Pemograman Berorientasi Objek

Febri Damatraseta Fairuz, S.T, M.Kom

Agenda

01
Section

Introduction
OOP vs Procedural

02
Section

Documentation UML
USE CASE

03
Section

Documentation UML
CLASS DIAGRAM

04
Section

Documentation UML
ACTIVITY DIAGRAM

05
Section

CASE STUDY

06
Section

CASE STUDY

Agenda

07
Section
PROJECT
REVIEW JOURNAL

08
Section
PROJECT
REVIEW JOURNAL

09
Section
CASE STUDY

10
Section
CASE STUDY

11
Section
PROJECT
JAVA APPS

12
Section
PROJECT
JAVA APPS II

Kampus Merdeka
INDONESIA JAYA

INSTITUT BISNIS DAN INFORMATIKA KESATUAN
KESATUAN
BOGOR

01

Prosedural VS OOP

Perbedaan antar metode

Difference Metode Programming

Kampus Merdeka INDONESIA JAYA

INSTITUT BISNIS DAN INFORMATIKA KESATUAN BOGOR

Prosedural

Metode pemograman yang teridiri dari Fungsi dan Logic

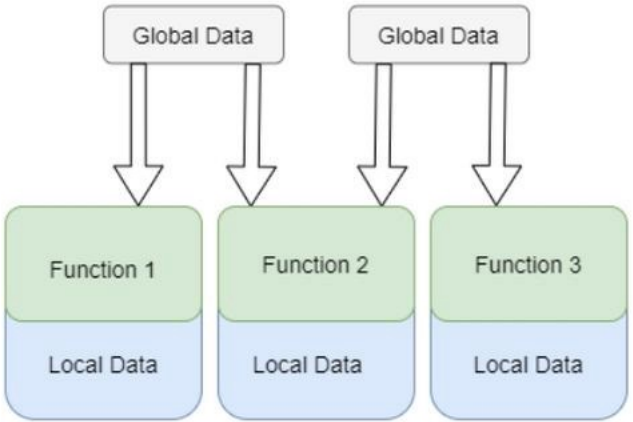
OOP

Metode pemograman yang berorientasi pada Objek. Dimana memiliki variable dan fungsi yang dibungkus kedalam Objek atau Class

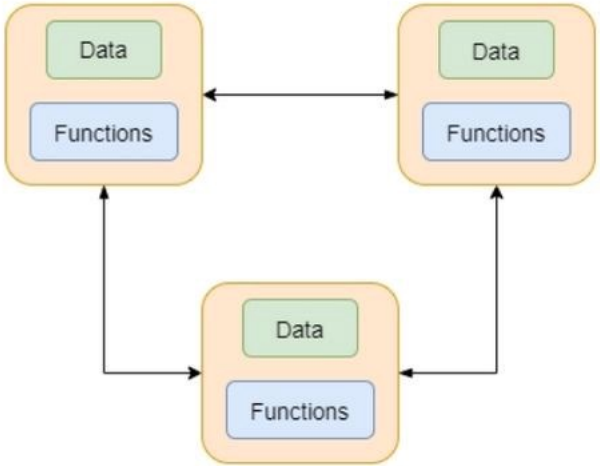
Prosedural vs OOP

Architecture

Procedural Oriented Programming



Object Oriented Programming



Prosedural vs OOP

Programming

Prosedural Code

```
#include <iostream>
using namespace std;

void add(){
  …
}

int main() {
  add();
  return 0;
}
```

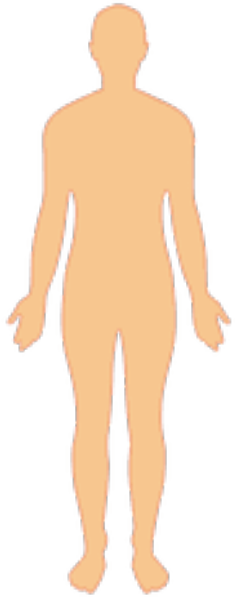
OOP Code

```
class Operator{
   void add(){
     …
   }
}

Object = Operator();
```

Prosedural

Programming

Prosedural Code

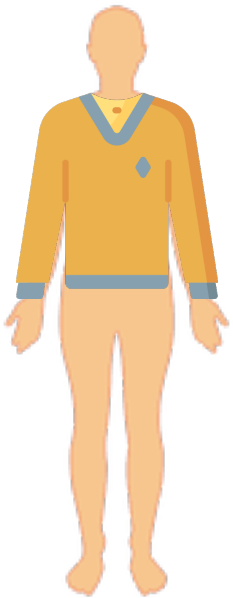


```
#include <iostream>
using namespace std;

void person(){
  string name = “Mahesa”;
  cout << “My Name is ” << name;
}

int main() {
  person();
  return 0;
}
```

Prosedural

Programming

Prosedural Code



```
#include <iostream>
using namespace std;

void person(){
   string name = “Mahesa”;
  cout << “My Name is ” << name;
}

void clothes(){
  cout << “Sweater orange”;
}

int main() {
  person();
  clothes();
  return 0;
}
```

Prosedural

Programming

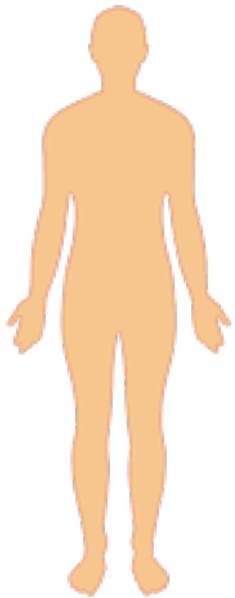
Prosedural Code

```
#include <iostream>
using namespace std;

void person(){
  string name = “Mahesa”;
  cout << “My Name is ” << name;
}

void clothes(){
  cout << “Sweater orange”;
}

void pants(){
  cout << “Short pants”;
}

int main() {
  person();
  clothes();
  pants();
  return 0;
}
```

OOP

Programming

OOP Code



```
class Person{
  String name = “Mahesa”;
  float weight = 65.5;
  int height = 60;

  void Biodata(){
    System.out.println(“My Name is”+name);
  }
}
```

OOP Code

```
class Clothes{
  String color = “orange”;
  char size = “M”;

  void Sweater(){
    System.out.println(“I have sweater ”+color);
  }

  void Jacket(){
    System.out.println(“My jacket ”+color+” has size ”+size);
  }
}
```

OOP

Programming

OOP Code

```
class Pants{
  String type = “joger”;

  void Short(){
    System.out.println(“I am wearing short pants”);
  }

  void Long(){
    System.out.println(“I am wearing a ”+type);
  }
}
```

OOP

Programming

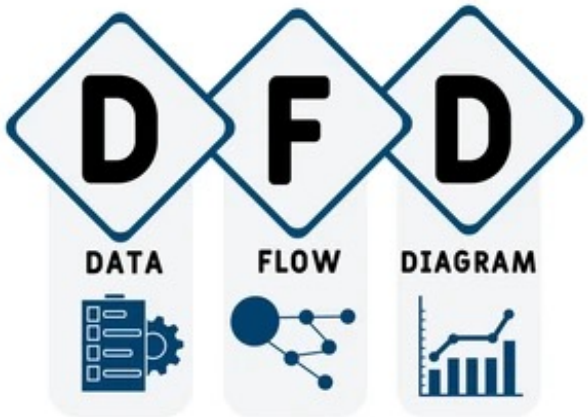
OOP Code

```
class Human{
   Person person = new Person();
   Clothes clothes = new Clothes();
   Pants pants = new Pants();

   static void main(String[] args){
     person.Biodata();
     clothes.Sweater()
     pants.Short();
   }
}
```

