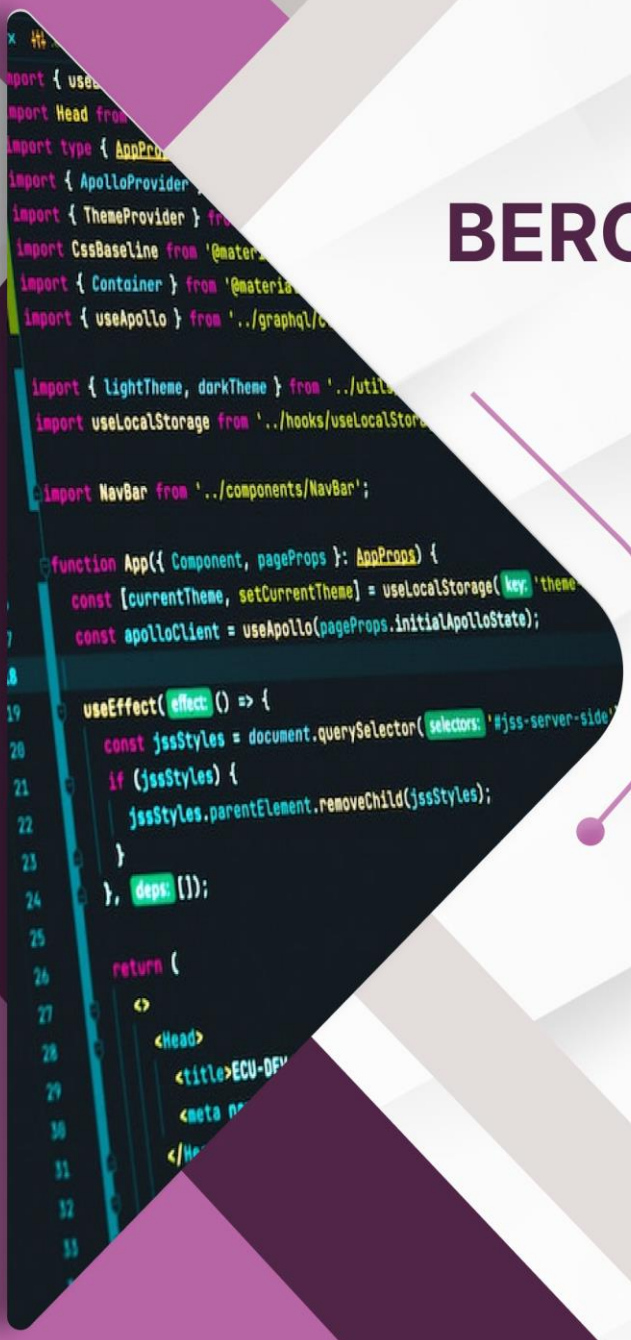
FIKOM

MODUL PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK



2023/2024

Tim Penyusun

- Mardiyyah Hasnawi, S.Kom., M.T., MTA
- Syariful Mujaddid, S.Kom., M.T
- Lutfi Budi Ilmawan, S.Kom., M.Cs., MTA
- Amaliah Faradibah, S.Kom., M.Kom., MTA., MCF
- Ir. Abdul Rachman Manga, S.Kom., M.T., MTA., MCF
- Ir. Huzain Azis, S.Kom., M.Cs., MTA
- Tim Asisten Laboratorium

Laboratorium Terpadu
Fakultas Ilmu Komputer
Universitas Muslim Indonesia

KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Kuasa, yang telah memberikan rahmat-Nya sehingga Modul Praktikum **Pemrograman Berorientasi Objek** untuk mahasiswa/i Program Studi Teknik Informatika dan Sistem Informasi Fakultas Ilmu Komputer Universitas Muslim Indonesia ini dapat diselesaikan dengan sebaik-baiknya.

Modul praktikum ini dibuat sebagai pedoman dalam melakukan kegiatan praktikum **Pemrograman Berorientasi Objek** yang merupakan kegiatan penunjang mata kuliah pada Program Studi Teknik Informatika dan Sistem Informasi. Modul praktikum ini diharapkan dapat membantu mahasiswa/i dalam mempersiapkan dan melaksanakan praktikum dengan lebih baik, terarah, dan terencana. Pada setiap topik telah ditetapkan capaian pembelajaran mata kuliah pelaksanaan praktikum dan semua kegiatan yang harus dilakukan oleh mahasiswa/i serta teori singkat untuk memperdalam pemahaman mahasiswa/i mengenai materi yang dibahas.

Penyusun menyakini bahwa dalam pembuatan Modul Praktikum **Pemrograman Berorientasi Objek** ini masih jauh dari sempurna. Oleh karena itu penyusun mengharapkan kritik dan saran yang membangun guna penyempurnaan modul praktikum ini dimasa yang akan datang.

Akhir kata, penyusun mengucapkan banyak terima kasih kepada semua pihak yang telah membantu baik secara langsung maupun tidak langsung.

Makassar, Maret 2024

Tim Penyusun

TATA TERTIB PELAKSANAAN PRAKTIKUM

Tata Tertib Pelaksanaan Praktikum pada Laboratorium Terpadu Fakultas Ilmu Komputer UMI adalah sebagai berikut:

1. Seluruh Pengguna laboratorium harus dalam keadaan sehat tidak menunjukkan gejala sakit (batuk, hidung tersumbat, dan suhu badan diatas 37°C).
2. Praktikan hanya diizinkan melaksanakan praktikum apabila :
   a. Pria
      - Berpakaian rapi memakai kemeja putih polos;
      - Menggunakan celana kain berwarna hitam bukan dari bahan jeans/semi jeans;
      - Rambut rapi dan tidak panjang;
   b. Wanita
      - Berpakaian rapi memakai kemeja tunik putih polos (tidak transparan)
      - Memakai Jilbab Segitiga Hitam (bukan pasmina) dan menutupi dada.
      - Menggunakan Rok Panjang berwarna hitam yang tidak terbelah dan tidak span serta bukan dari bahan jeans/semi jeans;
      - Memakai kaos kaki dengan tinggi minimal 10 cm di atas mata kaki;
3. Ketika memasuki dan selama berada dalam ruangan, praktikan diwajibkan :
   - Tenang, tertib, dan sopan;
   - Tidak mengganggu praktikan lain yang sedang melaksanakan praktikum;
   - Tidak diperbolehkan merokok, membawa makanan / minuman senjata tajam dan senjata api ke dalam ruangan praktikum;
   - Tidak diperbolehkan membawa *handphone* ke meja praktikum dan *handphone* dalam mode senyap;
   - Tidak diperbolehkan membawa media penyimpanan eksternal atau *flashdisk* ke meja praktikum tanpa seizin Dosen Pengampu atau Asisten;
4. Dilarang membawa, mengambil, serta memindahkan perangkat yang digunakan pada saat praktikum tanpa instruksi dari Dosen Pengampu atau Asisten.
5. Toleransi keterlambatan praktikan maksimal 5 menit.
6. Praktikan berada diarea laboratorium dengan mengikuti jadwal yang telah ditentukan oleh Kepala Laboratorium.
7. Penggunaan fasilitas Laboratorium menyesuaikan dengan kapasitas ruang Laboratorium.

8. Segala pelanggaran yang dilakukan oleh praktikan akan berakibat pada penutupan dan penghentian penggunaan seluruh fasilitas laboratorium dan ditindak sesuai dengan aturan yang berlaku.

SANKSI-SANKSI

Sanksi terhadap pelanggaran **TATA TERTIB**:

Dosen Pengampu dan Asisten laboratorium berhak menjatuhkan sanksi, sesuai dengan aturan yang berlaku di Laboratorium Terpadu Fakultas Ilmu Komputer UMI apabila :

1. Praktikan merusak peralatan praktikum (*Personal Computer*) secara sengaja, maka praktikan bertanggung jawab untuk mengganti kerusakan tersebut.
2. Praktikan tidak mematuhi dan mentaati aturan praktikum maka tidak diperkenankan mengikuti praktikum.

Pelanggaran point lainnya dikenakan sanksi teguran, dikeluarkan/dicoret namanya dalam kegiatan praktikum (mengulang mata kuliah sesuai dengan semester berjalan) sampai sanksi akademik.

Kepala Laboratorium Terpadu,

Ir. Abdul Rachman Manga', S.Kom., M.T., MTA., MCF

DAFTAR ISI

CAPAIAN PEMBELAJARAN MATA KULIAH (CPMK)

1. CPMK1 : Mahasiswa mampu memahami konsep pemrograman berorientasi objek dengan menggunakan bahasa pemrograman Java sebagai pengantar.
2. CPMK2 : Mahasiswa mampu dapat merancang class sesuai konsep pemrograman berorientasi objek
3. CPMK3 : Mahasiswa mampu membuat program berdasarkan konsep pemrograman berorientasi objek menggunakan bahasa pemrograman Java.

MODUL 1 – SYNTAX DASAR JAVA

A. Sub Capaian Pembelajaran Mata Kuliah (CPMK)

1. Mahasiswa dapat memahami syntax java dasar dan mempu mengimplementasikannya
2. Mahasiswa dapat memahami pengkondisian dan perulangan dan dapat mengimplementasikannya menggunakan bahasa pemprograman java

B. Instrument dan Prosedur

1. Instrument

a) Perangkat komputer
b) Sistem Operasi Windows
c) *Java Development KIT* (JDK)
d) IDE Netbeans

2. Prosedur

a) Baca dan pahami semua tahapan praktikum dengan cermat.
b) Gunakan fasilitas yang disediakan dengan penuh rasa tanggung jawab.
c) Rapihkan kembali setelah menggunakan komputer (mouse, keyboard, kursi, dll)
d) Perhatikan sikap anda untuk tidak mengganggu rekan praktikan lain.
e) Pastikan diri anda tidak menyentuh sumber listrik.

C. Tugas Pendahuluan

1. Jelaskan apa yang dimaksud dengan tipe data primitf dan tipe data referensi. Tuliskan tipe data yang termasuk tipe data primitif dan tipe data referensi
2. Berikan contoh penginisialisasian tipe data dalam Java. Minimal 5 contoh untuk tipe data primitif dan 1 contoh tipe data referensi
3. Sebutkan apa perbedaan pengkondisian *if* dan *switch* didalam Java. Bagaiman pengimplementasian didalam program?
4. Sebutkan apa perbedaan *for loop, while loop,* dan *do while loop*. Bagaiman pengimplementasian didalam program?
5. Berikan contoh program input dan output sederhana didalam Java!

D. Teori Dasar

1. Tipe Data

Tipe data adalah konsep fundamental dalam bahasa Pemrograman Java yang digunakan untuk menyimpan dan memanipulasi nilai dalam program. Dalam Java, terdapat dua jenis tipe data yaitu tipe data primitif dan tipe data referensi.

a. Tipe Data Primitif

Tipe data primitif adalah tipe data dasar dalam Java yang direpresentasikan oleh nilai tunggal. Tipe data primitif dikelompokkan menjadi beberapa kategori yaitu:

1. Tipe data numerik*: byte, short, int, long, float, double.*
2. Tipe data karakter*: char.*
3. Tipe data pengkondisian: *boolean.*

Deklarasi tipe data primitif:

```
int a = 10;
char b = 'B';
boolean benar = true;
```

b. Tipe Data Referensi

Tipe data referensi adalah tipe data yang mereferensikan objek yang dibuat dari class tertentu. Contoh dari tipe data referensi adalah *String* dan *Array*

1. *String*

String merupakan tipe data yang digunakan untuk merepresentasikan teks atau karakter. *String* dalam Java sebenarnya adalah *Object* yang diinisialisasi dari kelas '*String*'.

Contoh penginisialisasian *String*

```
String name = "Nasrullah";
```

2. PENGKONDISIAN

a. Pengkondisian *If*

If adalah kondisi yang digunaan untuk memeriksa kondisi tertentu dan melakukan aksi jika kondisi tersebut benar.

Bentuk umum if:

```
if (kondisi){
    // statement
}
```

Contoh :

```
int nilai = 50;
int nilai2 = 40;
int hasil = nilai + nilai2;

if (hasil >= 85){
   System.out.println(x: "SELAMAT ANDA LULUS DENGAN NILAI A" );
}else if (hasil >= 75 && hasil<= 84){
    System.out.println(x: "SELAMAT ANDA LULUS DENGAN NILAI B");
}else {
    System.out.println(x: "MAAF ANDA TIDAK LULUS");
}
```

b. Pengkondisian *Switch*

Switch adalah sebuah kondisi yang digunakan untuk memeriksa nilai tertentu dan melakukan aksi berbeda berdasarkan nilai tersebut"

Bentuk umum *switch*:

```
switch(kondisi){
    case 1 :
        // statement
}
```

Contoh:

```
int kondisi = 3;

switch(kondisi){
    case 1 :
        System.out.println(x: "kondisi 1");
        break;
    case 2 :
        System.out.println(x: "kondisi 2");
        break;
    default :
        System.out.println(x: "kondisi tidak tersedia");
        break;
}
```

3. PERULANGAN

Perulangan atau looping merupakan salah satu konsep penting dalam pemprograman, termasuk dalam pemprograman Java. Perulangan digunakan untuk melakukan eksekusi satu atau lebih sebuah perintah secara berulang-ulang sampai dengan kondisi yang kita inginkan tercapai. Java memiliki beberapa jenis perulangan yaitu:

1. Perulangan *While*

   Perulangan *while* digunakan untuk melakukan eksekusi terhadap satu atau lebih perintah selama kondisi perulangan masih bernilai *true* dan akan berhenti jika kondisi perulangan bernilai *false*;

   Bentuk umum perulangan while:

```
while (kondisi){
    // statement
}
```

   Contoh:

```
int i = 1;
while (i <= 5){
    System.out.println(x: i);
    i++;
}
```

2. Perulangan *do-while*

   Sama halnya dengan perulangan *while*, namun yang membedakan antara keduanya adalah *do-while* akan mengeksekusi terlebih dahulu perintah dalam blok perulangan lalu memeriksa kondisi dari perulangan tersebut. Perulangan *do-while* akan berhenti jika kondisi yang diberikan telah bernilai *false*;

   Bentuk umum perulangan *do-while*:

```
do {
    // statement
}while (kondisi);
```

Contoh :

```
int i = 1;
do {
    System.out.print(i);
    i++;
}while(i <= 5);
```

3. Perulangan *For*

   Perulangan *for* digunakan untuk mengeksekusi satu atau lebih perintah dalam blok *for* secara berulang-ulang. Perulangan *for* biasanya digunakan untuk mengulang beberapa perintah dengan jumlah iterasi yang telah ditentukan.

   Bentuk umum

```
for (int k = 0; k <= banyakIterasi; k++){
    // statement
}
```

   Contoh

```
// banyak perulangan yang akan dilakukan
int banyakIterasi = 10;

for (int k = 1; k <= banyakIterasi; k++){
    System.out.println("iterasi ke-" + k);
}
```

4. Perulangan *For-Each*

   Perulangan *for-each* digunakan untuk mengulang sejumlah kode program pada semua elemen yang ada didalam sebua *array*. Perulangan *for-each* hanya bisa digunakan untuk *object* yang mengimplementasikan *interfase iterable*.

