OBJECT

Teaching Team

Algorithm and Data Structure

2024/2025

Learning Outcome

After finishing this material, students must be able to:

- Understand the concept of **object** and **class**
- Declare class, attribute and method
- Create object form class (**instantiation**)
- Access the attributes and methods of an object
- Calling the constructor
- Understand the concept of objects and write them in class diagram form

Object Oriented Programming Basic Concept (1)

- In Basic Programming Course, programming is written using **procedural** approach, means by writing **procedures** or **methods/functions** that perform operations on data, while **Object Oriented Programming will** construct program containing **objects** that have **data** and **methods/functions**.
- Java is a programming language that fully implements the concept of Object-Oriented Programming (OOP).
- OOP is a programming paradigm that views a program consists of a collection of **objects** that interact with each other
- When you want to create a program, we must firstly identify **objects** which are in the system

Object Oriented Programming Basic Concept (2)

	OOP	Procedural
Point of View	Programs are considered as collections of **interacting objects**	Programs are considered as a collection of **processes** or **procedures**
Main Focus	Object	Process/Procedur/Function
Example	Banking System: • Customer • Account account • Transaction • Tellers • Money	Banking System: • Change PIN • Transfer • Cash withdrawal • Deposit

Object Oriented Programming Basic Concept (3)

- **Class** and **object** are the two main aspects of OOP
- Apart from these two aspects, there are some other basic concepts, such as Encapsulation, Inheritance, Polymorphism, etc. which will be explained in more detail in the OOP in Semester 3
- In this course, we will focus on the most basic concept of OOP that are **Class** and **Object**

Object

Object represents **real thing**

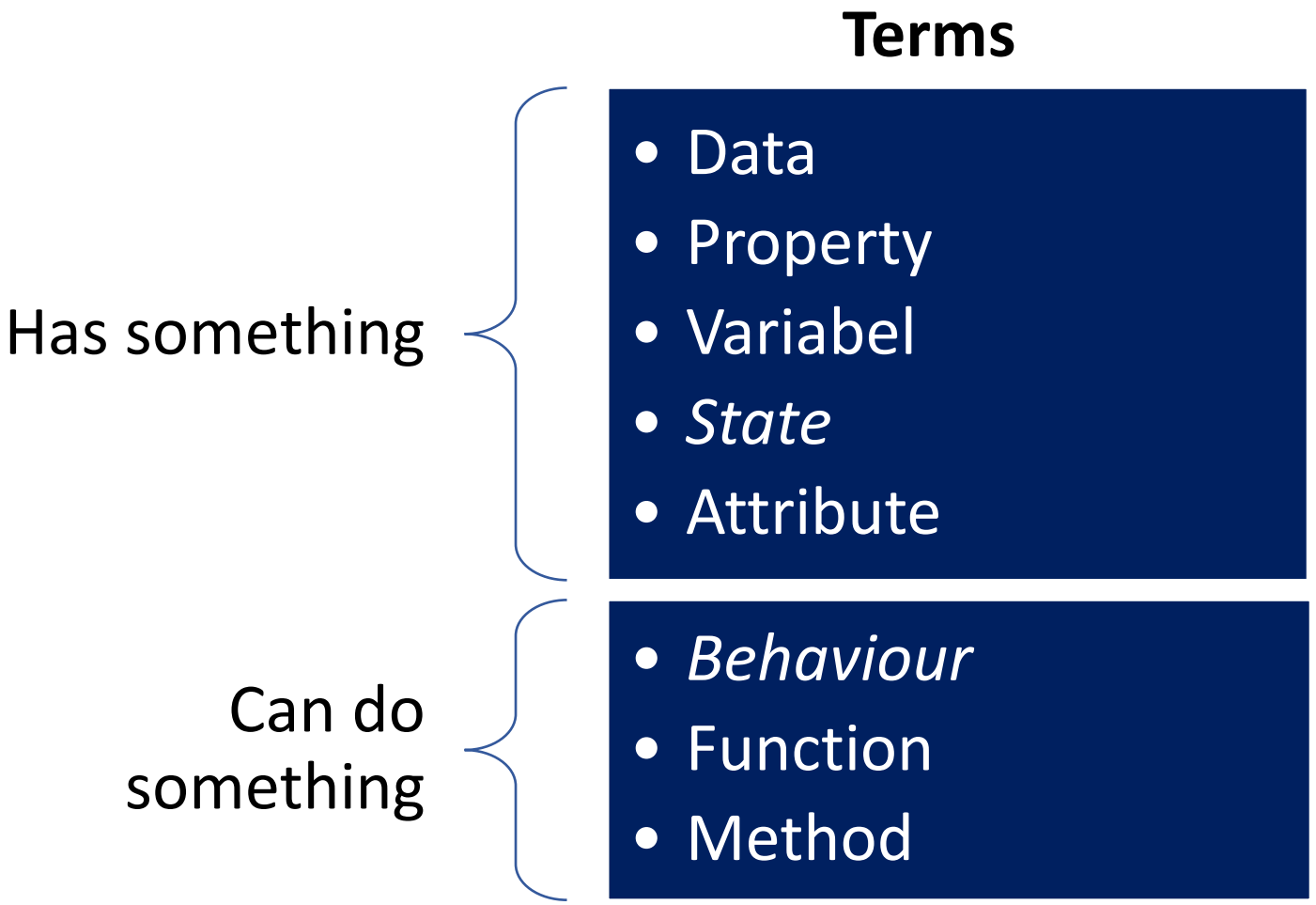
Examples of objects in the bedroom:

