Konsep OOP

IF2210 – Semester II 2022/2023

Tim Pengajar IF2210

Tujuan

- Mahasiswa memahami :
  - Definisi sebuah perangkat lunak yang dibangun berorientasi Objek
  - Perbedaan kelas dan objek
  - Siklus hidup objek: *definition*, *declaration*, penciptaan, *manipulation*, pemusnahan
  - Manipulasi objek:
    - *Object comparison*
    - *Assignment*, *clone* dan *deep clone*

Definisi OOP

- **[Meyer98]:** Sebuah sistem yang dibangun berdasarkan metoda berorientasi objek adalah sebuah sistem yang komponennya di-enkapsulasi menjadi kelompok data dan fungsi, yang dapat mewarisi atribut dan sifat dari komponen lainnya, dan komponen-komponen tersebut saling berinteraksi satu sama lain.

Memrogram Secara OOP

- Merancang program secara OO "*fundamentally different*" dibandingkan pendekatan structural. [Booch91]
- Tidak lagi membagi persoalan ke dalam data dan fungsi/prosedur.
- Melainkan bagaimana membagi ke dalam objek-objek yang memiliki peran & tanggung jawab masing-masing.
- Berpikir dalam terminologi objek akan berefek terhadap kemudahan mendesain program sebab dunia nyata terdiri atas objek-objek.

Objek & Kelas

Apa itu Objek? (1)

- **[West04]**: Object is *"the quanta from which the universe is constructed."*
- Objek adalah benda (*thing*) atau sebuah entitas
  contoh objek: mobil, buku, *bank account*, gajah, lagu, film, dll.
- Di dunia komputer, window, mouse, menu, textbox, button adalah objek.
- Objek bisa berbentuk objek fisik atau intangible seperti lagu.

Apa itu Objek? (2)

- Objek terbentuk atas objek-objek lain.
  - Contoh: objek mobil terdiri atas objek mesin, objek chassis, objek body, dst.
    - Objek mesin terdiri atas objek blok silinder, objek busi, objek piston, dst.
- Dalam konsep OO, “semua” adalah objek—termasuk integer, character, dll.
  - Kebanyakan bahasa pemrograman menganggap integer dst. sebagai tipe data primitif (bukan objek) karena alasan kinerja.

Objek

- Objek memiliki perilaku tertentu untuk memenuhi suatu tanggung jawab yang disepakati. (“Layanan” yang tersedia.)
  - Objek harus memiliki akses terhadap informasi yang dibutuhkan untuk menjalankan tanggung jawabnya.
    - Informasi tersebut bisa dimiliki sendiri, ataupun ditanyakan ke objek lain.
- Tanggung jawab sebuah objek dapat berupa:
  - Memberi informasi tertentu bagi objek lain yang meminta.
  - Melakukan perhitungan (komputasi).
  - Memberi tahu perubahan state dirinya.
  - Mengkoordinir objek-objek lain.

Mendefinisikan Objek

- Pada implementasi di bahasa pemrograman, objek didefinisikan dengan sekelompok **atribut** dan **method**.
  - **Atribut** adalah informasi yang menjadi bagian dari objek yang dimaksud.
    - Karena “semua adalah objek” maka atribut pun berupa objek.
    - Nilai atribut menentukan karakteristik dan state suatu objek.
  - **Method** adalah aksi atau perilaku suatu objek.
    - Method dieksekusi untuk memenuhi tanggung jawab suatu objek.

Objek vs. ADT

- Objek memiliki atribut (≈data) dan method (≈prosedur/fungsi). Jadi apa bedanya dari ADT?
  - Sebuah objek tidak mengekspos isi perutnya ke objek lain.
    - Objek lain tidak boleh tahu bagaimana sebuah objek mengelola informasi yang dimilikinya secara internal.
  - Method merefleksikan ekspektasi pada domain persoalan sedangkan fungsi merefleksikan detail implementasi pada program.
- Perancangan objek dimulai dari tanggung jawab (method) apa saja yang dimiliki suatu objek, dilanjutkan dengan memutuskan informasi (atribut) apa saja yang diperlukan objek untuk menjalankan tanggung jawab tersebut.

Message

- **Message** adalah komunikasi formal yang dikirim oleh sebuah objek ke objek lainnya untuk meminta sebuah layanan (meminta objek lain memenuhi tanggung jawabnya).
- Jenis message:
  - *Imperative*: menyuruh objek melakukan perubahan state.
  - *Informational*: hanya "memberitahu", tidak mengharapkan adanya perubahan state.
  - *Interrogatory*: meminta informasi.
- Objek menginvokasi method yang bersesuaian dengan message yang diterima. Misal (C++, Java):
  - `Stack.push(10); // mengirim message push dengan argumen 10 ke objek Stack.`
    - Objek Stack menginvokasi method `void push(int item)`

Protocol

- **Protocol** (kadang disebut **interface**) adalah spesifikasi message apa saja yang dapat ditangani oleh sebuah objek.
  - Dalam C++ langsung bersesuaian dengan *header file* (xxx.h).
  - Pada Java, dapat dilihat pada dokumentasi (Javadocs).

