Pemrograman Berbasis Obyek

Pengenalan OOP

Oleh Politeknik Elektronika Negeri Surabaya

2017

Politeknik Elektronika Negeri Surabaya
Departemen Teknik Informatika dan Komputer

Konten

- Pengenalan konsep OOP
- Paradigma Pemrograman
- Perbedaan OOP dan terstruktur
- Capaian Pembelajaran
- Konsep utama OOP
- Konsep obyek, kelas, variabel, dan method
- Mengorganisasi program menjadi element yang disebut sebagai class, dan bagaimana class ini digunakan untuk membuat obyek.

Deskripsi

- Mata kuliah ini berisi tentang paradigma pemrograman yang berorientasi obyek.
- Bagaimana cara menyusun langkah-langkah pemecahan masalah menggunakan konsep class dan obyek, aturan-aturan berorientasi obyek, dan menggunakan tool bahasa pemrograman Java & class diagram beserta berbagai studi kasusnya dalam praktikum

Kompetensi

- Mampu membandingkan paradigma OOP dengan Struktural
- Mampu menjelaskan konsep utama OOP
- Mampu melakukan langkah-langkah pemecahan masalah menggunakan OOP
- Mampu membuat program menggunakan konsep OOP dengan bahasa Java
- Mampu menggunakan berbagai fitur Java lainnya secara umum
- Mampu menggunakan class diagram pada studi kasus

Generasi Bahasa Pemrograman

- Generasi 1: Machine / Binary
- Generasi 2: Assembly
- Generasi 3: (High Level)
  - Java, C++, Pascal, C
- Generasi 4 (Special Purpose Language)
  - Report Generator: Crystal Reports, RAVE
  - Forms Generator: Delphi, VB, C#, FoxPro
  - CASE: Rational Rose, Poseidon
  - DBMS: FoxPro, Oracle, SQL

Paradigma Pemrograman

- Suatu cara berpikir dalam membuat program komputer yang direpresentasikan dalam sejumlah konsep dan teknik pemrograman
- Terdapat banyak paradigma pemrograman
- Suatu bahasa pemrograman bisa mendukung lebih dari satu paradigma pemrograman

Contoh Bahasa Pemrograman

- Procedural
  - Pascal, C, COBOL, Fortran, ALGOL, Basic, PHP, dll.
- Object Oriented
  - Java, C#, C++, Objective C, PHP, Visual Basic.Net, Object Pascal, dll.
- Pada C++, bisa procedural, bisa juga Object Oriented !

OOP

- Paradigma pemrograman yg menggunakan pendekatan berorientasi pada obyek
- Jadi permasalahan yang ada dipandang sebagai obyek
- Obyek => suatu bentuk nyata yang dapat dibayangkan, memiliki segala sesuatu yang memang melekat padanya, dan dapat melakukan tindakan tertentu