   Bentuk umum

```
Integer[] banyakIterasi = new Integer[10];

for(var value : banyakIterasi){
    // statement
}
```

   Contoh

```
Integer[] banyakIterasi = {
  54,43,64,23,64,234
};

for(var value : banyakIterasi){
    System.out.println(x: value);
}
```

E. Kegiatan Praktikum

1. Studi Kasus A

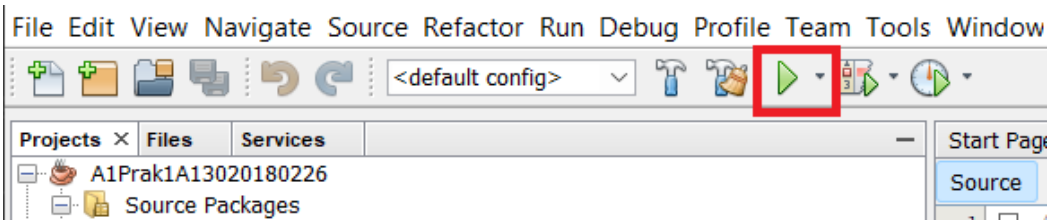
a) Buka Aplikasi *Netbeans*, buat *project* baru dengan nama sesuai dengan format berikut :

KelasPrak1ASTB. Contoh: A1Prak1A13020180226

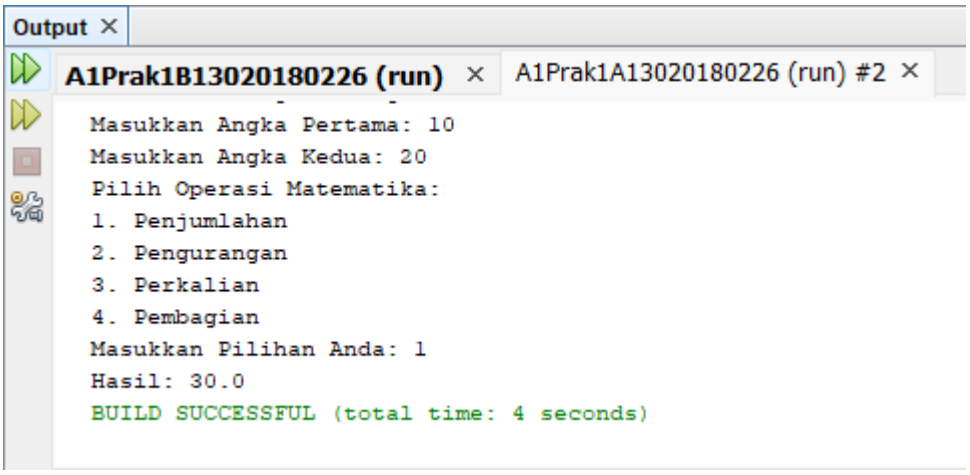
b) Pada *Function* Main dalam *Class* **A1Prak1A13020180226** tuliskan kodingan sebagai berikut :

```
package a1prak1a13020180226;

import java.util.Scanner;

public class A1Prak1A13020180226 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.println("Selamat Datang di Program Kalkulator
Sederhana");
        System.out.print("Masukkan Angka Pertama: ");
        double angka1 = input.nextDouble();

        System.out.print("Masukkan Angka Kedua: ");
        double angka2 = input.nextDouble();

        System.out.println("Pilih Operasi Matematika: ");
        System.out.println("1. Penjumlahan");
        System.out.println("2. Pengurangan");
        System.out.println("3. Perkalian");
        System.out.println("4. Pembagian");

        System.out.print("Masukkan Pilihan Anda: ");
        int pilihan = input.nextInt();

        double hasil = 0;

        switch (pilihan) {
            case 1:
                hasil = angka1 + angka2;
                break;

            case 2:
                hasil = angka1 - angka2;
                break;

            case 3:
                hasil = angka1 * angka2;
                break;

            case 4:
                hasil = angka1 / angka2;
                break;

            default:
                System.out.println("Operasi Matematika Tidak Tersedia");
                break;
        }

        System.out.println("Hasil: " + hasil);

    }

}
```

a. Selanjutnya tekan tombol *run* di bagian atas yang berbentuk tombol *play* berwarna hijau seperti gambar di bawah atau dengan menekan *shortcut* (*Shift*+f6) untuk melakukan *compile* dan *run*.



b. Setelah itu ketika program berjalan tanpa ada *error* maka akan tampil *output* seperti gambar dibawah ini:



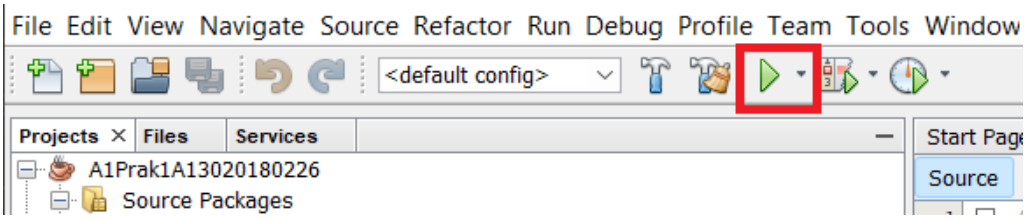
2. Studi Kasus B

a) Buka Aplikasi *Netbeans*, buat *project* baru dengan nama sesuai dengan format berikut :

KelasPrak1BSTB. Contoh: A1Prak1B13020180226

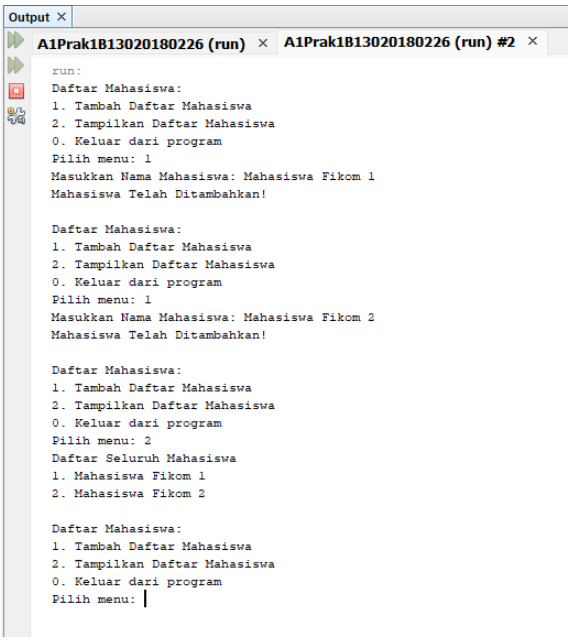
b) Pada *Function* Main dalam *Class* **A1Prak1A13020180226** tuliskan kodingan sebagai berikut :

c) Selanjutnya tekan tombol *run* di bagian atas yang berbentuk tombol *play* berwarna hijau seperti gambar di bawah atau dengan menekan *shortcut* (*Shift*+f6) untuk melakukan *compile* dan *run*.



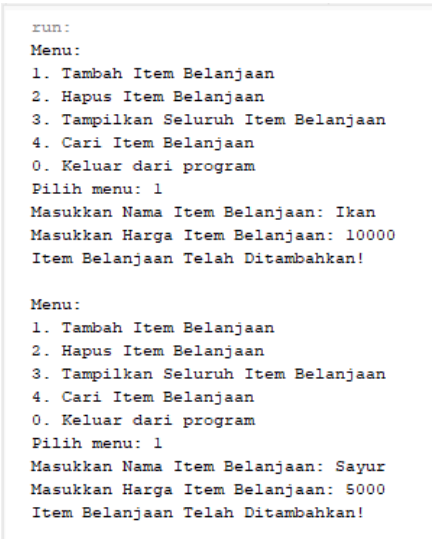
d) Setelah itu ketika program berjalan tanpa ada *error* maka akan tampil *output* seperti gambar di bawah ini :

```
package com.mycompany.alprak1b13020180226;

import java.util.ArrayList;
import java.util.Scanner;

public class AlPrak1B13020180226 {

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

        ArrayList<String> mahasiswa = new ArrayList<>();

        double total = 0;
        int pilihan;

        do{
            System.out.println("Daftar Mahasiswa :");
            System.out.println("1. Tambah Daftar Mahasiswa");
            System.out.println("2. Tampilkan Daftar Mahasiswa");
            System.out.println("0. Keluar dari Program");

            System.out.print("Pilih Menu : ");
            pilihan = input.nextInt();
            input.nextLine();

            switch (pilihan){
                case 1 :
                    System.out.println("Masukkan Nama Mahasiswa : ");
                    String mhs = input.nextLine();

                    mahasiswa.add(mhs);

                    System.out.println("Mahasiswa Telah DiTembahkan");
                    break;
                case 2 :
                    System.out.println("Daftar Seluruh Mahasiswa");
                    for (var value : mahasiswa){
                        System.out.println("Nama : " + value);
                    }
                    break;
                case 0 :
                    break;
                default :
                    System.out.println("Menu Tidak Tersedia");
                    break;
            }
            System.out.println();
        }while (pilihan != 0);
    }
}
```



LEMBAR EVALUASI PRAKTIKUM

1. Analisis program pada Studi Kasus B serta jelaskan cara kerja program tersebut?
2. Jelaskan penggunaan *ArrayList* pada program?
3. Jelaskan mengapa pada program menggunakan *do-while* bukannya *while*?
4. Buatkan satu program JAVA menggunakan *netbeans* untuk menghasilkan *output* seperti gambar di bawah :
   Testing ketika *user* memilih menu



   Testing ketika user memilih menu 2

```
Menu:
1. Tambah Item Belanjaan
2. Hapus Item Belanjaan
3. Tampilkan Seluruh Item Belanjaan
4. Cari Item Belanjaan
0. Keluar dari program
Pilih menu: 2
Masukkan Nama Item Belanjaan yang Akan Dihapus: Ubi
Tidak ada item yang di hapus harap masukkan nama item yang sesuai!!!

Menu:
1. Tambah Item Belanjaan
2. Hapus Item Belanjaan
3. Tampilkan Seluruh Item Belanjaan
4. Cari Item Belanjaan
0. Keluar dari program
Pilih menu: 2
Masukkan Nama Item Belanjaan yang Akan Dihapus: Ikan
Item Belanjaan Telah Dihapus!
```

Testing ketika user memiliih menu 3:

```
Menu:
1. Tambah Item Belanjaan
2. Hapus Item Belanjaan
3. Tampilkan Seluruh Item Belanjaan
4. Cari Item Belanjaan
0. Keluar dari program
Pilih menu: 3
Seluruh Item
=============
Ikan - Rp10000.0
Sayur - Rp5000.0
Total Harga: Rp15000.0
```

Testing ketika user memilih menu 4:

```
Menu:
1. Tambah Item Belanjaan
2. Hapus Item Belanjaan
3. Tampilkan Seluruh Item Belanjaan
4. Cari Item Belanjaan
0. Keluar dari program
Pilih menu: 4
Masukkan Nama Item Belanjaan yang Dicari: Ikan
Item Belanjaan Tidak Ditemukan!

Menu:
1. Tambah Item Belanjaan
2. Hapus Item Belanjaan
3. Tampilkan Seluruh Item Belanjaan
4. Cari Item Belanjaan
0. Keluar dari program
Pilih menu: 4
Masukkan Nama Item Belanjaan yang Dicari: Sayur
Sayur - Rp5000.0
```

Testing ketika user memilih menu 0:

```
Menu:
1. Tambah Item Belanjaan
2. Hapus Item Belanjaan
3. Tampilkan Seluruh Item Belanjaan
4. Cari Item Belanjaan
0. Keluar dari program
Pilih menu: 0
```

Evaluasi Praktikum 1:

No	Indikator	Skor Penilaian				
		Sangat Kurang (E) <=40	Kurang (D) 41-55	Cukup (C) 55-65	Baik (B) 66-85	Sangat Baik (A) >=86
1.	Pemahaman Tenang *Syntax* Dasar Java					
2.	Pemahaman Tentang Tipe Data					
3.	Pemahaman Tentang Pengkondisian					
4.	Pemahaman Perulangan					

Catatan Asisten:

Asisten 1 :_____

Asisten 2 : _____

MODUL 2 – ARRAY DAN METHOD

A. Sub Capaian Pembelajaran Mata Kuliah (CPMK)

Mahasiswa mampu menggunakan struktur data statis (*array*) dan dinamis dalam bahasa pemrograman berorientasi objek

B. *Instrument* dan Prosedur

1. Instrument

a) Perangkat computer
b) Sistem Operasi Windows
c) *Java Development KIT* (JDK)
d) IDE Netbeans

2. Prosedur

a) Baca dan pahami semua tahapan praktikum dengan cermat.
b) Gunakan fasilitas yang disediakan dengan penuh rasa tanggung jawab.
c) Rapihkan kembali setelah menggunakan komputer *(mouse, keyboard*, kursi, dll)
d) Perhatikan sikap anda untuk tidak mengganggu rekan praktikan lain.
e) Pastikan diri anda tidak menyentuh sumber listrik.

C. Tugas Pendahuluan

1. Bagaimana cara mendeklarasikan sebuah array dan mengakses elemen di dalam array?
2. Buatlah sebuah sebuah array dengan panjang array 5 dan tampilkan seluruh elemen array?
3. Apa perbedaan antara method void dan method non-void di Java? Bagaimana cara mengembalikan nilai dari sebuah method?

D. Teori Dasar

1. Array

Array adalah kumpulan elemen-elemen data dengan tipe data yang sama, yang diatur dalam urutan tertentu dan dapat diakses menggunakan indeks atau nomor urut.

Bentuk umum :

```
//bentuk umum array
tipeData[] namaArray = new tipeData[ukuran_array];
```

Contoh array :

```
String[] fakultas = new String[2];

fakultas[0] = "Ilmu komputer"

fakultas[1] = "Agama"
```

Pada penginisialisasian *array* diatas kita langsung menentukan berapa panjang sebuah *array* dan penginisialisasian fakultas[0] menunjukan elemen *array* ke berapa yang akan diisikan.

```
String[] fakultas = {"Ilmu komputer", "Agama"}
```

Contoh diatas adalah penginisialisasian *array*, namun yang membedakan adalah Jika contoh pertama kita perlu menentukan panjang *array* dan perlu inisialisasi satu persatu, namun dengan contoh kedua kita dapat langsung menginisialisasian value yang ada didalam array tersebut.

2. ArrayList

ArrayList merupakan salah satu struktur data pada bahasa Pemrograman yang digunakan untuk menyimpan kumpulan data dengan tipe yang sama secara dinamis. Artinya, ukuran *ArrayList* dapat berubah-ubah sesuai dengan kebutuhan dan jumlah data yang ingin disimpan. Untuk menggunakan *ArrayList* kita perlu memanggil class nya yaitu *ArrayList*

Bentuk umum :

```
ArrayList<tipeData> namaObject = new ArrayList<>();
```

Contoh implementasi dari class ArrayList :

```
ArrayList<String> fakultas = new ArrayList<>();

fakultas.add("Ilmu komputer");
```

contoh diatas adalah memasukan value ke dalam *arraylist* dengan menggunakan method bawaan dari class *arraylist* yaitu add

3. Method

Method adalah sebuah blok kode didalam program yang terdiri dari serangkaian dan perintah tertentu yang dirancang untuk melakukan tugas tertentu. Sebuah *method* dapat menerima input dari parameter dan dapat juga mengembalikan *output* dalam bentuk

nilai pengembalian.

Bentuk umum *method* yang tidak berparameter :

```
Acces_Modifier jenisMethod namaMethod( ){
  //statement
}
```

Contoh bentuk *method* tidak berparameter :

```
public void setName( ){
  System.out.print("Halo, selamat datang di lab")
}
```

Bentuk umum *method* yang menggunakan parameter :

```
Acces_Modifier jenisMethod namaMethod(tipeData namaVariable){
  //statement
}
```

Contoh bentuk *method* yang menggunakan parameter :

```
public void setName(String name){
  System.out.print("Halo, selamat datang " + name)
}
```

Bentuk umum *method overloading :*

```
// bentuk umum overloading method
access_modifier jenisMethod nameMethod(){
  //statment
}
access_modifier jenisMethod nameMetod(tipeData parameter){
  //statment
}
access_modifier jenisMethod nameMethod(tipeData parameter,tipeData parameter){
  //statment
}
```

Contoh *method overloading :*

```
// bentuk umum overloading method
public void setName(){
  //statment
}
public void setName(String name){
  //statment
}
public void setName(String name,String name2){
  //statment
}
```

E. Kegiatan Praktikum

1. Studi Kasus A

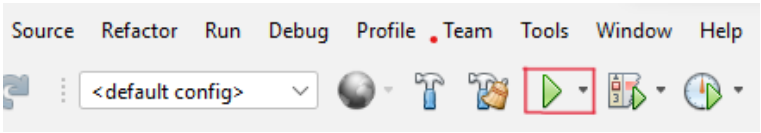
a) Buka Aplikasi Netbeans, buat project baru dengan nama sesuai dengan format berikut:

KelasPrak2ASTB. Contoh: A1Prak2A13020180226

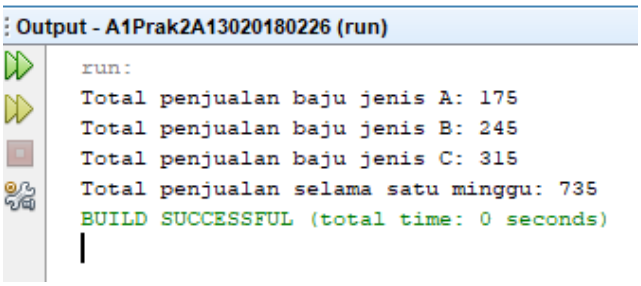
b) Pada *Function* Main dalam **Class A1Prak2A13020180226** tuliskan kodingan sebagai berikut dan lengkapi:

```java
package a1prak2a13020180226;


public class A1Prak2A13020180226 {

    int[][] dataPenjualan = {
        {..., 20, 30}, // jumlah penjualan baju jenis A, B, C pada hari 1
        {15, 25, 35}, // jumlah penjualan baju jenis A, B, C pada hari 2
        {20, ..., 40}, // jumlah penjualan baju jenis A, B, C pada hari 3
        {25, 35, 45}, // jumlah penjualan baju jenis A, B, C pada hari 4
        {30, 40, 50}, // jumlah penjualan baju jenis A, B, C pada hari 5
        {35, ..., 55}, // jumlah penjualan baju jenis A, B, C pada hari 6
        {..., 50, 60}  // jumlah penjualan baju jenis A, B, C pada hari 7
    };

    public int[] hitungTotalPenjualanPerJenisBaju() {
        int[] totalPenjualanPerJenisBaju = new int[dataPenjualan[0].length];
        for (int i = 0; i < //Lengkapi; i++) {
            for (int j = 0; j < dataPenjualan[i].length; j++) {
                totalPenjualanPerJenisBaju[j] += dataPenjualan[i][j];
            }
        }
        return totalPenjualanPerJenisBaju;
    }

    public int hitungTotalPenjualanSelamaSatuMinggu() {
        int[] totalPenjualanPerJenisBaju = hitungTotalPenjualanPerJenisBaju();
        int totalPenjualan = 0;
        for (int i = 0; i < totalPenjualanPerJenisBaju.length; i++) {
            // Lengkapi
        }
        return totalPenjualan;
    }

    public static void main(String[] args) {
        A1Prak2A13020180226 penjualanBaju = new A1Prak2A13020180226();
        int[] totalPenjualanPerJenisBaju = penjualanBaju.hitungTotalPenjualanPerJenisBaju();
        int totalPenjualan = penjualanBaju.hitungTotalPenjualanSelamaSatuMinggu();

        System.out.println("Total penjualan baju jenis A: " + totalPenjualanPerJenisBaju[...]);
        System.out.println("Total penjualan baju jenis B: " + totalPenjualanPerJenisBaju[...]);
        System.out.println("Total penjualan baju jenis C: " + totalPenjualanPerJenisBaju[...]);
        System.out.println("Total penjualan selama satu minggu: " + // Lengkapi);
    }

}
```

c) Selanjutnya tekan tombol *run* di bagian atas yang berbentuk tombol *play* berwarna hijau seperti gambar di bawah atau dengan menekan *shortcut* (*Shift*+f6) untuk melakukan *compile* dan *run*.



