MATA KULIAH – ANALISIS DAN PERANCANGAN SISTEM INFORMASI

# TOPIK 5: Object Modeling (Bagian Satu)

Disusun oleh:

Wahyu Indra Satria, S.Kom., M.Kom.

# *Learning Objectives*

**(Tujuan Pembelajaran)**

- **Memahami pengenalan dari *objects*, *attributes*, *methods*, *messages*, dan *classes***
- **Memahami relasi/hubungan antara *objects* dan *classes***

## ➢ *Overview* Analisis *Object–Oriented*:

- Pengembangan sistem paling populer adalah *structured–analysis*, *object–oriented analysis*, dan *agile methods*.
- Analisis *object–oriented* (O–O) menggambarkan suatu sistem informasi dengan mengidentifikasi hal-hal yang disebut **objek.**
- **Objek** mewakili orang, tempat, peristiwa, atau transaksi nyata.
- Contohnya ketika seorang pasien membuat janji bertemu dokter, pasien adalah objek, dokter adalah objek, dan janji itu sendiri adalah objek.
- Analisis O-O adalah pendekatan populer yang melihat sistem dari sudut pandang objek itu sendiri saat mereka berfungsi dan berinteraksi.
- Produk akhir dari analisis O-O adalah model objek, yang merepresentasikan sistem informasi dalam bentuk objek dan konsep O-O.

**Terkait materi topik ini, silahkan telusuri pada buku referensi 1 dimulai dari halaman 180.**

## ➢ *Overview* Analisis *Object–Oriented*: (lanjutan…)

- Analisis O–O adalah metode analisis dengan memeriksa *requirement* (syarat/keperluan) yang harus dipenuhi sebuah sistem, dari sudut pandang kelas–kelas dan objek–objek yang ditemui dalam ruang lingkup yang berhubungan

- *Unified Modeling Language* (UML–bahasa pemodelan terpadu) adalah metode yang banyak digunakan untuk memvisualisasikan dan mendokumentasikan sistem informasi.

**Terkait materi topik ini, silahkan telusuri pada buku referensi 1 dimulai dari halaman 180.**

- Dalam bab ini, UML digunakan untuk mengembangkan model objek.

- Langkah pertama adalah memahami istilah O-O dasar, termasuk *objects, attributes, methods, messages,* dan *classes*.

- Bab ini menunjukkan bagaimana analis sistem (*system analyst*) menggunakan istilah tersebut untuk menggambarkan suatu sistem informasi.

# Pengenalan *Objects*, *Attributes*, *Methods*, *Messages*, dan *Classes* (lanjutan...)

## Pengenalan *Objects*:

- Objek mewakili seseorang, tempat, peristiwa, atau transaksi yang signifikan bagi suatu sistem informasi.
- Pada bab sebelumnya, DFD telah dibuat yang memperlakukan data dan proses secara terpisah, dimana sebuah objek, bagaimanapun, termasuk data dan proses yang mempengaruhi data tersebut.

**Terkait materi topik ini, silahkan telusuri pada buku referensi 1 dimulai dari halaman 180.**

- Misalnya, objek pelanggan memiliki nama, alamat, nomor rekening, dan saldo saat ini.
- Objek pelanggan juga dapat melakukan tugas-tugas tertentu, seperti menempatkan pesanan, membayar tagihan, dan mengubah alamat mereka.
- Pertimbangkan contoh sederhana tentang bagaimana UML dapat menggambarkan sebuah *family* dengan *PARENT* dan *CHILDREN*.
- UML mewakili objek sebagai persegi panjang dengan nama objek di atas, diikuti oleh atribut dan metode objek.