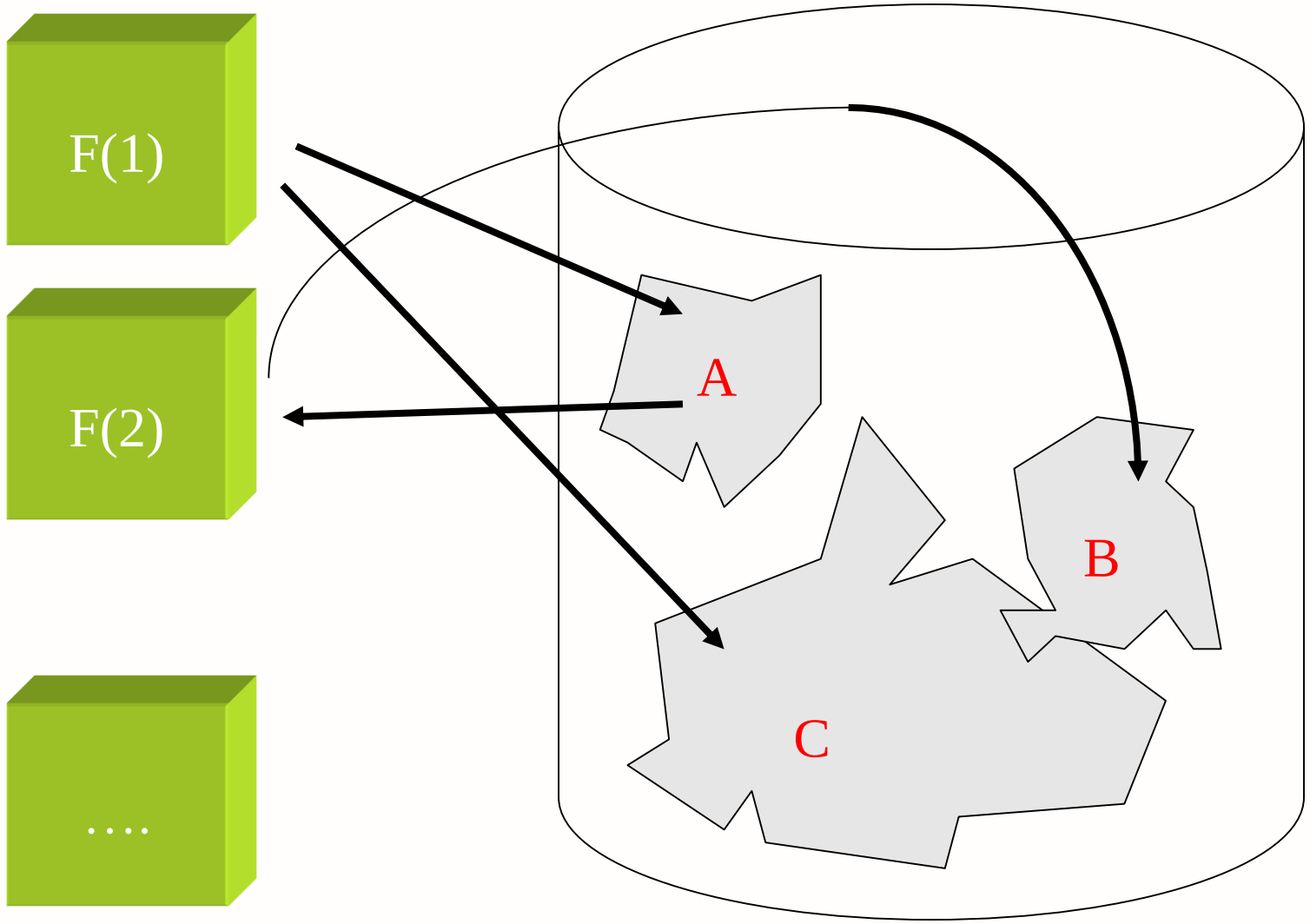
Pemrograman Berorientasi Obyek

- Fungsi dan data bukan menjadi dua hal yang terpisah.
- Fungsi dan data menjadi satu kesatuan yang disebut sebagai obyek aktif.
- Cara pandang → program adalah serangkaian obyek yang bekerjasama untuk menyelesaikan suatu problem.