d) Setelah itu ketika program berjalan tanpa ada *error* maka akan tampil *output* seperti gambar di bawah ini :



LEMBAR EVALUASI PRAKTIKUM

1. Apa yang menjadi tujuan dari program diatas?.
2. Apa yang dilakukan pada baris berikut?seperti gambar di bawah :

```
Buku[] books = new Buku[10];
```

3. Apa yang bisa ditingkatkan dari program ini untuk membuatnya lebih baik? Berikan penjelasan dan contohnya.
4. Andi adalah seorang pengelola perpustakaan Utsman bin Affan. Ia memutuskan untuk membuat sebuah program yang dapat membantu mengelola perpustakaannya. Program tersebut menggunakan bahasa pemrograman Java dan memiliki beberapa fitur, di antaranya:
   a. Tambah Buku: Andi dapat menambahkan buku baru ke dalam daftar buku yang tersedia di perpustakaannya. Setiap buku memiliki judul dan penulis.
   b. Lihat Buku: Andi dapat melihat daftar buku yang tersedia di perpustakaannya.
   c. Pinjam Buku: Anggota perpustakaan dapat meminjam buku yang tersedia. Jika buku yang ingin dipinjam sedang tidak tersedia, maka peminjaman tidak dapat dilakukan. Setiap peminjaman harus mencatat nama peminjam dan tanggal peminjaman.
   d. Lihat Peminjaman: Andi dapat melihat daftar buku yang sedang dipinjam beserta informasi peminjamnya.
   e. Kembalikan Buku: Anggota perpustakaan dapat mengembalikan buku yang sedang dipinjam. Setiap pengembalian harus mencatat tanggal pengembalian.

   Andi meminta Anda untuk membuat program tersebut. Anda diminta untuk membuat kodingan program tersebut menggunakan bahasa pemrograman Java.

Evaluasi Praktikum 2:

No	Indikator	Skor Penilaian				
		Sangat Kurang (E) <=40	Kurang (D) 41-55	Cukup (C) 55-65	Baik (B) 66-85	Sangat Baik (A) >=86
1.	Dapat Memahami Konsep Dari Array					
2.	Dapat Membuat sebuah Array					
3.	Memahami Konsep Dari Method					
4.	Dapat Membuat Method					

Catatan Asisten:

Asisten 1 :_____

Asisten 2 : _____

MODUL 3 – CLASS, OBJECT DAN ENCAPSULATION

A. Sub Capaian Pembelajaran Mata Kuliah (CPMK)

1. Mahasiswa mampu memahami konsep dasar dan bahasa pemrograman berorientasi objek
2. Mahasiswa mampu membuat program berorientasi objek dengan memanfaatkan class

B. *Instrument* dan Prosedur

1. Instrument

a) Perangkat komputer
b) Sistem Operasi Windows
c) *Java Development KIT* (JDK)
d) IDE Netbeans

2. Prosedur

a) Baca dan pahami semua tahapan praktikum dengan cermat.
b) Gunakan fasilitas yang disediakan dengan penuh rasa tanggung jawab.
c) Rapikan kembali setelah menggunakan komputer (*mouse*, *keyboard*, kursi, dll)
d) Perhatikan sikap anda untuk tidak mengganggu rekan praktikan lain
e) Pastikan diri anda tidak menyentuh sumber listrik.

C. Tugas Pendahuluan

1. Bagaimana pembuatan sebuah package? Berikan contoh pembuatan package dan cara mengimport package ke dalam program?
2. Jelaskan perbedaan antara Class dan Object beserta contoh sederhananya
3. Bagaimana cara membuat object?
4. Implementasikan object yang telah dibuat pada soal nomor 3 di dalam sebuah class
5. Buatkan sebuah class dengan nama “Mahasiswa” dengan minimal 3 field (nama, stb, jurusan) didalamnya lalu buatlah sebuah class dan implementasikan object Class “Mahasiswa” didalamnya sesuai dengan data diri praktikan

D. Teori Dasar

1. Package

Package dalam bahasa Pemrograman Java adalah sebuah cara untuk mengorganisasi kode program yang terdiri dari Class-class dan Object-Object.
Manfaat dari penggunaan package dalam pengembangan program adalah memudahkan kita untuk mengorganisasikan Class dan Object yang telah kita buat dan memudahkan kita untuk memaintenance program.Dalam membuat package

1. klik kanan pada mouse
2. klik new
3. lalu klik folder
4. lalu akan muncul laman isi nama folder
5. isikan sesuai dengan nam package

2. Class

Class adalah sekelompok *Object-Object* yang memiliki karakteristik yang sama atau biasa diartikan sebagai sebuah *blueprint* atau cetak biru dari dari sebuah *Object*.Bentuk umum

Contoh

a) Konstruktor

Konstruktor adalah sebuah *method* khusus yang digunakan untuk membuat Object dari sebuah *Class*. Sebuah konstruktor harus memiliki nama yang sama dengan kelasnya. Konstruktor juga bisa memiliki parameter dan juga tidak.

Bentuk umum

```
[Acces Modifier] NamaClass (Parameter)
{ //code block
}
```

Contoh

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

b) MainFunction

Main Function adalah sebuah method khusus yang digunakan sebagai titk awal dari semua program di Java. Didalam main function semua kode akan dieksekusi.

Bentuk umum

```
public static void main (String[] args){
  //statement
}
```

Contoh

```
public static void main (String[] args){
  System.out.print("Selamat datang")
}
```

3. Object

Object adalah sebuah *instance* dari sebuah *class*. *Object* dapat memiliki *state* dan *behavior* yang didefinisikan oleh *Class*-nya. *Object* digunakan untuk berelasi dengan *object* lainnya.

Bentuk umum

```
// Membuat sebuah object class
NamaClass namaObject = new NamaClass();
```

Contoh

```
// Membuat object dari class TeknikInformatika
TeknikInformatika teknikInformatika = new TeknikInformatika()
```

4. Enkapsulasi

Enkapsulasi adalah salah satu prinsip dasar dalam Pemrograman *object* yang memungkinkan untuk menyembunyikan detail implementasi internal dari sebuah *object* dan hanya mengekspos *function* atau *method*

Bentuk umum

```
public class NamaClass{
  private int namaVariabel;
  public void setVariabelPrivate(int value){
    //statement
  }
  public int getVariabelPrivate(){
    //statement
  }
}
```

Contoh

```
public class Fakultas{
    private String name;
    private String alamat;
    public String getName(){
      return name;
    }
    public void setName(String name){
      this.name = name;
    }
    public String getAlamat(){
      return alamat;
    }
    public void setAlamat(String alamat){
      this.alamat = alamat;
    }
}
```

Di dalam contoh ini, field name dan alamat di set sebagai private, sehingga tidak dapat diakses dari luar class. Kita hanya perlu membuatkan method yang dapat mensetting dan memanggil nama dan alamat yang dapat diakses dari luar class.

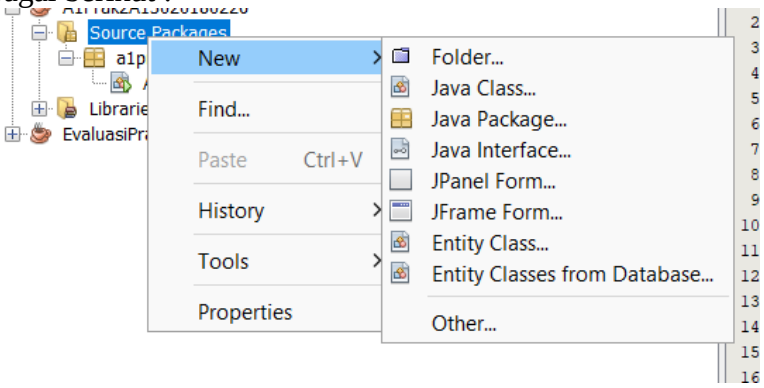
E. Kegiatan Praktikum

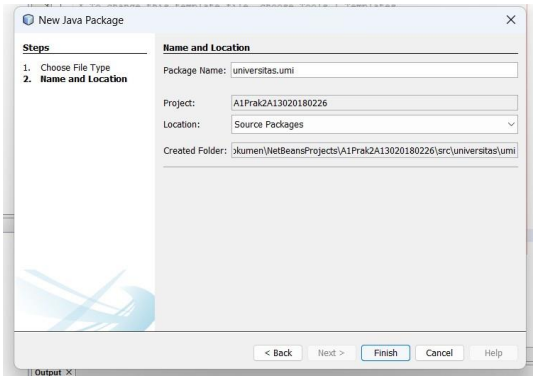
1. Studi Kasus A

a) Buka Aplikasi Netbeans, buat project baru dengan nama sesuai dengan format berikut :

KelasPrak2ASTB. Contoh: A1Prak2A13020180226

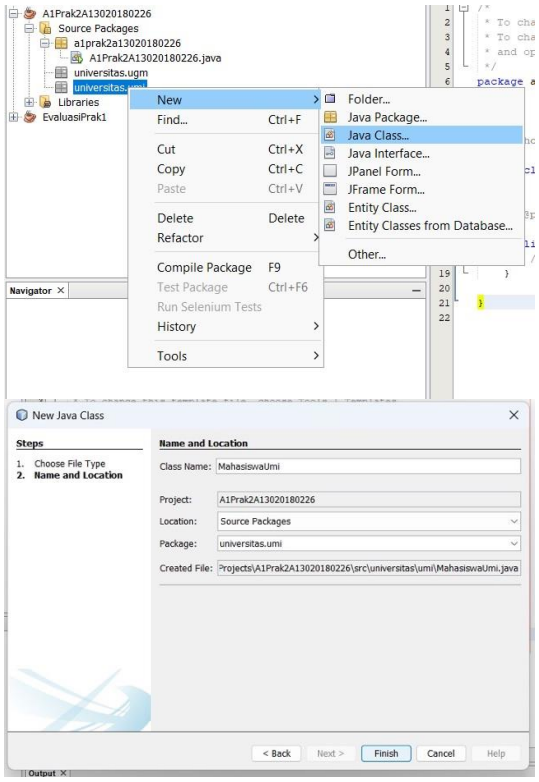
b) Pada *Function* Main dalam **Class A1Prak2A13020180226** tuliskan kodingan sebagai berikut :





c) Setelah itu buat lagi *package* baru dengan nama universitas.ugm dengan carayang sama degnan gambar diatas.

d) di dalam package universitas umi buat sebuah class dengan namaMahasiswa Umi seperti contoh gambar di bawah ini :



e) didalam *class* mahasiswa MahasiswaUmi buat kodingan seperti berikut

```
package universitas.umi;

public class MahasiswaUmi {

    public String universitas = "Universitas"
            + " Muslim Indonesia";
    public String name;
    public String stb;

    public MahasiswaUmi(String name, String stb) {
        this.name = name;
        this.stb = stb;
    }
}
```

f) setelah itu buat buat *class* baru di dalam *package* universitas.ugm dengan nama *class* MahasiswaUgm dengan cara yang sama pada gambar sebelumnya

g) didalam *class* MahasiswaUgm buat kodingan seperti berikut :

```
package universitas.ugm;

public class MahasiswaUgm {

    public String universitas = "Universitas Gajah "
            + "Mada";
    public String nama;
    public String stb;

    public MahasiswaUgm(String nama, String stb) {
        this.nama = nama;
        this.stb = stb;
    }
}
```

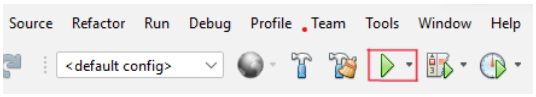
h) setelah itu pergi ke *function* main yang ada pada *class* **A1Prak2A13020180226**dan *import* semua *package* yang telah di buat sebelumnya seperti pada gambardi bawah :

```
6   package alprak2a13020180226;
7
    import universitas.ugm.MahasiswaUgm;
    import universitas.umi.MahasiswaUmi;
10  /**
```

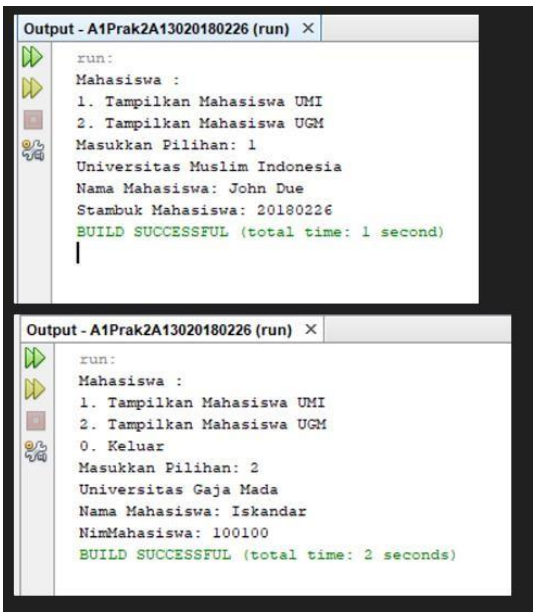
i) setelah melakukan *import* silahkan tulis kodingan berikut didalam *function* main :

```
package alprak2a13020180226;

import java.util.Scanner;
import universitas.ugm.MahasiswaUgm;
import universitas.umi.MahasiswaUmi;

public class A1Prak2A13020180226 {

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

        MahasiswaUmi mhsUmi = new MahasiswaUmi("John Due",
20180226);MahasiswaUgm mhsUgm = new MahasiswaUgm("Iskandar", 100100);

        System.out.println("Mahasiswa :");
        System.out.println("1. Tampilkan Mahasiswa UMI");
        System.out.println("2. Tampilkan Mahasiswa UGM");
        System.out.print("Masukkan Pilihan: ");
        int pil = input.nextInt();

        switch(pil){
            case 1:
                System.out.println(mhsUmi.universitas);
                System.out.println("Nama Mahasiswa: "+mhsUmi.name);
                System.out.println("Stambuk Mahasiswa:
"+mhsUmi.stb);  break;
            case 2:
                System.out.println(mhsUgm.universitas);
                System.out.println("Nama Mahasiswa: "+mhsUgm.name);
                System.out.println("NimMahasiswa: "+mhsUgm.nim);
                break;

            default:
                System.out.println("Menu tidak tersedia!");
        }

    }
}
```

j) Selanjutnya tekan tombol *run* di bagian atas yang berbentuk tombol *play* berwarna hijau seperti gambar di bawah atau dengan menekan *shortcut* (*Shift*+f6) untuk melakukan *compile* dan *run*.



k) Setelah itu ketika program berjalan tanpa ada *error* maka akan tampil *output* seperti gambar di bawah ini