Prosedural vs OOP

Documentation

Prosedural



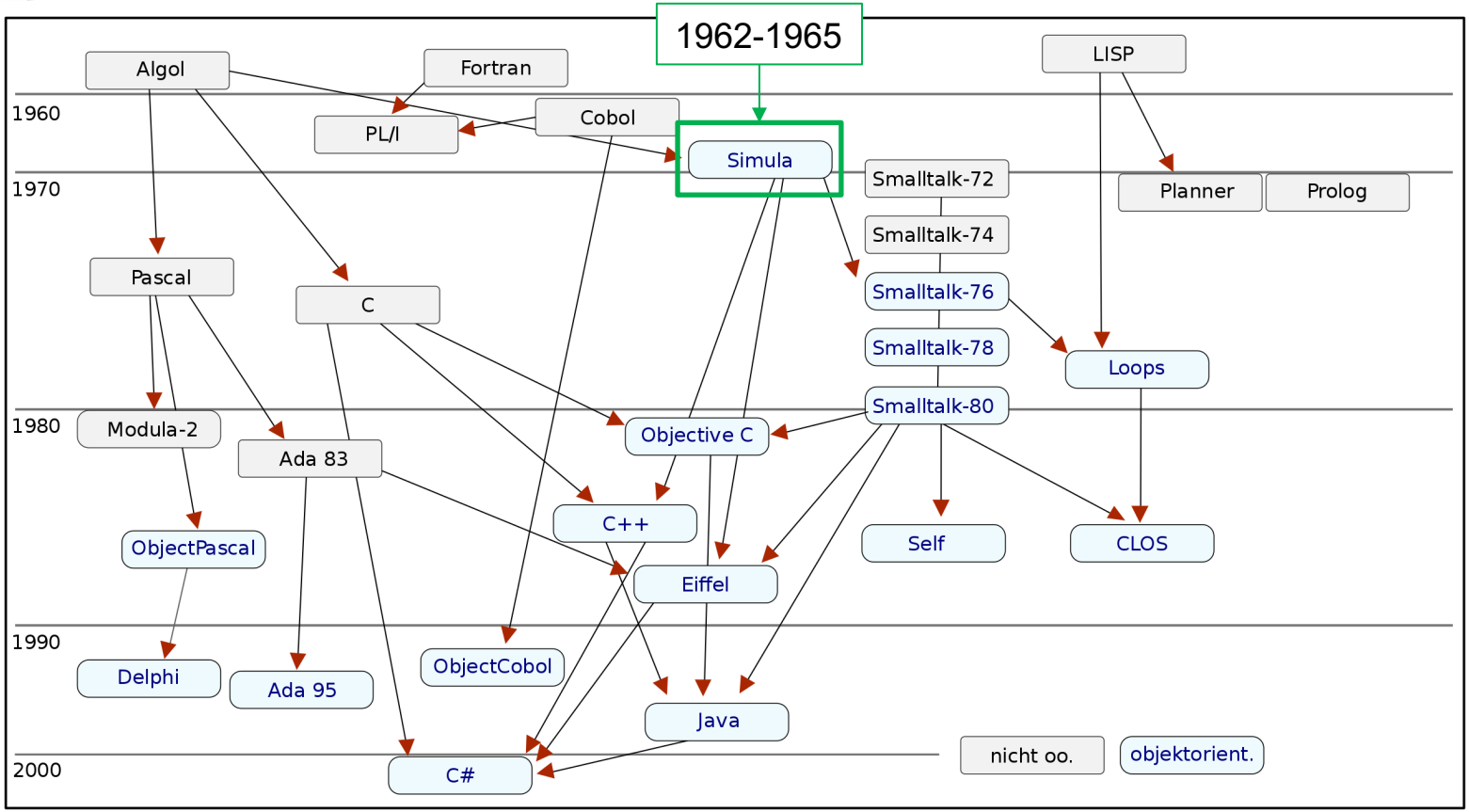
Object Oriented Programming

Creator of OOP

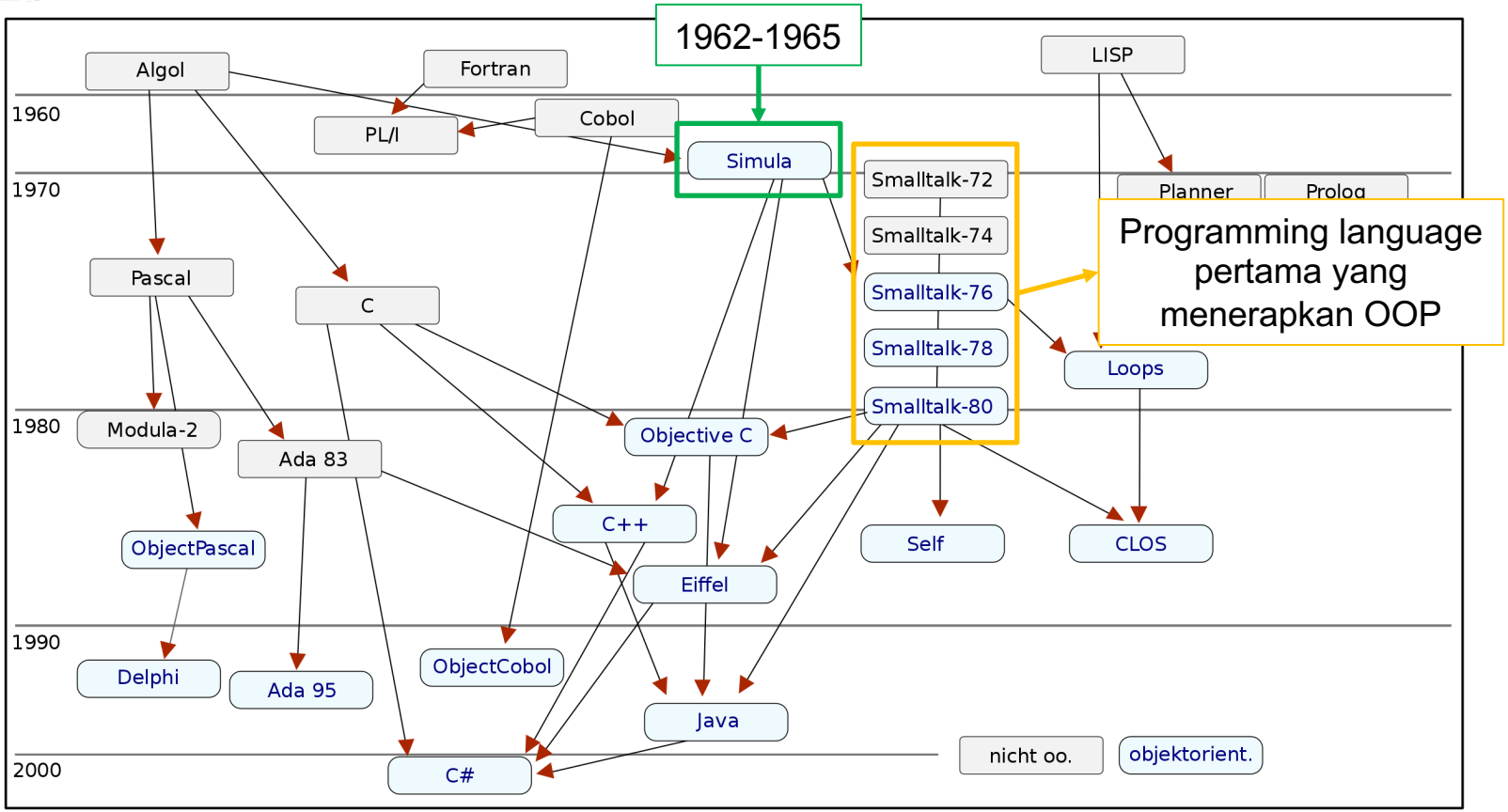
Kristen Nygaard

Ole-Johan

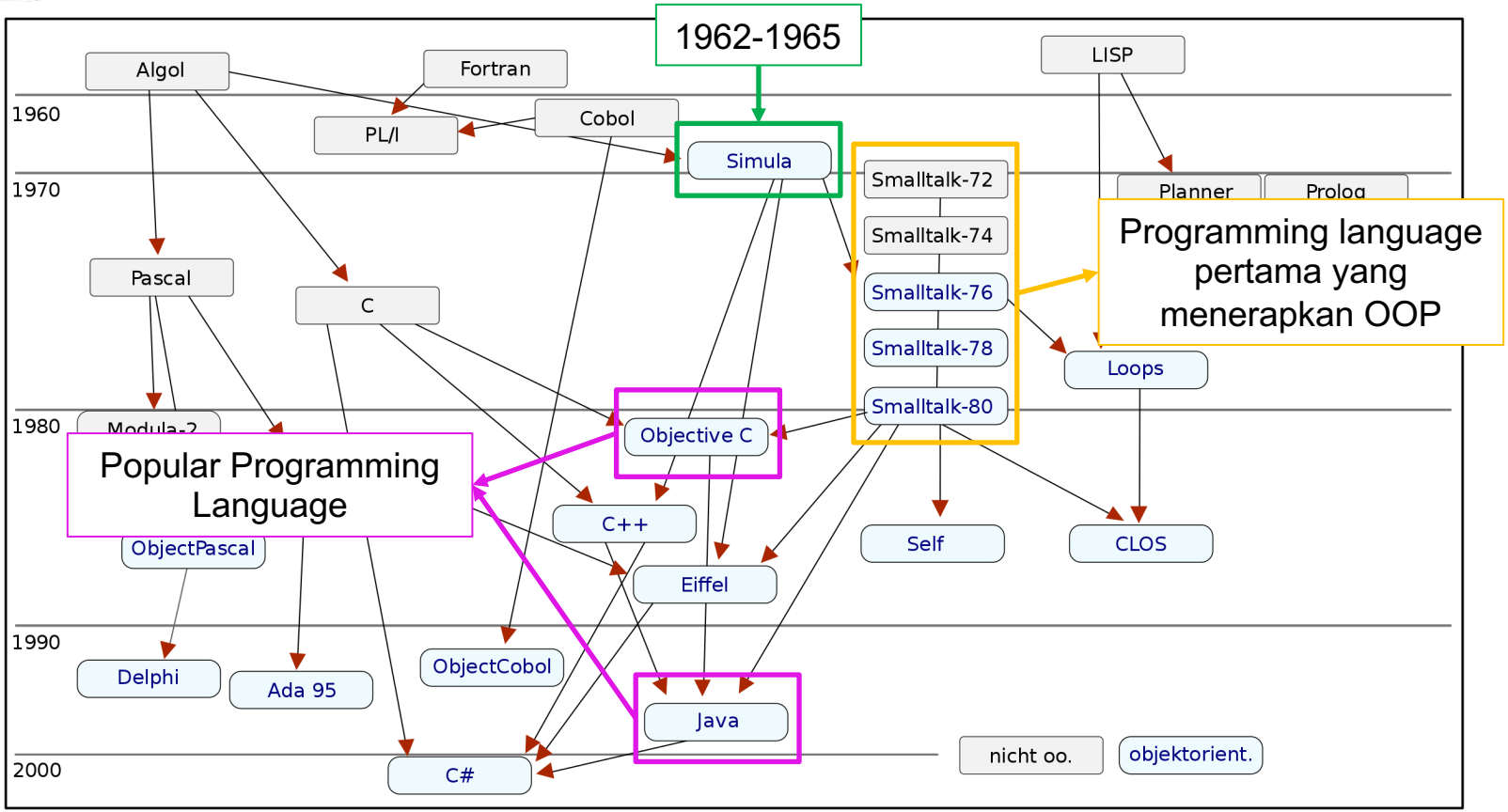
History of OOP



History of OOP



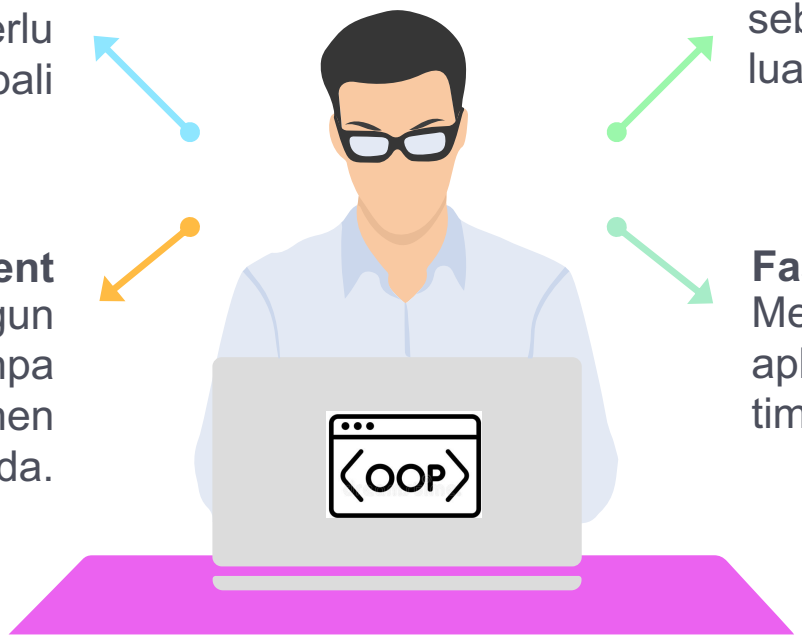
History of OOP



OOP Concepts

Reusable
Dapat menggunakan memanggil script dari code sebelumnya tanpa perlu menulis ulang kembali

Parallel Development
Dapat membangun komponen tersendiri, tanpa perlu mengubah komponen yang telah ada.



Scalability
Mempermudah pengembangan kebutuhan sebuah program yang lebih luas atau rumit

Fast Development
Mempercepat pembuatan aplikasi dan meminimalisir timbulnya bug

OOP Structure

Metode pemograman yang berorientasi pada ***Objek***.
Dimana memiliki ***variable*** dan ***method*** yang dibungkus kedalam ***Objek*** atau ***Class***

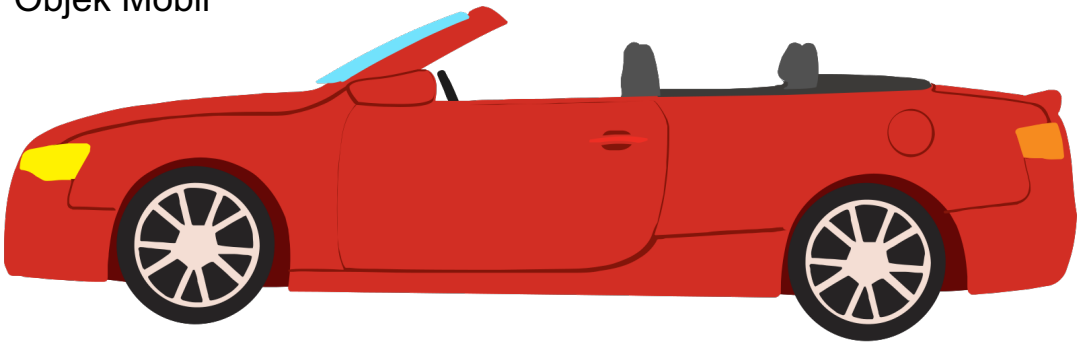
Class dan Object
Class bertugas untuk mengumpulkan prosedur/fungsi dan variabel dalam satu tempat.
Class merupakan *blueprint* dari sebuah objek atau cetakan untuk membuat objek

Atribut
Atribut merupakan bagian dari sebuah kelas yang masih berhubungan erat dari kelas tersebut. Atribut bisa juga disebut sebagai properti atau *properties* dari sebuah *class*.

Method
Method berperan menjelaskan bagaimana suatu atribut beraksi.
Peran yang dimaksud berupa tingkah laku (*behavior*) yang dapat digambarkan oleh suatu *method*.