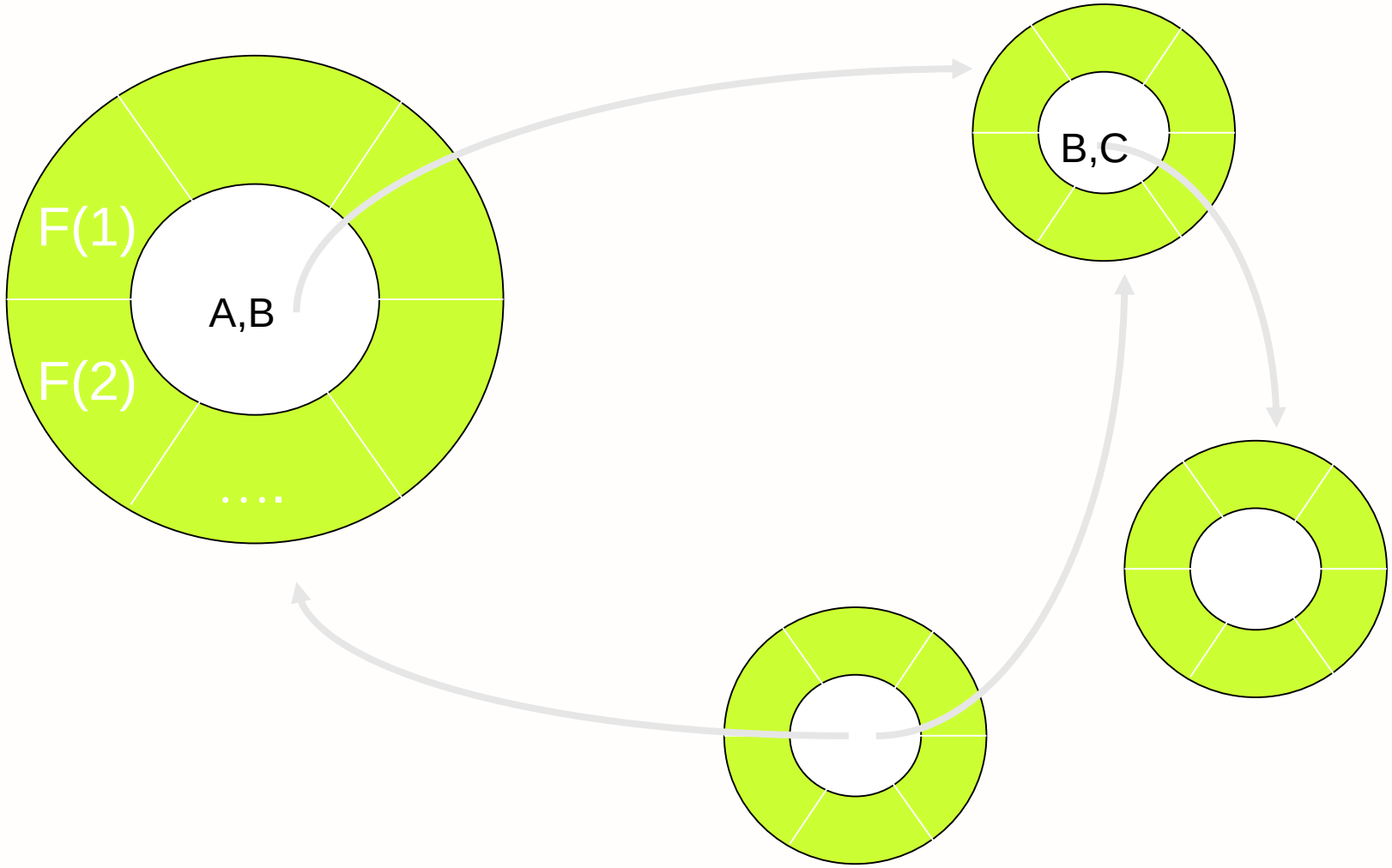
Procedural / Struktural vs OOP

- Procedural
  - Menyusun langkah-langkah untuk menyelesaikan suatu masalah
  - Misal: menghitung luas bangun segi empat “X”
  - Langkahnya:
    - Input panjang dan lebar segi empat “X”
    - Cari luas segi empat “X” dengan cara kalikan panjang dan lebar
    - Tampilkan luas segi empat “X”

Pemrograman Prosedural



Pemrograman berorientasi obyek



Procedural / Struktural vs OOP

- Object Oriented
  - Menyusun / merancang obyek yang akan dioperasikan
  - Segi empat “X” memiliki panjang dan lebar
  - Segi empat “X” bisa dihitung luasnya dengan panjang * lebar
  - Langkahnya:
    - Buat Obyek segi empat “X”, isikan data panjang dan lebar
    - Meminta obyek segi empat “X” menghitung luasnya

Obyek segi empat ”X” dibuat dari class Segi Empat

- Class: Segi Empat
- Atribut: sifat atau identitas yg melekat
  - Panjang
  - Lebar
- Behaviour: tingkah laku / kegiatan
  - Hitung Luas
  - Hitung Keliling