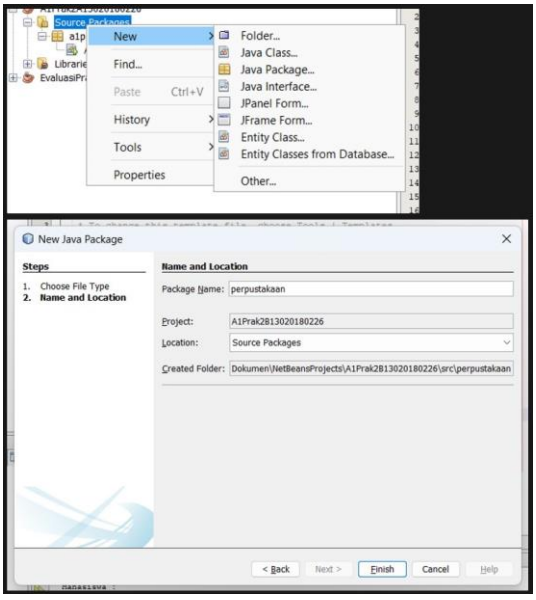


2. Studi Kasus B

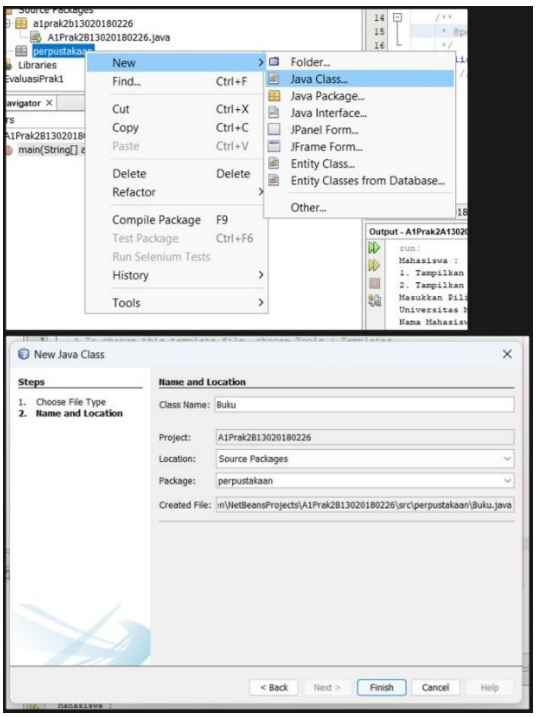
a) Buka Aplikasi *Netbeans*, buat *project* baru dengan nama sesuai dengan format berikut :

KelasPrak2BSTB. Contoh: A1Prak2B13020180226

b) Buat sebuah *package* dengan nama perpustakaan seperti pada contoh gambar di bawah



c) di dalam *package* perpustakaan buat sebuah *class* dengan nama Buku seperti contoh gambar di bawah ini :



d) didalam *class* Buku buat kodingan seperti berikut :

```
package perpustakaan;

public class Buku {
    public static int count = 0;
    public String title;
    public String author;

    public Buku(String title, String author)
{       this.title = title;
        this.author = author;
        count++;
    }
}
```

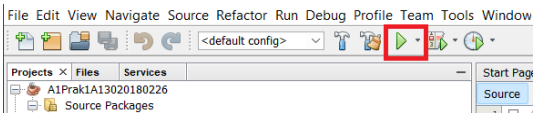
e) setelah itu pergi ke *function* main yang ada pada *class* **A1Prak2B13020180226** dan *import package* yang telah di buat sebelumnya seperti pada gambar di bawah :

```
import java.util.Scanner;
import perpustakaan.Buku;
```

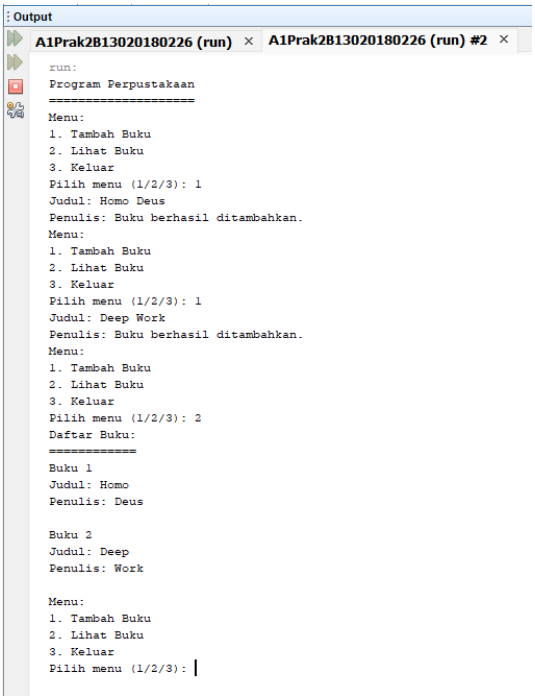
f) setelah melakukan *import* silahkan tulis kodingan berikut didalam *function* main :

```
package a1prak2b13020180226;

import java.util.Scanner;
import perpustakaan.Buku;

public class A1Prak2B13020180226 {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Buku[] books = new Buku[10];

        System.out.println("Program Perpustakaan");
        System.out.println("====================");

        while (true) {
            System.out.println("Menu:");
            System.out.println("1. Tambah Buku");
            System.out.println("2. Lihat Buku");
            System.out.println("3. Keluar");
            System.out.print("Pilih menu (1/2/3): ");
            int menu = scanner.nextInt();

            if (menu == 1) {
                if (Buku.count >= 10) {
                    System.out.println("Maaf, maksimum buku sudah
tercapai.");          continue;
                }

                System.out.print("Judul: ");
                String title = scanner.next();
                System.out.print("Penulis: ");
                String author = scanner.next();

                books[Buku.count] = new Buku(title, author);

                System.out.println("Buku berhasil ditambahkan.");
            } else if (menu == 2) {
                System.out.println("Daftar Buku:");
                System.out.println("============");
                for (int i = 0; i < Buku.count; i++) {
                    Buku book = books[i];
                    System.out.println("Buku " + (i+1));
                    System.out.println("Judul: " + book.title);
                    System.out.println("Penulis: " + book.author);
                    System.out.println();
                }
            } else if (menu == 3) {
                System.out.println("Terima kasih.");
                break;
            } else {
                System.out.println("Menu tidak valid.");
            }
        }
    }

}
```

g) Selanjutnya tekan tombol *run* di bagian atas yang berbentuk tombol *play* berwarna hijau seperti gambar di bawah atau dengan menekan *shortcut* (*Shift*+f6) untuk melakukan *compile* dan *run*.



h) Setelah itu ketika program berjalan tanpa ada error maka akan tampil *output* seperti gambar di bawah ini :



LEMBAR EVALUASI PRAKTIKUM

1. Apa yang menjadi tujuan dari program ini?
2. Jelaskan struktur *class* Buku dan variabel statis *count* yang ada di dalamnya.
3. Apa yang dilakukan pada baris berikut?seperti gambar di bawah :

```
Buku[] books = new Buku[10];
```

4. Apa yang bisa ditingkatkan dari program ini untuk membuatnya lebih baik? Berikan penjelasan dan contohnya.
5. Andi adalah seorang pengelola perpustakaan Utsman bin Affan. Ia memutuskan untuk membuat sebuah program yang dapat membantu mengelola perpustakaannya.

   Program tersebut menggunakan bahasa pemrograman Java dan memiliki beberapa fitur, di antaranya:

a. Tambah Buku: Andi dapat menambahkan buku baru ke dalam daftar buku yang tersedia di perpustakaannya. Setiap buku memiliki judul dan penulis.
b. Lihat Buku: Andi dapat melihat daftar buku yang tersedia di perpustakaannya.
c. Pinjam Buku: Anggota perpustakaan dapat meminjam buku yang tersedia. Jika buku yang ingin dipinjam sedang tidak tersedia, maka peminjaman tidak dapat dilakukan. Setiap peminjaman harus mencatat nama peminjam dan tanggal peminjaman.
d. Lihat Peminjaman: Andi dapat melihat daftar buku yang sedang dipinjam beserta informasi peminjamnya.

e. Kembalikan Buku: Anggota perpustakaan dapat mengembalikan buku yang sedang dipinjam. Setiap pengembalian harus mencatat tanggal pengembalian.
Andi meminta Anda untuk membuat program tersebut. Anda diminta untuk membuat kodingan program tersebut menggunakan bahasa pemrograman Java.

Evaluasi Praktikum 3:

No	Indikator	Skor Penilaian				
		Sangat Kurang (E) <=40	Kurang (D) 41-55	Cukup (C) 55-65	Baik (B) 66-85	Sangat Baik (A) >=86
1.	Dapat Memahami Konsep Dari Class					
2.	Dapat membuat Sebua Class					
3	Memahami Konsep Dari Object					
4	Dapat Membuat Object					
5	Dapat Memahami Konsep Dari Enkapsulasi					
6	Dapat Membuat Enkapsulasi					

Catatan Asisten:

Asisten 1 :_____

Asisten 2 : _____

MODUL 4 – INHERITANCE DAN POLYMORPHISM

A. Sub Capaian Pembelajaran Mata Kuliah (CPMK)

1. Mahasiswa dapat memahami konsep dari *inheritence* dan *overriding*, serta mampu menerapkan dan mengaplikasikanya menggunakan bahasa java
2. Mahasiswa dapat memahami konsep dari *access modifier* serta mampu menerapkan dengan menggunakan bahasa Pemrograman java

B. Instrument dan Prosedur

1. Instrument

a) Perangkat komputer
b) Sistem Operasi Windows
c) *Java Development KIT* (JDK)
d) IDE Netbeans

2. Prosedur

a) Baca dan pahami semua tahapan praktikum dengan cermat.
b) Gunakan fasilitas yang disediakan dengan penuh rasa tanggung jawab.
c) Rapihkan kembali setelah menggunakan komputer (*mouse, keyboard,* kursi, dll)
d) Perhatikan sikap anda untuk tidak mengganggu rekan praktikan lain.
e) Pastikan diri anda tidak menyentuh sumber listrik.

C. Tugas Pendahuluan

1. Jelaskan apa yang dimaksud dengan *inheritence*
2. Sebutkan dan jelaskan *access modifier* yang ada di Java
3. Jelaskan apa yang dimaksud dengan *Polymorphism*
4. Buatlah program sederhana yang mengimplementkan *inheritence* dan *override*

D. Teori Dasar

1. Inheritance

Inheritance adalah sebuah konsep dimana sebuah *class* dapat "diwarisi" properti dan metode dari *class* lain yang lebih umum disebut *superClass* atau *parent Class*. *Class* yang menerima warisan tersebut disebut *subClass* atau *child Class*.

Sebagai contoh kita memiliki *Class* Fakultas dan *Class* TeknikInformatika.
Class Fakultas sebagai *parent* dan TeknikInformatika sebagai *child*.

```
public class TeknikInformatika extends Fakultas {
```

Perhatikan kode diatas, dengan kita menambahkan *extends* pada *class* TeknikInformatika maka secara otomatis semua *field* dan *fungtion* yang ada didalam *Class* Fakultas akan bisa diakses menggunakan *Class* TeknikInformatika.

2. *Overriding*

Method overriding adalah ketika sebuah *subClass* atau turunan mengimplementasikan ulang atau memodifikasi *method* yang sudah ada disuperclass atau induknya dengan nama dan parameter yang sama.

Contoh kita mempunyai sebuah *Class* Fakultas dengan *method setName*()

```
public class Fakultas {

    String name;

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

}
```

Kemudian kita mempunyai *class* TeknikInformatika yang merupakan *subClass* dari Fakultas dan ingin memodifikasi implementasi *method setName*();

```
public class TeknikInformatika extends Fakultas {

    @Override
    public void setName(String name) {
        super.setName(name);
    }
```

Super didalam *code* artinya dia memanggil *method* yang ada *diparent class*

3. *Access modifier*

Access modifier adalah sebuah kata kunci yang digunakan untuk mengatur hak akses terhadap *class*, *field*, dan *method* pada sebuah program. *Access modifier* terdiri dari 4 jenis yaitu:

a. *Public*

Public adalah *access modifier* yang paling luas, artinya *class*, *field*, ataupun *method* yang dideklarasikan dengan *public* dapat diakses dari mana saja, bahkan dari *package* lain.

```
// sebuah method dengan access modifier public
public void setName(){
    // statement
}
```

b. *Private*

Private adalah *access modifier* yang paling ketat, artinya *class*, *field*, ataupun *method* yang dideklarasikan dengan *private* hanya bisa diakses didalam *class* itu sendiri. *Private* digunakan untuk membatasi akses terhadap data atau *method* yang tidak bisa diubah atau diakses secara langsung dari luar *class*

```
// sebuah method dengan access modifier private
private void setName(){
    // statement
}
```

c. *Protected*
Protected adalah *access modifier* yang digunakan untuk membatasi akses pada *package* dan *subClass*-nya. Artinya seluruh *method, field*, dan *class* tidak bisa diakses dari luar *package* dimana *class* ini dideklarasikan.

```
// sebuah method dengan access modifier protected
protected void setName(){
    // statement
}
```

d. *Default*
Default adalah *access modifier* yang didefinisikan secara *implisit* jika tidak ada *access modifier* yang ditentukan terhadap *class*, *field*, ataupun *method*. Jika sebuah *class, field*, ataupun *method* yang diberi *access modifier default* maka hanya bisa diakses dipackage yang sama dengan *class* dimana *class* ini dideklarasikan

```
// sebuah method dengan access modifier default
void setName(){
    // statement
}
```

4. *Polymorphism*
Polymorphism dalam OOP adalah sebuah prinsip di mana class dapat memiliki banyak "bentuk" *method* yang berbeda-beda meskipun namanya sama."Bentuk" di sini dapat kita artikan, isinya berbeda, parameternya berbeda, dan tipe datanya berbeda.*Polimorfisme* pada Java ada dua macam:

1. *Static Polymorphism* (Polimorfisme statis)
   Overloading adalah mendefinisikan dua atau lebih *method* di dalam kelas yang sama, dengan nama yang sama, namun dengan deklarasi parameter yang berbeda.

2. *Dynamic Polymorphism* (Polimorfisme dinamis).
   Method subclass override terhadap *method superclass* ketika *subclass* mendeklarasikan method yang signaturenya serupa ke *method* dalam *superclass*. *Method* ini hanya terjadi jika merupakan implementasi dari pewarisan.

Beda dari keduanya terletak pada cara membuat polimorfismenya.Polimorfisme statis menggunakan method *overloading* sedangkan *polimorfisme* dinamis menggunakan *method overriding*

E. Kegitan Praktikum

1. Studi Kasus A

a) Buka *IDE Apache NetBeans* dan buat *project* baru dengan nama *project* sesuai format berikut**: KelasPraktikum5ANim**. Contoh: **A8Praktikum5A13020190418**

b) Buatlah sebuah *class* baru yang diberi nama Pegawai, dengan *atribut* seperti gambar dibawah ini

```
package a8praktikum5a13020190418;

public class Pegawai {
    String nama;
    String jabatan;

    void tampilkanFakultas() {
        System.out.println("Fakultas Ilmu Komputer");
    }
}
```

c) Kemudian buat 2 class yang bernama Dosen dan Staff dan kelas tersebut merupakan turunan dari kelas Pegawai yang sudah dibuat sebelumnya

```
package a8praktikum5a13020190418;

public class Dosen extends Pegawai {

}
```

```
package a8praktikum5a13020190418;

public class Staff extends Pegawai {

}
```

d) Selanjutnya, lakukan instansiasi untuk objek Dosen dan Staff dan tampilkan

```java
package a8praktikum5a13020190418;

public class A8Praktikum5A13020190418 {

    public static void main(String[] args) {
        Pegawai pegawai = new Pegawai();
        Dosen dosen = new Dosen();
        Staff staff = new Staff();

        dosen.nama = "Dra. Anisah";
        staff.nama = "Indri Ihsaniy";

        System.out.println("Nama Dosen : " + dosen.nama);
        System.out.print("Fakultas Dosen : ");
        dosen.tampilkanFakultas();

        System.out.println("\nNama Staff : " + staff.nama);
        System.out.print("Fakultas Staff: ");
        staff.tampilkanFakultas();
    }
}
```

e) Jalankan program dan lihat hasilnya

```
Output - A8Praktikum5A13020190418 (run)
run:
Nama Dosen : Dra. Anisah
Fakultas Dosen : Fakultas Ilmu Komputer

Nama Staff : Indri Ihsaniy
Fakultas Staff: Fakultas Ilmu Komputer
BUILD SUCCESSFUL (total time: 0 seconds)
```

f) Terapkan *method overriding* pada *class* Pegawai

```java
package a8praktikum5a13020190418;

public class Pegawai {
    String nama;
    String jabatan;

    void tampilkanFakultas() {
        System.out.println("Fakultas Ilmu Komputer");
    }
}
```

g) Jalankan program dan lihat hasilnya

```
Output - A8Praktikum5A13020190418 (run)
run:
Nama Dosen : Dra. Anisah
Fakultas Dosen : Fakultas Ilmu Komputer

Nama Staff : Indri Ihsaniy
Fakultas Staff: Fakultas Teknik
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Studi Kasus B

a) Buka IDE Apache NetBeans dan buat project baru dengan nama project sesuai format berikut: **KelasPraktikum5BNim**. Contoh**: A8Praktikum5B13020190418**

b) Buatlah sebuah *class* yang diberi nama Buku, dan seluruh *attribut* dari *class* tersebut menggunakan *access modifier private*

