



Asia Cyber
University

Algoritma & Pemrograman 1

OOP

Tujuan

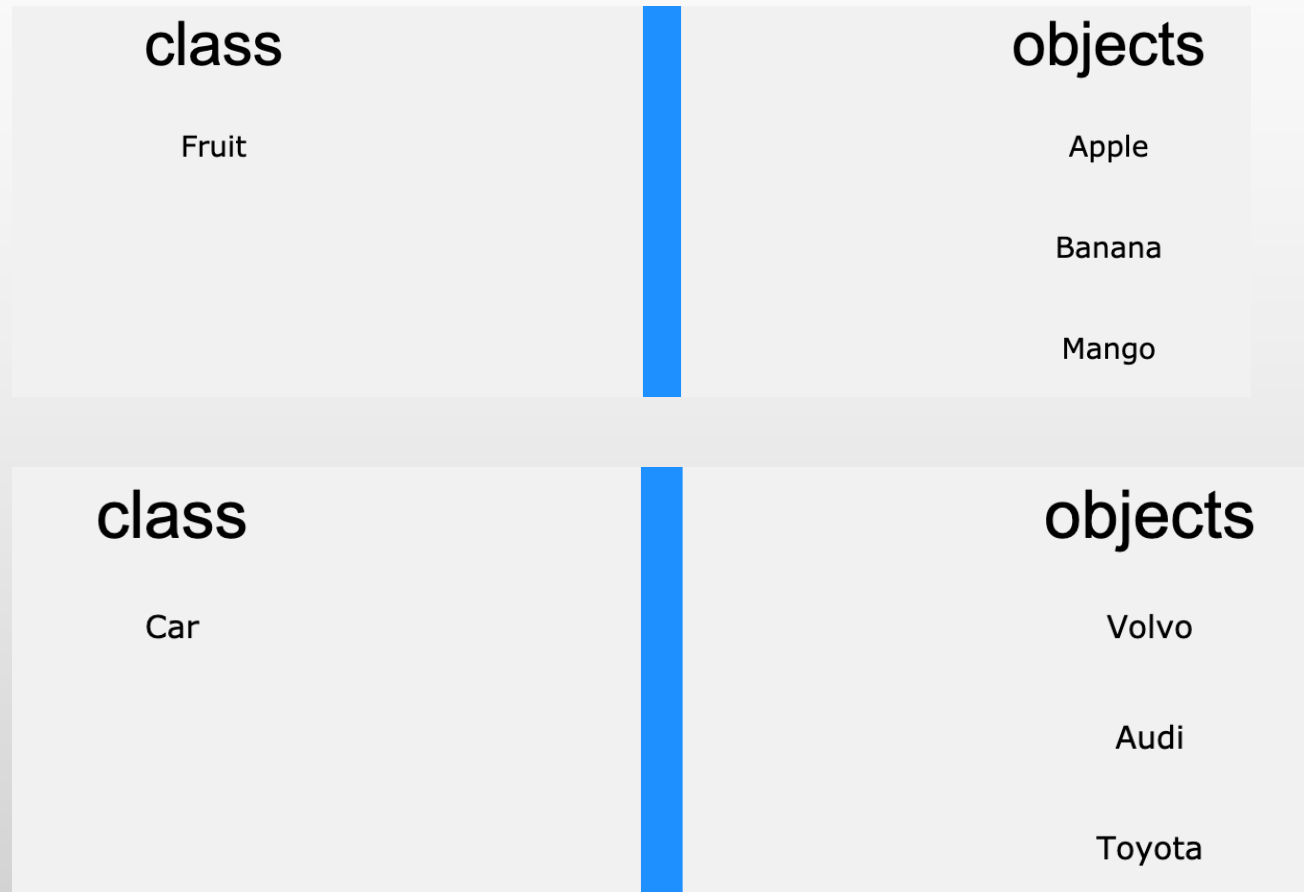
- Mahasiswa mampu memahami konsep Pemrograman berorientasi Object
- Mampu membuat program berbasis object