Obyek dalam PBO

- Segi Empat merupakan salah satu dari sekian banyak bangun 2 dimensi lainnya
  - Segitiga
  - Lingkaran
  - Belah Ketupat
  - Segilima
  - Segienam
- Kebanyakan atribut dan behaviour untuk benda 2 dimensi adalah mirip
- Namun bisa juga spesifik untuk masing-masing bendanya
- Sangat tergantung bendanya

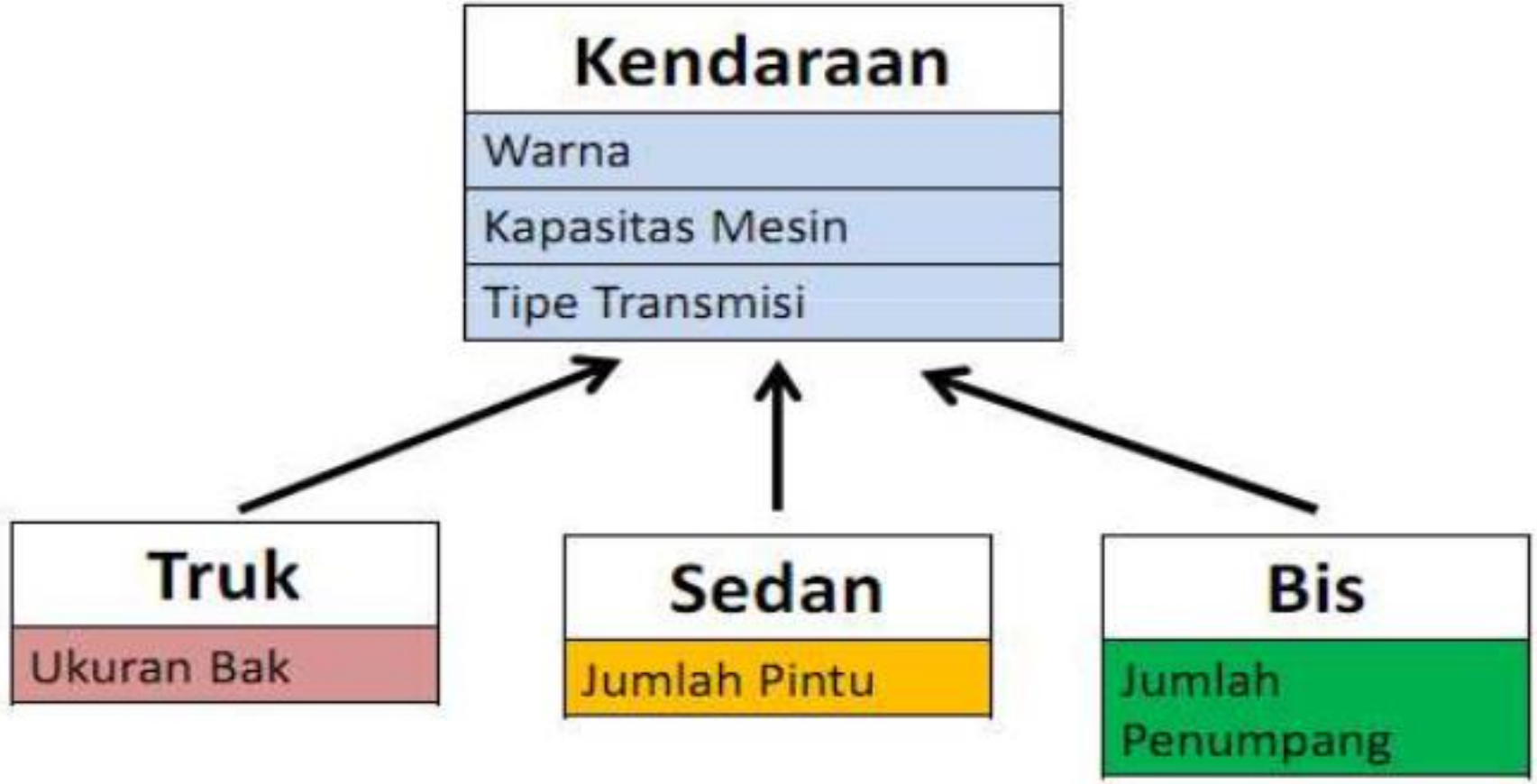
Contoh OOP : Kendaraan

Truk
Warna
Kapasitas Mesin
Tipe Transmisi
Ukuran Bak

Sedan
Warna
Kapasitas Mesin
Tipe Transmisi
Jumlah Pintu

Bis
Warna
Kapasitas Mesin
Tipe Transmisi
Jumlah Penumpang

Class Diagram OOP: Inheritance



Procedural / Struktural vs OOP

- Procedural
  - Lebih cepat untuk memecahkan masalah-masalah berskala kecil
  - Mudah membuatnya
- Object Oriented
  - Scalable, cocok untuk masalah-masalah berskala besar
  - Pengembangannya mudah

Pemrograman Berorientasi Obyek

- Fungsi dan data bukan menjadi dua hal yang terpisah.
- Fungsi dan data menjadi satu kesatuan yang disebut sebagai obyek aktif.
- Cara pandang → program adalah serangkaian obyek yang bekerjasama untuk menyelesaikan suatu problem.

Kata kunci OOP

- **Class** → template untuk membuat obyek**.**
- **Objek** → dapat berupa Class atau Instances. Harus berasal dari entitas atau konsep dunia nyata.
- **Atribut** → identitas unik dari obyek
- **Metode** → fungsi untuk pengaksesan atribut atau tugas tertentu
- **Enkapsulasi** → menyembunyikan struktur data dan implementasi suatu class.
- **Inheritansi** → merepresentasikan keterhubungan struktural antar class
- **Polymorphism** → kemampuan untuk merepresentasikan 2 bentuk obyek yang berbeda

Apakah Class?

- Definisi class: merupakan template untuk membuat obyek.
- Definisi class: merupakan prototipe / blue prints yang mendefinisikan variabel – variabel dan method – method secara umum.