```
package a8praktikum5b13020190418;

public class Buku {
    private String judul;
    private String penulis;
    private int tahunRilis;
    private int harga;
}
```

c) Pembuatan *method* pada *class* Buku yang berfungsi untuk mengatur nilai dan mengambil nilai dari *attribute* dari *class*

```
public String getJudul() {
  return judul;
}

public void setJudul(String judul) {
  this.judul = judul;
}

public String getPenulis() {
  return this.penulis;
}

public void setPenulis(String penulis) {
  this.penulis = penulis;
}

public int getTahunRilis() {
  return this.tahunRilis;
}

public void setTahunRilis(int tahunRilis) {
  this.tahunRilis = tahunRilis;
}

public int getHarga() {
  return this.harga;
}

public void setHarga(int harga) {
  this.harga = harga;
}
```

d) Melakukan instansiasi objek dari *class* Buku di fungsi main

```
package a8praktikum5b13020190418;

public class A8Praktikum5B13020190418 {

    public static void main(String[] args) {
        Buku buku = new Buku();

        buku.setJudul("Pemrograman Berorientasi Objek");
        buku.setPenulis("Budi Rahardjo");
        buku.setHarga(60000);
        buku.setTahunRilis(2019);

        String judul = buku.getJudul();
        String penulis = buku.getPenulis();
        double harga = buku.getHarga();
        int tahunRilis = buku.getTahunRilis();

        System.out.println("Judul Buku: " + judul);
        System.out.println("Penulis Buku: " + penulis);
        System.out.println("Harga Buku: " + harga);
        System.out.println("Tahun Rilis Buku: " + tahunRilis);
    }
}
```

e) Dalam melakukan instansiasi, kita tidak lagi memberikan nilai dari *attribut* melalui nama *attribute* nya, dalam hal ini, saat kita ingin memberikan nilai dari *attribut*, kita bisa memanfaatkan *method set* dan *method get* untuk mengambil nilai dari *attribut* nya

f) Jalankan program dan lihat hasilnya

```
Output - A8Praktikum5B13020190418 (run)
run:
Judul Buku: Pemrograman Berorientasi Objek
Penulis Buku: Budi Rahardjo
Harga Buku: 60000.0
Tahun Rilis Buku: 2019
BUILD SUCCESSFUL (total time: 0 seconds)
```

LEMBAR EVALUASI PRAKTIKUM

1. Terapkan akses modifier pada atribut dan lakukan proses *enkapsulasi* pada suatu kelas dan juga Implementasikan method setter dan getter pada attribute kelas yang sudah diberikan akses modifier **private**
2. Buatlah satu kelas yang nantinya kelas tersebut memiliki 3 kelas turunan, dan juga terapkan method *overriding* pada kelas turunannya.

Evaluasi Praktikum 4:

No	Indikator	Skor Penilaian				
		Sangat Kurang (E) <=40	**Kurang (D) 41-55**	**Cukup (C) 55-65**	**Baik (B) 66-85**	**Sangat Baik (A) >=86**
1.	**Memahami Konsep *Inheritenance, Polymorphism, Access Modifier,* dan *Override***					
2.	**Dapat Mengimpelmentasikan *Inheritenance, Polymorphism, Access Modifier,* dan *Override***					

Catatan Asisten:

Asisten 1 :_____

Asisten 2 _____

MODUL 5 – INTERFACE DAN ABSTRACT

A. Sub Capaian Pembelajaran Mata Kuliah (CPMK)

1. Mahasiswa dapat menjelaskan konsep *polymorphism* serta menerapkan menggunakan bahasa Pemrograman java
2. Mahasiswa dapat menjelaskan konsep abstrak menggunakan bahasa Pemrograman java
3. Mahasiswa dapat menjelaskan konsep *interface* dan mengimplementasikannya menggunakan bahasa Pemrograman java

B. Instrument dan Prosedur

1. Instrument

a) Perangkat komputer
b) Sistem Operasi Windows
c) Java Development KIT (JDK)
d) IDE Netbeans

2. Prosedur

a) Baca dan pahami semua tahapan praktikum dengan cermat.
b) Gunakan fasilitas yang disediakan dengan penuh rasa tanggung jawab.
c) Rapihkan kembali setelah menggunakan komputer (*mouse, keyboard*, kursi, dll)
d) Perhatikan sikap anda untuk tidak mengganggu rekan praktikan lain.
e) Pastikan diri anda tidak menyentuh sumber listrik.

C. Tugas Pendahuluan

1. Jelaskan apa yang dimaksud dengan *interface*
2. jelaskan apa yang dimaksud dengan *abstract*
3. sebutkan dan jelaskan apa perbedaan dari *interface* dan *abstract*
4. Buatlah program sederhana yang mengimplementkan *interface* dan *abstract*
5. Buatkan program sederhana pengimplementasian method *override*

D. Teori Dasar

1. Interface

Interface di Java adalah sebuah blueprint atau template untuk sebuah *class*. *Interface* menyediakan cara bagi kita untuk menentukan metode yang harus dimiliki oleh suatu class, namun tidak memberikan implementasi dari metode tersebut, *Interface* seperti sebuah kontrak yang harus dipenuhi oleh sebuah *class*. Ketika sebuah class mengimplementasikan sebuah *interface*, *class* tersebut harus menyediakan implementasi untuk semua metode yang didefinisikan dalam *interface* tersebut,untuk mengakses *interface* memerlukan kata kunci "*implements*".

```
//membuat interface
public interface interfaces {
    // interface method(tidak memiliki badan)
    public void methodInterfaces();
}
```

```java
//Class labJaringan "implements" interface Lab
public class implementasiInterface implements interfaces{
    @Override
    public void methodInterfaces() {
    //statement
    }
    public static void main(String[] args) {
        //buat objec class
    }
}
```

pada metode *interface* dapat melakukan banyak implementasi, catatan: Untuk mengimplementasikan banyak antarmuka, pisahkan dengan koma

```java
//Class labJaringan "implements" interface Lab
public class implementasiInterface implements interfaces, interfaces2
{   @Override
    public void methodInterfaces() {
    //statement
    }
    @Override
    public void methodInterfaces2() {
        //statement
    }
    public static void main(String[] args) {
        //buat objec class
    }
}
```

Ada beberapa catatan dalam membuat *interface*:

1. Jangan buat variabel di dalam *interface*, tapi membuat konstanta boleh,
2. Jangan mengisi *method*-nya, cukup tuliskan nama *method*, tipe data, dan parameter saja. Tapi untuk *default method* boleh punya isi.
3. Metode *interface* tidak memiliki badan,badan disediakan oleh *class* yang di "*implements*"
4. Saat implementasi *interface*,semua methodnya harus diimplementasi pada *class* yang di "*implements*"

2. Abstract

Sebuah *class abstract* adalah *class* yang tidak dapat di-*instantiate*. *Method* ini dalam *class abstract* yang tidak mempunyai implementasi dinamakan *method abstract*. Untuk membuat *method abstract*, tinggal menulis deklarasi *method* tanpa tubuh *class* dan digunakan menggunakan kata kunci *abstract.Class abstract* berfungsi sebagai kerangka atau template yang menyediakan definisi metode-metode yang harus dimiliki oleh *subclass* (*class* turunan). Dalam *class abstract*, kita dapat mendefinisikan metode-metode yang harus dimiliki oleh *subclass*, namun tidak memberikan implementasi untuk metode tersebut.cara membuat *abstract*:

```java
//abstract class
public abstract class Abstract{
    // Method abstract (tidak memiliki implementasi)
    abstract int namaAbstract();
}
```

```java
//Class implementasiAbstract mewarisi class Abstract
public class implementasiAbstract extends Abstract
{
    @Override
    int namaAbstract(){
    //statment
    }
}
```

E. Kegiatan Praktikum

1. Studi Kasus A

a) Buka *IDE Apache NetBean*s dan buat *project* baru dengan nama project sesuai format berikut**: KelasPraktikum6ANim**. Contoh: **A8Praktikum6A13020190418**

b) Buatlah sebuah kelas *interface* yang bernama Transaksi, kelas tersebut berisi 2 *method* kontrak yaitu setor dan tarik, 2 method tersebut mempunya parameter yang bertipe integer dan diberi nama saldo

```java
package a8praktikum6a13020190418.transaksi;

public interface Transaksi {
    void tarik(int nominal);
    void setor(int nominal);
}
```

c) Selanjutnya, buat 2 class yang bernama bernama TransaksiGiro dan TransaksiTabungan, class tersebut akan mengimplement method dari class kontrak Transaksi

```java
package a8praktikum6a13020190418.transaksi;

public class TransaksiTabungan implements Transaksi {

    @Override
    public void tarik(int nominal) {
    }

    @Override
    public void setor(int nominal) {
    }
}
```

```java
package a8praktikum6a13020190418.transaksi;

public class TransaksiGiro implements Transaksi{

    @Override
    public void tarik(int nominal) {
    }

    @Override
    public void setor(int nominal) {
    }
}
```

d) Pada class Transaksi dan Tabungan, buatlah atribut saldo dan buat juga *method cocnstructor* pada masing masing *class* yang berguna untuk melakukan inisialisasi atribut saldo di tersebut.

```java
package a8praktikum6a13020190418.transaksi;

public class TransaksiGiro implements Transaksi{

    int saldo;

    public TransaksiGiro(int saldo) {
        this.saldo = saldo;
    }

    @Override
    public void tarik(int nominal) {
    }

    @Override
    public void setor(int nominal) {
    }
}
```

e) Selanjutnya, pada *class* TransaksiTabungan dan TransaksiGiro, tambahkan logika sederhana terkait transaksi penarikan dan penyetoran uang pada *method* setor dan tarik

```
package a8praktikum6a13020190418.transaksi;

public class TransaksiTabungan implements Transaksi {

    int saldo;

    public TransaksiTabungan(int saldo) {
        this.saldo = saldo;
    }

    @Override
    public void tarik(int nominal) {
    }

    @Override
    public void setor(int nominal) {
    }
}
```

```
package a8praktikum6a13020190418.transaksi;

public class TransaksiGiro implements Transaksi{

    int saldo;

    public TransaksiGiro(int saldo) {
        this.saldo = saldo;
    }

    @Override
    public void tarik(int nominal) {
        if (nominal > saldo) {
            System.out.println("Maaf, saldo tabungan anda tidak mencukupi");
        } else {
            saldo = saldo - nominal;
            System.out.println("Penarikan berhasil, saldo giro saat ini : " + saldo);
        }
    }

    @Override
    public void setor(int nominal) {
        saldo = saldo + nominal;
        System.out.println("Setoran berhasil, saldo giro saat ini : " + saldo);
    }
}
```

```
package a8praktikum6a13020190418.transaksi;

public class TransaksiTabungan implements Transaksi {

    int saldo;

    public TransaksiTabungan(int saldo) {
        this.saldo = saldo;
    }

    @Override
    public void tarik(int nominal) {
        if (nominal > saldo) {
            System.out.println("Maaf, saldo tabungan anda tidak mencukupi");
        } else {
            saldo = saldo - nominal;
            System.out.println("Penarikan berhasil, saldo tabungan saat ini : " + saldo);
        }
    }

    @Override
    public void setor(int nominal) {
        saldo = saldo + nominal;
        System.out.println("Setoran berhasil, saldo tabungan saat ini : " + saldo);
    }
}
```

f) Lakukan instansiasi untuk objek TransaksiGiro dan TransaksiTabungan pada fungsi main

```
package a8praktikum6a13020190418;

import a8praktikum6a13020190418.transaksi.TransaksiGiro;
import a8praktikum6a13020190418.transaksi.TransaksiTabungan;

public class A8Praktikum6A13020190418 {

    public static void main(String[] args) {

        TransaksiGiro transaksiGiro = new TransaksiGiro(5000000);
        transaksiGiro.setor(2000000);
        transaksiGiro.tarik(4000000);

        System.out.println("\n");

        TransaksiTabungan transaksiTabungan = new TransaksiTabungan(2000000);
        transaksiTabungan.setor(3000000);
        transaksiTabungan.tarik(250000);
    }
}
```

g) Jalankan program dan lihat hasilnya

```
Output - A8Praktikum6A13020190418 (run)
run:
Setoran berhasil, saldo giro saat ini : 7000000
Penarikan berhasil, saldo giro saat ini : 3000000


Setoran berhasil, saldo tabungan saat ini : 5000000
Penarikan berhasil, saldo tabungan saat ini : 4750000
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Studi Kasus B

a) Buka *IDE Apache NetBeans* dan buat project baru dengan nama project sesuai format berikut**: KelasPraktikum6BNim**. Contoh**: A8Praktikum6B13020190418**

b) Buatlah sebuah *class abstract* yang bernama Karyawan, dan implementasikan *attribut* dan *method*, seperti yang ada pada gambar dibawah ini:

```
package a8praktikum6b13020190418;

public abstract class Karyawan {
    int id;
    String nama;
    int gaji;
    int bonus;

    public Karyawan(int id, String nama, int gaji, int bonus) {
        this.id = id;
        this.nama = nama;
        this.gaji = gaji;
        this.bonus = bonus;
    }

    public void totalGaji() {
        int total = this.gaji += this.bonus;
        System.out.println("Total Gaji : " + total);
    }

    public abstract void updateJabatan(String jabatan);
    public abstract void updateGaji(int gaji);
    public abstract void updateBonus(int bonus);

    public void detailKaryawan() {
        System.out.println("ID    : " + this.id);
        System.out.println("Nama  : " + this.nama);
        System.out.println("Gaji  : " + this.gaji);
        System.out.println("Bonus : " + this.bonus);
    }
}
```

c) Sealjutanya, buatlah 2 class baru yang nantinya akan menurunkan sifat-sifat dari *class abstract* Karyawan yang telah dibuat sebelumny

```java
package a8praktikum6b13020190418;

public class Dosen extends Karyawan {

    public String jabatan;

    public Dosen(int id, String nama, int gaji, int bonus) {
        super(id, nama, gaji, bonus);
    }

    public String getJabatan() {
        return this.jabatan;
    }

    public void setJabatan(String jabatan) {
        this.jabatan = jabatan;
    }

    @Override
    public void updateJabatan(String jabatan) {
        this.jabatan = jabatan;
    }

    @Override
    public void updateGaji(int gaji) {
        super.gaji = gaji;
    }

    @Override
    public void updateBonus(int bonus) {
        super.bonus = bonus;
    }
}
```

```java
package a8praktikum6b13020190418;

public class Staff extends Karyawan {

    String jabatan;

    public Staff(int id, String nama, int gaji, int bonus) {
        super(id, nama, gaji, bonus);
    }

    public String getJabatan() {
        return this.jabatan;
    }

    public void setJabatan(String jabatan) {
        this.jabatan = jabatan;
    }

    @Override
    public void updateJabatan(String jabatan) {
        this.jabatan = jabatan;
    }

    @Override
    public void updateGaji(int gaji) {
        super.gaji = gaji;
    }

    @Override
    public void updateBonus(int bonus) {
        super.bonus = bonus;
    }
}
```

d) Lakuan instansiasi objek Dosen dan Staff yang telah dibuat sebelumnya ke dalam fungsi main

```
package a8praktikum6b13020190418;

public class A8Praktikum6B13020190418 {

    public static void main(String[] args) {
        Dosen dosen = new Dosen(1, "Mohandes Gandhi", 5000000, 500000);
        dosen.setJabatan("Dosen Tetap");
        dosen.detailKaryawan();
        System.out.println("Jabatan : " + dosen.getJabatan());
        dosen.totalGaji();

        System.out.println("\n");

        Staff staff = new Staff(2, "John Dhoe", 2500000, 200000);
        staff.detailKaryawan();
        staff.updateBonus(300000);
        staff.totalGaji();
    }
}
```

e) Jalankan program dan lihat hasilnya

```
Output - A8Praktikum6B13020190418 (run)
run:
ID    : 1
Nama  : Mohandes Gandhi
Gaji  : 5000000
Bonus : 500000
Jabatan : Dosen Tetap
Total Gaji : 5500000


ID    : 2
Nama  : John Dhoe
Gaji  : 2500000
Bonus : 200000
Total Gaji : 2800000
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Studi Kasus C

a) Buka *IDE Apache NetBeans* dan buat *project* baru dengan nama project sesuai format berikut: **KelasPraktikum6CNim**. Contoh**: A8Praktikum6C13020190418**

b) Buatlah sebuah class yang bernama Transportasi, kemudian implementasikan *attribut* dan *method* seperti gambar dibawah ini

```
package a8praktikum6c13020190418;

public class Transportasi {
    private String jenis;

    public Transportasi(String jenis) {
        this.jenis = jenis;
    }

    public void jenisTransportasi() {
        System.out.println("Jensi Transportasi : " + jenis);
    }
}
```

c) Selanjutnya buatlah 2 *class* diberi nama Mobil, Pesawat *class* tersebut akan menjadi *class* turunan dari *class* Transportasi yang telah dibuat sebelumnya

