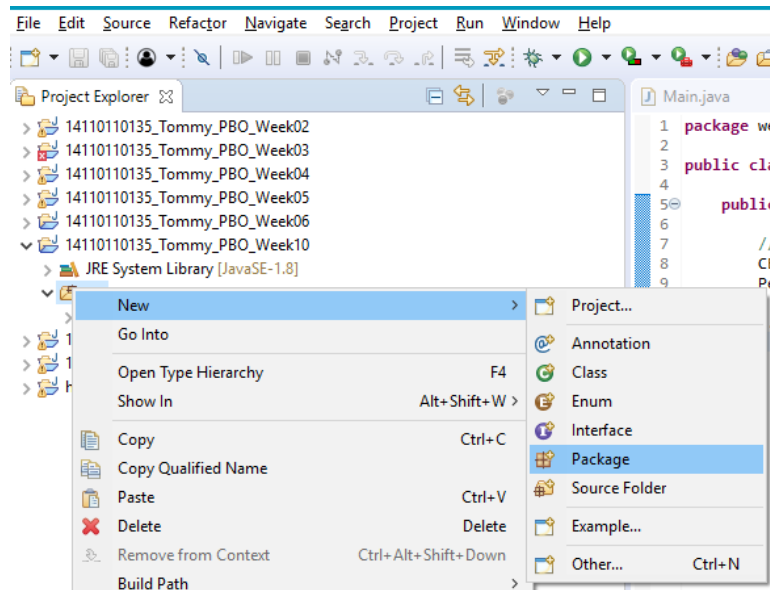
Package-package ini terdiri dari sejumlah besar class yang merupakan bagian dari Java API. Beberapa package yang biasanya digunakan adalah :

1. java.lang : Berisi class dukungan (misalnya class yang mendefinisikan tipe data primitif, operasi matematika). Paket ini diimpor secara otomatis.
2. java.io : Berisi digolongkan untuk mendukung operasi input/output.
3. java.util : Berisi class utilitas yang mengimplementasikan struktur data seperti linked list, kamus dan dukungan untuk operasi tanggal / waktu.
4. java.applet : berisi class untuk membuat Applet.
5. java.awt : berisi class untuk mengimplementasikan komponen untuk antarmuka pengguna grafis seperti button, menu, dll).
6. java.net : berisi class untuk mendukung operasi jaringan.

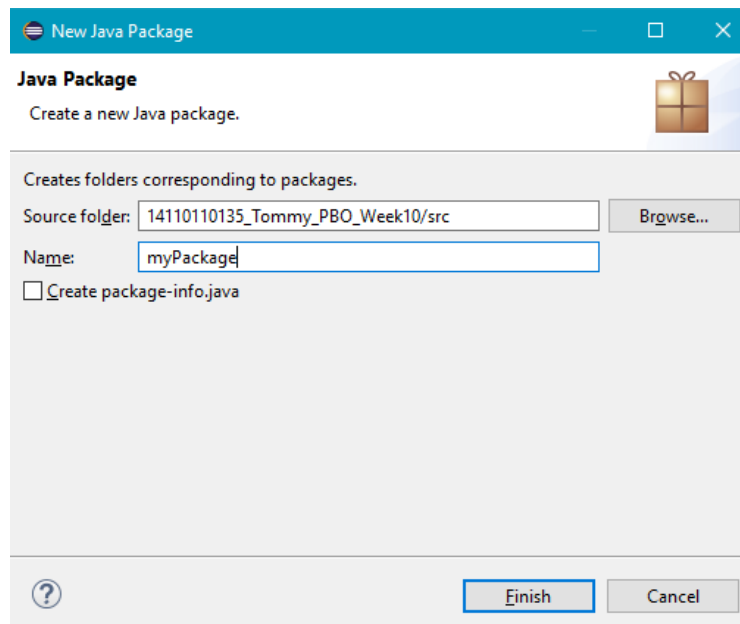
- User Defined Packages

User Defined packages adalah packages yang dibuat oleh pengguna. Berikut adalah contoh cara pembuatan packages.

1. Klik kanan pada folder src, pilih New > Package.



2. Beri nama packages, contoh beri nama package dengan nama myPackage.





3. Buatlah class dengan nama MyClass yang berisikan code seperti gambar dibawah ini.

```
package myPackage;

public class MyClass {
    public void getNames(String s) {
        System.out.println(s);
    }
}
```

4. Buatlah class di luar package myPackage dengan nama PrintName yang berisikan code seperti gambar dibawah ini.

```
import myPackage.MyClass;

public class PrintName {

    public static void main(String[] args) {
        String name = "Pemrograman Berorientasi Objek";

        MyClass obj = new MyClass();

        obj.getNames(name);
    }
}
```

5. Untuk dapat mengakses class yang didalam package myPackage perlu menuliskan code import.[nama package].[nama class]. Dalam contoh menuliskan import.myPackage.MyClass.

## Tugas

### Tugas 1

Lanjutkan program pembelian handphone dan voucher pada week 6. Terapkan upcasting, downcasting dan package pada program tersebut. Berikut perubahan dan penerapan pada program.

- Ubah id pada class Barang, Handphone dan Voucher menjadi String.
- Buatlah package Model yang berisi class Barang, Handphone, Voucher dan Order.
- Simpan objek Handphone dan Voucher dalam array class Barang.

## Menu Barang Baru

-----Menu Toko Voucher & HP-----

1. Pesan Barang  
2. Lihat Pesanan  
3. Barang Baru  
Pilihan : 3  
Voucher / Handphone (V/H): h  
Nama : Samsung Note 9  
Harga : 13000000  
Stok : 10  
Warna : Hitam  
Handphone telah berhasil diinput

-----Menu Toko Voucher & HP-----

1. Pesan Barang  
2. Lihat Pesanan  
3. Barang Baru  
Pilihan : 3  
Voucher / Handphone (V/H): v  
Nama : Google Play  
Harga : 20000  
Stok : 100  
PPN : 0.1  
Voucher telah berhasil diinput

## Menu Pesang Barang

-----Menu Toko Voucher & HP-----

1. Pesan Barang  
2. Lihat Pesanan  
3. Barang Baru  
Pilihan : 1  
Daftar Barang Toko Voucher & HP  
ID : H01  
Nama : Samsung Note 9 Hitam  
Stok : 9  
Harga : 13000000

-----

ID : V01  
Nama : Google Play  
Nominal : 20000  
Stok : 95  
Harga : 22000

-----

Ketik 0 untuk batal  
Pesan Barang (ID) : s0  
Barang tidak ditemukan

-----Menu Toko Voucher & HP-----

1. Pesan Barang
2. Lihat Pesanan
3. Barang Baru

Pilihan : 1

Daftar Barang Toko Voucher & HP

ID : H01

Nama : Samsung Note 9 Hitam

Stok : 10

Harga : 13000000

ID : V01

Nama : Google Play

Nominal : 20000

Stok : 100

Harga : 22000

Ketik 0 untuk batal

Pesan Barang (ID) : H01

Masukkan Jumlah : 1

1 @ Samsung Note 9 dengan total harga 13000000

Masukkan jumlah uang : 15000000

berhasil dipesan

-----Menu Toko Voucher & HP-----

1. Pesan Barang
2. Lihat Pesanan
3. Barang Baru

Pilihan : 1

Daftar Barang Toko Voucher & HP

ID : H01

Nama : Samsung Note 9 Hitam

Stok : 9

Harga : 13000000

ID : V01

Nama : Google Play

Nominal : 20000

Stok : 100

Harga : 22000

Ketik 0 untuk batal

Pesan Barang (ID) : V01

Masukkan Jumlah : 5

5 @ Google Play dengan total harga 110000

Masukkan jumlah uang : 110000

berhasil dipesan

## Menu Lihat Pesanan

-----Menu Toko Voucher & HP-----

1. Pesan Barang
2. Lihat Pesanan
3. Barang Baru

Pilihan : 2

Daftar Pesanan Toko Multiguna

ID : OH01-0

Nama : Samsung Note 9 Hitam

Jumlah : 1

Total : 13000000

-----  
ID : OV01-1

Nama : Google Play

Nominal : 20000

Jumlah : 5

Total : 110000  
-----

## REFERENSI

<https://www.javatpoint.com/package>

<https://www.geeksforgeeks.org/packages-in-java/>

# MODUL 9

## (FUNCTION OVERRIDING & FUNCTION OVERLOADING)



### DESKRIPSI TEMA

Function Overriding & Function Overloading.

### CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

Mahasiswa mempelajari konsep function overloading dan function overriding.

### PENUNJANG PRAKTIKUM

1. Eclipse.

### LANGKAH-LANGKAH PRAKTIKUM

#### Function Overriding dan Function Overloading

Setelah mengenal konsep inheritance, berikutnya adalah polymorphism. Dengan polimorfisme, kita bisa membuat method dengan nama yang sama tapi isi dan eksekusinya berbeda (function overloading) dan method dengan nama yang sama pada parent dan child class tapi masing-masing memiliki isi dan eksekusi yang berbeda (function overriding).

Berikut adalah contoh code function overloading dan overriding, silahkan diketik dan dicoba

```
public class Polygon {
    protected int width;
    protected int height;
    //Contoh function overloading constructor
    //perhatikan jumlah parameternya
    public Polygon() {
        this.width = 0;
        this.height = 0;
    }

    public Polygon(int width, int height) {
        this.width = width;
        this.height = height;
    }

    public void askWidth() {
        System.out.println("Object ini memiliki width dengan nilai "+this.width);
    }

    public void countSide() {
        System.out.println("Object ini jumlah sisinya tidak jelas");
    }
}
```

```
public class Rectangle extends Polygon{

    public Rectangle() {}
    //contoh overriding
    public void countSide() {
        System.out.println("Object ini memiliki 4 sisi");
    }
}

public class Triangle {

    public Triangle() {}
    //contoh overriding
    public void countSide() {
        System.out.println("Object ini memiliki 3 sisi");
    }
}

public class Main {

    public static void main(String[] args) {
        Polygon poly1 = new Polygon();
        poly1.askWidth();
        Polygon poly2 = new Polygon(1,2);
        poly2.askWidth();

        Rectangle rect1 = new Rectangle();
        rect1.countSide();

        Triangle tri1 = new Triangle();
        tri1.countSide();

    }
}
```

Berikut adalah hasil dari code function overloading dan overriding.

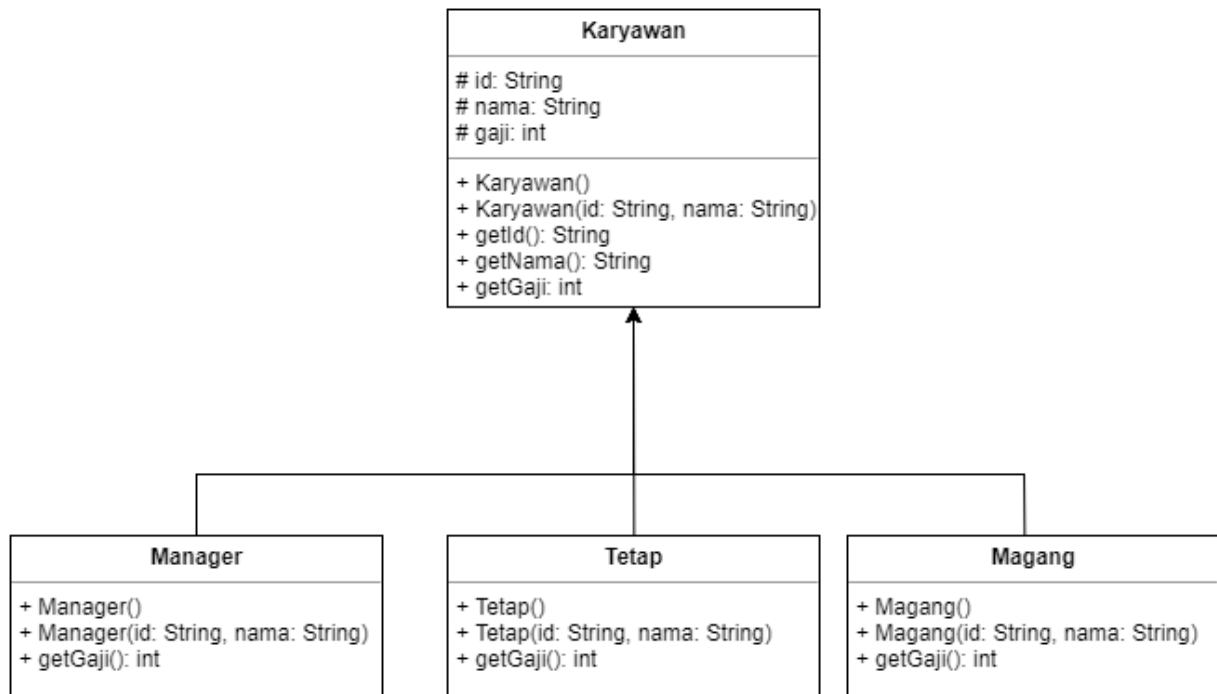
```
Object ini memiliki width dengan nilai 0
Object ini memiliki width dengan nilai 1
Object ini memiliki 4 sisi
Object ini memiliki 3 sisi
```

Hal yang perlu diingat dalam function overloading dan function overriding adalah :

1. Nama method sama
2. Method yang di-overloading bisa memiliki return type yang berbeda, sedangkan method yang di-overriding return type harus sama
3. Method yang di-overloading bisa memiliki jumlah dan tipe parameter yang berbeda, sedangkan method yang di-overriding jumlah dan tipe parameter harus sama

## Tugas

### Tugas 1



Buatlah program sederhana untuk menyimpan data karyawan. Fitur pada program data karyawan ini adalah :

1. Program dapat menambahkan data karyawan baru. Program akan menerima *input* nama dan status karyawan (manajer, karyawan tetap dan karyawan magang).
2. Program dapat melihat semua data karyawan yang sudah dimasukkan. Menampilkan id, nama dan gaji. Gaji pada karyawan tidak perlu di-*input*.