Sample of Class

Objek Mobil



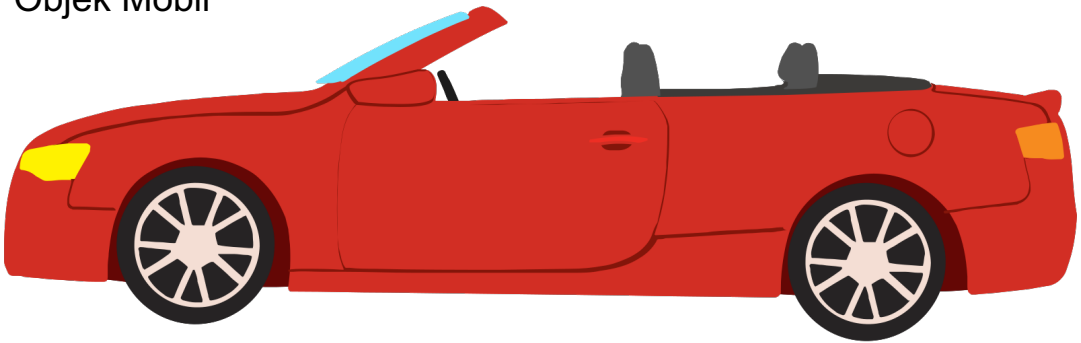
Class

Attribute

Method

Sample of Class

Objek Mobil



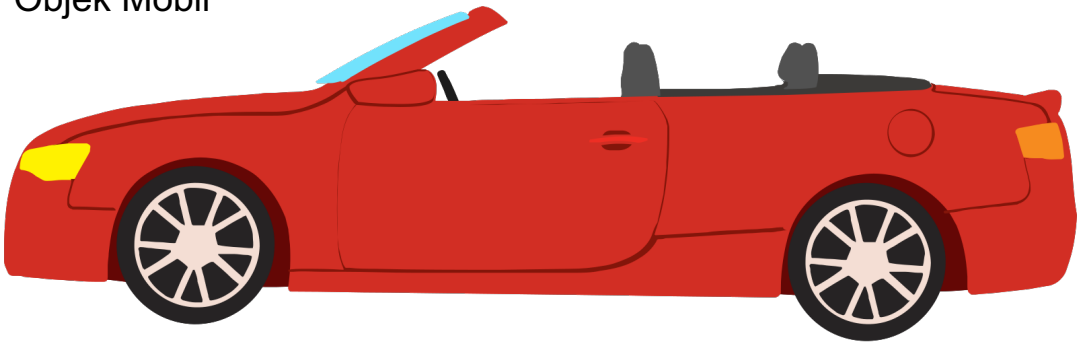
Class
Mobil

Attribute

Method

Sample of Class

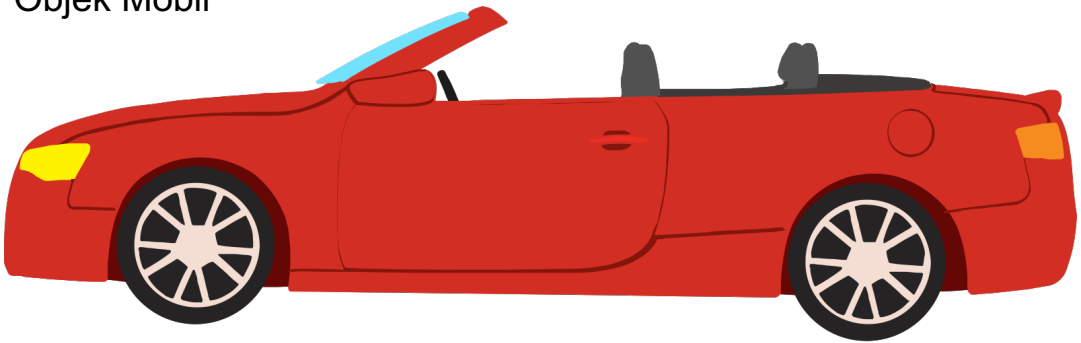
Objek Mobil



Class	Attribute	Method
Mobil	Warna, Roda, Merek, Kecepatan	

Sample of Class

Objek Mobil



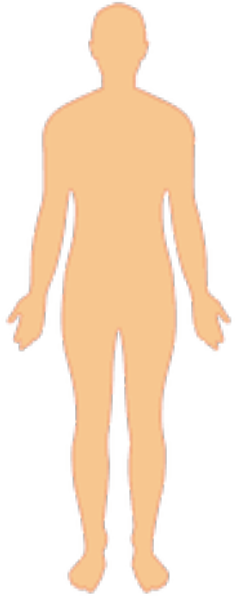
Class
Mobil

Attribute
Warna, Roda, Merek, Kecepatan

Method
Bergerak Maju/Mundur, Bunyikan Klakson, MengisiBensin

Questions

Objek Manusia



Katarina

Class

...

Attribute

...

Fauzan

Method

...

Adrian

Implement of Class

```
class Mobil{
    String Warna;
    Integer Roda;
    Float Merek;
    Long Kecepatan;

    void Bergerak(){
      …
    }

    boolean MenyalakanKlakson(){
      return true;
    }

    int MengisiBensin(){
      return 0;
    }
}
```

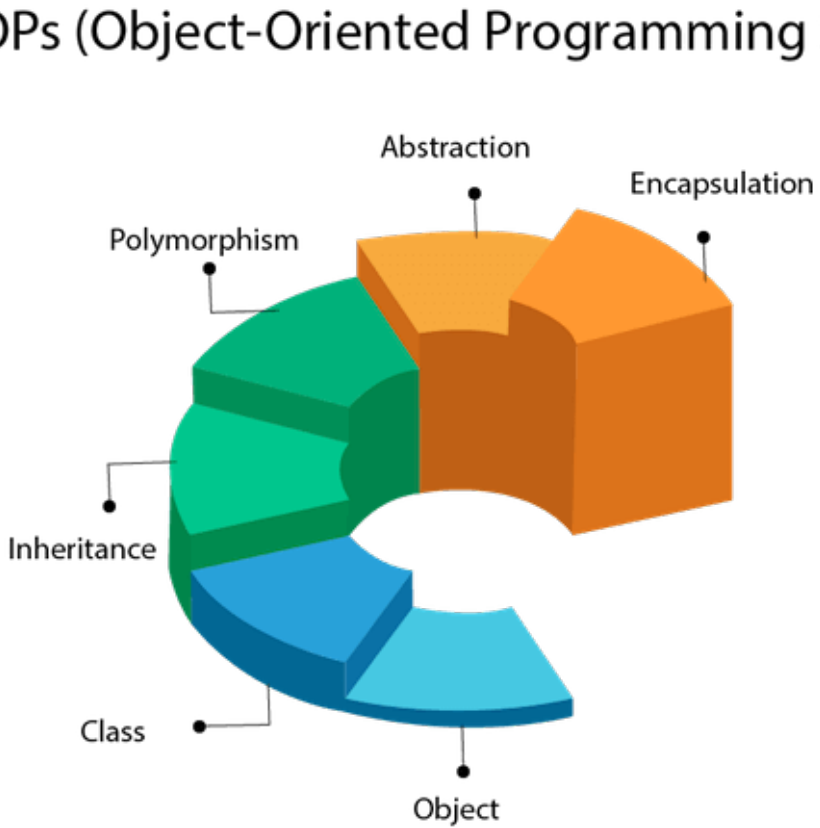
Sample Code

Mobil
Warna : string Roda : Integer Merek : Float Kecepatan : Long
Bergerak() MenyalakanKlakson() MengisiBensin()

UML

CLASS DIAGRAM

OOP Principles



OOP PRINCIPLES

Encapsulation

a way to restrict the direct access to some components of an object,

Abstraction

process of hiding the implementation details from the user

Inheritance

mechanism of basing an object or class upon another object or class, retaining similar implementation.

Polymorphism

provision of a single interface to entities of different types

02

Documentation

UML

Documentation for OOP

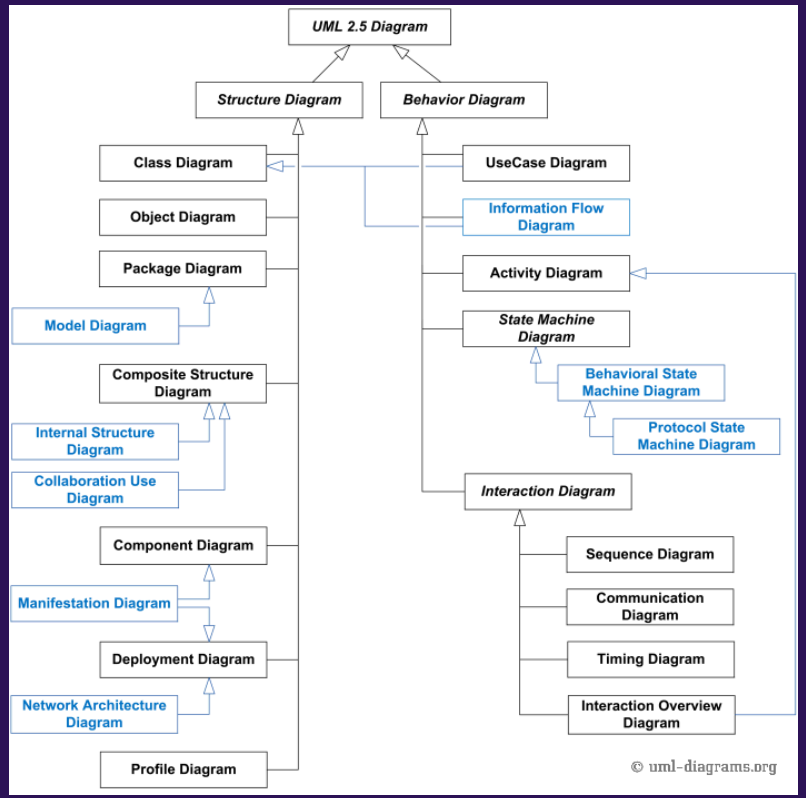
UML 2.5 Diagrams Overview

Metode dalam pemodelan secara visual yang digunakan sebagai sarana perancangan sistem berorientasi objek.

Bahasa standar visualisasi, perancangan, dan pendokumentasian sistem, atau dikenal juga sebagai bahasa standar penulisan *blueprint* sebuah *software.*

https://www.uml-diagrams.org/uml-25-diagrams.html

Classification of UML 2.5 Diagrams



Dua jenis utama diagram UML:

diagram **Structure** dan diagram **Behavior**.

Diagram **Structure** menunjukkan struktur statis dari sistem dan bagian-bagiannya pada tingkat abstraksi dan implementasi yang berbeda dan bagaimana mereka terkait satu sama lain.

Diagram **Behavior** menunjukkan perilaku dinamis dari objek dalam suatu sistem, yang dapat digambarkan sebagai serangkaian perubahan sistem dari waktu ke waktu.

Diagram UML you should Know..!!