- Obyek merupakan hasil instansiasi dari suatu kelas.
- Proses pembentukan obyek dari suatu class disebut dengan ***instantiation***.
- Obyek disebut juga ***instances.***

Apakah Obyek?

- Semua benda yang ada di dunia nyata dapat dianggap sebagai obyek.
- Contoh : rumah, mobil, sepeda motor, gelas, komputer, meja dll.

Karakteristik Obyek

- Setiap obyek memiliki state sebagai status (atribut).
- Setiap obyek memiliki tingkah laku (method)
- Contoh: obyek sepeda
  - Memiliki atribut →: pedal, roda, jeruji, warna, jumlah roda.
  - Memiliki method → : kecepatanya menaik, kecepatannya menurun, perpindahan gigi sepeda.

Karakteristik Obyek

- Penggambaran pemrograman berorientasi obyek = penggambaran pada dunia nyata.
- Pada pemrograman berorientasi obyek:
  - State disimpan pada → variabel
  - Tingkah laku disimpan pada → method

Atribut

- Definisi atribut : adalah **data** yang membedakan antara obyek satu dengan yang lain.
- Contoh: VolcanoRobot → A volcanic exploration vehicle, mempunyai atribut sebagai berikut:
  - Status → exploring, moving, returning home
  - Speed →1, 2, 3 dll in miles per hour)
  - Temperature → 100, 120, 130 dll (in Fahrenheit degrees)
- Dalam class atribut disebut juga dengan **variabel**.

Atribut

- Instance variable: adalah atribut untuk tiap obyek dari class yang sama.
- Tiap obyek mempunyai dan menyimpan nilai atributnya sendiri.
- Jadi tiap obyek dari class yang sama boleh mempunyai nilai yang sama atau beda.

- Class variable: adalah atribut untuk semua obyek yang dibuat dari class yang sama.
- Semua obyek mempunyai nilai atribut yang sama.
- Jadi semua obyek dari class yang sama mempunyai hanya satu nilai yang value nya sama.