Berikut adalah contoh gambar pada program.

#### - Menu Utama

```

-----Program Data Karyawan-----
1. Lihat Karyawan
2. Tambah Karyawan
3. Keluar
Pilih :
  
```

## - Menu Tambah Karyawan

```
-----Program Data Karyawan-----
1. Lihat Karyawan
2. Tambah Karyawan
3. Keluar
Pilih : 2
-----Tambah Karyawan-----
1. Manajer
2. Karyawan Tetap
3. Karyawan Magang
Pilih : 1
Nama : Ridwan
Manajer baru telah ditambahkan
```

## - Menu Lihat Karyawan

```
-----Program Data Karyawan-----
1. Lihat Karyawan
2. Tambah Karyawan
3. Keluar
Pilih : 1
-----Daftar Manajer-----
ID   : M1
Nama : Ridwan
Gaji : Rp. 10.000.000,00
-----
-----Daftar Pegawai Tetap-----
Tidak ada
-----Daftar Pegawai Magang-----
Tidak ada

-----Program Data Karyawan-----
1. Lihat Karyawan
2. Tambah Karyawan
3. Keluar
Pilih : 1
-----Daftar Manajer-----
ID   : M1
Nama : Ridwan
Gaji : Rp. 10.000.000,00
-----
-----Daftar Pegawai Tetap-----
ID   : R1
Nama : Pendi
Gaji : Rp. 3.000.000,00
-----
-----Daftar Pegawai Magang-----
ID   : I1
Nama : Mario
Gaji : Rp. 1.500.000,00
-----
```

## REFERENSI



# MODUL 10

## (EXCEPTION HANDLING)



### DESKRIPSI TEMA

- Jenis-jenis Exception
- Exception Handling

### CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

Mahasiswa mempelajari konsep *exception*.

### PENUNJANG PRAKTIKUM

1. Eclipse.

### LANGKAH-LANGKAH PRAKTIKUM

#### Exception

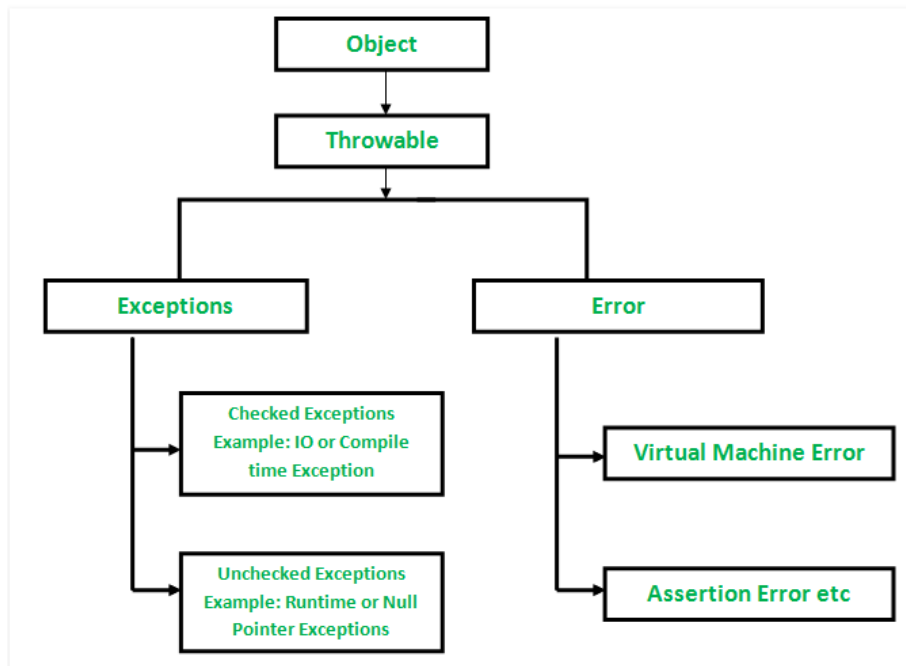
Exception adalah peristiwa yang tidak diinginkan atau tidak terduga, yang terjadi selama pelaksanaan program yang pada saat run time, yang mengganggu aliran normal dari instruksi program.

#### Error vs Exception

- Error menunjukkan masalah serius yang seharusnya tidak coba ditangkap oleh aplikasi yang wajar.
- Exception menunjukkan ketentuan bahwa aplikasi yang wajar mungkin mencoba untuk menangkap

#### Hierarki Exception

Semua jenis exception dan error adalah sub class dari class throwable, yang merupakan class dasar hierarki. Satu cabang dikepalai oleh Exception. Class ini digunakan untuk kondisi luar biasa yang harus ditangkap oleh program pengguna. NullPointerException adalah contoh exception semacam itu. Cabang lain, Error digunakan oleh Java run-time system (JVM) untuk menunjukkan kesalahan yang berkaitan dengan lingkungan run-time itu sendiri (JRE). StackOverflowError adalah contoh kesalahan seperti itu.



### Default Exception Handling

Setiap kali di dalam suatu method, jika exception telah terjadi, method tersebut menciptakan Objek yang dikenal sebagai Objek Exception dan menyerahkannya ke sistem run-time (JVM). Objek exception berisi nama dan deskripsi exception, dan keadaan saat ini dari program di mana exception telah terjadi. Membuat Objek Exception dan menanganinya ke sistem run-time disebut throwing an Exception. Mungkin ada daftar method yang telah dipanggil untuk menuju ke method di mana exception terjadi.

Daftar ini memerintahkan metode disebut Call Stack. Sekarang prosedur berikut akan terjadi.

- Sistem run-time mencari tumpukan panggilan untuk menemukan method yang berisi blok kode yang dapat menangani pengecualian yang terjadi. Blok kode itu disebut Exception handler.
- Sistem run-time mulai mencari dari method di mana exception terjadi, berlanjut melalui call stack dalam urutan terbalik di mana method dipanggil.
- Jika menemukan handler yang sesuai maka ia melewati pengecualian yang terjadi. Penangan yang tepat berarti jenis objek exception yang dilempar cocok dengan jenis objek exception yang dapat ditangani.
- Jika sistem run-time mencari semua method di Call Stack dan tidak dapat menemukan penangan yang sesuai, maka sistem run-time menyerahkan Objek Exception ke penangan exception default, yang merupakan bagian dari sistem run-time. Penangan ini mencetak informasi exception dalam format berikut dan menghentikan program secara tidak normal.

### Contoh Default Exception Handling :

```
public class ThrowException {

    public static void main(String[] args) {
        String str = null;
        System.out.println(str.length());
    }

}
```

Output :

```
java.lang.NullPointerException
    at ThrowException.main(ThrowException.java:7)
```

Contoh yang mengilustrasikan bagaimana sistem run-time mencari penangan exception code pada call stack :

```
// Program java untuk mendemonstrasikan exception dilempar
// bagaimana sistem run-time mencari call stack
// untuk mencari penanganan exception yang sesuai.
public class ExceptionThrown {
    // Melempar Exception(ArithmeticException).
    // Appropriate Exception handler is not found within this method.
    static int divideByZero(int a, int b){
        // Statement ini akan menyebabkan ArithmeticException(/ oleh nol)
        int i = a/b;
        return i;
    }

    // Sistem run-time mencari penanganan exception yang sesuai
    // dalam method ini juga tetapi tetapi tidak bisa ditemukan. Jadi melihat kedepan
    // pada call stack.
    static int computeDivision(int a, int b) {
        int res =0;
        try
        {
            res = divideByZero(a,b);
        }
        // tidak sesuai dengan ArithmeticException
        catch(NumberFormatException ex)
        {
            System.out.println("NumberFormatException is occured");
        }
        return res;
    }

    // Dalam method ini mencari Exception handler yang sesuai.
    // yaitu mencocokkan blok catch.
    public static void main(String args[]){
        int a = 1;
        int b = 0;
        try
        {
            int i = computeDivision(a,b);
        }
        // sesuai ArithmeticException
        catch(ArithmeticException ex)
        {
            // getMessage akan mem-print deskripsi exception(ini / oleh nol)
            System.out.println(ex.getMessage());
        }
    }

}
```

Output :  
/ by zero

## Customized Exception Handling

Penangan java exception dikelola melalui lima kata kunci : try, catch, throw, throws, finally.

Secara singkat inilah cara kerjanya:

Pernyataan program yang menurut Anda dapat meningkatkan *exception* terkandung dalam blok try. Jika *exception* terjadi dalam blok try, itu akan dilempar. Kode anda dapat menangkap *exception* ini (menggunakan blok catch) dan menanganinya dengan beberapa cara rasional. Pengecualian yang dihasilkan sistem secara otomatis dilempar oleh sistem run-time Java. Untuk melempar *exception* secara manual, gunakan kata kunci throw. Setiap *exception* yang dilempar dari suatu *method* harus ditentukan seperti itu oleh kata kunci throw. Setiap kode yang benar-benar harus dijalankan setelah blok coba selesai dimasukkan ke blok finally.

Contoh customized exception handling :

```
public class CustomizedException {  
  
    public static void main(String[] args) {  
        try {  
            int[] arr = new int[4];  
            int i = arr[4];  
  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Output :

```
java.lang.ArrayIndexOutOfBoundsException: 4  
    at CustomizedException.main(CustomizedException.java:7)
```

## Tugas

### Membuat Custom Exception

Buatlah sebuah package bernama 'exceptions' di dalam package 'src', Buatlah sebuah class bernama 'AuthenticationException' dan isi dari dapat dilihat seperti pada gambar dibawah ini.

```
package exceptions;

public class AuthenticationException extends Exception{

    public AuthenticationException() {
        super("Anda telah mencapai jumlah batas login");
    }

    public AuthenticationException(String message) {
        super(message);
    }
}
```

Buatlah sebuah class bernama 'InvalidPropertyException' seperti pada gambar di bawah ini.

```
package exceptions;

public class InvalidPropertyException extends Exception{

    public InvalidPropertyException() {
        super("Input data tidak valid");
    }

    public InvalidPropertyException(String message) {
        super(message);
    }
}
```

Buatlah sebuah class bernama 'ExcessiveFailedLoginException' seperti pada gambar di bawah ini.

```
package exceptions;

public class ExcessiveFailedLoginException extends Exception{

    public ExcessiveFailedLoginException() {
        super("Anda telah mencapai jumlah batas login");
    }

    public ExcessiveFailedLoginException(String message) {
        super(message);
    }
}
```

## Java Class yang melempar Exceptions

Buatlah sebuah class bernama 'User' di dalam package 'src'. Isi dari class ini dapat dilihat seperti pada gambar berikut.

```
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import javax.xml.bind.DatatypeConverter;
import exceptions.ExcessiveFailedLoginException;
import exceptions.InvalidPropertyException;

public class User {

    private String firstName;
    private String lastName;
    private Character gender;
    private String address;
    private String userName;
    private String password;
    private MessageDigest digest;

    private static final int maxLoginAttempts = 3;
    private static int loginAttempts;
```

```

private String hash(String strToHash) {
    try {
        digest = MessageDigest.getInstance("SHA-256");
        byte[] hash = digest.digest(strToHash.getBytes(StandardCharsets.UTF_8));
        return DatatypeConverter.printHexBinary(hash);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return "";
}

public User(String firstName, String lastName, Character gender, String address,
            String userName, String password) {

    this.firstName = firstName;
    this.lastName = lastName;
    this.gender = gender;
    this.address = address;
    this.userName = userName;
    this.password = hash(password);
}

public boolean login(String username, String password) throws ExcessiveFailedLoginException {
    if(this.userName.equals(username)) {
        if(LoginAttempts == maxLoginAttempts) {
            LoginAttempts++;
            throw new ExcessiveFailedLoginException();
        } else if(LoginAttempts > maxLoginAttempts) {
            throw new ExcessiveFailedLoginException("Anda telah mencapai batas login");
        }

        if(this.password.equals(hash(password))) {
            LoginAttempts = 0;
            return true;
        } else {
            LoginAttempts++;
        }
    }
    return false;
}

public String greeting() {
    String greet = "Selamat Datang!";
    switch (gender) {
        case 'M' : greet+="Tuan ";break;
        case 'F' : greet+="Nona ";break;
    }
    greet += this.firstName + " " + this.lastName;

    return greet;
}

public String getUserName() {
    return userName;
}
}

```

## Tugas Mandiri

1. Buatlah method initialize() pada class 'Main' yang melakukan penambahan sebuah user baru dengan identitas berikut!

Nama Depan	John
Nama Belakang	Doe
Jenis Kelamin	L
Alamat	Jl. Merpati No. 1 RT 1 RW 1, Banten
Username	admin
Password	admin

2. Buatlah sebuah method handleLogin() pada class 'Main' yang akan memanggil method login pada setiap user di listOfUser. Jika proses iterasi listOfUser selesai dan proses login belum berhasil, fungsi akan melempar sebuah AuthenticationException! Jangan lupa untuk meng-handle ExcessiveFailedLoginException dan AuthenticationException!
3. Buatlah method handleSignUp() pada class 'Main' untuk mendaftarkan user baru pada listOfUser. Berikut adalah contoh program.

### Menu Utama