```java
package a8praktikum6c13020190418;

public class Mobil extends Transportasi {

    public Mobil(String jenis) {
        super(jenis);
    }
}
```

```java
package a8praktikum6c13020190418;

public class Pesawat extends Transportasi {

    public Pesawat(String jenis) {
        super(jenis);
    }
}
```

d) Selanjutnya, lakukan instansiasi objek dari class yang telah dibuat dan juga terapkan proses *Polymorphism* untuk 3 *class* yang sudah dibuat

```java
package a8praktikum6c13020190418;

public class A8Praktikum6C13020190418 {

    public static void main(String[] args) {

        Transportasi transportasi = new Transportasi("Kendaraan Laut");
        transportasi.jenisTransportasi();

        transportasi = new Mobil("Kendaraan Darat");
        transportasi.jenisTransportasi();

        transportasi = new Pesawat("Kendaraan Udara");
        transportasi.jenisTransportasi();
    }
}
```

e) Jalankan program dan lihat hasilnya

```
Output - A8Praktikum6C13020190418 (run)
run:
Jensi Transportasi : Kendaraan Laut
Jensi Transportasi : Kendaraan Darat
Jensi Transportasi : Kendaraan Udara
BUILD SUCCESSFUL (total time: 0 seconds)
```

LEMBAR EVALUASI PRAKTIKUM

1. Berikan satu contoh penggunaan *polymorphism*
2. Tuliskan pendapatmu kelebihan dari konsep *polymorphism* pada pemrograman berorientasi objek?

Evaluasi Praktikum 5:

No	Indikator	Skor Penilaian				
		Sangat Kurang (E) <=40	**Kurang (D) 41-55**	**Cukup (C) 55-65**	**Baik (B) 66-85**	**Sangat Baik (A) >=86**
1.	**Memahami Konsep Dari *Interface***					
2	**Dapta Membuat *Interfaces***					
3	**Memahami Konsep dari *Abstract***					
4	**Dapat membuat *Class Abstract***					

Catatan Asisten:

Asisten 1 :_____

Asisten 2 : _____

MODUL 6 – INNER CLASS DAN EXCEPTION HANDLING

A. Sub Capaian Pembelajaran Mata Kuliah (CPMK)

1. Mahasiswa dapat memahami konsep dari *Inner Class* dan *Exception Handling* di Java
2. Mahasiswa dapat mengimplementasikan konsep dari *Inner Class* dan *Exception Handling* dengan menggunakan Bahasa Pemrograman Java

B. Instrument dan Prosedur

1. Instrument

a) Perangkat komputer
b) Sistem Operasi Windows
c) Java Development KIT (JDK)
d) IDE Netbeans

2. Prosedur

a) Baca dan pahami semua tahapan praktikum dengan cermat.
b) Gunakan fasilitas yang disediakan dengan penuh rasa tanggung jawab.
c) Rapihkan kembali setelah menggunakan komputer (*mouse, keyboard*, kursi, dll)
d) Perhatikan sikap anda untuk tidak mengganggu rekan praktikan lain.
e) Pastikan diri anda tidak menyentuh sumber listrik.

C. Tugas Pendahuluan

1. Jelaskan pengertian dari *inner class*?
2. Buatkan contoh paling sederhana dari inner class
3. Jelaskan apa yang dimaksud dari *exception handling?*
4. Buatkan contoh paling sederhana dari pengemplementasian exception handling

D. Tugas Pendahuluan

1. Inner Class

Inner Class adalah kelas yang dideklarasikan didalam kelas lain. Inner Class memiliki akses kesemua anggota kelas yang memuatnya, termasuk field yang memiliki access modifier private.

Inner Class dapat digunakan untuk mengorganisir kode secara lebih terstruktur dan meningkatkan keterbacaan kode.

Contoh penginisialisasian Inner Class:

```java
package a8prak6a13020190418;

import java.util.ArrayList;
import java.util.List;

public class Mahasiswa {
    private String nama;
    private String nim;
    private String prodi;
    private List<MataKuliah> mataKuliah;

    public Mahasiswa(String nama, String nim, String prodi) {
        this.nama = nama;
        this.nim = nim;
        this.prodi = prodi;
        this.mataKuliah = new ArrayList<>();
    }


    public static class MataKuliah{
        private String namaMataKuliah;
        private int sks;

        public MataKuliah(String namaMataKuliah, int sks) {
            this.namaMataKuliah = namaMataKuliah;
            this.sks = sks;
        }
    }


}
```

Cara pemanggilan inner class

```java
Mahasiswa mahasiswa = new Mahasiswa(nama, nim, prodi);

Mahasiswa.MataKuliah pbo = new Mahasiswa.MataKuliah("Pemrograman Berorientasi Objek", 3);
mahasiswa.tambahMataKuliah(pbo);
```

2. Exception

Exception didalam java adalah sebuah object yang mewakili situasi abnormal yang terjadi saat program dieksekusi. Exception biasanya terjadi karen masalah pada runtime seperti pembagian dengan nol, akses file yang tidak ada, koneksi jaringa terputus dan sebagainya.

Ada banyak macam exception contohnya yaitu ArithmeticException, NullPonterException, ArrayIndexOutOfBoundsException, dan lain sebagainya.

Contoh penginisialisasian salah satu exception

```java
try {
    // Berisikan kode program yang berpotensi menghasilkan error
} catch (Exception exc) {
    // Akan menangkap error yang terjadi pada kode yang ada di dalam blok try
} finally {
    // Kode yang tetap dijalankan baik terjadi error maupun tidak terjadi error
}
```

Contoh penerapan exception handling

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

       double angka1;
       double angka2;
       try {
           System.out.print("Input angka pertama : ");
           angka1 = input.nextInt();
           System.out.print("Input angka kedua : ");
           angka2 = input.nextInt();

           double hasil = angka1 + angka2;
           System.out.println("Hasil : " + hasil);
       } catch (InputMismatchException exc) {
           System.out.println("ERROR, TERDAPAT KESALAHAN DALAM MELAKUKAN INPUT");
       } catch (Exception exc) {
           System.out.println("PROGRAM ERROR");
       }
   }
```

3. Generic Parameter

   Generic di Java adalah sebuah mekanisme yang memungkinkan pengguna untuk membuat kelas, interface, dan metode yang dapat menerima parameter dengan tipe yang bervariasi tanpa harus menuliskan ulang class, method ataupun field.

   Generic parameter dapat digunakan diclass, method, maupun field didalam sebuah class.

Contoh pengimplementasian generic di class

```
public class MyData<T> {

    private T data;

    public MyData(T data) {
        this.data = data;
    }

    public T getData() {
        return this.data;
    }

    public void setData(T data) {
        this.data = data;
    }

}
```

```
public static void main(String[] args) {
    MyData<Integer> data1 = new MyData<>(10);
    MyData<String> data2 = new MyData<>("John");

    Integer data1Value = data1.getData();
    String data2Value = data2.getData();

    System.out.println("Data 1 : " + data1Value);
    System.out.println("Data 2 : " + data2Value);

}
```

Contoh pengimplementasian generic di sebuah method

```
public class ArrayHelper {

    public static <T> void showArray(T[] array) {
        for (var value : array) {
            System.out.println("Element : " + value);
        }
    }

    public static <T> int countArrayLength(T[] array) {
        return array.length;
    }
}
```

```
public static void main(String[] args) {

    String[] stringArray = {"John", "Dhoe"};
    Integer[] integerArray = {1, 2, 3, 4, 5};

    ArrayHelper.showArray(stringArray);
    System.out.println("String Array Length : " + ArrayHelper.countArrayLength(stringArray));

    System.out.println("");

    ArrayHelper.showArray(integerArray);
    System.out.println("Integer Array Length : " + ArrayHelper.countArrayLength(integerArray));
}
```

E. Kegiatan Praktikum

Buka Aplikasi *Netbeans*, buat *project* baru dengan nama sesuai dengan format berikut : **A8Prak6A13020190418**

a. Buat kelas yang dinamakan **Mahasiswa.java** yang mempunyai atribut dan method seperti contoh dibahawa ini

```
public class Mahasiswa {

    private String nama;
    private String nim;
    private String prodi;
    private List<MataKuliah> mataKuliah;
```

```
    public Mahasiswa(String nama, String nim, String prodi) {
        this.nama = nama;
        this.nim = nim;
        this.prodi = prodi;
        this.mataKuliah = new ArrayList<>();
    }

    public void tambahMataKuliah(MataKuliah newMataKuliah) {
        this.mataKuliah.add(newMataKuliah);
    }
```

```
    public void infoMahasiswa() {
        System.out.println("Nama : " + this.nama);
        System.out.println("Nim : " + this.nim);
        System.out.println("Prodi : " + this.prodi);

        System.out.println("====== MATA KULIAH ======");
        for (MataKuliah mataKuliah : this.mataKuliah) {
            System.out.println(mataKuliah.namaMataKuliah + " (" + mataKuliah.sks + " SKS)");
        }
    }

    public static class MataKuliah{
        private String namaMataKuliah;
        private int sks;

        public MataKuliah(String namaMataKuliah, int sks) {
            this.namaMataKuliah = namaMataKuliah;
            this.sks = sks;
        }
    }
```

b. Implementasikan kelas yang sudah dibuat pada method **main**

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

    System.out.println("==== DAFTAR MAHASISWA ====");
    System.out.print("Input Nama  : ");
    String nama = input.nextLine();
    System.out.print("Input NIM   : ");
    String nim = input.nextLine();
    System.out.print("Input Prodi : ");
    String prodi = input.nextLine();

    Mahasiswa mahasiswa = new Mahasiswa(nama, nim, prodi);
```

```
    String pilih;
    do {
        System.out.println("\n=== Pilih Mata Kuliah ===");
        System.out.println("1. Pemrograman Berorientasi Objek");
        System.out.println("2. Algoritma dan Pemrograman 2");
        System.out.println("3. Jaringan Komputer");
        System.out.println("4. Selesai");
        System.out.print("Pilih MataKuliah : ");
        pilih = input.nextLine();

        switch (pilih) {
            case "1":
                Mahasiswa.MataKuliah pbo = new Mahasiswa.MataKuliah("Pemrograman Berorientasi Objek", 3);
                mahasiswa.tambahMataKuliah(pbo);
                System.out.println("Berhasil menambahkan matakuliah");
                break;
            case "2":
                Mahasiswa.MataKuliah alpro = new Mahasiswa.MataKuliah("Algoritma dan Pemrograman 2", 3);
                mahasiswa.tambahMataKuliah(alpro);
                System.out.println("Berhasil menambahkan matakuliah");
                break;
            case "3":
                Mahasiswa.MataKuliah jarkom = new Mahasiswa.MataKuliah("Jaringan Komputer", 3);
                mahasiswa.tambahMataKuliah(jarkom);
                System.out.println("Berhasil menambahkan matakuliah");
                break;
            default:
                System.out.println("Menu tidak ditemukan");
                break;
        }

    } while (!pilih.equals("4"));

    System.out.println("\n");
    mahasiswa.infoMahasiswa();
```

LEMBAR EVALUASI PRAKTIKUM

1. Berikan contoh penerapan generic class dan generic method
2. Tuliskan jenis-jenis exception error pada bahasa pemrograman java dan berikan penjelasan pada masing-masing jenis exception
3.
```
int[] arrays = {5, 2, 1, 5, 10};
```
Sebutkan jenis exception yang akan terjadi jika kita mengakses index ke-5 pada contoh array di atas

Evaluasi Praktikum 6:

No	Indikator	Skor Penilaian				
		Sangat Kurang (E) <=40	**Kurang (D) 41-55**	**Cukup (C) 55-65**	**Baik (B) 66-85**	**Sangat Baik (A) >=86**
1.	**Memahami Konsep Dari *Inner Class***					
2	**Dapta Membuat *Inner Class***					
3	**Memahami Konsep dari *Exception Handling***					
4	**Dapat membuat *Class Exception Handling***					

Catatan Asisten:

Asisten 1 : _____

Asisten 2 : _____

MODUL 7 – GRAPHICAL USER INTERFACE (GUI)

A. Sub Capaian Pembelajaran Mata Kuliah (CPMK)

1. Mahasiswa dapat memahami konsep dari GUI di Java
2. Mahasiswa dapat mengimplementasikan konsep dari GUI dengan menggunakan bahasa Pemrograman Java

B. Instrument dan Prosedur

1. **Instrument**
   a) Perangkat komputer
   b) Sistem Operasi Windows
   c) Java Development KIT (JDK)
   d) IDE Netbeans
2. **Prosedur**
   a) Baca dan pahami semua tahapan praktikum dengan cermat.
   b) Gunakan fasilitas yang disediakan dengan penuh rasa tanggung jawab.
   c) Rapihkan kembali setelah menggunakan komputer (*mouse, keyboard*, kursi, dll)
   d) Perhatikan sikap anda untuk tidak mengganggu rekan praktikan lain.
   e) Pastikan diri anda tidak menyentuh sumber listrik.

C. Teori Dasar

GUI adalah singkatan dari *Graphical User Interface*, yaitu sebuah antarmuka yang memungkinkan pengguna untuk berinteraksi dengan program melalui tampilan visual, seperti tombol, teks, gambar, dan elemen-elemen grafis lainnya,untuk membuat GUI dalam Java, Anda harus membuat sebuah kelas yang meng-*extend* kelas JFrame atau JDialog, kemudian menambahkan komponen-komponen GUI ke dalamnya, seperti JLabel, JButton, JTextField, dan sebagainya. Komponen-komponen ini dapat diatur posisinya dengan menggunakan *layout manager*, seperti *BorderLayout*, *GridLayout*, atau *FlowLayout*

a. Jlabel

JLabel adalah salah satu komponen GUI (*Graphical User Interface*) di Java yang digunakan untuk menampilkan teks atau gambar pada sebuah *frame* atau *panel*.

b. JTextField

JTextField adalah komponen grafis pada pemrograman Java yang digunakan untuk membuat sebuah kotak inputan atau input text yang dapat diisi oleh pengguna dengan teks atau angka.

c. JPasswordField

JPasswordField adalah sebuah komponen pada bahasa pemrograman Java yang digunakan untuk mengambil input kata sandi (*password*) dari pengguna aplikasi. Komponen ini digunakan untuk menyembunyikan karakter yang dimasukkan oleh pengguna, sehingga karakter-karakter tersebut tidak akan ditampilkan di layar.

d. JButton

JButton digunakan untuk membuat tombol yang bisa diklik oleh pengguna untuk melakukan sesuatu, seperti menyimpan data, membatalkan aksi, atau melanjutkan ke langkah selanjutnya.

e. JToggleButton

JToggleButton adalah sebuah komponen antarmuka grafis pada Java yang dapat berfungsi sebagai tombol on/off atau tombol yang dapat ditoggle atau dinyalakan dan dimatikan. JToggleButton biasanya digunakan pada aplikasi desktop yang membutuhkan kontrol untuk mengaktifkan atau menonaktifkan suatu fitur atau opsi tertentu.

f. JTable

JTable biasanya digunakan untuk menampilkan data dalam bentuk tabel seperti pada program aplikasi database atau aplikasi bisnis yang membutuhkan tampilan data dalam bentuk tabel. jTable dapat menampilkan data dalam format yang berbeda-beda, seperti format angka, teks, tanggal, atau gambar.

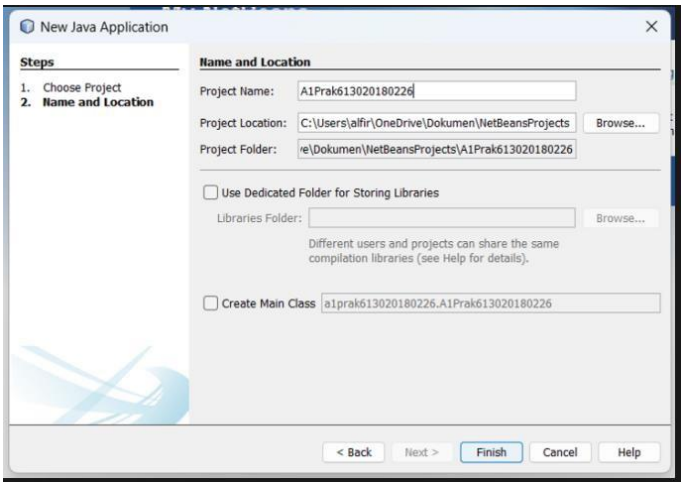
D. Kegiatan Praktikum

1. Studi Kasus A

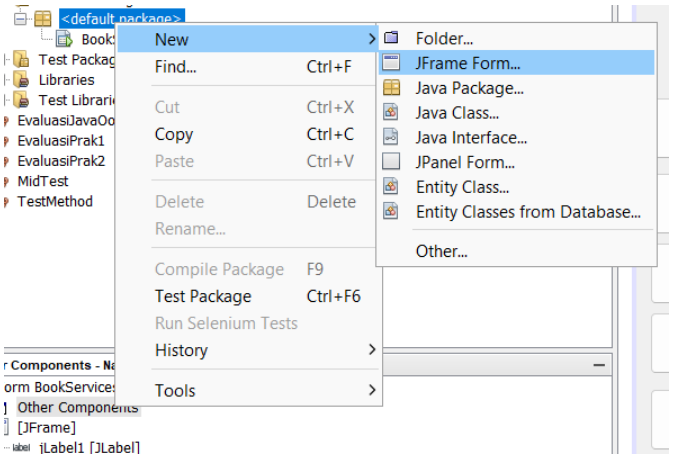
a. Buka Aplikasi Netbeans, buat project baru dengan nama sesuai dengan format berikut :

KelasPrak6STB. Contoh: A1Prak613020180226

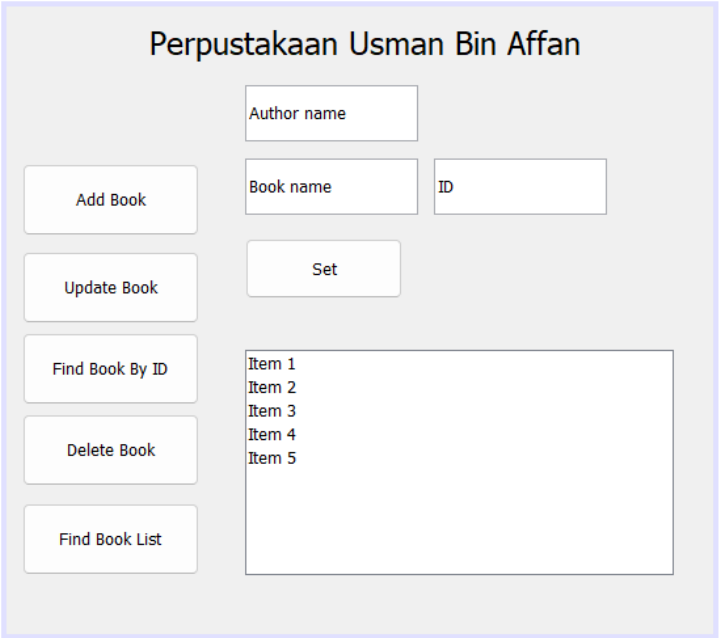
b. Kemudian *uncheck Create* Main *Class* seperti pada gambar berikut :



c. kemudian buat sebuah *Package* sesuai dengan nama *project* yang telah di buat kemudian di dalam *package* yang dibuat tambahkan file baru berupa JFremeForm dengan nama **BookServices** seperti pada contoh gambar berikut :



d. Setelah itu buat UI sesuai dengan gambar di bawah ini :

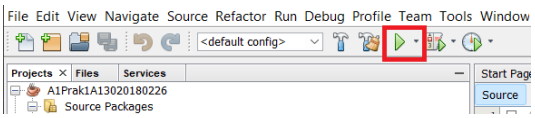


e. Selanjutnya ikuti kodingan berikut :

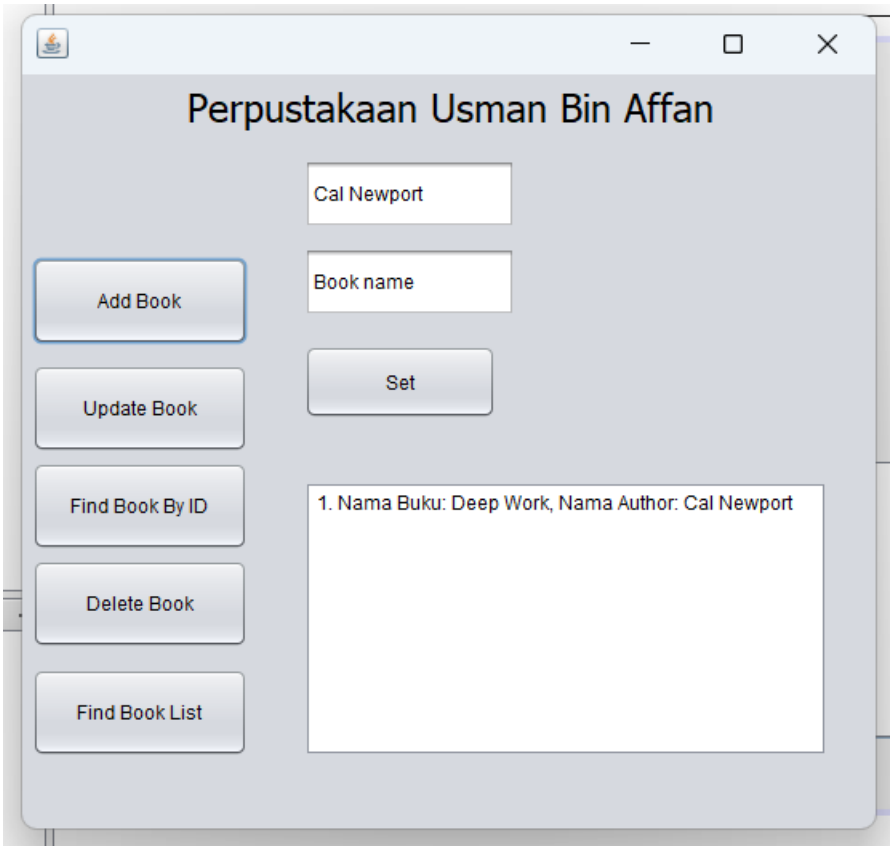
```
import java.util.ArrayList;
import javax.swing.DefaultListModel;
import javax.swing.JList;