Use Case

rangkaian proses bisnis yang harus atau dapat dilakukan oleh beberapa sistem pengguna

Class Diagram

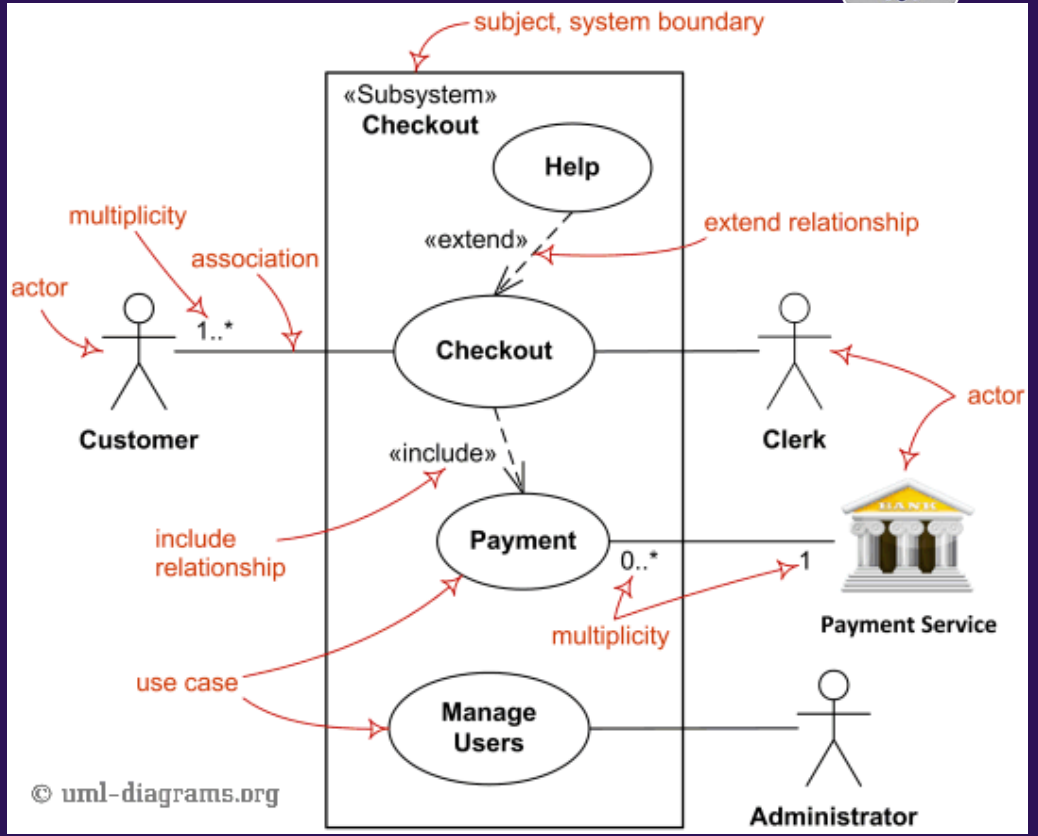
Menunjukkan struktur sistem, subsistem atau komponen yang dirancang sebagai kelas terkait

Activity Diagram

Menunjukkan urutan dan kondisi untuk mengoordinasikan perilaku pengklasifikasi

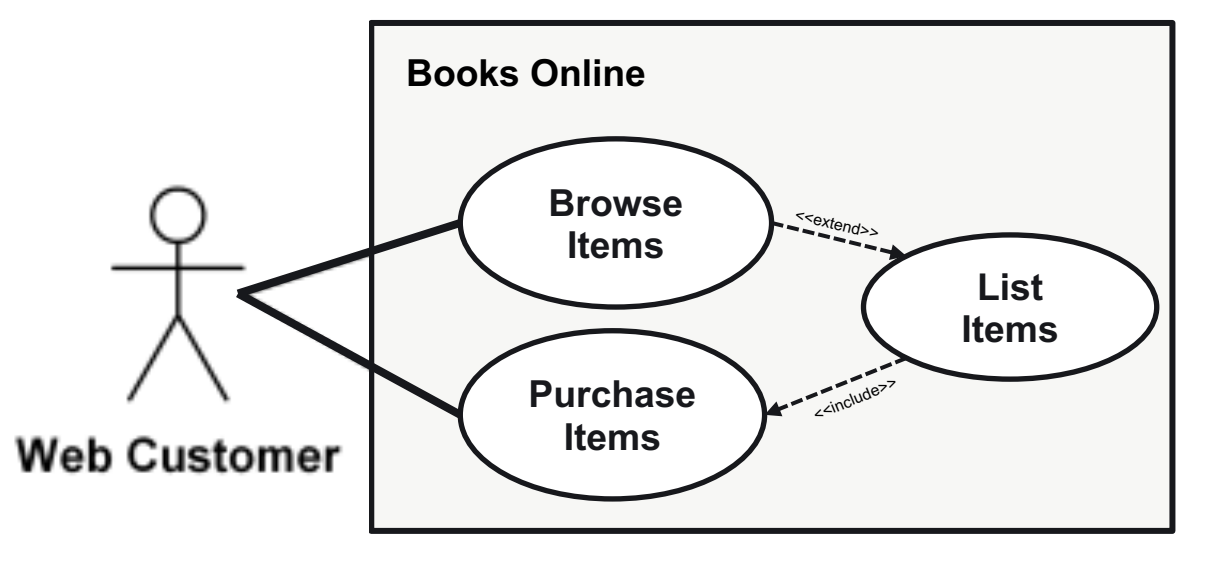
USE CASE

Diagram use case biasanya disebut sebagai diagram perilaku yang digunakan untuk menggambarkan serangkaian tindakan (use case) yang harus atau dapat dilakukan oleh beberapa sistem atau sistem (subjek) dalam kolaborasi dengan satu atau lebih pengguna eksternal sistem (aktor).
Setiap use case harus memberikan beberapa hasil yang dapat diamati dan berharga bagi para aktor atau pemangku kepentingan lain dari sistem.

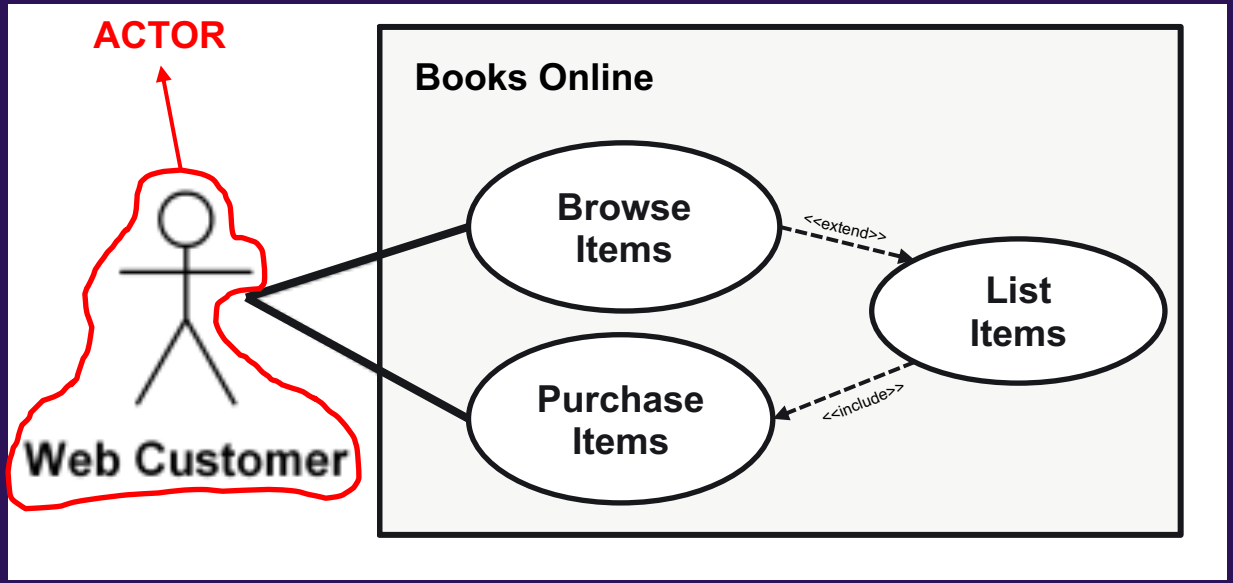


USE CASE – Example

Contoh usecase system aplikasi Books Online:



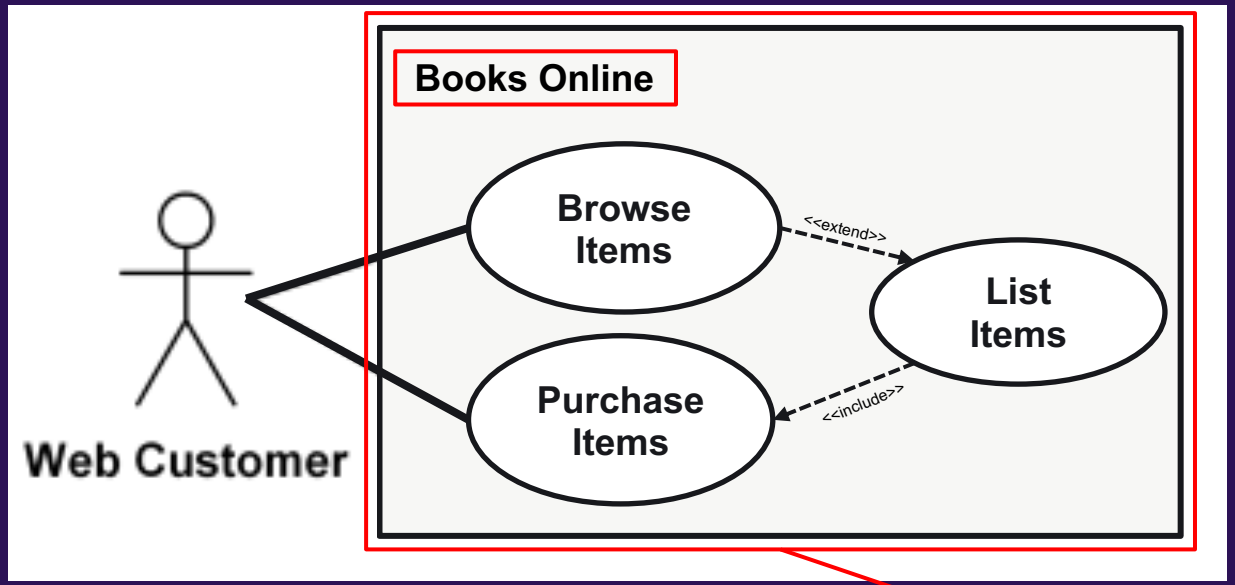
USE CASE – Actor



Note:

Aktor adalah pengklasifikasi perilaku yang menentukan peran yang dimainkan oleh entitas eksternal yang berinteraksi dengan subjek pengguna manusia dari sistem yang dirancang, beberapa sistem atau perangkat keras lain yang menggunakan layanan subjek.

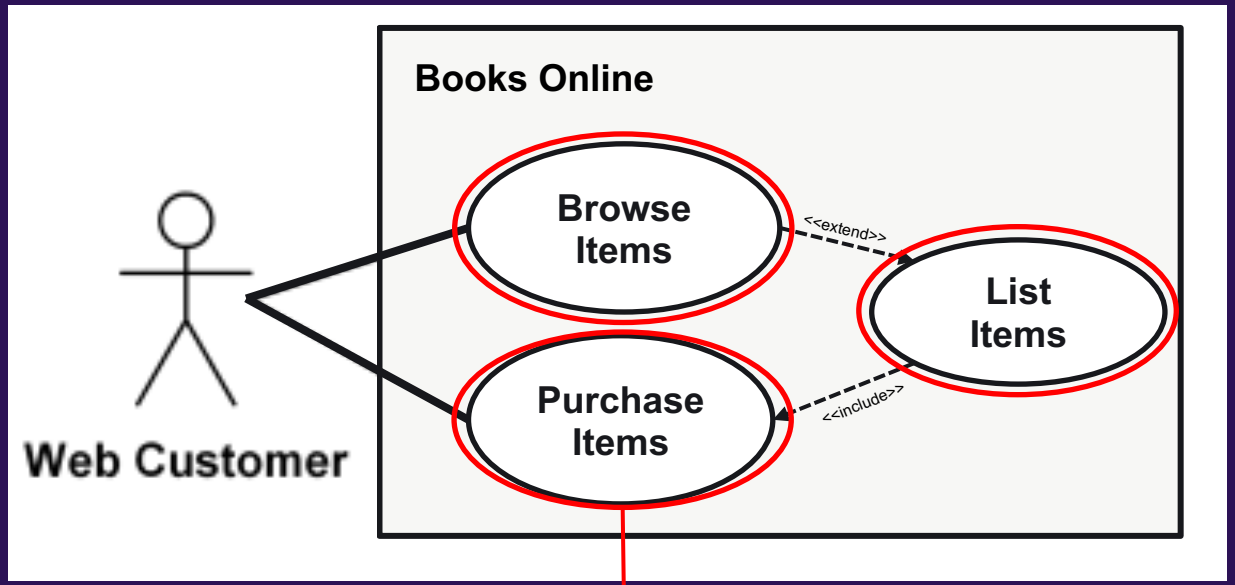
USE CASE – System Boundering



Note:

Subjek dari use case mendefinisikan dan mewakili batas-batas bisnis, sistem perangkat lunak, sistem atau perangkat fisik, subsistem, komponen atau bahkan kelas tunggal dalam kaitannya dengan pengumpulan dan analisis persyaratan.