Contoh Objek

- Sebuah lampu lalu lintas (LLL) adalah sebuah objek.
  - Tanggung jawab: memberitahukan perubahan state dirinya kepada objek yang “berlangganan”:
    - Pengemudi mengirim message “subscribe” terhadap layanan LLL ketika mendekati perempatan.
  - Pengemudi sebagai pengguna layanan dari LLL, tidak perlu tahu bagaimana LLL “menghitung” kapan harus berubah state.
    - Apakah berdasarkan timer sederhana? ⟶ LLL memiliki atribut “Timer”.
    - LLL cerdas yang menghitung kepadatan dari semua arah supaya bisa menentukan siapa yang harus diberi prioritas? ⟶ LLL memiliki atribut sensor atau berkoordinasi dengan objek sensor.
  - Pengemudi sebagai objek harus patuh terhadap tanggung jawabnya: berhenti jika menerima message “merah”, dst.

Mind Exercise

- Definisikan apa saja tanggung jawab objek Vending Machine!
- Tentukan atribut apa saja yang diperlukan objek Vending Machine untuk memenuhi tanggung jawabnya!

Kelas

- **Kelas** adalah *blueprint* yang mendeskripsikan objek-objek.
  - Definisi atribut dan method.
- Misalkan kelas *lampu lalu lintas dengan timer sederhana*, mendefinisikan bahwa setiap objek dari kelas ini memiliki timer di dalamnya.
  - Setiap *instance* dari kelas ini adalah objek lampu lalu lintas yang memiliki timer, namun konfigurasi timer setiap *instance* dapat berbeda.
- Karena “semua adalah objek”, secara konseptual Kelas pun adalah sebuah objek yang bertanggung jawab menciptakan objek-objek yang sesuai spesifikasi yang dimilikinya.
  - Di bahasa pemrograman, umumnya kelas bukan objek.

Analogi

- *Blueprint* dari sebuah rumah bukanlah rumah.
- Arsitek membuat *blueprint*. Kontraktor membuat rumah dari blueprint.
- *Programmer* mendefinisikan kelas yang nantinya akan dibentuk objek dari kelas tersebut. Objek adalah instansiasi kelas.
- *Programmer* dapat bersudut pandang sebagai arsitek atau kontraktor.
  - Jika sudut pandang bercampur-campur, desain kelas dapat menjadi tidak “bersih”.

Kelas dan Objek

- Program akan menciptakan objek-objek dari sebuah kelas
- Dari sebuah kelas bisa diciptakan banyak objek
- Objek dibuat berdasarkan spesifikasi kelas
- Setiap objek dimiliki oleh suatu kelas

- Program utama pada OOP “seharusnya” hanya bertugas menciptakan satu objek.
  - Objek ini akan menciptakan dan mengkoordinir objek-objek lain.

Kelas vs Objek

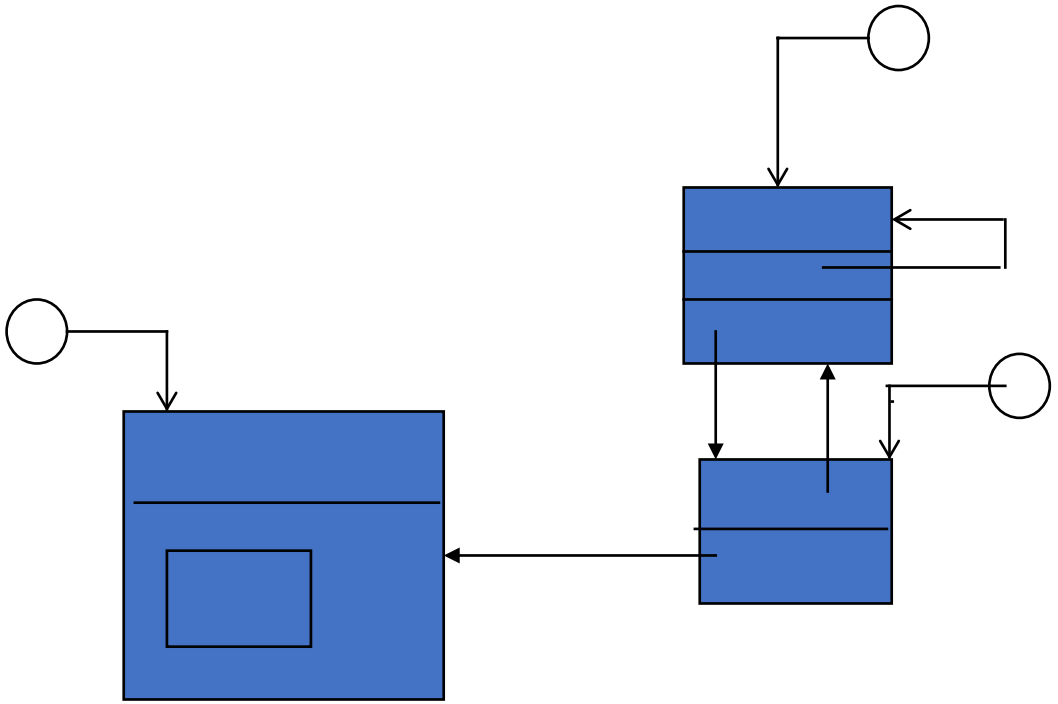
- Kelas ~ “type”
  - static definition, defined in source code
- Objek ~ “variable”
  - instance of class, exists during run-time (execution)
- Siklus hidup objek:
  - creation, manipulation, destruction
- OOP:
  - Define class, create objects
  - Manage objects, their life cycle and states