```
1. Login
2. Sign Up
Pilihan : |
```

### Menu Login

```
1. Login
2. Sign Up
Pilihan : 1
Username : admin
Password : admin
Selamat Datang! John Doe
```



```
1. Login
2. Sign Up
Pilihan : 1
Username : admin
Password : a
Username / password salah
Username : admin
Password : nimda
Username / password salah
Username : admin
Password : 123456
Username / password salah
Username : admin
Password : admin123
Anda telah mencapai jumlah batas login
```

### Menu Lihat Sign Up

```
1. Login
2. Sign Up
Pilihan : 2
Nama Depan : Jack
Nama Belakang : Wallace
Jenis Kelamin (L/P) : L
Alamat : Curug Permai
Username : jackwallace
Password : Thedeathwish74
User telah berhasil didaftarkan
```

```
1. Login
2. Sign Up
Pilihan : 2
Nama Depan : Jack
Nama Belakang : Wallace
Jenis Kelamin (L/P) : L
Alamat : Curug Permai
Username : jack
Username harus lebih dari 8 karakter
Nama Depan : Jack
Nama Belakang : Wallace
Jenis Kelamin (L/P) : L
Alamat : Curug Permai
Username : jackwallace
Password : thedeathwish
Password harus mengandung huruf besar, angka, minimum 6 karakter dan maksimum 16 karakter
```

## REFERENSI

<https://www.geeksforgeeks.org/exceptions-in-java/>

# MODUL 11

## (KONSEP GUI, LAYOUT, CONTAINER, FRAME, PANEL, UI COMPONENTS LAINNYA)

### DESKRIPSI TEMA

- Konsep GUI
- Layout
- Container
- Frame
- Panel
- UI Components.

### CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

Mahasiswa mempelajari membuat java program dengan UI dengan konsep OOP.

### PENUNJANG PRAKTIKUM

1. Eclipse.

### LANGKAH-LANGKAH PRAKTIKUM

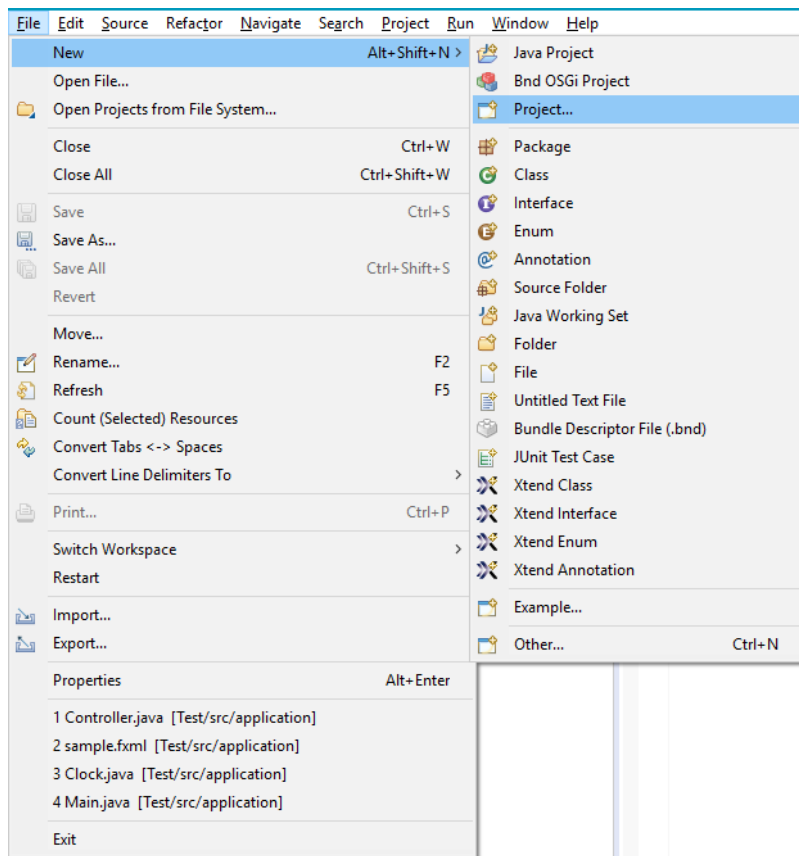
#### Petunjuk

Download eclipse dengan JavaFX pada link dibawah ini :

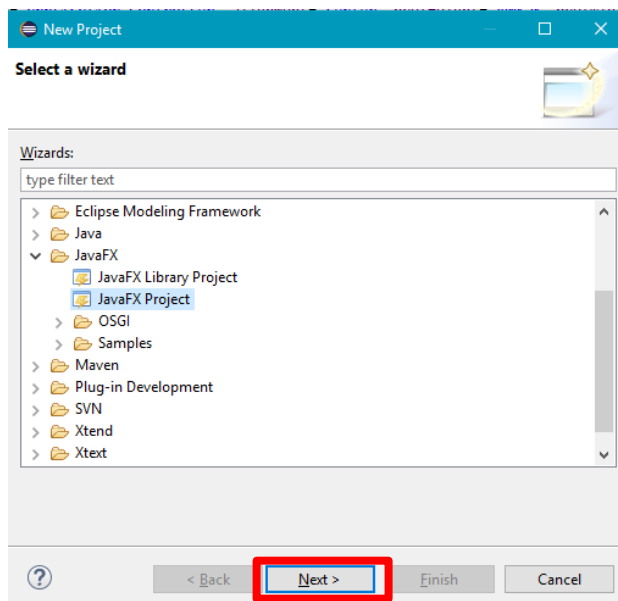
[http://downloads.efxclipse.bestsolution.at/downloads/released/2.4.0/sdk/eclipse-SDK-4.6.0-win32-x86\\_64-distro-2.4.0.zip](http://downloads.efxclipse.bestsolution.at/downloads/released/2.4.0/sdk/eclipse-SDK-4.6.0-win32-x86_64-distro-2.4.0.zip)