USE CASE – Notasi



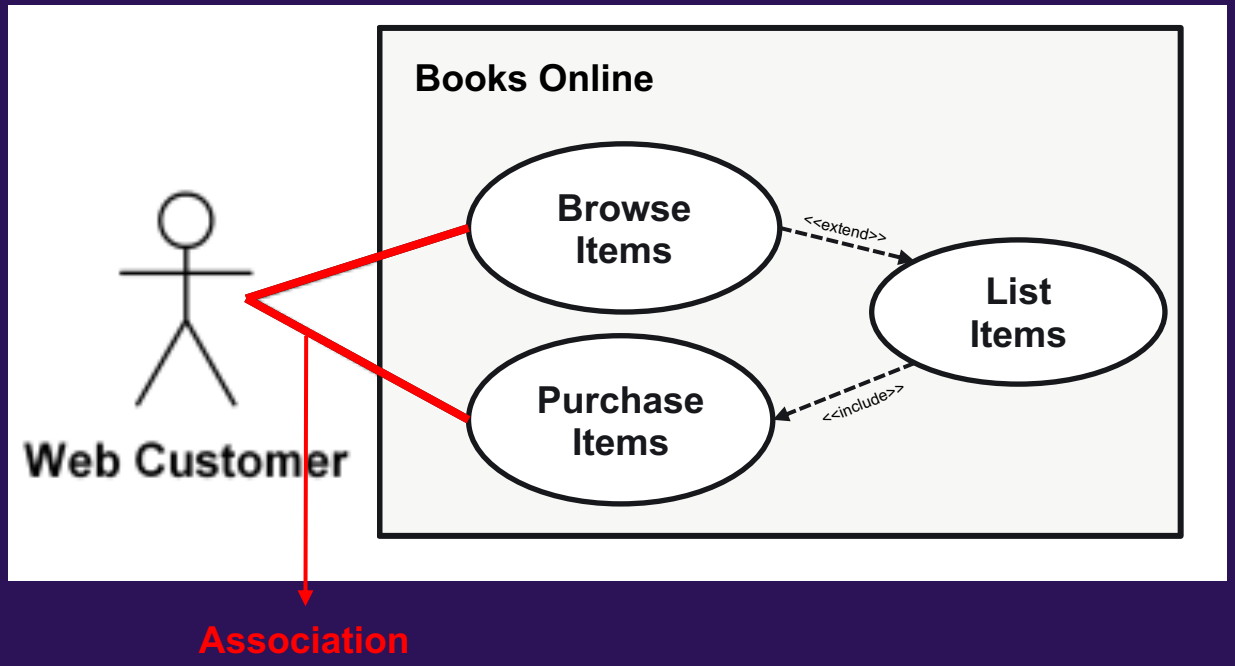
Note:

Notasi ini harus memiliki hubungan relasi dengan actor

Notasi harus dinamakan dengan menggunakan KATA KERJA.

Jumlah Notasi pada 1 use case maximum adalah 6+1.

USE CASE – Association

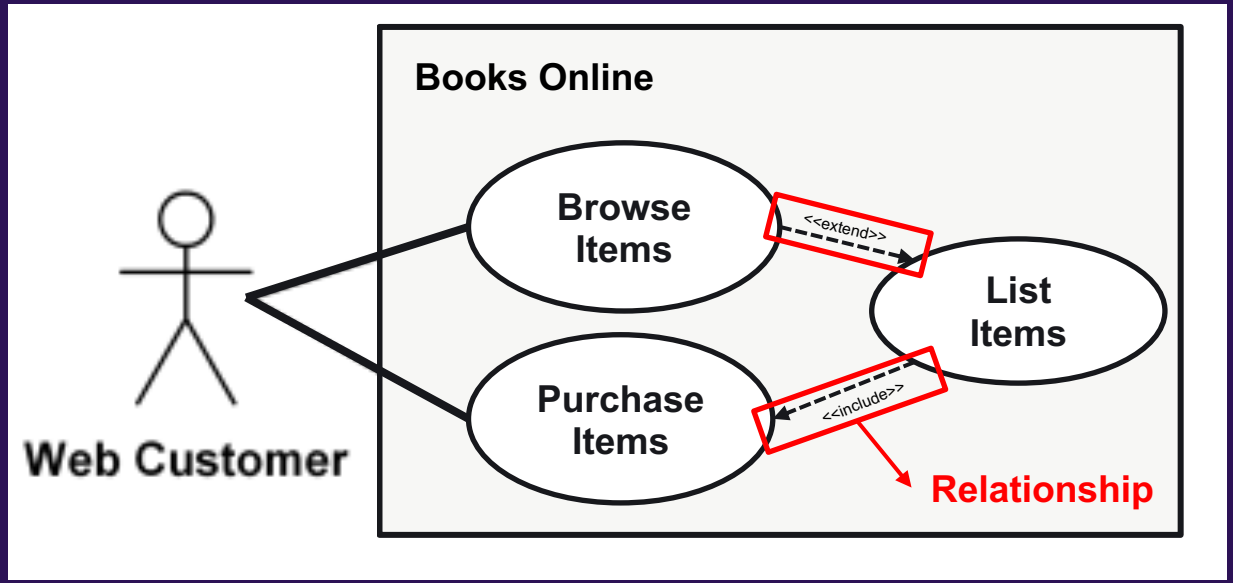


Note:

Setiap use case mewakili unit fungsionalitas berguna yang diberikan subjek kepada aktor.

Asosiasi antara aktor dan use case menunjukkan bahwa aktor dan use case entah bagaimana berinteraksi atau berkomunikasi satu sama lain.

USE CASE – Relationship

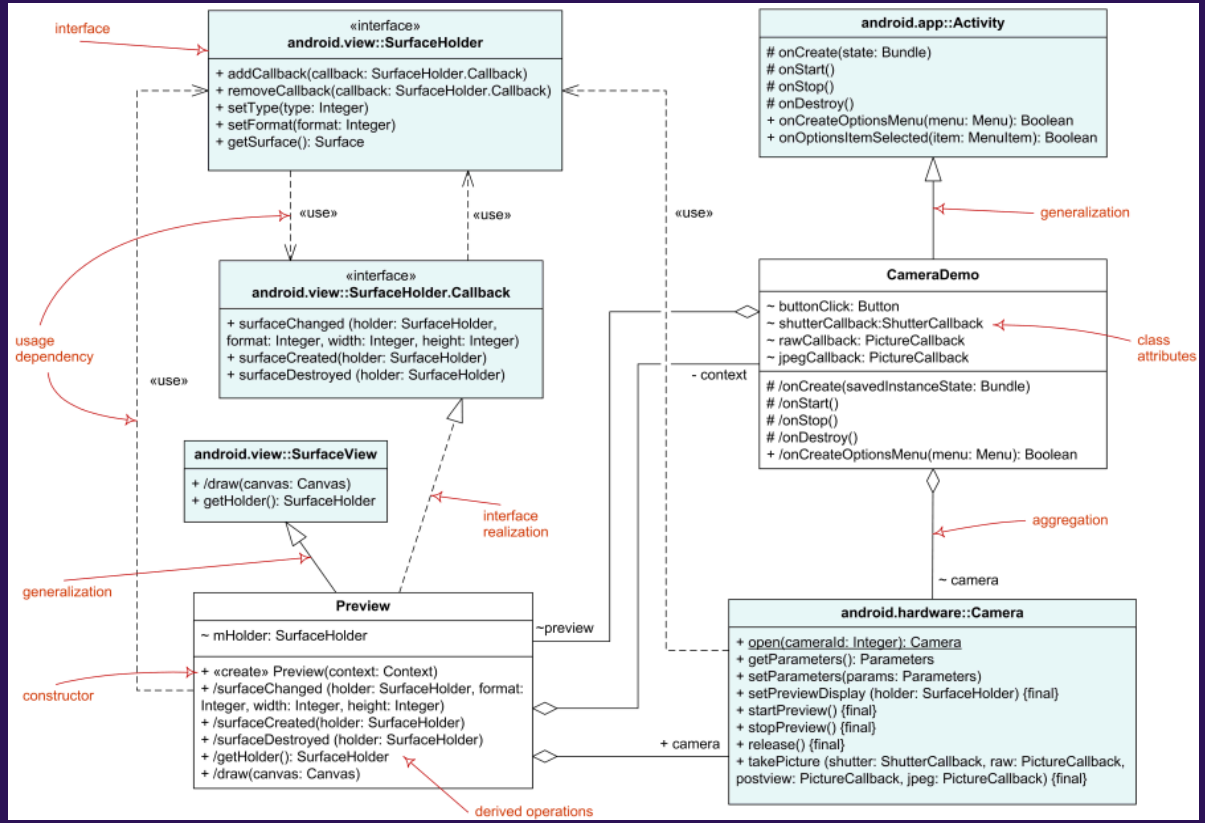


Note:

Extend adalah hubungan terarah antara dua notasi yang digunakan untuk menunjukkan bahwa perilaku notasi yang tidak harus disertakan ke dalam perilaku notasi termasuk (optional).

Include adalah hubungan terarah antara dua notasi yang digunakan untuk menunjukkan bahwa perilaku notasi yang disertakan dimasukkan ke dalam perilaku notasi termasuk (wajib).

CLASS DIAGRAM



Implement of Class

```
class Mobil{
    String Warna;
    Integer Roda;
    Float Merek;
    Long Kecepatan;

    void Bergerak(){
      …
    }

    boolean MenyalakanKlakson(){
      return true;
    }

    int MengisiBensin(){
      return 0;
    }
}
```

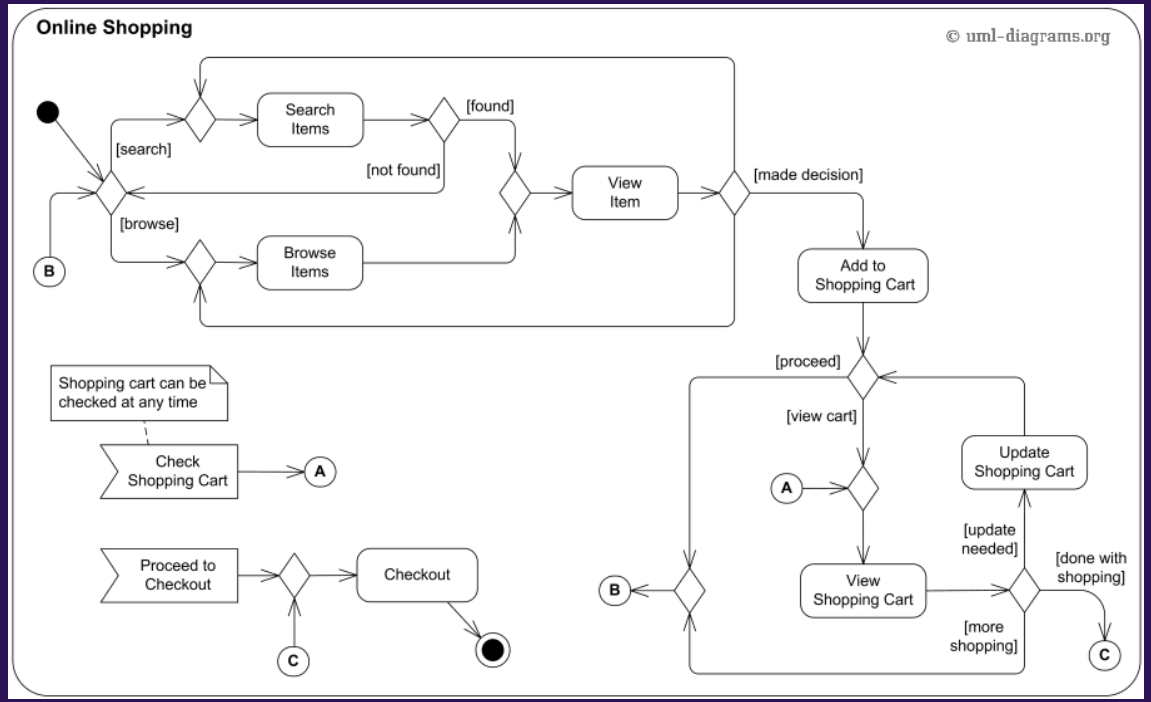
Sample Code

Mobil
Warna : string Roda : Integer Merek : Float Kecepatan : Long
Bergerak() MenyalakanKlakson() MengisiBensin()