- Mattress
- Study desk
- Pillow1, Pillow2, etc

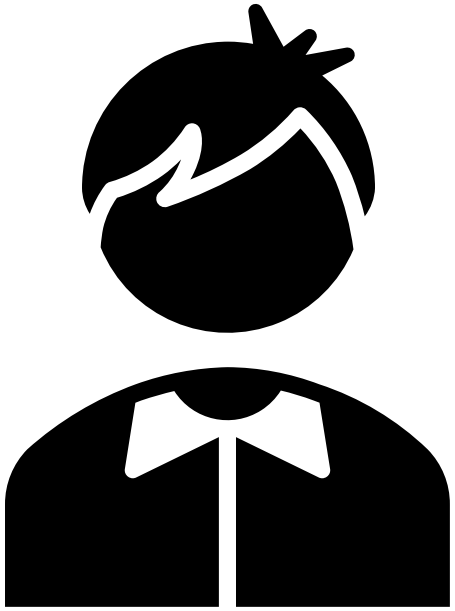
Examples of objects in a class:

- Student1, Student2, Student3, etc
- Whiteboard
- PC1, PC2, PC3, etc

Object Characteristics



Example Object (1)



Student1

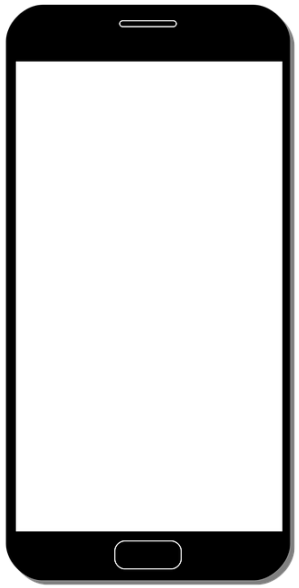
Attribute

- NIM → 2414210045
- Name → Will Dafoe
- IPK → 3.8
- Address → Malang

Method

- Do Examination
- View KHS
- View Schedule
- Presence
- Submit assignment

Example Object (2)



HP1

Attribute

- Merk → Samsung
- Type → S23 Ultra
- DisplaySize → 6.8
- Price → 20.000.000

Method

- Send Message
- Accept Call
- Open document
- Connect bluetooth

Class

Class is a **template** to create object

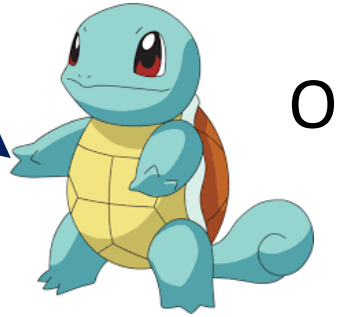
- All objects must come from a **plan/design/template/class**
- The process of creating an **object** from a **class** is called **instantiation**
- No class means no objects, and classes will be unusefull if there is no object created from it

Class

Class: **Pokemon**

Object: **Pokemon1**

Object: **Pokemon2**



Object: **Pokemon3**

Class VS Object

	Class	Object
Description	Still as a **plan/ template/ design/ blueprint**	**Real thing** that is already created from a class
Characteristic	General	Specific
Example	• Student • Lecturer • Course	• Student 1, Student 2, etc • Lecturer A, Lecturer B, etc • Course Operating System, Course Database

Does class have attribute and method?

- Because class must be created before object, of course **class** has **attributes** and **methods**, which belongs to object after it is created.
- However, the **attributes** and **methods** on **object** are **real** (can be filled with certain values), while the **attributes** and **methods** in **class** are still **design** (cannot be filled with values).