Cara memasang eclipse dengan JavaFX dan membuat project JavaFX :

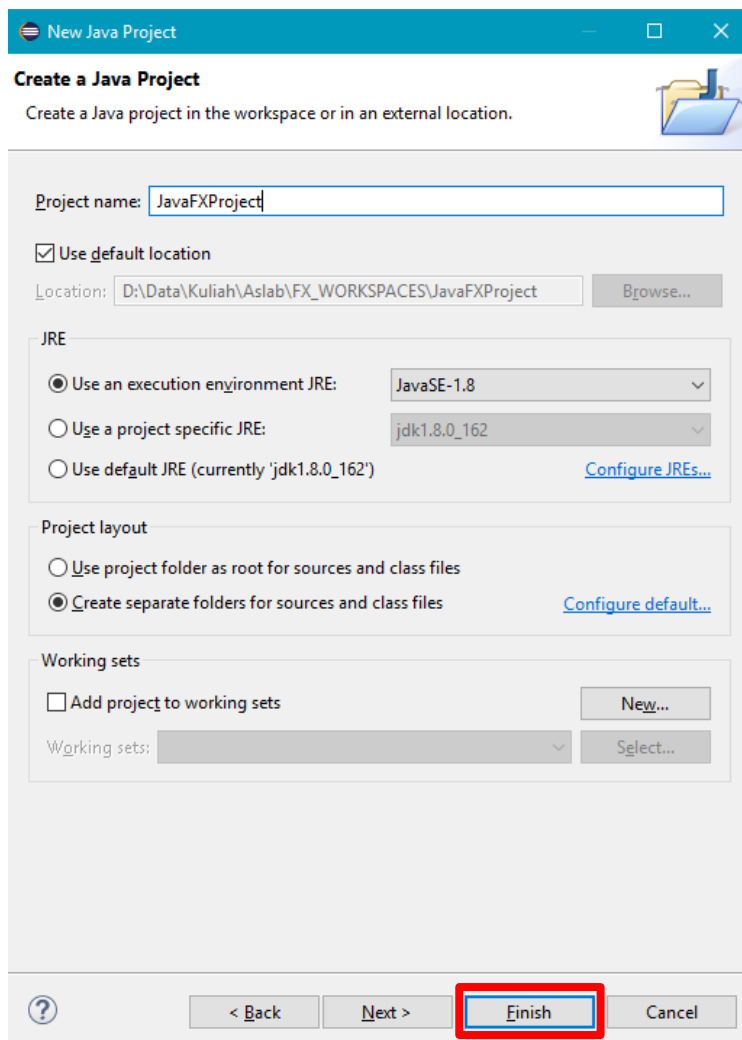
1. Extract zip file pada link
2. Buka eclipse
3. Buka menu proyek baru dengan buka File > New > Project



4. Klik folder JavaFX > JavaFX Project, bila sudah klik Next



5. Beri nama projek JavaFX, bila sudah klik Finish.

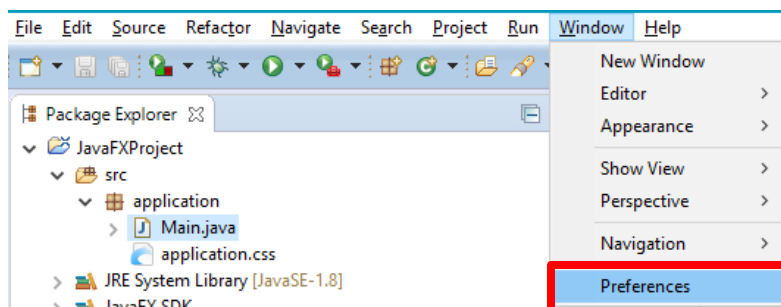


Download Java Scene Builder pada link dibawah ini :

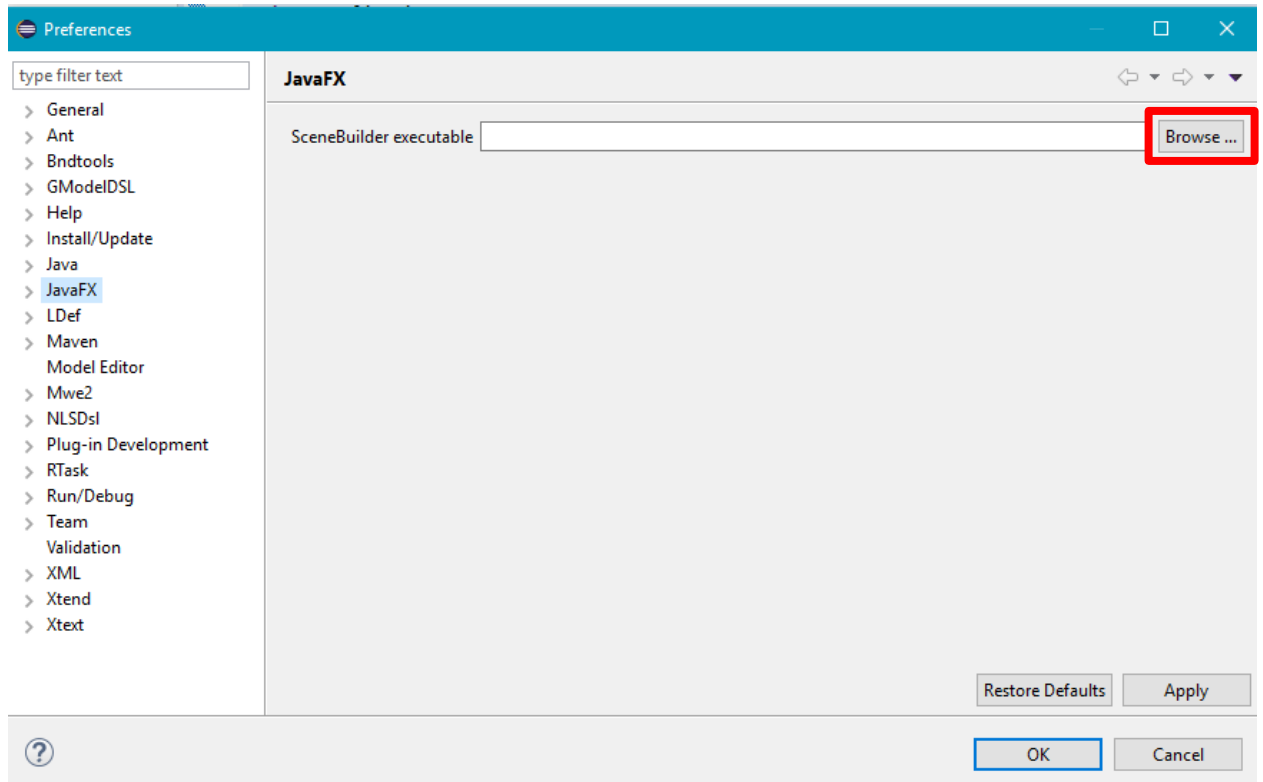
[http://download.oracle.com/otn-pub/java/javafx/scenbuilder/2.0-b20/javafx\\_scenbuilder-2.0-windows.msi](http://download.oracle.com/otn-pub/java/javafx/scenbuilder/2.0-b20/javafx_scenbuilder-2.0-windows.msi)

Cara memasang Java Scene Builder dan membuat UI dengan SceneBuilder:

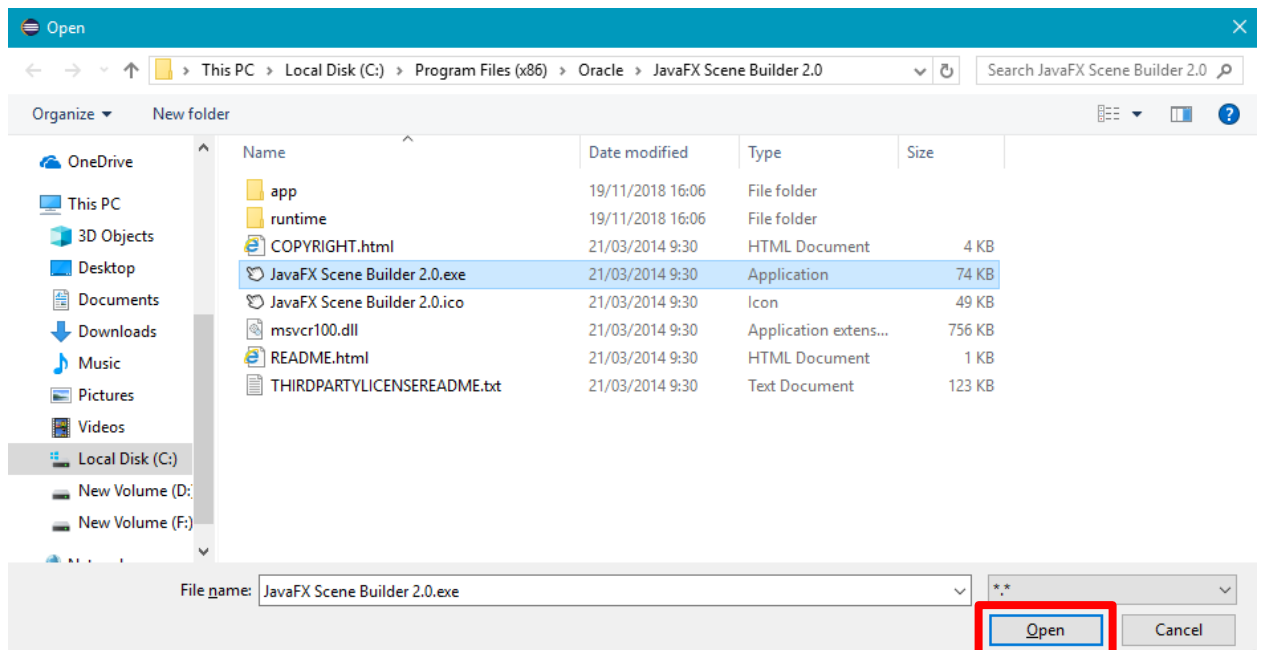
1. Buka eclipse
2. Buka preferences pada Windows > Preferences



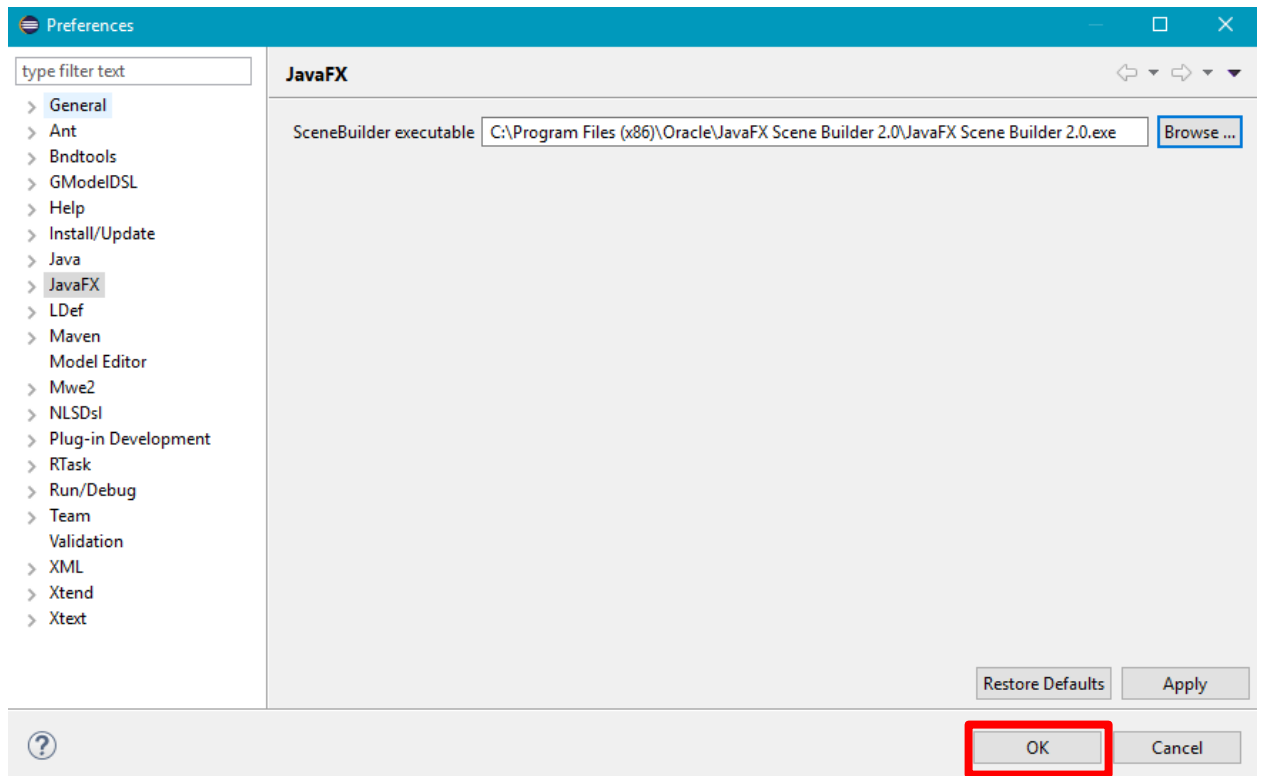
- Klik JavaFX dan klik browse



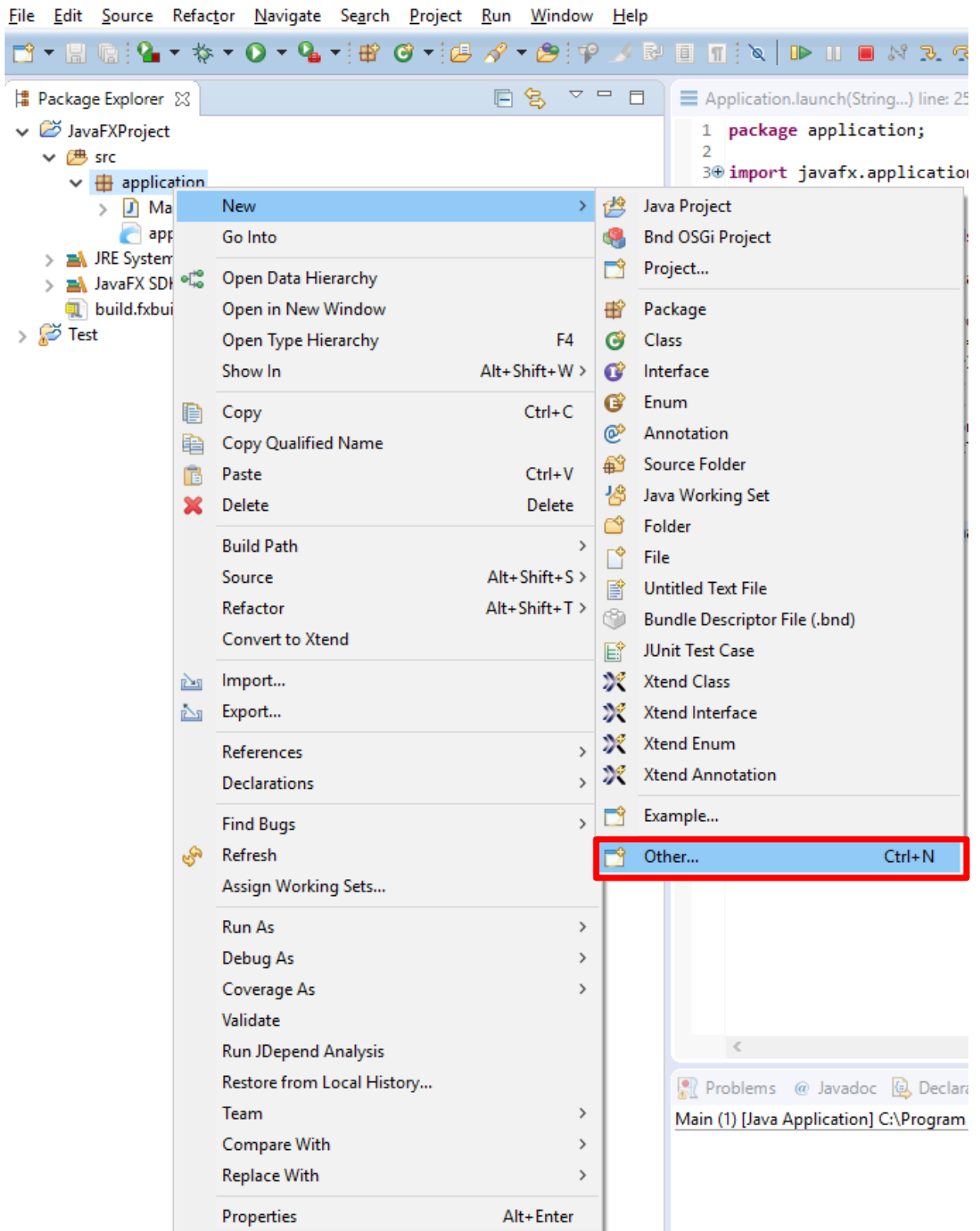
- Cari dimana JavaSceneBuilder diinstall, kemudian pilih klik JavaSceneBuilder.exe dan klik Open



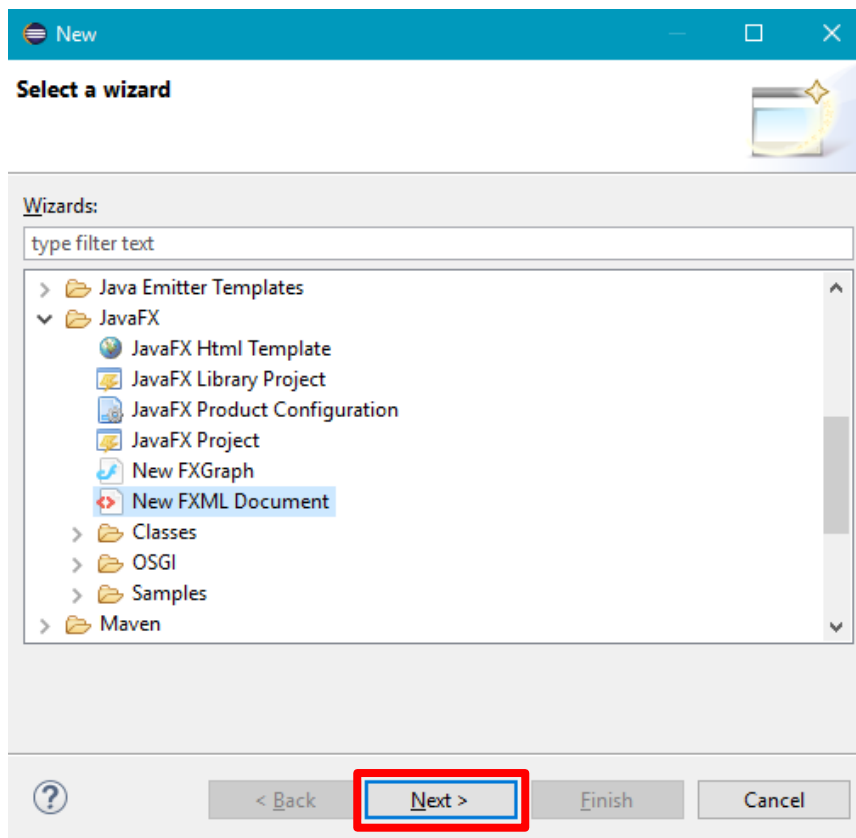
- Setelah itu akan kembali ke menu preferences dan klik Ok.



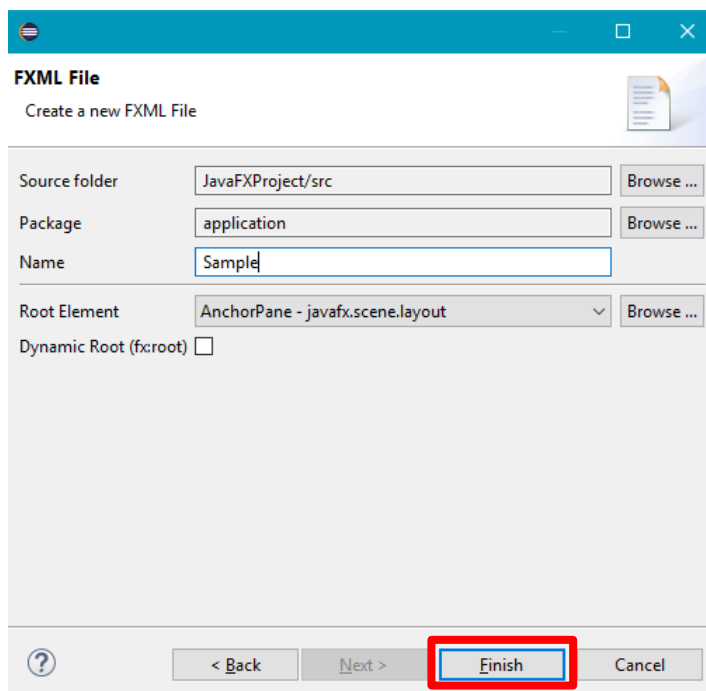
6. Klik kanan pada package application pilih New > Other... atau Ctrl+N



7. Pada menu New pilih folder JavaFX > New FXML Document, kemudian klik Next.

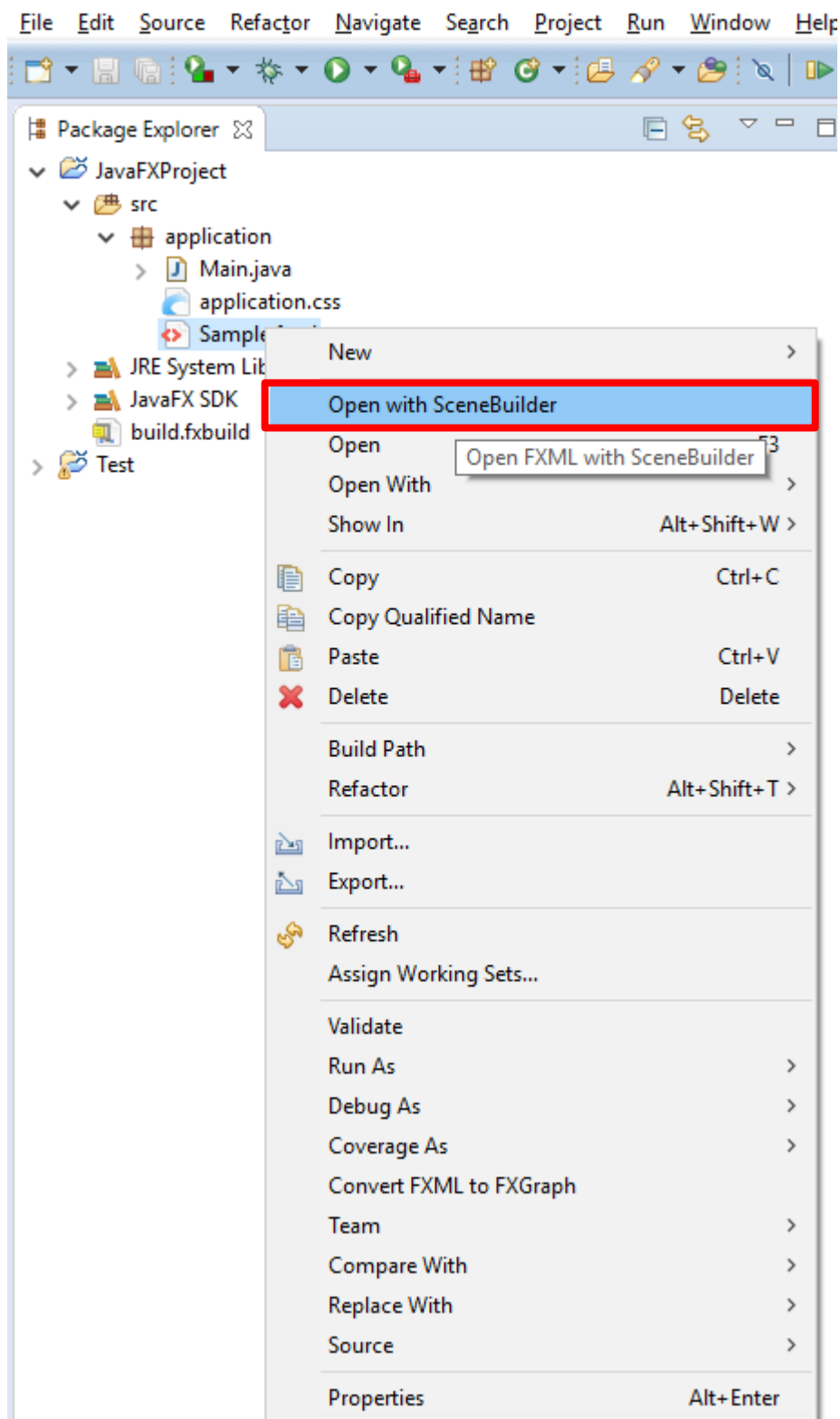


8. Beri nama file fxml, kemudian klik Finish.

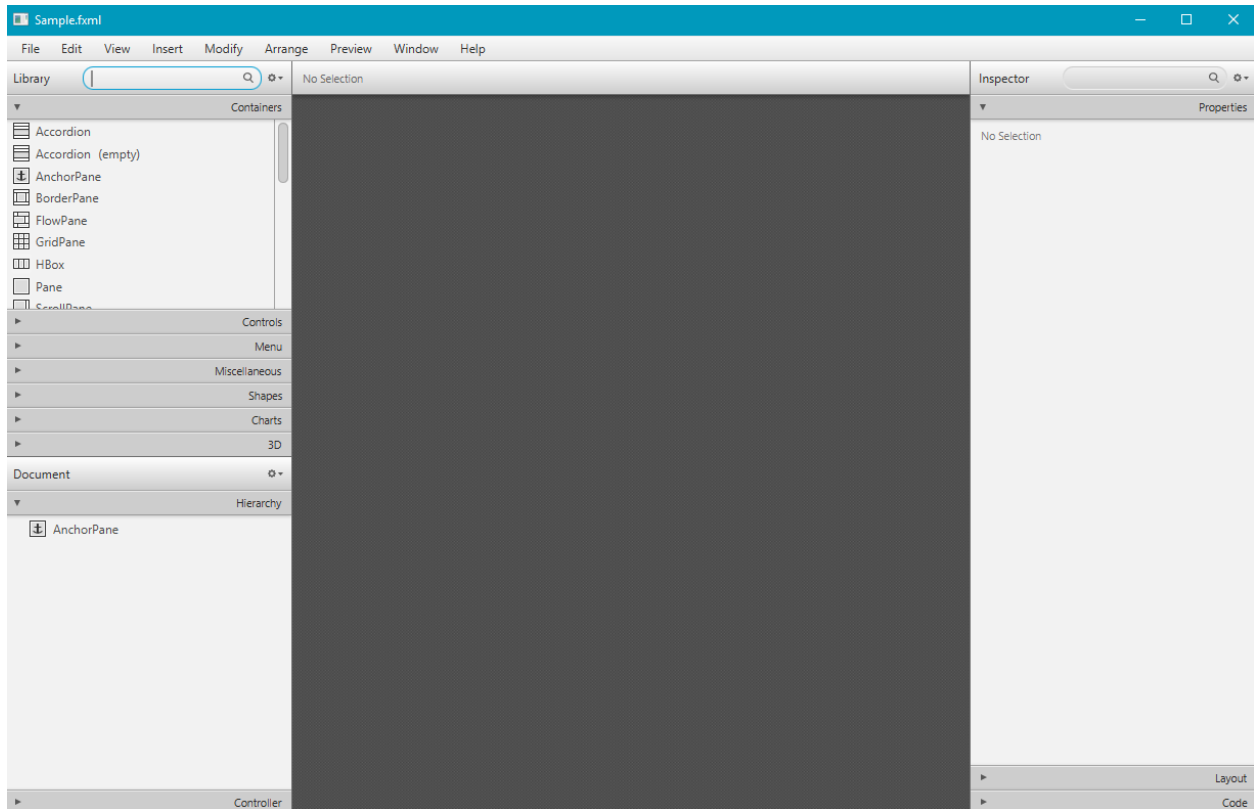




9. Klik kanan pada file fxml kemudian pilih 'Open with SceneBuilder'



10. SceneBuilder akan terbuka. Setelah membuat UI dengan SceneBuilder harus di save agar semua pekerjaan tidak hilang. Tampilan SceneBuilder seperti pada gambar dibawah ini.



## Tutorial Pratikum

Buat sebuah proyek JavaFX dengan nama Tutorial1. Kemudian buatlah class dengan nama 'Clock'. Pada class 'Clock' tuliskan potongan kode Gambar di bawah ini.