UML

CLASS DIAGRAM

ACTIVITY DIAGRAM



Note:

Jumlah dari activity diagram mengikuti jumlah Use Case yang dibuat

03

Programming Language

Programming Language OOP

Programming Language for OOP

Popular pure OOP

Ruby

Ruby Programming

JADE

Jade Programming

Scala

Scala Programming

Emerald

Emerald Programming

Kampus Merdeka INDONESIA JAYA

INSTITUT BISNIS DAN INFORMATIKA KESATUAN BOGOR

Programming Language for OOP

Primarily for OOP

JAVA

JAVA Programming

PHYTON

Phyton Programming

C++

C++ Programming

Kampus Merdeka INDONESIA JAYA

INSTITUT BISNIS DAN INFORMATIKA KESATUAN BOGOR

Programming Language for OOP

Use in this course

JAVA

JAVA Programming

WHY JAVA ?

JAVA

1. Lahir sebagai program yang structural
2. Bahasa yang Multiplatform
3. Memiliki open source library yang sangat lengkap
4. Java run on BILLIONS of devices around the planet




JAVA Tools

Java SE Development Kit Version 18
https://www.oracle.com/java/technologies/downloads

JAVA IDE

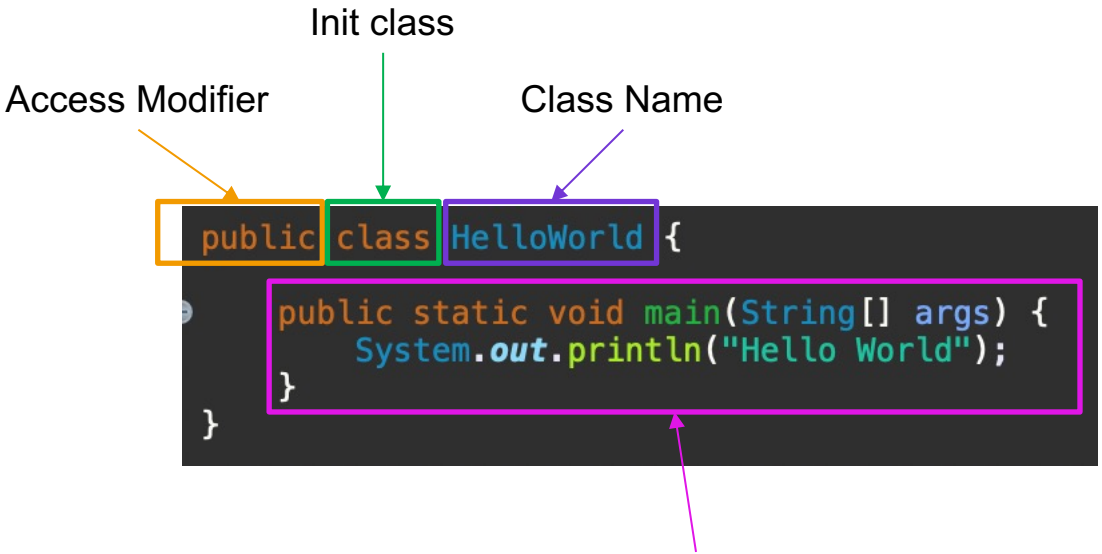
VSCode

https://www.eclipse.org/downloads/

Sistem Versioning Code

https://prod.liveshare.vsengsaas.visualstudio.com/join?31D414B3A144161620EE5A19F9605BA1BDD2

JAVA Script



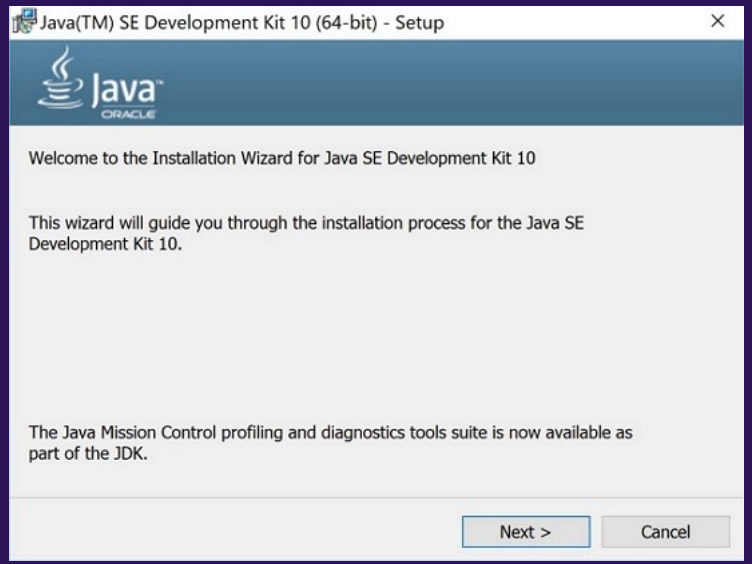
Pemrosesan program Java dimulai dari metode main() yang merupakan bagian wajib dari setiap program Java.

Setup JAVA

Java SE Development Kit Version 18

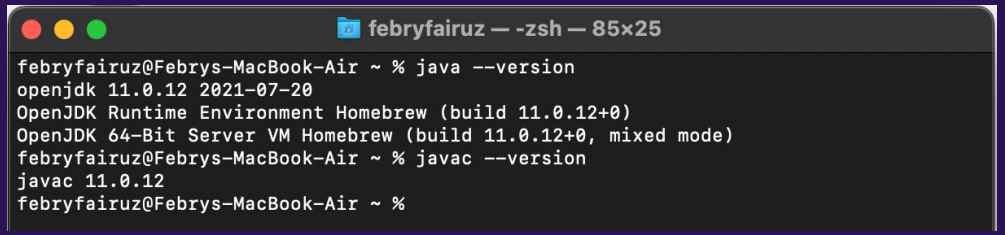
1. Download:

https://www.oracle.com/java/technologies/downloads



2. Install

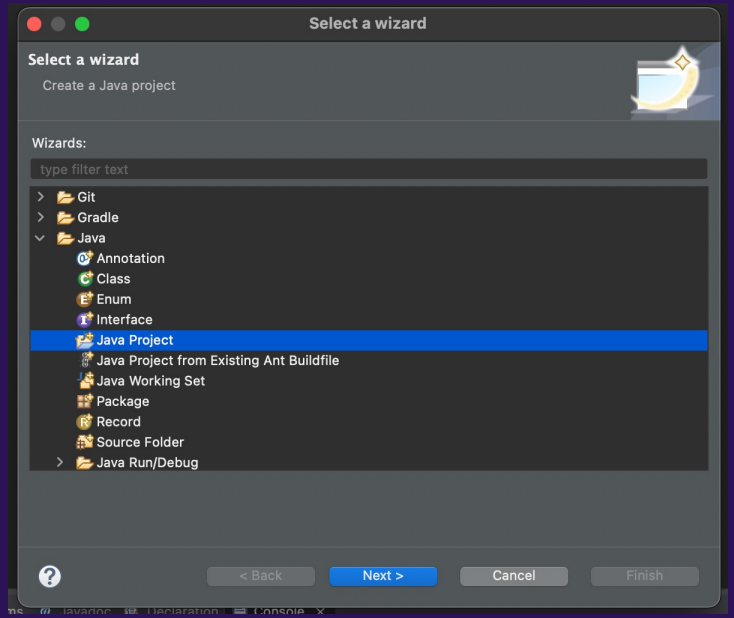
3. Cek java environment



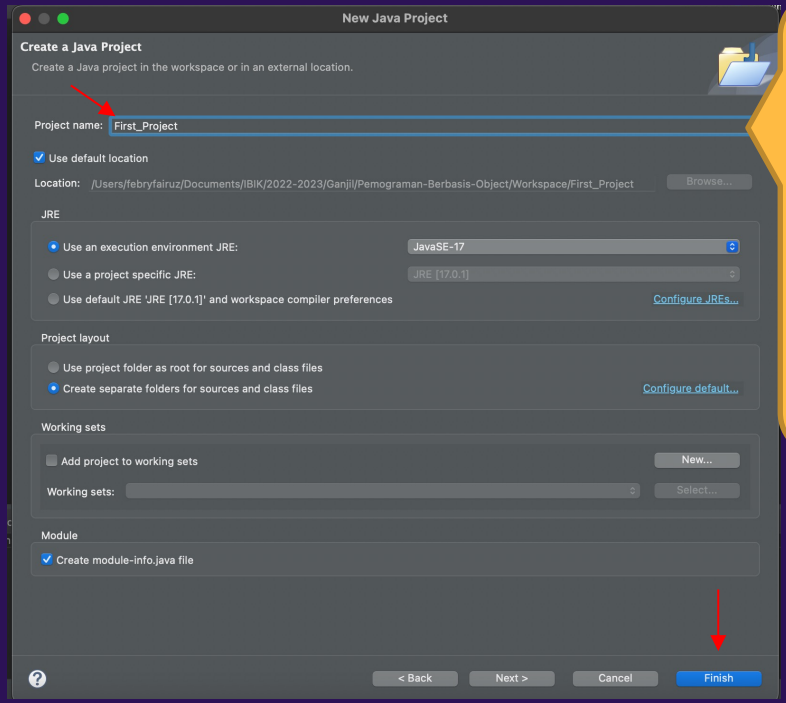
a) Buka terminal masukan syntax *java --version* lalu enter
b) Masukan Kembali *javac --version* lalu enter

How To Create Project?