Apa itu OOP

- OOP adalah Object-Oriented Programming atau dalam bahasa Indonesia Pemrograman Berorientasi Object.
- Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions.
- Object-oriented programming has several advantages over procedural programming:
 - OOP is faster and easier to execute
 - OOP provides a clear structure for the programs
 - OOP helps to keep the C++ code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time

Apa itu Classes dan Object

- Classes and objects are the two main aspects of object-oriented programming.
- Look at the following illustration to see the difference between class and objects :



C++ Classes / Object

- C++ is an object-oriented programming language.
- Everything in C++ is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an **object**. The car has **attributes**, such as weight and color, and **methods**, such as drive and brake.
- Attributes and methods are basically **variables** and **functions** that belongs to the class. These are often referred to as "class members".
- A class is a user-defined data type that we can use in our program, and it works as an object constructor, or a "blueprint" for creating objects.

Membuat Class

- In C++, an object is created from a class. We have already created the class named MyClass, so now we can use this to create objects.
- To create an object of MyClass, specify the class name, followed by the object name.
- To access the class attributes (myNum and myString), use the dot syntax (.) on the object:

Example

Create a class called "MyClass":

```
class MyClass {           // The class
    public:                // Access specifier
        int myNum;         // Attribute (int variable)
        string myString;   // Attribute (string variable)
};
```

Example



Asia Cyber
University

Create an object called "**myObj**" and access the attributes:

```
class MyClass {           // The class
    public:                // Access specifier
        int myNum;         // Attribute (int variable)
        string myString;   // Attribute (string variable)
};

int main() {
    MyClass myObj;        // Create an object of MyClass

    // Access attributes and set values
    myObj.myNum = 15;
    myObj.myString = "Some text";

    // Print attribute values
    cout << myObj.myNum << "\n";
    cout << myObj.myString;
    return 0;
}
```

Multiple Object

```
// Create a Car class with some attributes
class Car {
public:
    string brand;
    string model;
    int year;
};

int main() {
    // Create an object of Car
    Car carObj1;
    carObj1.brand = "BMW";
    carObj1.model = "X5";
    carObj1.year = 1999;

    // Create another object of Car
    Car carObj2;
    carObj2.brand = "Ford";
    carObj2.model = "Mustang";
    carObj2.year = 1969;

    // Print attribute values
    cout << carObj1.brand << " " << carObj1.model << " " << carObj1.year << "\n";
    cout << carObj2.brand << " " << carObj2.model << " " << carObj2.year << "\n";
    return 0;
}
```


Class Method

Methods are **functions** that belongs to the class.

There are two ways to define functions that belongs to a class:

- Inside class definition
- Outside class definition

In the following example, we define a function inside the class, and we name it "myMethod".

Note: You access methods just like you access attributes; by creating an object of the class and by using the dot syntax (.):



Inside Example

```
class MyClass {           // The class
    public:                // Access specifier
        void myMethod() { // Method/function defined inside the class
            cout << "Hello World!";
        }
};

int main() {
    MyClass myObj;        // Create an object of MyClass
    myObj.myMethod();     // Call the method
    return 0;
}
```



Outside Example

```
class MyClass {           // The class
    public:                // Access specifier
        void myMethod();  // Method/function declaration
};

// Method/function definition outside the class
void MyClass::myMethod() {
    cout << "Hello World!";
}

int main() {
    MyClass myObj;        // Create an object of MyClass
    myObj.myMethod();     // Call the method
    return 0;
}
```

[Run example »](#)

Parameters

You can also add parameters:

Example

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

class Car {
public:
    int speed(int maxSpeed);
};

int Car::speed(int maxSpeed) {
    return maxSpeed;
}

int main() {
    Car myObj; // Create an object of Car
    cout << myObj.speed(200); // Call the method with an argument
    return 0;
}
```

Terimakasih