Class Implementation

- Basic form:

```
class ClassName{
      //attribute declaration
      //method declaration
}
```

- Example:

```
class HP{



}
```

Attribute

Attribute is **data/property/variable** that belongs to an object

- **Attribute** names are usually **nouns**
- Examples of attributes of:
- Student → NIM, name, IPK, address
- HP → brand, type, screen size, price
- Book → title, author, pages, publisher

Attribute Implementation

- Basic form:

```
dataType attributeName;
```

- Example:

```
String merk;
String type;
float screenSize;
int price;
```

Method

Method is **process/behaviour/function** that can be done by object

- Methods are used by objects to **interact with other objects**
- Method names are usually marked with **verbs**
- Examples of methods of students: **taking exams**, **viewing KHS**, **viewing schedules**, **taking attendance**, **collecting assignments**

Method Implementation

- Basc Form:

```
dataType methodName(dataType parameter){
      //method body
}
```

- Example:

```
void checkCondition(boolean c){
      if(c==true)
            System.out.println("This HP is a second-hand\n");
      else
            System.out.println("This HP is new\n");
}
```

Instantiation

Instantiation is a process **to create object** from class

- The keyword of instantiation is **new**
- Basic form:

```
ClassName objectName = new Constructor();
```

- Example:

```
HP phone1 = new HP();
```

Constructor has the same name as class name, which will be discussed in more detail on the next slide

Accessing Attribute and Method of an Object

- After creating object, then we can access its attribute as well as calling its method
- Accessing attribute:

```
objectName.attributeName = value;
```

- Accessing method:

```
objectName.methodName();
```

- Example:

```
phone1.merk = “Samsung”;
phone1.screenSize = 6.8f;
phone1.checCondition(false);
```

Source Code of Class HP

Class Declaration

Attribute

Method

```
public class HP {
    String merk;
    String tipe;
    float ukuranLayar;
    int harga;

    void cekKondisi(boolean c) {
        if (c == true)
            System.out.println(x:"HP ini second\n");
        else
            System.out.println(x:"HP ini masih baru\n");
    }

    void tampilInformasi() {
        System.out.printf(format:"HP merk %s tipe %s dengan ukuran layar %.1f\n", merk, tipe, ukuranLayar);
    }

    void mengirimPesan(String pesan, String penerima, String paketData) {
        if (paketData.equalsIgnoreCase(anotherString:"ada")) {
            System.out.printf(format:"Pesan %s berhasil dikirim ke %s", pesan, penerima);
        } else {
            System.out.printf(format:"Pengirima pesan ke %s gagal", penerima);
        }
    }
}
```

Source Code of Creating Object **phone1** from Class HP

```
public class HPMain {
    Run | Debug
    public static void main(String[] args) {
        HP phone1 = new HP();
        phone1.merk = "Samsung";
        phone1.tipe = "S23 Ultra";
        phone1.ukuranLayar = 6.8f;
        phone1.tampilInformasi();
        phone1.cekKondisi(c:false);
    }
}
```

Instantiation

Accessing attribute **merk** of object **phone1** and give **Samsung** value on it

Call method from object **phone1**

Constructor

Constructor is a **special method** that is used in an object instantiation

Why **special**:

- The method name is the same as the **class name**
- Does not have a method **data type**
- Can only run/called during the **instantiation** process
- Can have parameters

Constructor

Default Constructor — Constructor that has no parameter

- Example:

```
public HP(){

}
```

Parametric Constructor — Constructor that has parameter

- Example:

```
public HP(String mr, String tp, float ukuran){

}
```

Calling Constructor on Instantiation

- Constructor can only run at instantiation process
- Example of instantiation object using default constructor:

```
HP phone1 = new HP();
```

- Example of instantiation object using parametric constructor :

```
HP phone1 = new HP("Samsung", "S23 Ultra", 6.8f);
```

Example of Constructor

```
public class HP {
    String merk;
    String tipe;
    float ukuranLayar;
    int harga;

    public HP(String mr, String tp, float ukuran) {
        merk = mr;
        tipe = tp;
        ukuranLayar = ukuran;
    }

    void cekKondisi(boolean c) {
        if (c == true)
            System.out.println(x:"HP ini second\n");
        else
            System.out.println(x:"HP ini masih baru\n");
    }

    void tampilInformasi() {
        System.out.printf(format:"HP merk %s tipe %s dengan ukuran layar %.1f\n", merk, tipe, ukuranLayar);
    }

    void mengirimPesan(String pesan, String penerima, String paketData) {
        if (paketData.equalsIgnoreCase(anotherString:"ada")) {
            System.out.printf(format:"Pesan %s berhasil dikirim ke %s", pesan, penerima);
        } else {
            System.out.printf(format:"Pengirima pesan ke %s gagal", penerima);
        }
    }
}
```

Parametric Constructor

Example of Source Code that Call Constructor at Instantiation Process

```
public class HPMain {
    Run | Debug
    public static void main(String[] args) {
        HP phone1 = new HP();
        phone1.merk = "Samsung";
        phone1.tipe = "S23 Ultra";
        phone1.ukuranLayar = 6.8f;
        phone1.tampilInformasi();
        phone1.cekKondisi(c:false);
        HP phone2 = new HP(mr:"iPhone",tp:"15 Plus", ukuran:6.69f);
    }
}
```