5. Pada pop up window pilih **Java** → **Java Project** lalu klik tombol **Next**



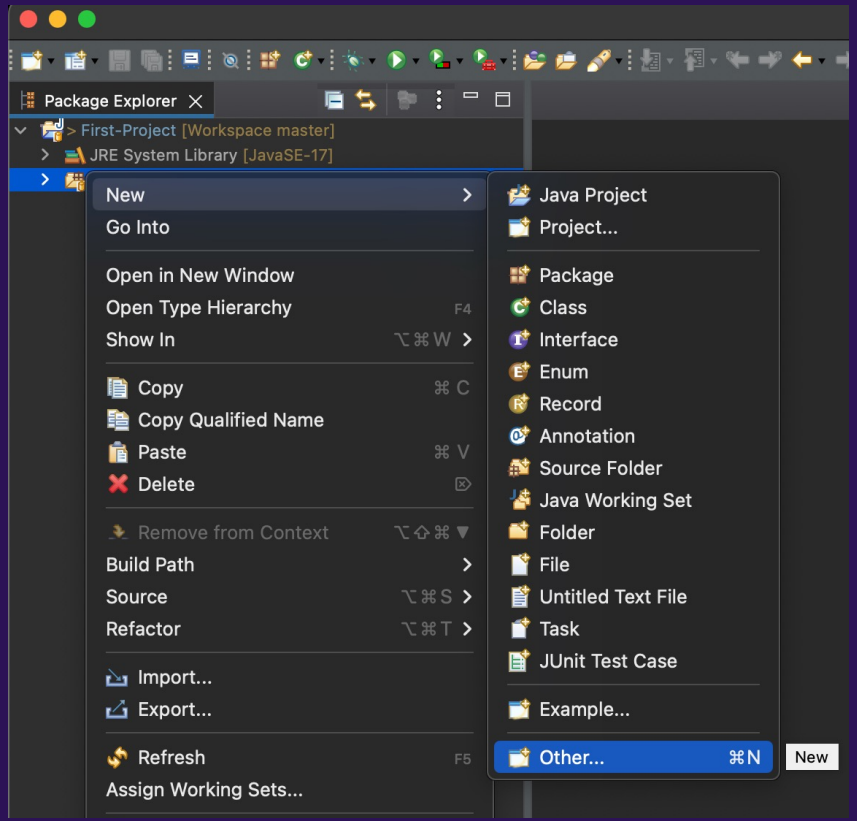
6. Pada isian Project name, tuliskan nama project kalian. Contoh: **First-Project**



Penamaan Projek Tidak boleh mengandung spasi gunakan tanda strip (-) atau underline (_) atau digabungkan.
Contoh:
First-Project
First_Project
FirstProject
1stProject

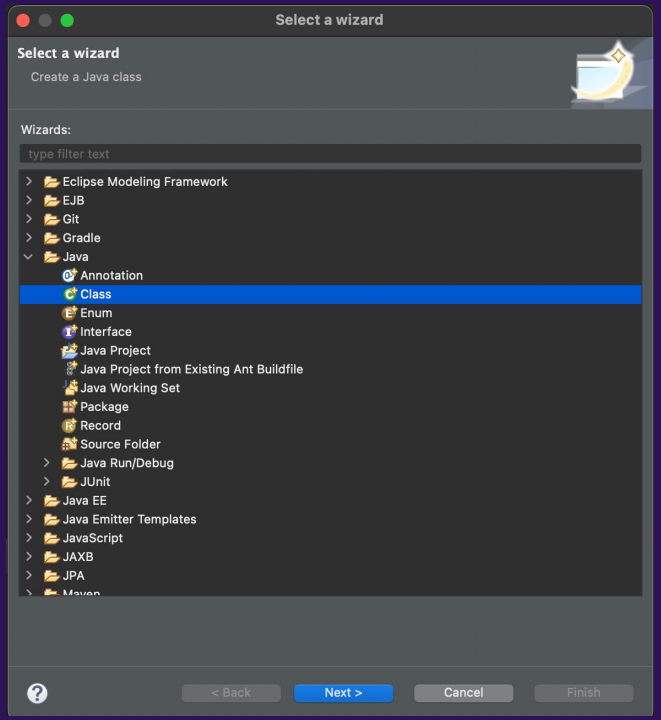
How To Create Project?

7. Pada collation view sebelah kiri, terdapat Package Explorer. Dimana akan menampilkan seluruh projek Java yang telah dibuat.
Didalam projek tersebut terdapat folder src. Pada folder tersebut klik kanan pilih **New → Other…**

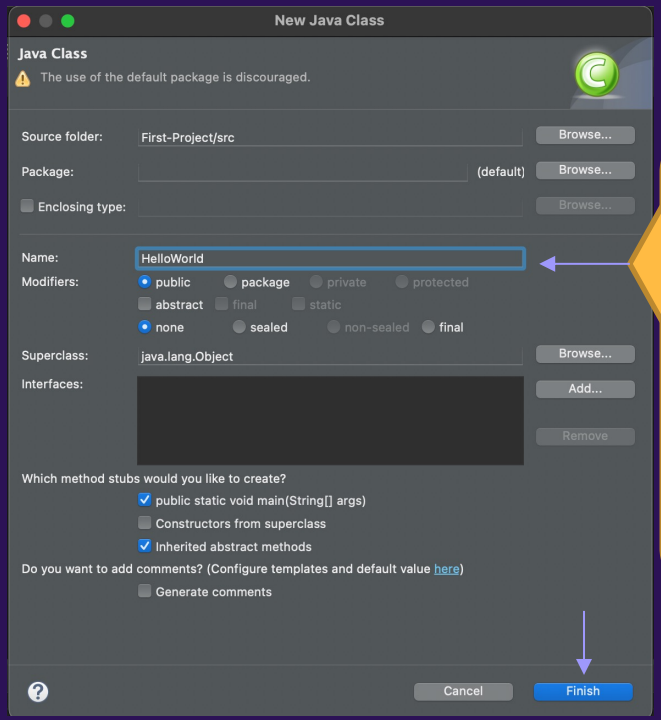


How To Create Project?

8. Pada pop up window, pilih **Java → Class** lalu **Next**



9. Pada isian Name, tulislah *nama class* yang akan dibuat. Contoh **HelloWorld** lalu **Finish**



Penamaan Class Tidak boleh mengandung **spasi**, **symbol** atau pun **angka**. Penulisan harus diawali dengan *Huruf Besar..!!*

How To Create Project?

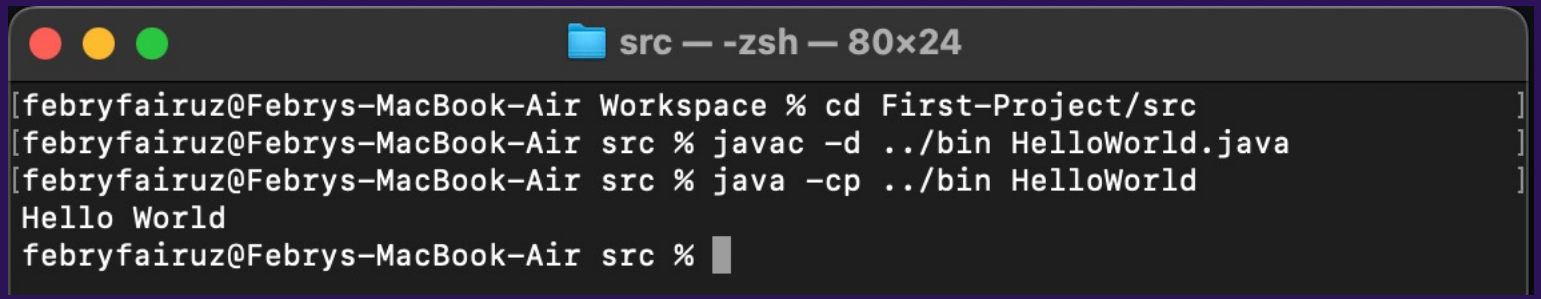
10. Tuliskan script dibawah ini di HelloWorld.java untuk membuat tulisan Hello World pada console terminal.

HelloWorld.java

```
1
2 public class HelloWorld {
3     public static void main(String[] args) {
4         System.out.println("Hello World");
5     }
6 }
7
```

How To Run ?

1. Buka terminal atau command line
2. Masukan syntax untuk masuk kedalam directory First-Project
3. Tulislah syntax compiler: ***javac –d ../bin HelloWorld.java*** lalu enter
4. Tulislah syntax run: ***java –cp ../bin HelloWorld*** lalu enter



04 ASSIGNMENT section

Assignment 1

Object 1

Object 2

Object 3

TOYOTA

HONDA

Object 4

1. Buatlah bentuk Class Diagram berdasarkan ke empat Objek diatas. Tentukan **Class Name**, **Atribute** dan **Method** berdasarkan objek diatas.

Assignment 2

Jual/Beli

Penyewaan

Pegadaian

2. Pilihlah salah satu Sistem Aplikasi diatas, dan buatlah **Use Case** dari Sistem Aplikasi tersebut berdasarkan objek-objek sebelumnya.

Assignment 2

Service Auto

Game Drift

Samsat Digital

2. Pilihlah salah satu Sistem Aplikasi diatas, dan buatlah **Use Case** dari Sistem Aplikasi tersebut berdasarkan objek-objek sebelumnya.

Final Assignment

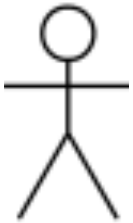
Aturan

- Buatlah 1 kelompok maximum 4 orang.
- Setiap kelompok memilih satu jenis Sistem Aplikasi yang telah ditentukan (tidak boleh sama), dimana 1 kelompok perlu membuat class diagram berdasarkan objek-objek pada Assignment 1 yang sesuai dengan Sistem Aplikasi yang telah dipilih.
- Dan membuat Use Case dari Sistem Aplikasi tersebut.

Pengumpulan

1. Dikumpulkan dalam bentuk Word, yang berisi Class Diagram dan Use Case dari Sistem Aplikasi terpilih, beserta nama anggotanya (NPM-Nama).
2. Dikumpulkan ke LMS Pemograman Berorientasi Objek pengumpulan hanya perwakilan kelompok.
3. Paling lambat hari Jumat 16 September 2022 pukul 23.00.