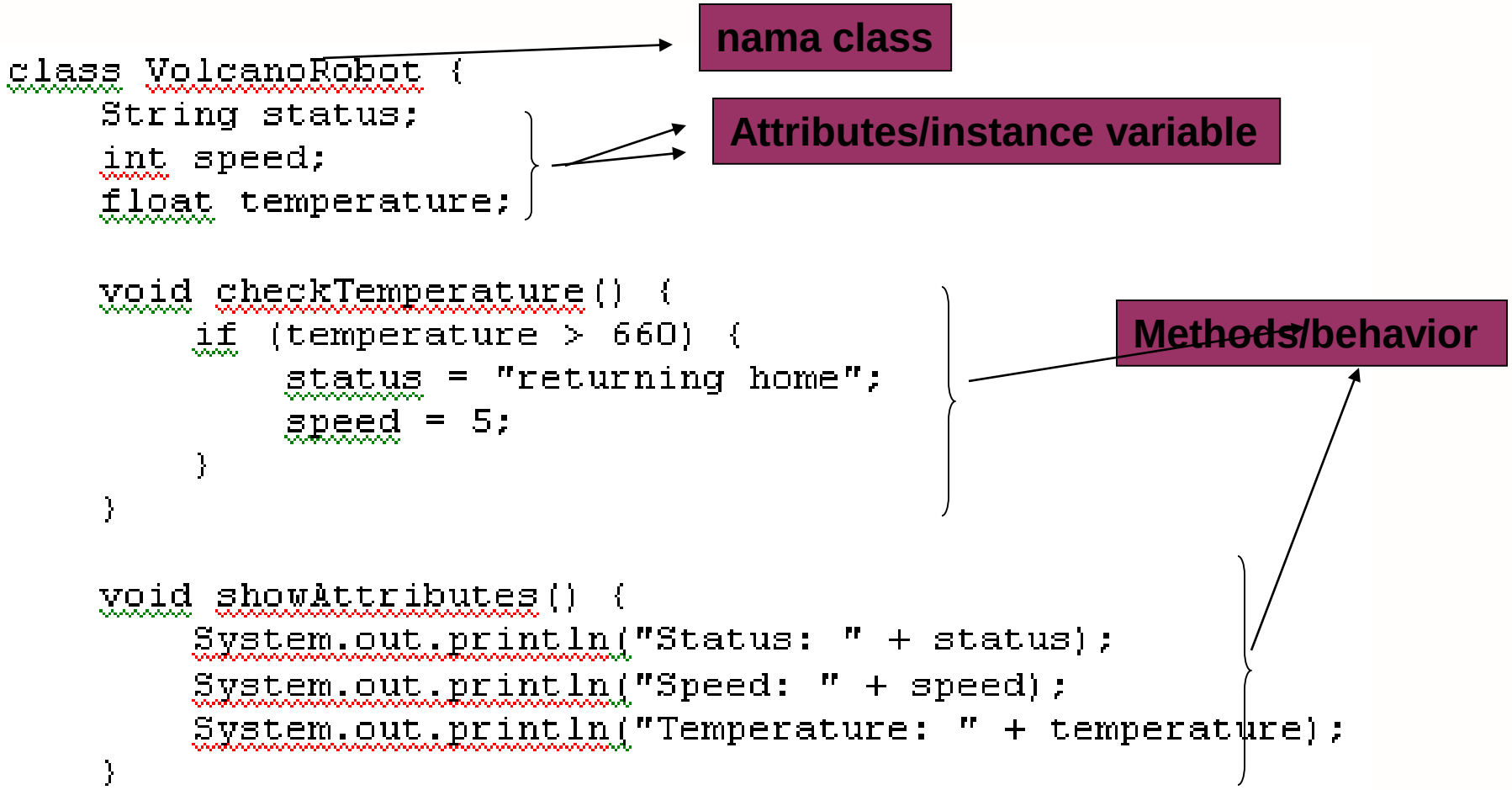
Tingkah Laku

- Tingkah laku adalah hal – hal yang bisa dilakukan oleh obyek dari suatu class.
- Tingkah laku dapat digunakan untuk mengubah nilai atribut suatu obyek, menerima informasi dari obyek lain, dan mengirim informasi ke obyek lain untuk melakukan suatu task.
- Contoh: VolcanoRobot
  - Check current temperature
  - Begin a survey
  - Report its current location
- Dalam class, tingkah laku disebut juga sebagai **method**.