```
package application;

public class Clock {
    private int hours;
    private int minutes;
    private int seconds;

    public Clock(){
        this.hours = this.minutes = this.seconds;
    }

    public Clock(int hours, int minutes, int seconds) {
        super();
        this.hours = hours;
        this.minutes = minutes;
        this.seconds = seconds;
    }

    public void addSecond(){
        this.seconds++;
        seconds%=60;
        if(seconds == 0){
            addMinute();
        }
    }

    public void addMinute(){
        this.minutes++;
        minutes%=60;
        if(minutes == 0){
            addHour();
        }
    }

    public void addHour(){
        this.hours++;
    }

    public String getTime(){
        StringBuilder builder = new StringBuilder();
        builder.append(this.hours);
        builder.append(":");
        builder.append(this.minutes);
        builder.append(":");
        builder.append(this.seconds);
        return builder.toString();
    }
}
```

Pada class Controller, anda dapat menyiapkan beberapa metode yang dapat di-bind pada tampilan UI yang anda miliki. Tuliskan potongan kode dibawah ini pada class Controller.

```
package application;

import java.net.URL;
import java.util.ResourceBundle;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Label;

public class Controller {
    Clock clock;
    @FXML Label lblClock;

    public Controller(){
        clock = new Clock();
    }

    @FXML protected void handleBtnSecond(ActionEvent event){
        clock.addSecond();
        updateLabel();
    }

    @FXML protected void handleBtnMinute(ActionEvent event){
        clock.addMinute();
        updateLabel();
    }

    @FXML protected void handleBtnHour(ActionEvent event){
        clock.addHour();
        updateLabel();
    }

    private void updateLabel(){
        lblClock.setText(clock.getTime());
    }
}
```

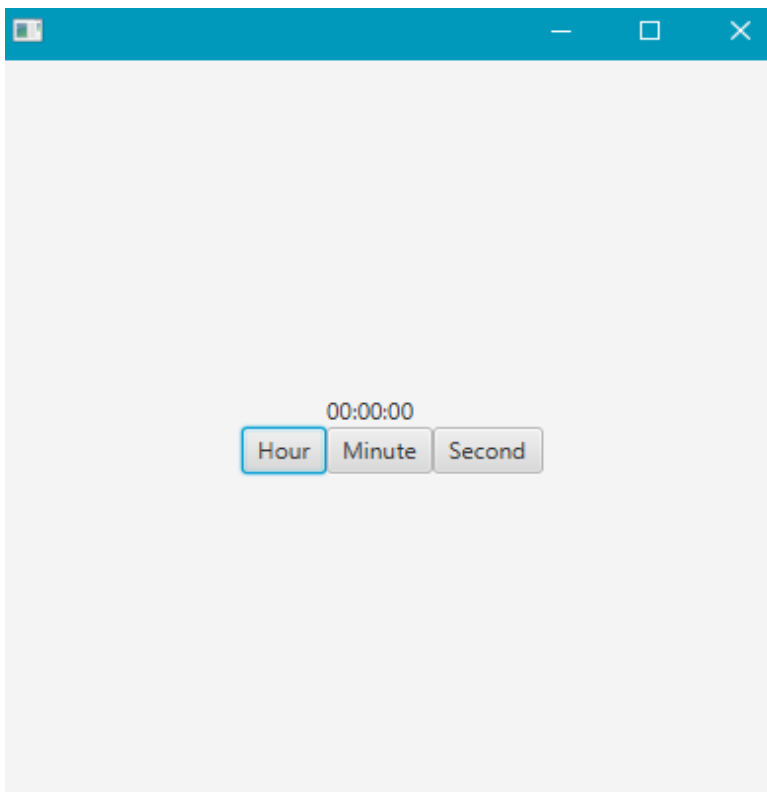
Pada JavaFX terdapat beberapa cara untuk menampilkan UI, salah satu cara yang dapat dilakukan adalah dengan menggunakan file dengan ekstensi fxml. Pada percobaan kali ini kita akan mencoba untuk menggunakan GridPane. Tuliskan potongan kode pada di bawah ini.

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.*?>
<?import java.lang.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>

<GridPane fx:controller="application.Controller"
    alignment="center" prefHeight="400.0" prefWidth="600.0"
    xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/8">
    <Label text="00:00:00" fx:id="lblClock"
        GridPane.columnIndex = "2" GridPane.rowIndex="1"/>
    <Button fx:id="btnHour" text="Hour"
        GridPane.columnIndex="1" GridPane.rowIndex="2"/>
    <Button fx:id="btnMinute" text="Minute"
        GridPane.columnIndex="2" GridPane.rowIndex="2"
        onAction="#handleBtnMinute"/>
    <Button fx:id="btnSecond" text="Second"
        GridPane.columnIndex="3" GridPane.rowIndex="2"
        onAction="#handleBtnSecond"/>
</GridPane>
```