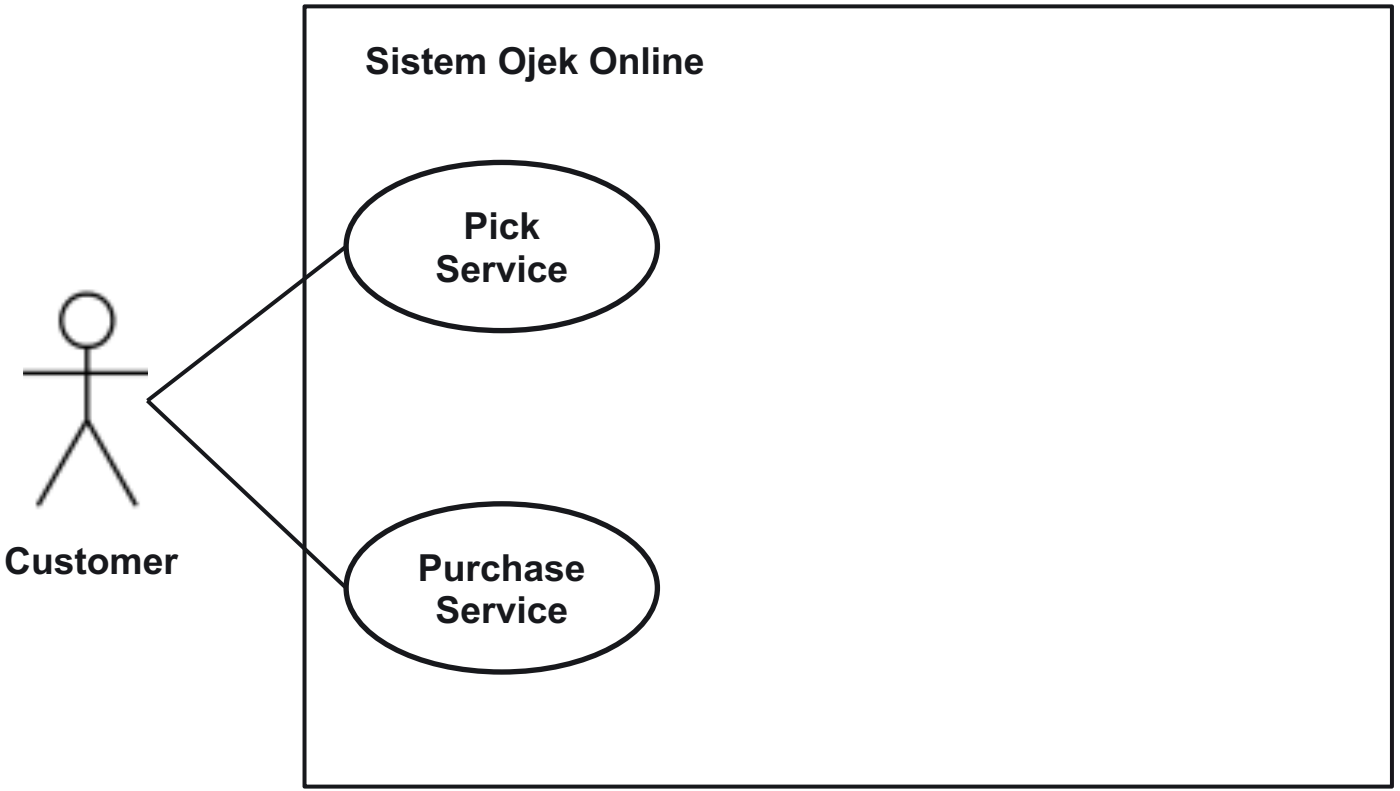
Use Case - OJOL



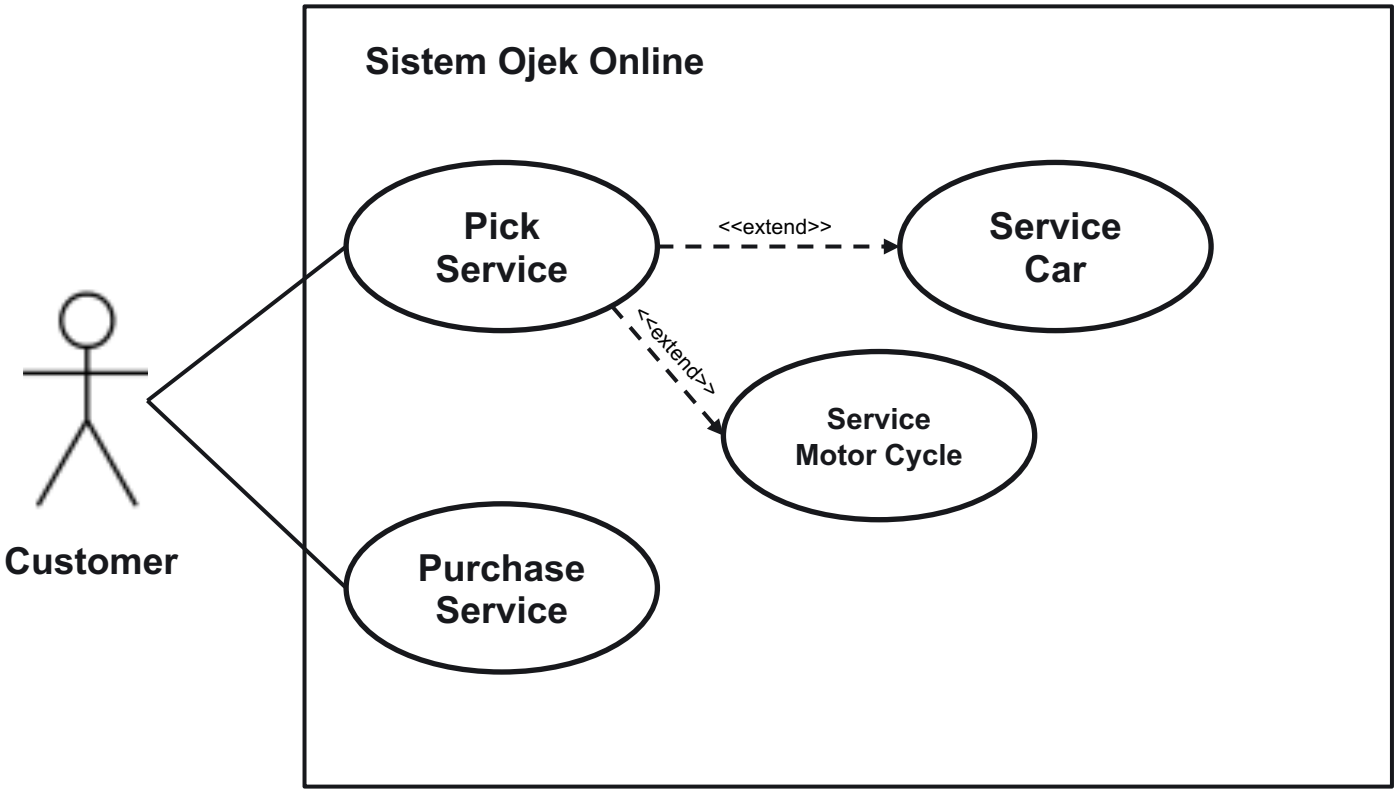
Customer

Sistem Ojek Online

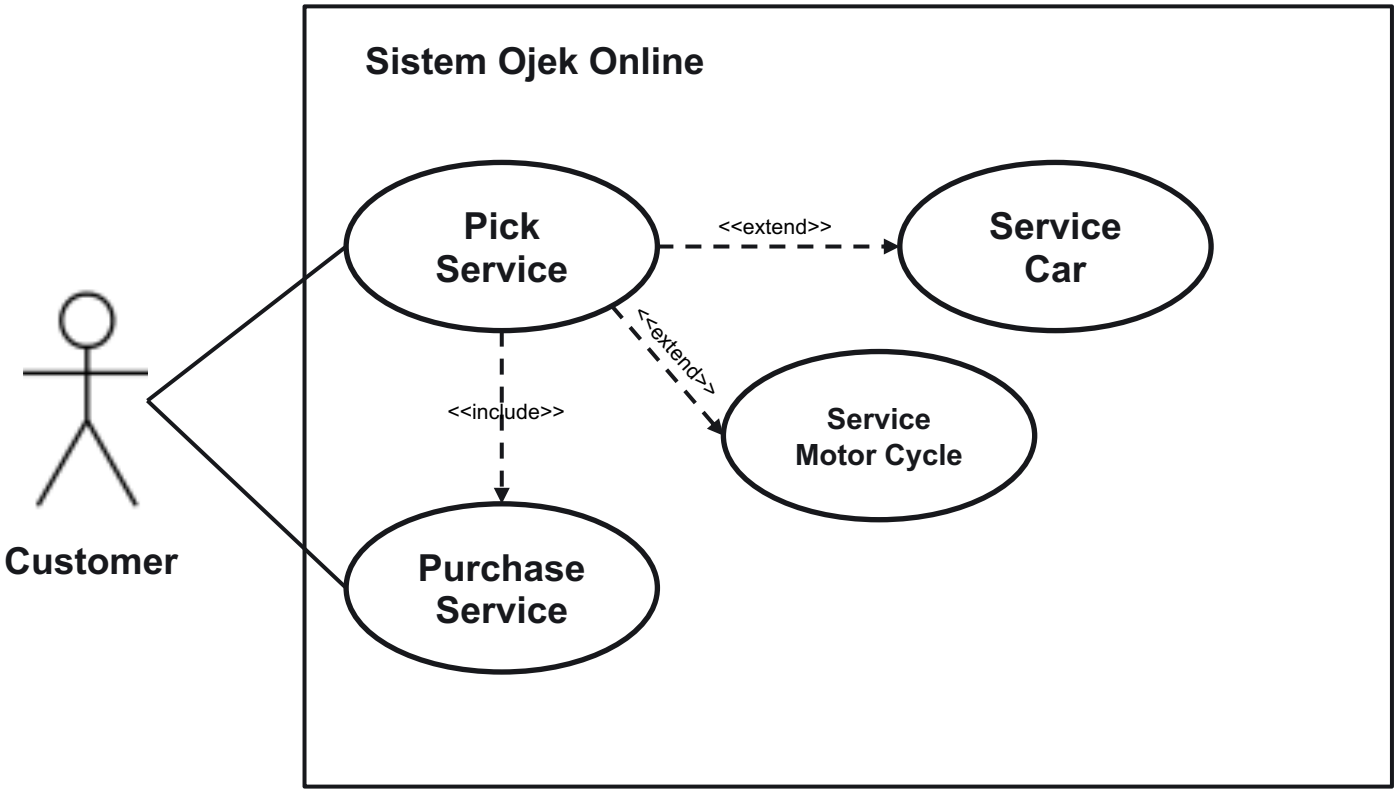
Use Case - OJOL



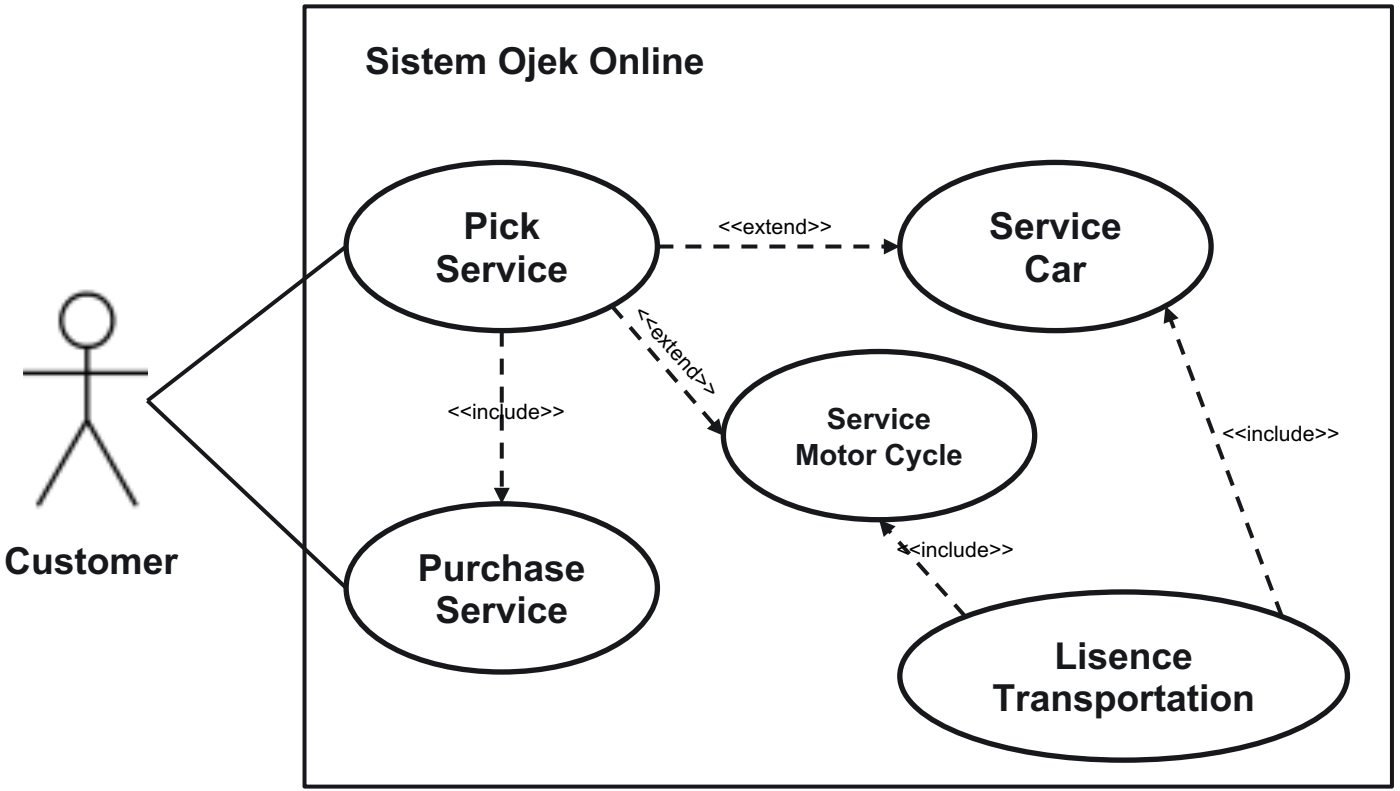
Use Case - OJOL



Use Case - OJOL



Use Case - OJOL



THANK YOU