Tingkah Laku

- Method: adalah serangkaian statements dalam suatu class yang menghandle suatu task tertentu.
- Cara obyek berkomunikasi dengan obyek lain adalah dengan menggunakan method.

Contoh: class, object, attributtes, dan behavior



Contoh: lanjutan

```
    public static void main(String[] arguments) {
        VolcanoRobot dante = new VolcanoRobot();
        dante.status = "exploring";
        dante.speed = 2;
        dante.temperature = 510;

        dante.showAttributes();
        System.out.println("Increasing speed to 3.");
        dante.speed = 3;
        dante.showAttributes();
        System.out.println("Changing temperature to 670.");
        dante.temperature = 670;
        dante.showAttributes();
        System.out.println("Checking the temperature.");
        dante.checkTemperature();
        dante.showAttributes();
    }
}
```

```
class Bicycle {
    int cadence = 0;
    int speed = 0;
    int gear = 1;

    void changeCadence(int newValue) {
                cadence = newValue;
    }

    void changeGear(int newValue) {
                gear = newValue;
    }

    void speedUp(int increment) {
                speed = speed + increment;
    }

    void applyBrakes(int decrement) {
                speed = speed - decrement;
    }

    void printStates() {
                System.out.println("cadence:" + cadence + " speed:" + speed + " gear:" + gear);
    }
}
```

```
class BicycleDemo {
    public static void main(String[] args) {
                // Create two different
                // Bicycle objects
                Bicycle bike1 = new Bicycle();
                Bicycle bike2 = new Bicycle();

                // Invoke methods on
                // those objects
                bike1.changeCadence(50);
                bike1.speedUp(10);
                bike1.changeGear(2);
                bike1.printStates();

                bike2.changeCadence(50);
                bike2.speedUp(10);
                bike2.changeGear(2);
                bike2.changeCadence(40);
                bike2.speedUp(10);
                bike2.changeGear(3);
                bike2.printStates();
    }
}
```

Konsep Dasar OOP

- Enkapsulasi (Encapsulation)
- Penurunan (Inheritance)
- Polimorfisme (Polymorphism)

Enkapsulasi

- Definisi enkapsulasi: Pembungkusan variabel dan method dalam sebuah obyek yang terlindungi serta menyediakan interface untuk mengakses variabel tersebut.
- Variabel dan method yang dipunyai suatu obyek, bisa ditentukan hak aksesnya.

Enkapsulasi

- Contoh: jam tangan
  - Penting sekali untuk mengetahui waktu.
  - Cara jam mencatat waktu dengan baik antara jam bertenaga baterai atau bertenaga gerak tidaklah penting kita ketahui.
- Dalam OOP, konsep enkapsulasi sebenarnya merupakan perluasan dari struktur dalam bahasa C.

Pewarisan

- Definisi : merupakan pewarisan atribut dan method dari sebuah class ke class lainnya.
- Class yang mewarisi → superclass
- Class yang diwarisi → subclass
- Subclass bisa berlaku sebagai superclass bagi class lainya → multilevel inheritance.