Default Constructor

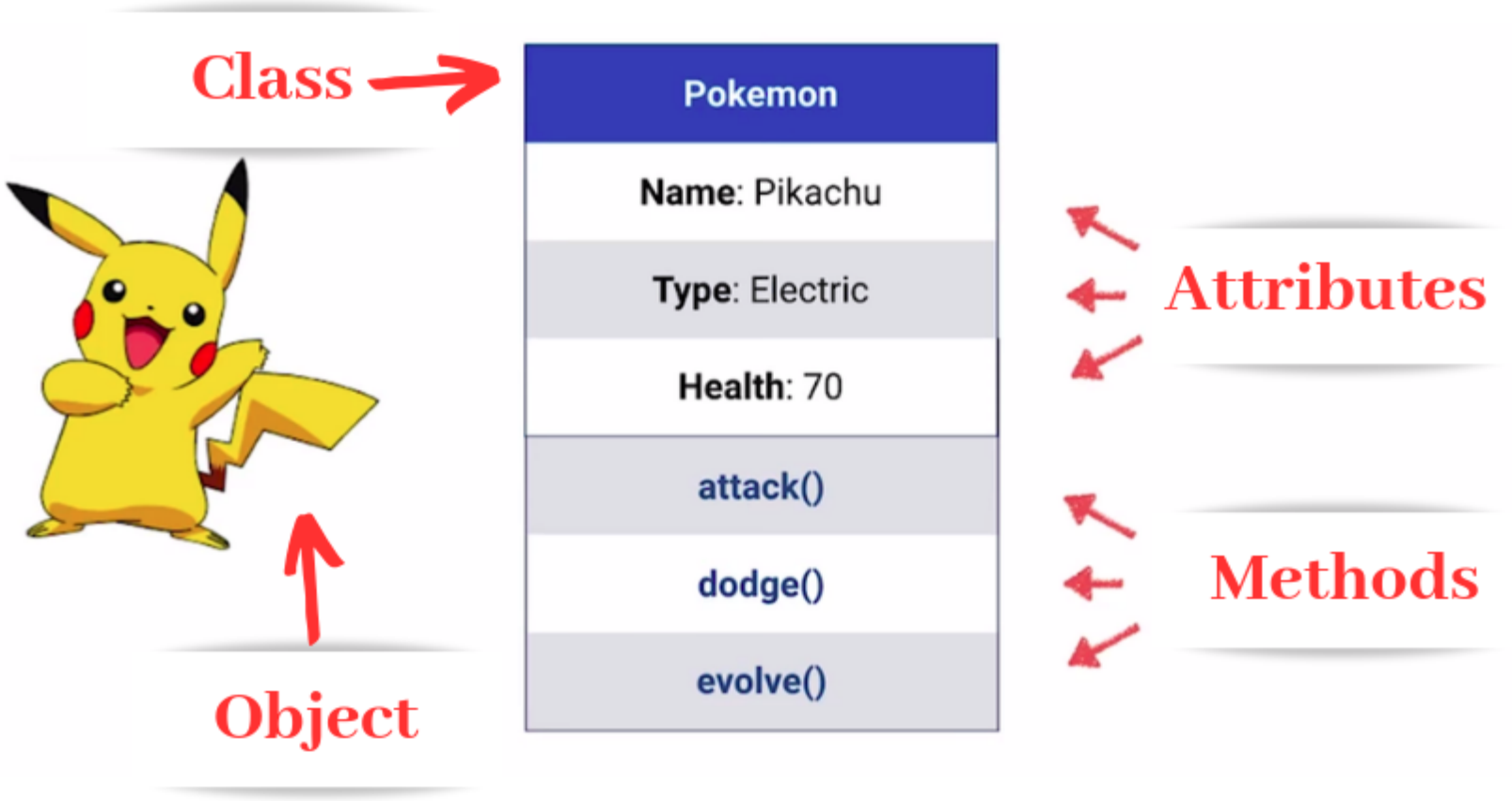
Parametric Constructor

Class Diagram (1)

- Represents a visual diagram for class design
- Example:

HP
merk: String type: String screenSize: float price: int
cekKondisi(c: boolean): void tampilInformasi(): void mengirimPesan(psn: String, penerima: String, pkt: String): void

Class Diagram (2)



https://blog.glugmvit.com/oops/

Task

1. Determine an object around you and determine the attributes (at least 4) and methods (at least 3) for this object!
2. In the JTI Classroom Loan Management Information System, determine what the objects are!
3. There is a class called Bang**u**nRuang, with two objects, **prismaSegitiga** and **Balok**, as in the following image. Create a class diagram for **BangunRuang**!