Jalankan program dan tampilan program akan seperti pada gambar ini



## Tugas Pratikum

Pada class Item tuliskan potongan kode pada Gambar dibawah ini.

```
package application;

import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;

public class Item {
    private int id;
    private String productName;
    private int price;
    private String category;
    private int stock;

    public Item(int id, String productName, int price, String category, int stock) {
        super();
        this.id = id;
        this.productName = productName;
        this.price = price;
        this.category = category;
        this.stock = stock;
    }

    public int getId() { return id; }

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

    public String getProductName() { return productName; }

    public void setProductName(String productName) { this.productName = productName; }

    public int getPrice() { return price; }

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

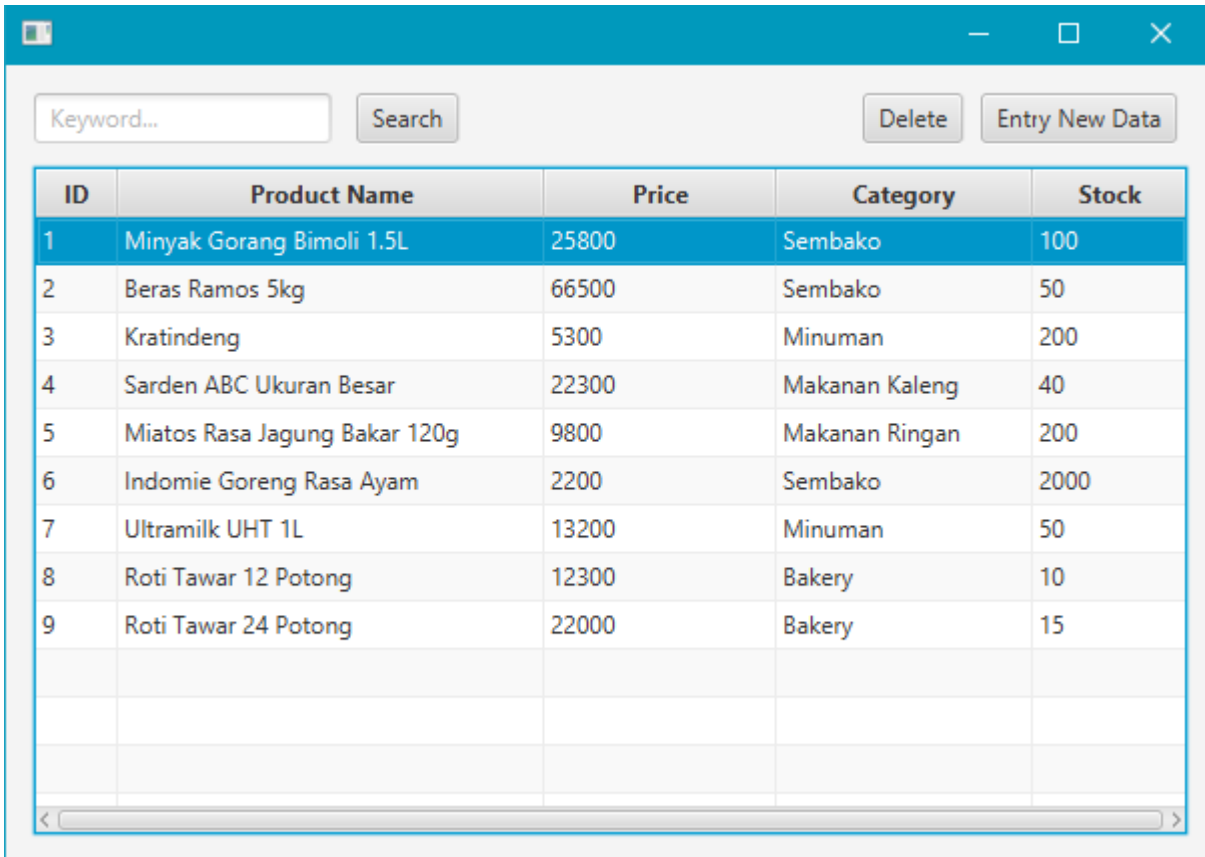
    public String getCategory() { return category; }

    public void setCategory(String category) { this.category = category; }

    public int getStock() { return stock; }

    public void setStock(int stock) { this.stock = stock; }
}
```

Bukalah file `itemList.fxml` pada SceneBuilder dan ubahlah tampilan dari `itemList.fxml` agar terlihat seperti pada gambar di bawah ini. Tampilan table dapat dibuat menggunakan TableView dan kolom pada TableView dapat ditambahkan dengan menggunakan TableColumn.



ID	Product Name	Price	Category	Stock
1	Minyak Gorang Bimoli 1.5L	25800	Sembako	100
2	Beras Ramos 5kg	66500	Sembako	50
3	Kratindeng	5300	Minuman	200
4	Sarden ABC Ukuran Besar	22300	Makanan Kaleng	40
5	Miatos Rasa Jagung Bakar 120g	9800	Makanan Ringan	200
6	Indomie Goreng Rasa Ayam	2200	Sembako	2000
7	Ultramilk UHT 1L	13200	Minuman	50
8	Roti Tawar 12 Potong	12300	Bakery	10
9	Roti Tawar 24 Potong	22000	Bakery	15

Setelah file `itemList.fxml` selesai dimodifikasi, save perubahan yang telah terjadi pada file dan set fx:id pada TextField `txtKeyword` dengan promptText `"Keyword..."`. Tambahkan juga fx:id `"tabellitem"` pada TableView dan `"colId"`, `"colProductName"`, `"colPrice"`, `"colCategory"`, dan `"colStock"` secara berturut-turut pada tiap TableColumn.

Buatlah file Controller seperti pada gambar dibawah ini.

```
package application;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.Stage;

public class Controller {

    @FXML private TableView<Item> tableItem;

    @FXML
    private TableColumn colId, colProductName, colPrice, colCategory, colStock;

    private ObservableList<Item> items = FXCollections.observableArrayList(
        new Item(1,"Minyak Gorang Bimoli 1.5L",25800,"Sembako",100),
        new Item(2,"Beras Ramos 5kg",66500,"Sembako",50),
        new Item(3,"Kratindeng",5300,"Minuman",200),
        new Item(4,"Sarden ABC Ukuran Besar",22300,"Makanan Kaleng",40),
        new Item(5,"Miatos Rasa Jagung Bakar 120g",9800,"Makanan Ringan",200),
        new Item(6,"Indomie Goreng Rasa Ayam",2200,"Sembako",2000),
        new Item(7,"Ultramilk UHT 1L",13200,"Minuman",50),
        new Item(8,"Roti Tawar 12 Potong",12300,"Bakery",10),
        new Item(9,"Roti Tawar 24 Potong",22000,"Bakery",15));

    public void initialize(){
        colId.setCellValueFactory(new PropertyValueFactory<Item,Integer>("id"));
        colProductName.setCellValueFactory(new PropertyValueFactory<Item,Integer>("productName"));
        colPrice.setCellValueFactory(new PropertyValueFactory<Item,Integer>("price"));
        colCategory.setCellValueFactory(new PropertyValueFactory<Item,Integer>("category"));
        colStock.setCellValueFactory(new PropertyValueFactory<Item,Integer>("stock"));
        tableItem.setItems(items);
    }

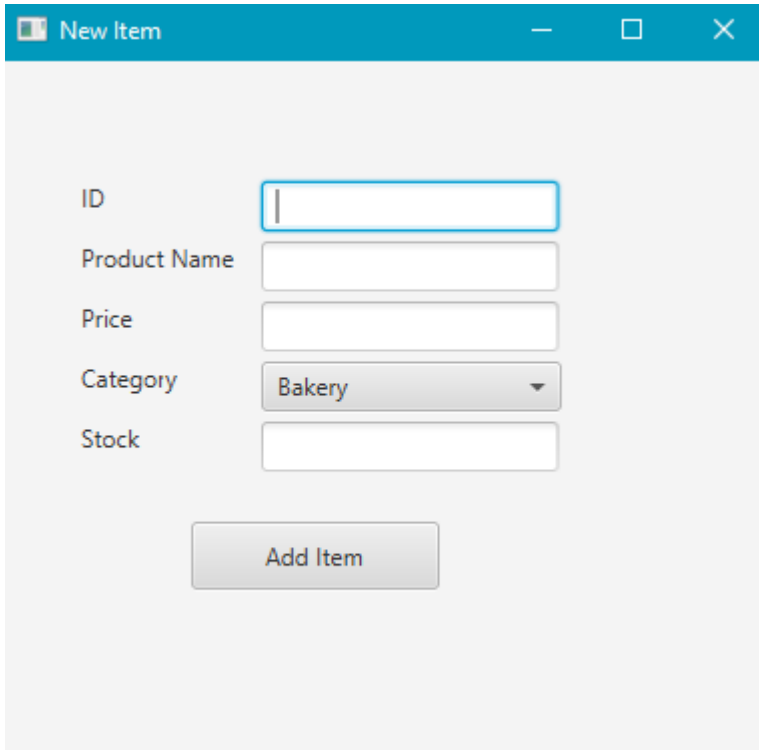
    @FXML void handleBtnEntry(){
        try{
            FXMLLoader loader = new FXMLLoader(getClass().getResource("entryForm.fxml"));
            Parent entryForm = loader.load();
            Stage entryStage = new Stage();
            entryStage.setTitle("New Item");
            entryStage.setScene(new Scene(entryForm, 384, 347));
            entryStage.show();
            entryStage.requestFocus();
        }catch(Exception ex){
            ex.printStackTrace();
        }
    }

    @FXML
    public void handleBtnDelete(){
        Item selectedItem = tableItem.getSelectionModel().getSelectedItem();
        items.remove(selectedItem);
    }
}
```



Pasangkan fungsi `handleBtnEntry` pada button dengan text "Entry New Data" dan juga fungsi `handleBtnDelete` pada button dengan text "Delete".

Buatlah sebuah file `entryForm.fxml` dan edit file menggunakan scene builder sehingga terlihat seperti pada gambar berikut.



Setelah membuat `entryForm.fxml`, ketiklah potongan kode berikut pada `entryForm.fxml` untuk mengisi pilihan pada ComboBox. Jangan lupa untuk mengimport `FXCollections` dengan kode `<? import javafx.collections.FXCollections ?>`

```
<ComboBox fx:id="cboCategory" layoutX="130.0" layoutY="150.0" prefWidth="150.0" >
  <items>
    <FXCollections fx:factory="observableArrayList">
      <String fx:value="Bakery"/>
      <String fx:value="Makanan Kaleng"/>
      <String fx:value="Makanan Ringan"/>
      <String fx:value="Makanan Instan"/>
      <String fx:value="Minuman"/>
      <String fx:value="Sembako"/>
    </FXCollections>
  </items>
  <value>
    <String fx:value="Bakery"/>
  </value>
</ComboBox>
```

Berikan juga fx:id pada setiap TextField dan ComboBox sesuai dengan caption miliknya dan tipe element (Contoh: txtProductName untuk TextField Product Name dan cboCategory untuk ComboBox Category)

Tambahkan sebuah Java class baru bernama InputFormController. Isi dari class ini dapat dilihat seperti pada gambar berikut.

```
package application;

import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.ComboBox;
import javafx.scene.control.TextField;

public class InputFormController {

    @FXML
    TextField txtId, txtProductName, txtPrice, txtStock;
    @FXML
    ComboBox<String> cboCategory;

    ObservableList<Item> items;

    @FXML
    public void handleBtnAdd(){
        int id = Integer.parseInt(txtId.getText());
        String productName = txtProductName.getText();
        int price = Integer.parseInt(txtPrice.getText());
        String category = cboCategory.getValue().toString();
        int stock = Integer.parseInt(txtStock.getText());
        Item i = new Item(id, productName, price, category, stock);
        items.add(i);
    }

    public void setListItem(ObservableList<Item> items){
        this.items = items;
    }
}
```

Setelah potongan kode di atas selesai dituliskan, update fungsi handleBtnEntry() agar terlihat seperti gambar berikut.

```
@FXML void handleBtnEntry(){
    try{
        FXMLLoader loader = new FXMLLoader(getClass().getResource("entryForm.fxml"));
        Parent entryForm = loader.load();
        Stage entryStage = new Stage();
        entryStage.setTitle("New Item");
        entryStage.setScene(new Scene(entryForm, 384, 347));
        entryStage.show();
        entryStage.requestFocus();
        InputFormController ec = loader.getController();
        ec.setListItem(items);
    }catch(Exception ex){
        ex.printStackTrace();
    }
}
```

## Tugas

### Tugas 1

Implementasikan fitur Search menggunakan txtKeyword dan button Search dimana list yang akan ditampilkan pada TableView hanyalah list yang memiliki nama yang mengandung keyword.

### Tugas 2

Pahami potongan kode pada pertemuan ini.

## REFERENSI

# MODUL 12

## (ACCORDION, TABS, TABLE)

### DESKRIPSI TEMA

Accordion, Tabs, dan Table.

### CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

Mahasiswa mempelajari membuat java program dengan UI dengan konsep OOP.