public class BookServices extends javax.swing.JFrame {

    private String menu = "";
    private ArrayList<String> bookName = new ArrayList<>();
    private ArrayList<String> authorName = new ArrayList<>();

    private void changeAllVisible(boolean value) {
        fieldBookName.setVisible(value);
        fieldAuthorName.setVisible(value);
        fieldId.setVisible(false);
        btnSet.setVisible(value);
    }

    private void changeAllVisibleUpdate(boolean value) {
        fieldBookName.setVisible(value);
        fieldAuthorName.setVisible(value);
        fieldId.setVisible(value);

        btnSet.setVisible(value);
    }

    void changeBookNameVisible(boolean value) {
        changeAllVisible(false);
        fieldBookName.setVisible(value);
        btnSet.setVisible(value);
    }

    void addBook() {
        bookName.add(fieldBookName.getText());
        authorName.add(fieldAuthorName.getText());
    }

    void findListBook() {

        DefaultListModel<String> model = new DefaultListModel<>();

        for (int i = 0; i < bookName.size(); i++) {
            String outputText = "";
            outputText += (i + 1) + ". ";
            outputText += "Nama Buku: " + bookName.get(i) + ", ";
            outputText += "Nama Author: " + authorName.get(i);
            model.addElement(outputText);
        }

        jList1.setModel(model);
    }

    void findListBookById(int i) {

        DefaultListModel<String> model = new DefaultListModel<>();

        String outputText = "";
        outputText += (i) + ". ";
        outputText += "Nama Buku: " + bookName.get(i - 1) + ", ";
        outputText += "Nama Author: " + authorName.get(i - 1);
        model.addElement(outputText);

        jList1.setModel(model);
    }

    void deleteBook(int i) {
        bookName.remove(i - 1);
    }

    void changeMenu(String menu) {
        this.menu = menu;
    }

    void updateBook() {
        int id = Integer.parseInt(fieldId.getText()) -1;
        bookName.set(id, fieldBookName.getText());
        authorName.set(id, fieldAuthorName.getText());
    }

    public BookServices() {
        initComponents();
        changeAllVisible(false);
        panelOutput.setVisible(true);
    }
```

```
private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {

        fieldBookName.setText("Book name");
        changeBookNameVisible(false);
        changeAllVisible(true);
        changeMenu("add");
    }

    private void btnFindBookActionPerformed(java.awt.event.ActionEvent evt) {

        findListBook();
        changeAllVisible(false);
        panelOutput.setVisible(true);

    }

    private void btnFindByIdActionPerformed(java.awt.event.ActionEvent evt) {

        fieldBookName.setText("ID Book");
        changeBookNameVisible(true);
        changeMenu("findById");
        findListBook();

    }

    private void btnUpdateActionPerformed(java.awt.event.ActionEvent evt) {

        changeAllVisible(false);

        changeAllVisibleUpdate(true);
        changeMenu("update");

    }

    private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {

        fieldBookName.setText("ID Book");
        changeAllVisible(false);

        changeBookNameVisible(true);
        changeMenu("delete");
    }

    private void fieldBookNameActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
    }

    private void fieldAuthorNameActionPerformed(java.awt.event.ActionEvent evt) {

        // TODO add your handling code here:
    }

    private void btnSetActionPerformed(java.awt.event.ActionEvent evt) {

        switch (menu) {
            case "add":
                addBook();
                break;

            case "delete":
                deleteBook(Integer.parseInt(fieldBookName.getText()));
                break;

            case "update":
                updateBook();
                break;

            case "findById":
                findListBookById(Integer.parseInt(fieldBookName.getText()));
                break;
        }
    }
```

f. Selanjutnya tekan tombol *run* di bagian atas yang berbentuk tombol *play* berwarna hijau seperti gambar di bawah atau dengan menekan *shortcut* (*Shift*+f6) untuk melakukan *compile* dan *run*.



g. Setelah itu ketika program berjalan tanpa ada error maka akan tampil output seperti gambar di bawah ini :



LEMBAR EVALUASI PRAKTIKUM

1. Jelaskan apa yang terjadi pada *function* clickHitung, clickButton, dan clickOperator
2. John Due adalah seorang mahasiswa teknik informatika yang tertarik dengan pengembangan aplikasi Java. Dia ingin membuat program sederhana untuk menghitung berat massa tubuh ideal seseorang berdasarkan tinggi dan berat badan mereka. Program ini akan menggunakan konsep GUI (*Graphical User Interface*) dan akan menampilkan hasil perhitungan berat massa tubuh *ideal* seseorang. bantu John Due dalam membuat aplikasinya.

   Note :

   Rumus untuk menghitung berat massa tubuh (BMI - Body Mass Index) adalah sebagai berikut:

   BMI = berat (kg) / (tinggi (m))^2

   Berikut adalah beberapa kategori BMI dan arti dari setiap kategori:

   - Kurang dari 18,5: Kurang berat badan
   - 18,5 - 24,9: Normal
   - 25 - 29,9: Kelebihan berat badan
   - 30 - 34,9: Obesitas kelas 1
   - 35 - 39,9: Obesitas kelas 2
   - 40 atau lebih: Obesitas kelas 3

Evaluasi Praktikum 7:

No	Indikator	Skor Penilaian				
		Sangat Kurang (E) <=40	Kurang (D) 41-55	Cukup (C) 55-65	Baik (B) 66-85	Sangat Baik (A) >=86
1.	Memahami Konsep GUI					
2.	Dapat Mengimplementasikan Konsep GUI					

Catatan Asisten:

Asisten 1 : _____

Asisten 2 : _____

MODUL 8 – JAVA DATABASE CONNECTIVITY (JDBC)

A. Sub Capaian Pembelajaran Mata Kuliah (CPMK)

1. Mahasiswa dapat memahami konsep dari JDBC
2. Mahasiswa dapat mengimplementasikan konsep dari JDBC menggunakan bahasa Pemrograman Java

B. Instrument dan Prosedur

1. Instrument

a) Perangkat komputer
b) Sistem Operasi Windows
c) Java Development KIT (JDK)
d) IDE Netbeans

2. Prosedur

a) Baca dan pahami semua tahapan praktikum dengan cermat.
b) Gunakan fasilitas yang disediakan dengan penuh rasa tanggung jawab.
c) Rapihkan kembali setelah menggunakan komputer (*mouse, keyboard,* kursi, dll)
d) Perhatikan sikap anda untuk tidak mengganggu rekan praktikan lain.
e) Pastikan diri anda tidak menyentuh sumber listrik.

C. Tugas Pendahuluan

1. Jelaskan apa yang dimaksud dengan JDBC?
2. Sebutkan jenis-jenis driver JDBC?
3. Jelaskan jenis-jenis JDBC?

D. Teori Dasar

JDBC (*Java Database Connectivity*) adalah sebuah API (*Application Programming Interface*) yang digunakan untuk mengakses dan mengelola basis data (database) menggunakan bahasa pemrograman Java. Dengan menggunakan JDBC, programmer dapat menghubungkan aplikasi Java ke basis data yang berbeda-beda, seperti MySQL, Oracle, Microsoft SQL Server, dan sebagainya.Untuk menggunakan JDBC, Anda perlu menambahkan driver JDBC untuk basis data yang Anda gunakan ke dalam proyek Java Anda. Setelah itu, Anda dapat membuat koneksi ke basis data dengan memanggil *method getConnection*() dari *class DriverManager*, dan menyediakan parameter seperti URL basis data, *username*, dan *password*.Setelah terhubung ke basis data, Anda dapat melakukan operasi CRUD (*Create*, *Read*, *Update*, *Delete*) pada tabel-tabel dalam basis data, seperti menambahkan data baru, membaca data yang sudah ada, memperbarui data yang sudah ada, atau menghapus data.

```
import java.sql.*;
public class JDBC {
    public static void main(String[] args) {
        try{
                Class.forName("com.mysql.jdbc.Driver");
                Connection con=DriverManager.getConnection(
                  "jdbc:mysql://localhost:3306/users","root","");
                   //disini users adalaha database, root sebagai username
                   //dan  string kosong sebagai password
}catch(Exception e){ System.out.println(e);}
}
    }
```

E. Kegiatan Praktikum

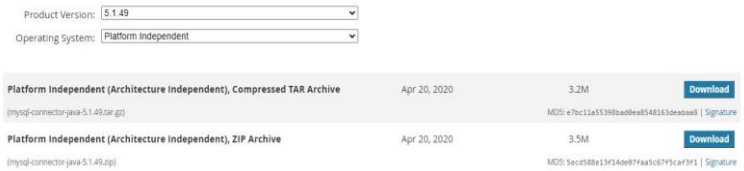
A. Studi Kasus A

a. Buka Aplikasi *Netbeans*, buat *project* baru dengan nama sesuai dengan format berikut :

KelasPrak7STB. Contoh: A1Prak713020190418

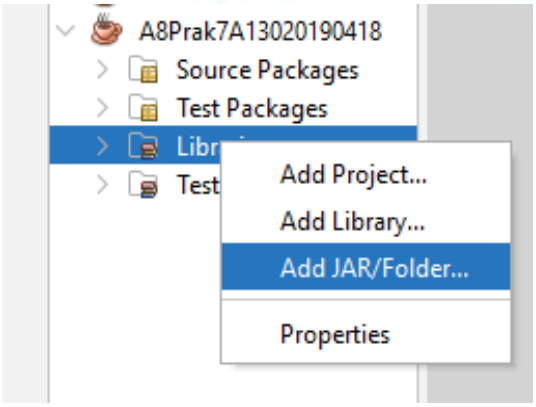
b. Langkah selanjutnya, download driver JDBC MySQL pada link berikut:

https://downloads.mysql.com/archives/c-j/

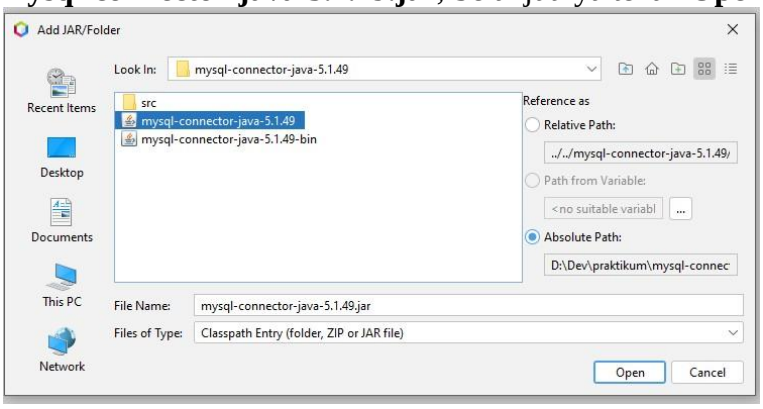


Download sesuai platform masing-masing. Pada praktikum kali ini kita menggunakan JDBC "**Platform Independent (Architecture Independent), ZIP. Archive**".

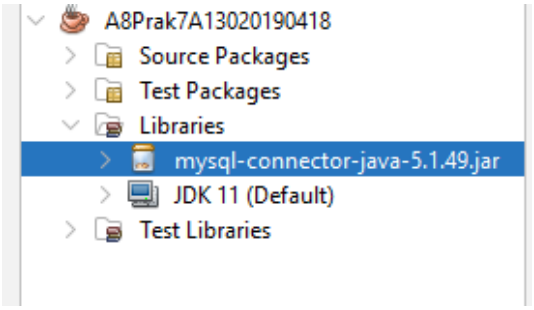
c. Setelah file selesai di download, silahkan lakukan extract file .zip
d. Selanjutnya, kembali ke project yang telah dibuat pada Aplikasi Netbeans. Kemudian klik kanan pada package **Libraries**, kemudian pilih menu **Add JAR/Folder.**



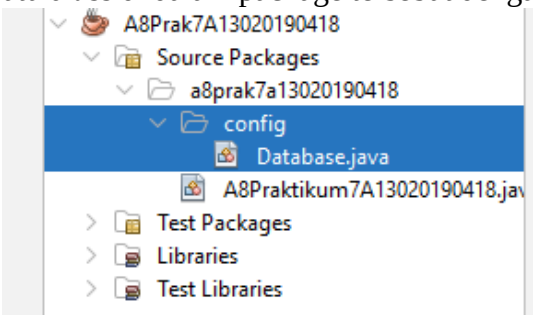
Kemudian masuk ke folder JDBC yang sudah di extract sebelumnya, kemudian pilih file **mysql-connector-java-5.1.49.jar**, Selanjutnya tekan **Open**



Pastikan file **mysql-connector-java-5.1.49.jar** sudah terdaftar pada package **Libraries**



e. Selanjutnya buatlah *package* yang bernama ***config***, kemudian di dalam *package* tersebut, buatlah satu *class* di dalam package tersebut dengan nama **Database.java**



Di dalam *class* **Database.java** yang telah di buat, nantinya akan berisi konfigurasi yang digunakan untuk membuat koneksi ke database

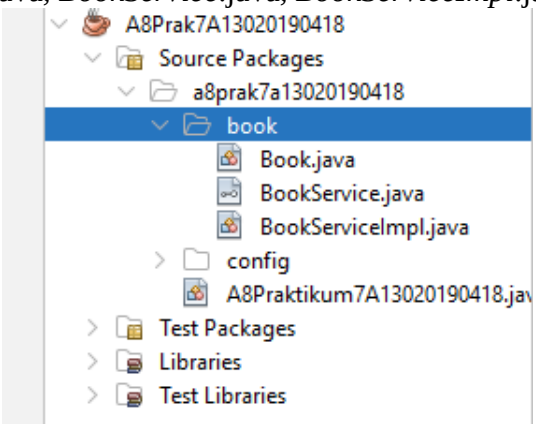
```
package a8prak7a13020190418.config;

import com.mysql.jdbc.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class Database {
    private static Connection connection;
    public static Connection DBConnection() {
        String dbUrl = "jdbc:mysql://localhost:3306/db_praktikum";
        String user = "root";
        String password = "";

        try {
            DriverManager.registerDriver(new com.mysql.jdbc.Driver());
            connection = (Connection) DriverManager.getConnection(dbUrl, user, password);
```