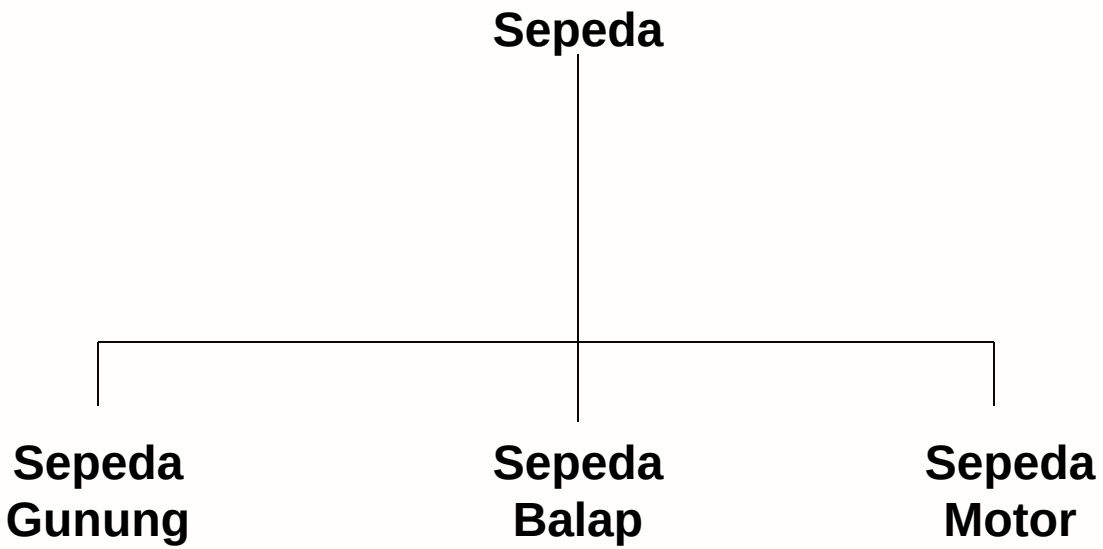
Pewarisan

Contoh : terdapat class sepeda dan sepeda gunung.

- Sepeda → superclass
- Sepeda gunung → subclass
- Sepeda gunung memiliki variabel dan method yang dimiliki oleh sepeda.

- Prinsip : Persamaan-persamaan yang dimiliki oleh beberapa kelas dapat digabungkan dalam sebuah class induk sehingga setiap kelas yang diturunkannya memuat hal-hal yang spesifik untuk kelas yang bersangkutan.

Pewarisan



Keuntungan pewarisan

- Subclass menyediakan state/behaviour yang spesifik yang membedakan dengan superclass → memungkinkan programmer untuk menggunakan ulang source code dari superclass yang telah ada.
- Programmer dapat mendefinisikan superclass khusus yang bersifat generik, yang disebut abstract class, untuk mendefinisikan class dengan tingkah laku dan state secara umum.

Single dan multiple inheritance

- C → multiple inheritance
- Suatu class diperbolehkan untuk mempunyai lebih dari satu superclass.
- Variabel dan method yang diwariskan merupakan kombinasi dari superclass-nya.

- Java → single inheritance
- Suatu class hanya boleh mempunyai satu superclass.

Multilevel inheritance

- Suatu subclass bisa menjadi superclass bagi class yang lain.

Polimorfisme

- Definisi: Kemampuan suatu obyek untuk mempunyai lebih dari satu bentuk .

Materi:

1. Dasar Pemrograman Java
2. Operator, Percabangan, Perulangan
3. Array, String
4. Class
5. Enkapsulasi
6. Inheritance
7. Overriding dan Overloading
8. Polimorphisme
9. Abstract Class & Interface
10. Exception Handling
11. Collection & Generic
12. GUI & Event Handling
13. Case Study
14. Case Study

Tugas

1. Buatlah makalah yang berisi tentang perkembangan teknologi Java dan uraikan berbagai macam teknologi Java serta aplikasinya saat ini.

1. Oracle Java Documentation, The Java™ Tutorials, https://docs.oracle.com/javase/tutorial/, Copyright © 1995, Oracle 2015.
2. Tita Karlita, Yuliana Setrowati, Rizky Yuniar Hakkun, Pemrograman Berorientasi Obyek, PENS-2012
3. Sun Java Programming, Sun Educational Services, Student Guide, Sun Microsystems, 2001.
4. John R. Hubbard, Programming With Java, McGraw-Hill, ISBN: 0-07-142040-1, 2004.
5. Patrick Niemeyer, Jonathan Knudsen, Learning Java, O'reilly, CA, ISBN: 1565927184, 2000.
6. Philip Heller, Simon Roberts, Complete Java 2 Certification Study Guide, Third Edition, Sybex, San Francisco, London, ISBN: 0-7821-4419-5, 2002.
7. Herbert Schildt, The Complete Reference, Java™ Seventh Edition, Mc Graw Hill, Osborne, ISBN: 978-0-07-163177-8, 2007