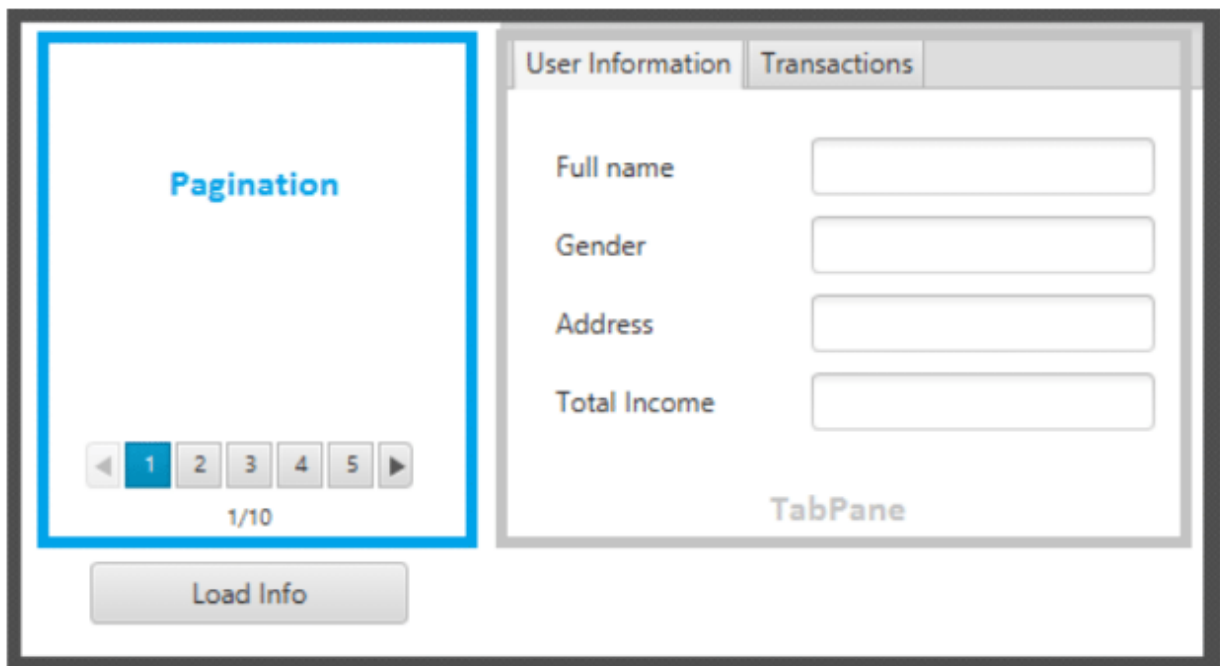
### PENUNJANG PRAKTIKUM

1. Eclipse.

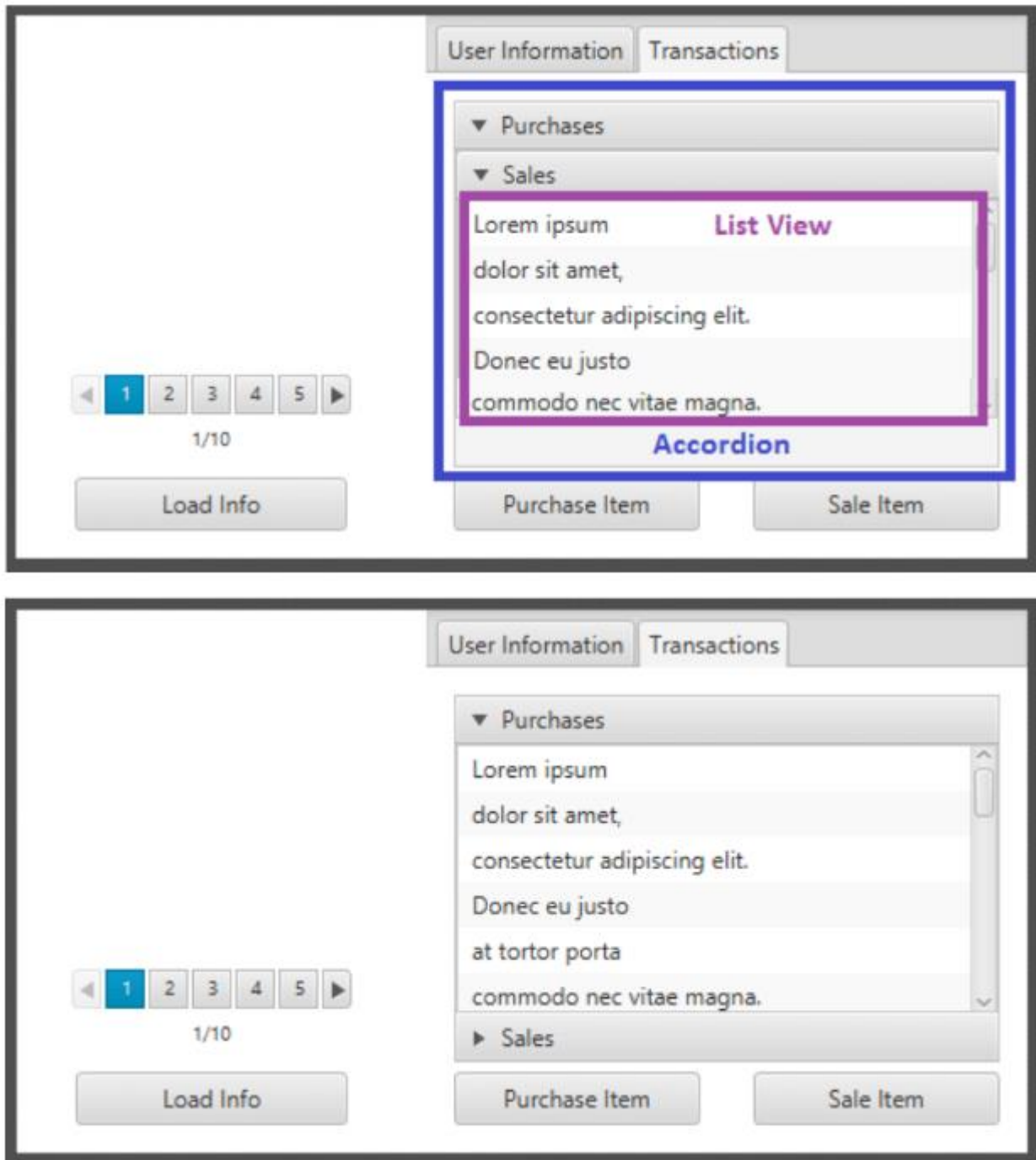
### LANGKAH-LANGKAH PRAKTIKUM

#### Tugas Praktikum

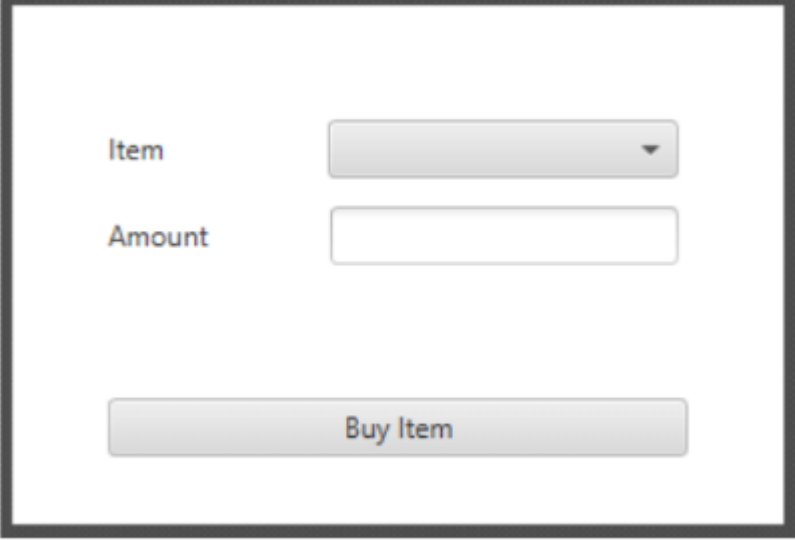
Buatlah tampilan UI 'main.fxml' agar terlihat seperti pada gambar di bawah ini. Hubungkan class 'Controller' ke 'main.fxml'.



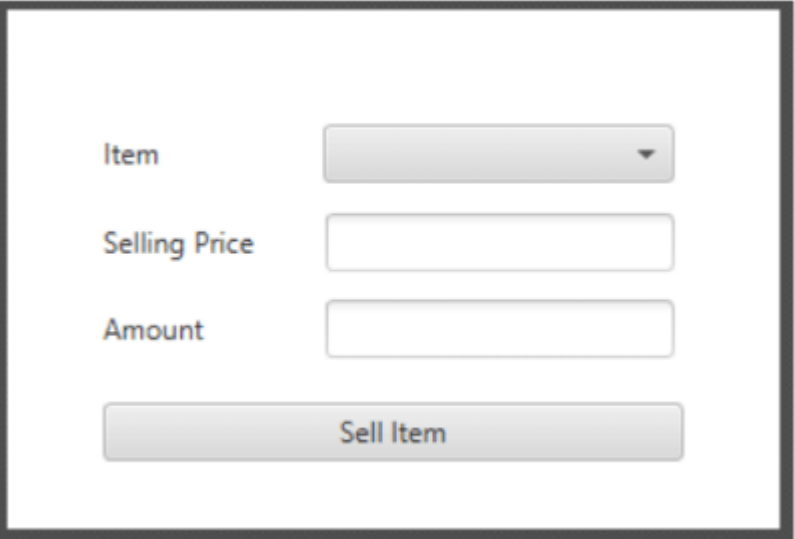
Detail tampilan pada tab kedia dilihat seperti pada gambar berikut.



Setelah tampilan 'main.fxml' selesai dibuat, buatlah dua buah form sederhana bernama 'purchaseForm.fxml' dan 'saleForm.fxml' yang dapat dilihat secara berturut pada gambar di bawah ini.



The Purchase Form UI consists of a rectangular frame containing three elements: a label 'Item' followed by a dropdown menu, a label 'Amount' followed by a text input field, and a 'Buy Item' button at the bottom.



The Sale Form UI consists of a rectangular frame containing four elements: a label 'Item' followed by a dropdown menu, a label 'Selling Price' followed by a text input field, a label 'Amount' followed by a text input field, and a 'Sell Item' button at the bottom.

Buatlah sebuah class bernama Item, isi dari class tersebut dapat dilihat seperti pada gambar di bawah ini.

```
package application;

public class Item {
    private String name;
    private Integer price;

    public Item(String name, Integer price) {
        this.name = name;
        this.price = price;
    }

    public String getName() { return name; }

    public Integer getPrice() { return price; }
}
```

Buatlah sebuah abstract class bernama 'Transaction' dan isi dari class tersebut dapat dilihat seperti pada gambar berikut.

```
package application;

public abstract class Transaction {
    protected Item item;
    protected Integer amount;

    public abstract Integer calculateTransaction();
    public abstract String getTransactionInfo();
}
```

Tambahkan class 'Purchase' dan class 'Sale' sebagai anak dari class 'Transaction'. Isi dari kedua kelas tersebut secara beruntun dapat dilihat pada gambar berikut.

```
package application;

public class Purchase extends Transaction{
    public Purchase(Item item, Integer amount){
        this.item = item;
        this.amount = amount;
    }
    @Override
    public Integer calculateTransaction() { return -item.getPrice()*amount; }

    @Override
    public String getTransactionInfo() { return item.getName()+"("+amount+")"; }
}
```

```
package application;

public class Sale extends Transaction{
    private Integer sellingPrice;
    public Sale(Item item, Integer amount, Integer sellingPrice){
        this.item = item;
        this.amount = amount;
        this.sellingPrice = sellingPrice;
    }
    @Override
    public Integer calculateTransaction() { return sellingPrice*amount; }

    @Override
    public String getTransactionInfo() {
        return item.getName()+"("+amount+" * "+sellingPrice+")";
    }
}
```



Setelah class 'Purchase' dan 'Sale' selesai dibuat, buatlah class 'User'. Class 'User' akan memiliki sebuah List<Transaction> yang digunakan untuk menyimpan informasi sederhana terkait dengan 'User' serta transaksi pembelian dan penjualan yang ia lakukan menggunakan tampilan aplikasi yang telah kita buat. Isi dari class 'User' dapat dilihat seperti pada gambar berikut.

```
package application;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.atomic.LongAdder;
import java.util.stream.Collectors;

public class User {
    private String firstName;
    private String lastName;
    private Character gender;
    private String address;

    private List<Transaction> transactions;

    public User(String firstName, String lastName, Character gender, String address) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.gender = gender;
        this.address = address;
        this.transactions = new ArrayList<>();
    }

    public String getFullName(){
        return this.firstName+" "+this.lastName;
    }

    public String getGender(){
        switch(gender){
            case 'M':return "Male";
            case 'F':return "Female";
            default: return "Other";
        }
    }

    public String getAddress(){
        return this.address;
    }

    public void addTransaction(Transaction transaction){
        transactions.add(transaction);
    }

    public List<Transaction> getPurchases(){
        return transactions
            .stream()
            .filter(transaction -> transaction instanceof Purchase)
            .collect(Collectors.toList());
    }
}
```

```

    public List<Transaction> getSales(){
        return transactions
            .stream()
            .filter(transaction -> transaction instanceof Sale)
            .collect(Collectors.toList());
    }

    public Long getIncome(){
        LongAdder income = new LongAdder();
        transactions.stream().forEach(t -> income.add(t.calculateTransaction()));
        return income.longValue();
    }
}