```
        } catch (SQLException exc) {
            System.out.println("Koneksi error : " + exc.getMessage());
        }
        return connection;
    }

    public static void closeConnection() {
        try {
        connection.close();
        } catch (SQLException exc) {
            System.out.println("FAILED TO CLOSE DATABASE CONNECTION : " + exc.getMessage());
        }
    }
}
```

f. Selanjutnya, jalankan mysql pada xampp, kemudian buat database yang bernama **db_praktikum**, kemudian buatlah table **book** dan mempunyai 3 kolom yaitu **id**, **title**, **author_name**.

```
MariaDB [db_praktikum]> CREATE table book (
    -> id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    -> title VARCHAR(50) NOT NULL,
    -> author_name VARCHAR(50)
    -> ) Engine = InnoDB;

MariaDB [db_praktikum]> DESC book;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| id          | int(11)     | NO   | PRI | NULL    | auto_increment |
| title       | varchar(50) | NO   |     | NULL    |                |
| author_name | varchar(50) | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
3 rows in set (0.015 sec)

MariaDB [db_praktikum]>
```

g. Setelah mengatur konfigurasi yang dibutuhkan untuk melakukan koneksi ke database, selanjutnya buatlah *package* bernama *book*, yang di dalamnya berisikan 3 *class* yaitu, *Book*.java, *BookService*.java, *BookServiceImpl*.java.



h. ***Book*.java** : digunakan sebagai model dari kolom yang ada di database

```java
package a8prak7a13020190418.book;

public class Book {

    private int id;
    private String authorName;
    private String title;

    public Book(String title, String authorName) {
        this.title = title;
        this.authorName = authorName;
    }

    public Book() {
    }

    public int getId() {
        return id;
    }

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

    public String getAuthorName() {
        return authorName;
    }

    public void setAuthorName(String authorName) {
        this.authorName = authorName;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }
}
```

i. ***BookService*.java** : digunakan sebagai class kontrak untuk operasi atau *method-method* apa saja yang digunakan nantinya.

```java
package a8prak7a13020190418.book;

import java.util.List;

public interface BookService {
    public void addBook(Book book);
    public List<Book> findBookList();
    public Book findBookById(int id);
    public void updateBook(Book book);
    public void removeBook(int id);
}
```

j. ***BookServiceImpl*.java** : digunakan sebagai *class* yang akan menjadi *class implements* dari *class BookService* yang dibuat sebelumnya.

```java
package a8prak7a13020190418.book;

import a8prak7a13020190418.config.Database;
import com.mysql.jdbc.Connection;
import com.mysql.jdbc.PreparedStatement;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class BookServiceImpl implements BookService {

    private Connection connection = Database.DBConnection();
    private PreparedStatement statement;

    @Override
    public void addBook(Book book) {
        try {
            String query = "INSERT INTO book VALUES (0, ?, ?)";
            statement = (PreparedStatement) connection.prepareStatement(query);
            statement.setString(1, book.getTitle());
            statement.setString(2, book.getAuthorName());
            statement.executeUpdate();

            System.out.println("Book has been added!\n");
            statement.close();
        } catch (SQLException exc) {
            System.out.println("FAILED TO ADD BOOK " + exc.getMessage());
        }
    }

    @Override
    public List<Book> findBookList() {
        List<Book> books = new ArrayList<>();

        try {

            String query = "SELECT * FROM book";
            statement = (PreparedStatement) connection.prepareStatement(query);
            ResultSet result = statement.executeQuery();
            while (result.next()) {
                Book book = new Book();
                book.setId(result.getInt("id"));
                book.setTitle(result.getString("title"));
                book.setAuthorName(result.getString("author_name"));

                books.add(book);
            }
            statement.close();
            return books;
        } catch (SQLException exc) {
            System.out.println("FAILED TO GET BOOK LIST: " + exc.getMessage());
        }
        return books;
    }

    @Override
    public Book findBookById(int bookId) {
        Book book = new Book();
        try {

            String query = "SELECT * FROM book WHERE id = ?";
            statement = (PreparedStatement) connection.prepareStatement(query);
            statement.setInt(1, bookId);

            ResultSet result = statement.executeQuery();
            if (result.next()) {
                int id = result.getInt("id");
                String title = result.getString("title");
                String authorName = result.getString("author_name");

                book.setId(id);
                book.setTitle(title);
                book.setAuthorName(authorName);

            }
            statement.close();
            return book;
        } catch (SQLException exc) {
            System.out.println("FAILED TO GET BOOK : " + exc.getMessage());
        }
        return book;
    }
```

```
    @Override
    public void updateBook(Book book) {
        try {

            String query = "UPDATE book SET title = ?, author_name = ? WHERE id = ?";
            statement = (PreparedStatement) connection.prepareStatement(query);
            statement.setString(1, book.getTitle());
            statement.setString(2, book.getAuthorName());
            statement.setInt(3, book.getId());
            statement.executeUpdate();

            System.out.println("Successfully updated the book with id = " + book.getId());
            System.out.println("\n");
            statement.close();
        } catch (SQLException exc) {
            System.out.println("FAILED TO UPDATE BOOK DATA : " + exc.getMessage());
        }
    }

    @Override
    public void removeBook(int id) {
        try {

            String query = "DELETE FROM book WHERE id = ?";
            statement = (PreparedStatement) connection.prepareStatement(query);
            statement.setInt(1, id);
            statement.executeUpdate();

            System.out.println("Successfully delete book!\n");
            statement.close();
        } catch (SQLException exc) {
            System.out.println("FAILED TO DELETE BOOK DATA : " + exc.getMessage());
        }
    }
}
```

k. Selanjutnya implementasikan *method* yang sudah dibuat pada *class* ***BookServiceImpl*.java** di fungsi main

```
package a8prak7a13020190418;

import a8prak7a13020190418.book.Book;
import a8prak7a13020190418.book.BookService;
import a8prak7a13020190418.book.BookServiceImpl;
import a8prak7a13020190418.config.Database;
import java.util.List;
import java.util.Scanner;
public class A8Praktikum7A13020190418 {
    public static void main(String[] args) {
        BookService bookService = new BookServiceImpl();
        int menuInput = 0;

        System.out.println("Library Program");
        System.out.println("===============");

       do {

            Scanner scanner = new Scanner(System.in);
            System.out.println("1. Add Book");
            System.out.println("2. Find Book List");
            System.out.println("3. Find Book By Id");
            System.out.println("4. Update Book");
            System.out.println("5. Delete Book");
            System.out.println("6. Exit");

            System.out.print("\nSelect Menu : ");
            menuInput = scanner.nextInt();
```

```
switch (menuInput) {
                case 1:
                    System.out.println("========");
                    System.out.println("Add Book");
                    System.out.println("========");

                    scanner.nextLine();

                    System.out.print("Book Title : ");
                    String title = scanner.nextLine();

                    System.out.print("Author Name : ");
                    String authorName = scanner.nextLine();

                    Book newBook = new Book(title, authorName);
                    bookService.addBook(newBook);
                    break;
                case 2:
                    System.out.println("==============");
                    System.out.println("Find Book List");
                    System.out.println("==============");
```

```
            List<Book> books = bookService.findBookList();
                if (books.size() < 1) {
                    System.out.println("No books yet\n");
                } else {
                    for (Book book : books) {
                        System.out.println("ID          : " + book.getId());
                        System.out.println("Title       : " + book.getTitle());
                        System.out.println("Author Name : " + book.getAuthorName());
                        System.out.println("\n");
                    }
                }
                break;
            case 3:
                System.out.println("==============");
                System.out.println("Find Book By Id");
                System.out.println("==============");

                System.out.print("Book id : ");
                int bookId = scanner.nextInt();
```

```
Book book = bookService.findBookById(bookId);
                if (book != null) {
                    System.out.println("ID          : " + book.getId());
                    System.out.println("Title       : " + book.getTitle());
                    System.out.println("Author Name : " + book.getAuthorName());

                    System.out.println("\n");
                } else {
                    System.out.println("No books yet\n");
                }
                break;
             case 4:
                System.out.println("==========");
                System.out.println("Update Book");
                System.out.println("==========");

                System.out.print("Update book id : ");
                int bookIdUpdate = scanner.nextInt();

                scanner.nextLine();

                System.out.print("\nBook Title : ");
                String newTitle = scanner.nextLine().substring(0);
                System.out.print("Author Name : ");
                String newAuthorName = scanner.nextLine();

                Book bookUpdate = new Book();
                bookUpdate.setId(bookIdUpdate);
                bookUpdate.setTitle(newTitle);
                bookUpdate.setAuthorName(newAuthorName);
```

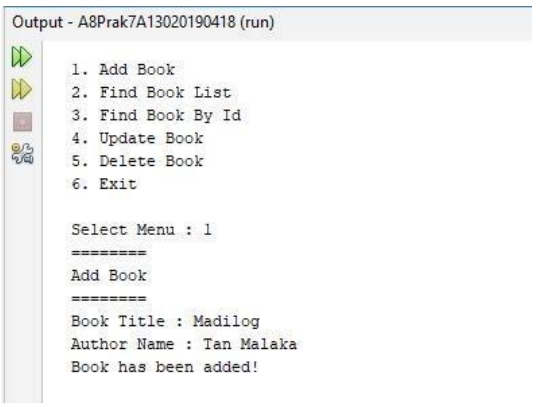
```
bookService.updateBook(bookUpdate);
                    break;
                case 5:
                    System.out.println("===========");
                    System.out.println("Remove Book");
                    System.out.println("===========");

                    System.out.print("Book id : ");
                    int bookIdRemove = scanner.nextInt();
                    bookService.removeBook(bookIdRemove);
                    break;
                case 6:
                    System.out.println("Program finished!");
                    Database.closeConnection();
                    break;
                default:
                    System.out.println("Invalid Menus!");
                    break;
            }
        } while (menuInput != 6);
    }
}
```

l. Jalankan program dan lihat hasilnya

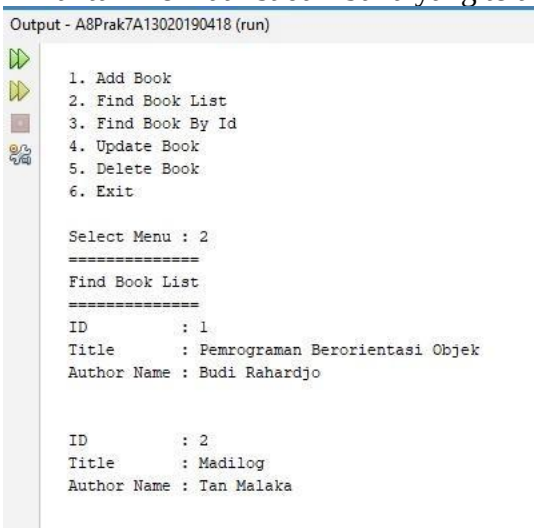
Jika kita memilih untuk menambahkan buku:

```
Output - A8Prak7A13020190418 (run)
run:
Library Program
===============
1. Add Book
2. Find Book List
3. Find Book By Id
4. Update Book
5. Delete Book
6. Exit

Select Menu : 1
========
Add Book
========
Book Title : Pemrograman Berorientasi Objek
Author Name : Budi Rahardjo
Book has been added!

1. Add Book
2. Find Book List
3. Find Book By Id
4. Update Book
5. Delete Book
6. Exit
```

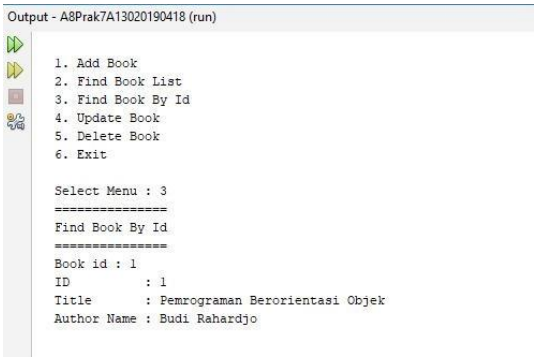
Kita memilih untuk menambahkan buku ke- 2:



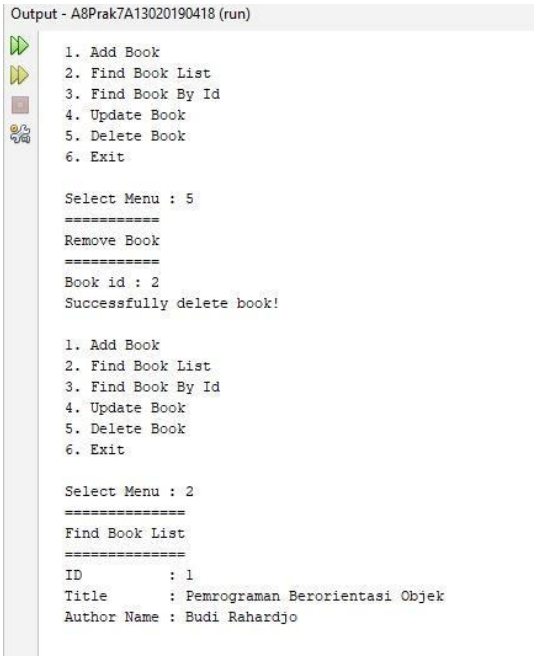
Lalu kita memilih untuk melihat list dari buku yang telah kita masukan:



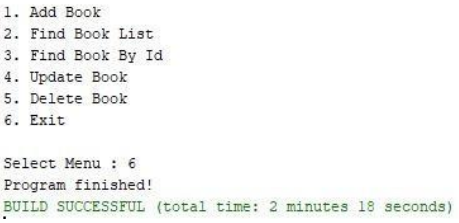
Lalu pilih menu untuk mencari buku berdasarkan id dari buku tersebut:



Lalu pilih menu untuk mengahapus buku berdasarkan id dari buku:



Lalu pilih menu untuk exit



LEMBAR EVALUASI PRAKTIKUM

1. Buatlah sebuah program yang terhubung dengan database yang dapat membuat, mengupdate, menghapus, mencari, dan list semua data yang ada didalam database!

Evaluasi Praktikum 8:

No	Indikator	Skor Penilaian				
		Sangat Kurang (E) <=40	**Kurang (D) 41-55**	**Cukup (C) 55-65**	**Baik (B) 66-85**	**Sangat Baik (A) >=86**
1.	**Dapat Memahami Konsep JDBC**					
2.	**Dapat Membuat Sebuah Program Yang TerhubungKe Database**					

Catatan Asisten:

Asisten 1 :_____

Asisten 2 : _____