Penciptaan Objek

- Objek diciptakan dengan mengirim message kepada kelas untuk menginvokasi suatu method khusus yang disebut **constructor** [ctor].
  - Dalam bahasa OO, nama ctor biasanya sama dengan nama kelas.
- ctor dipakai untuk menciptakan objek dan menginisialisasi atribut-atributnya.
  - Setiap bahasa memiliki aturan inisialisasi “default” jika pembuat kelas tidak menuliskan ctor.

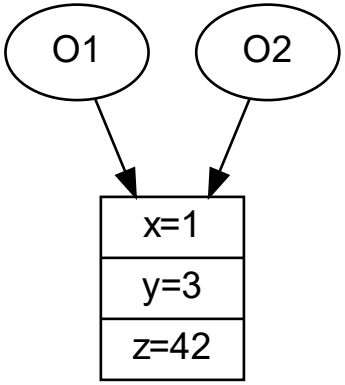
Penugasan Objek

- Assignment
- Copy
- Clone
- Deep Clone



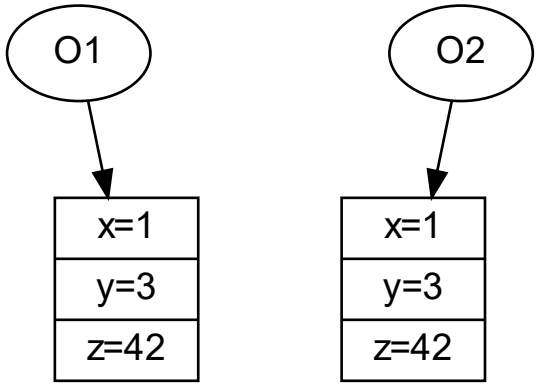
Membandingkan Objek (1)

- *Reference comparison*: membandingkan apakah O1 dan O2 adalah dua buah reference yang mengacu ke objek yang sama?



Membandingkan Objek (2)

- *Object comparison*: membandingkan apakah O1 dan O2 adalah dua buah objek yang identik kandungan informasinya?
  - Kasus menjadi rekursif jika atribut objek adalah objek yang mengandung objek lagi (bukan “tipe primitif”).



Penugasan & Pembandingan Objek

- Pada kebanyakan kasus, kita hanya akan peduli pada *reference comparison*.
  - Misal: apakah dua *reference* mengacu ke satu orang yang sama.
  - Umumnya kita tidak peduli apakah dua orang yang berbeda punya nama, tempat, dan tanggal lahir yang sama.
- Secara semantik, *object comparison* berbicara tentang dua objek yang berbeda tapi mirip ⟶ biasanya dapat distrukturkan ulang menjadi hierarki kelas dari pada *object comparison*.
  - Misal: sesama objek dari kelas X.

Karakteristik OOP

Karakteristik OOP

- Abstraction
- Encapsulation
- Pewarisan (inheritance)
- Composability
  - →Reuseability
- Specialization
- Generalization
- Communication between objects
- Polymorphism, dynamic binding

Konsep OOP: Enkapsulasi

- Menurut kamus: membungkus dengan kapsul
- Contoh: komputer, membungkus prosesor, memori, *motherboard*, kabel-kabel, dll
  - Pengguna berinteraksi melalui antarmuka.
- Contoh lain: TV, DVD player, kamera, mobil, dll
  - mobil membungkus mesin, chassis, body, dst.

Enkapsulasi

- Detil teknis disembunyikan dari pengguna ⟶ *information hiding*
- Data dan method, atribut dan fungsionalitas dikombinasikan dalam sebuah unit, sebuah kelas
- Akses data melalui method atau interface (tidak diakses langsung)
  - *In fact*, pada sebagian besar kasus, desain yang baik adalah data tidak dapat diakses sama sekali (kembali ke konsep “tanggung jawab” objek)

“Level” of complexity of OOP

- OOP using ADT
- OOP with generic class
- OOP with inheritance
- OOP with inheritance and polymorphism
- OOP – concurrent programming
- OOP – distributed, concurrent and parallel

Konkurensi hanya akan dibahas sebatas fitur bahasa Java. Kuliah fokus sampai dengan inheritance dan polymorphism.