```

Tuliskan potongan kode di bawah ini ke dalam class 'Controller' dan hubungkan komponen di 'main.fxml' sesuai dengan variabel digunakan di dalam Controller.

```

package application;

import java.util.*;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.ListView;
import javafx.scene.control.Pagination;
import javafx.scene.control.TextField;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class Controller {
    @FXML
    public Pagination pagination;

    @FXML
    protected TextField txtFullName, txtGender, txtAddress, txtIncome;

    @FXML
    protected ListView<String> lvPurchase, lvSale;

    protected int contentPerPage = 7;

    protected String selectedName;
    protected User selectedUser;

    List<User> lstOfUser = new ArrayList<>();
    List<Item> lstOfItem = new ArrayList<>();
}

```

```

public void seedUser(){
    lstOfUser.add(new User("John","Doe",'M',"Dove Street"));
    lstOfUser.add(new User("Van","Doe",'M',"Crow Street"));
    lstOfUser.add(new User("Tom","Doe",'M',"Eagle Street"));
    lstOfUser.add(new User("Chuan","Doe",'M',"Dove Street"));
    lstOfUser.add(new User("Pan","Doe",'M',"Dove Street"));
    lstOfUser.add(new User("San","Doe",'M',"Dove Street"));
    lstOfUser.add(new User("Ran","Doe",'M',"Dove Street"));
    lstOfUser.add(new User("Sam","Doe",'M',"Dove Street"));
}

public void seedItem(){
    lstOfItem.add(new Item("Helm",125000));
    lstOfItem.add(new Item("Obeng",12000));
    lstOfItem.add(new Item("Spion",18000));
    lstOfItem.add(new Item("Oli",30000));
}

@FXML
public void initialize(){
    selectedName = "";
    seedUser();
    seedItem();
    double pageCount = (double) lstOfUser.size() / contentPerPage;
    pageCount = Math.ceil(pageCount);
    pagination.setPageCount((int)pageCount);
    /* Tanpa lambda expression
    pagination.setPageFactory(new Callback<Integer, Node>(){

        @Override
        public Node call(Integer param) {
            return createPage(param);
        }

    });
    */
    pagination.setPageFactory(param -> createPage(param));
}

public ListView<String> createPage(int pageIndex){
    ListView<String> lvUser = new ListView<>();
    /* Tanpa lambda expression
    lvUser.setOnMouseClicked(new EventHandler<MouseEvent>(){
        @Override
        public void handle(MouseEvent event) {
            selectedName = lvUser.getSelectionModel().getSelectedItem();
        }
    });
    */
    lvUser.setOnMouseClicked(event->
        selectedName = lvUser.getSelectionModel().getSelectedItem());
    int minIndex = pageIndex * contentPerPage;
    int maxIndex = pageIndex+1 * contentPerPage;

```

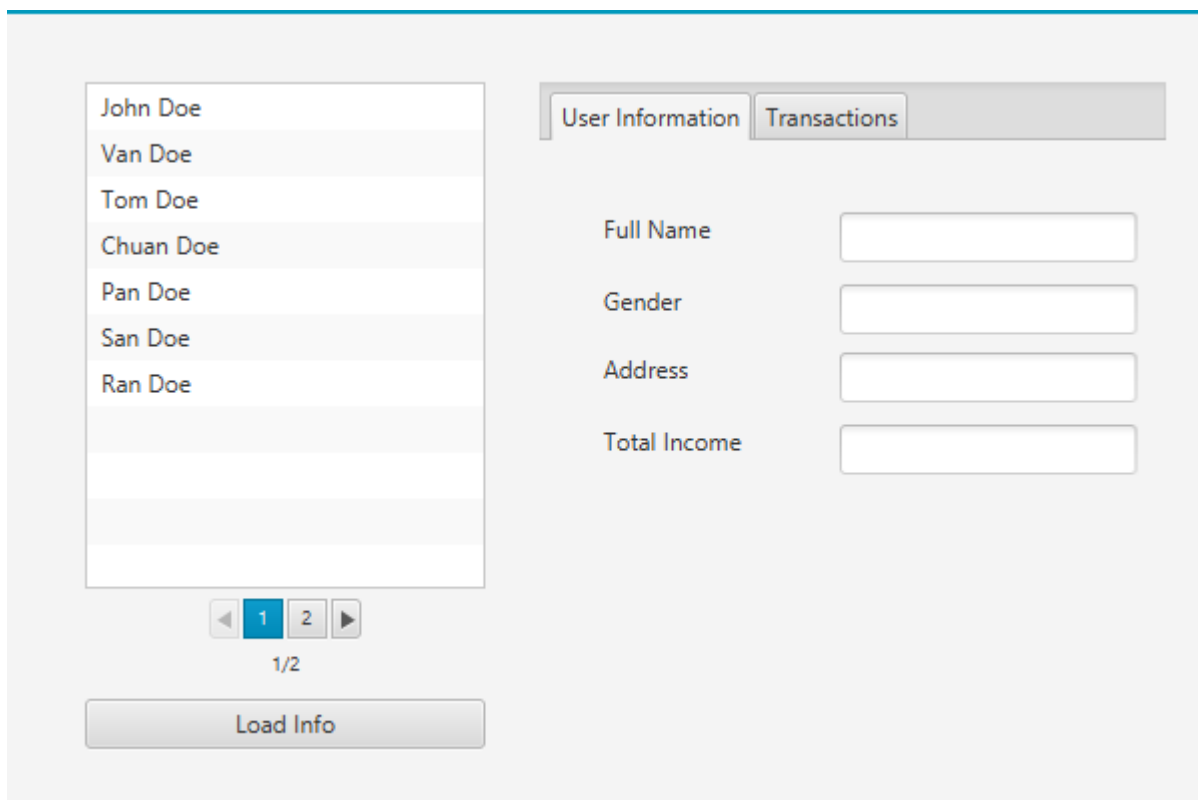
```

maxIndex = Math.min(maxIndex, lstOfUser.size());
LinkedList<String> lstOfNames =
    lstOfUser.subList(minIndex, maxIndex)
        .stream()
        .map(user->user.getFullName())
        .collect(Collectors.toCollection(LinkedList::new));

ObservableList<String> items = FXCollections.observableArrayList(lstOfNames);
lvUser.setItems(items);
return lvUser;
}

```

Jika potongan kode di atas telah dituliskan secara benar dan proses binding variabel pada Controller telah dilakukan, bagian “Pagination” dari form utama akan berisikan sebuah ListView yang menampilkan kumpulan nama dari user seperti pada gambar berikut.



Tambahkan juga potongan kode berikut pada class 'Controller'

```

public void handleLoadInfo(){
    try{
        Optional<User> userOptional =
            lstOfUser.stream()
                .filter(u -> u.getFullName().equals(selectedName))
                .findFirst();
        selectedUser = userOptional.get();
        txtFullName.setText(selectedUser.getFullName());
        txtGender.setText(selectedUser.getGender());
        txtAddress.setText(selectedUser.getAddress());
        refreshSaleListView();
        refreshPurchaseListView();
        refreshIncome();
    }catch(NoSuchElementException ex){
        System.out.println("No Selected element");
    }
}

public void refreshPurchaseListView(){
    lvPurchase.getItems().clear();
    for(Transaction p : selectedUser.getPurchases()){
        lvPurchase.getItems().add(p.getTransactionInfo());
    }
}

public void refreshSaleListView(){
    try{
        lvSale.getItems().clear();
        for(Transaction p : selectedUser.getSales()){
            lvSale.getItems().add(p.getTransactionInfo());
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}

public void refreshIncome(){
    txtIncome.setText(selectedUser.getIncome().toString());
}

```

Buatlah class 'PurchaseFormController' dan isi dari class tersebut dapat dilihat seperti pada gambar berikut.

```

package application;

import java.util.List;
import java.util.NoSuchElementException;

import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.ComboBox;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

```

```

public class PurchaseFormController {
    @FXML
    protected ComboBox<String> cboItem;

    @FXML
    protected TextField txtAmount;

    protected User user;
    protected List<Item> lstOfItem;
    public void setPurchaseFormData(User user, List<Item> lstOfItem){
        this.user = user;
        this.lstOfItem = lstOfItem;
        lstOfItem.stream().forEach(item -> cboItem.getItems().add(item.getName()));
    }

    public void addSale(){
        String title = "Warning Dialog";
        try{
            String selectedItemName = cboItem.getSelectionModel().getSelectedItem();
            Item item =
                lstOfItem
                .stream()
                .filter(i -> i.getName().equals(selectedItemName))
                .findFirst()
                .get();
            Integer amount = Integer.parseInt(txtAmount.getText());
            Purchase purchase = new Purchase(item, amount);
            user.addTransaction(purchase);
            Stage stage = (Stage)cboItem.getScene().getWindow();
            stage.close();
        }catch(NoSuchElementException e){
            String header = "No Selected item";
            String content = "Please select item from the item options";
            presentAlert(title,header,content);
        }catch(NumberFormatException e){
            String header = "Amount must be a number !";
            String content = "Please input number for amount field";
            presentAlert(title,header,content);
        }
    }

    private void presentAlert(String title, String header, String content){
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle(title);
        alert.setHeaderText(header);
        alert.setContentText(content);
        alert.show();
    }
}

```

Tambahkah juga class 'SaleFormController', isi dari class tersebut dapat dilihat seperti pada gambar di bawah ini.

```
public class SaleFormController {
    @FXML
    private ComboBox<String> cboItem;

    @FXML
    private TextField txtSellingPrice, txtAmount;

    private User user;
    private List<Item> lstOfItem;
    public void setSaleFormData(User user, List<Item> lstOfItem){
        this.user = user;
        this.lstOfItem = lstOfItem;
        lstOfItem.stream().forEach(item -> cboItem.getItems().add(item.getName()));
    }

    public void addSale(){
        String title = "Warning Dialog";
        try{
            String selectedItemName = cboItem.getSelectionModel().getSelectedItem();
            Item item =
                lstOfItem.stream()
                    .filter(i -> i.getName().equals(selectedItemName))
                    .findFirst()
                    .get();
            Integer amount = Integer.parseInt(txtAmount.getText());
            Integer price = Integer.parseInt(txtSellingPrice.getText());
            Sale sale = new Sale(item,amount,price);
            user.addTransaction(sale);
            Stage stage = (Stage)cboItem.getScene().getWindow();
            stage.close();
        }catch(NoSuchElementException e){
            String header = "No Selected item";
            String content = "Please select item from the item options";
            presentAlert(title,header,content);
        }catch(NumberFormatException e){
            String header = "Price and amount must be a number !";
            String content = "Please input number for price and amount field";
            presentAlert(title,header,content);
        }
    }

    private void presentAlert(String title, String header, String content){
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle(title);
        alert.setHeaderText(header);
        alert.setContentText(content);
        alert.show();
    }
}
```

Setelah kedua controller selesai dibuat, hubungkan 'PurchaseFormController' ke 'purchaseForm.fxml' dan 'SaleFormController' ke 'saleForm.fxml'. Jangan lupa untuk menghubungkan setiap variabel di tiap-tiap controller element yang bersangkutan di fxml-nya.

Langkah terakhir yang perlu kita lakukan hanyalah menghubungkan form 'main.fxml' ke 'purchaseForm.fxml' dan 'saleForm.fxml'. Tambahkan potongan kode di bawah ini pada class 'Controller' dan tambahkan kedua fungsi ke attribute onAction button 'Purchase item' dan 'Sale Item'.

```
public void handlePurchase(){
    if(selectedUser == null){
        String title = "Warning";
        String header = "No Selected user";
        String content = "Please select user from the list";
        presentAlert(title,header,content);
        return;
    }
    try{
        FXMLLoader loader = new FXMLLoader(getClass().getResource("purchaseForm.fxml"));
        Parent root = loader.load();
        Stage purchaseStage = new Stage();
        purchaseStage.initModality(Modality.APPLICATION_MODAL);
        purchaseStage.setTitle("Purhcase");
        purchaseStage.setScene(new Scene(root,333,222));

        PurchaseFormController controller = loader.getController();
        controller.setPurchaseFormData(selectedUser, lstOfItem);
        purchaseStage.showAndWait();
        refreshPurchaseListView();
        refreshIncome();
    }catch(Exception e){
        System.out.println("Failed to open form!");
    }
}
```



```

public void handleSale(){
    if(selectedUser == null){
        String title = "Warning";
        String header = "No Selected user";
        String content = "Please select user from the list";
        showAlert(title,header,content);
        return;
    }
    try{
        FXMLLoader loader = new FXMLLoader(getClass().getResource("saleForm.fxml"));
        Parent root = loader.load();
        Stage purchaseStage = new Stage();
        purchaseStage.initModality(Modality.APPLICATION_MODAL);
        purchaseStage.setTitle("Purhcase");
        purchaseStage.setScene(new Scene(root,333,222));

        SaleFormController controller = loader.getController();
        controller.setSaleFormData(selectedUser, lstOfItem);
        purchaseStage.showAndWait();
        refreshSaleListView();
        refreshIncome();
    }catch(Exception e){
        e.printStackTrace();
        System.out.println("Failed to open form!");
    }
}

private void showAlert(String title, String header,String content){
    Alert alert = new Alert(Alert.AlertType.WARNING);
    alert.setTitle(title);
    alert.setHeaderText(header);
    alert.setContentText(content);
    alert.show();
}

```

## Tugas Mandiri

### Tugas 1

Tugas anda hanyalah untuk memahami potongan kode di atas dan menanyakan potongan kode yang belum anda mengerti sebagai bentuk persiapan UAS Pratikum!

### Tugas 2

Kerjakan tugas akhir.

## REFERENSI