## ➢ Pengenalan *Objects*: (lanjutan…)

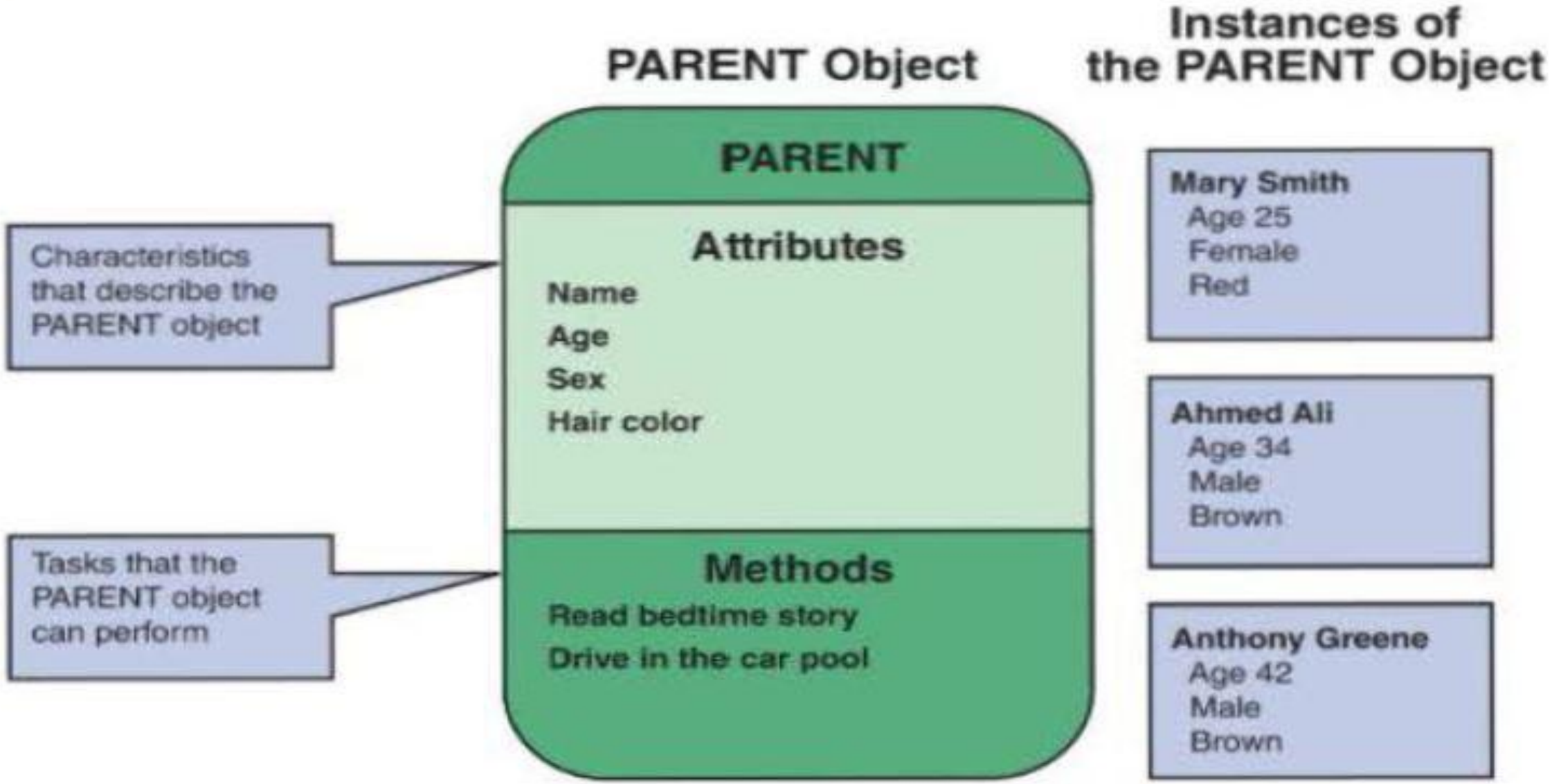



**FIGURE 6-1** The PARENT object has four attributes and two methods. Mary Smith, Ahmed Ali, and Anthony Greene are instances of the PARENT object.

## Pengenalan *Objects*: (lanjutan…)

- **FIGURE 6–1 (Pada *slide* 6)** menunjukkan objek *PARENT* dengan ***attributes*** tertentu seperti nama, umur, jenis kelamin, dan warna rambut.
- Jika ada dua *PARENT*, maka ada dua ***instances*** dari objek *PARENT*.
- Objek *PARENT* dapat melakukan ***methods***, seperti membacakan dongeng sebelum tidur, mengemudikan mobil van, atau menyiapkan makan siang sekolah.
- Ketika objek *PARENT* menerima ***messages***, ia melakukan *action* atau ***methods***.
- Contohnya pesan *GOOD NIGHT* dari *CHILD* mungkin memberitahukan objek *PARENT* untuk membacakan cerita pengantar tidur, sedangkan pesan *DRIVE* dari *PARENT* lain menandakan bahwa objek *PARENT* untuk mengemudi ke tempat parkir mobil.

**Terkait materi topik ini, silahkan telusuri pada buku referensi 1 dimulai dari halaman 180.**

## ➢ Pengenalan *Objects*: (lanjutan…)

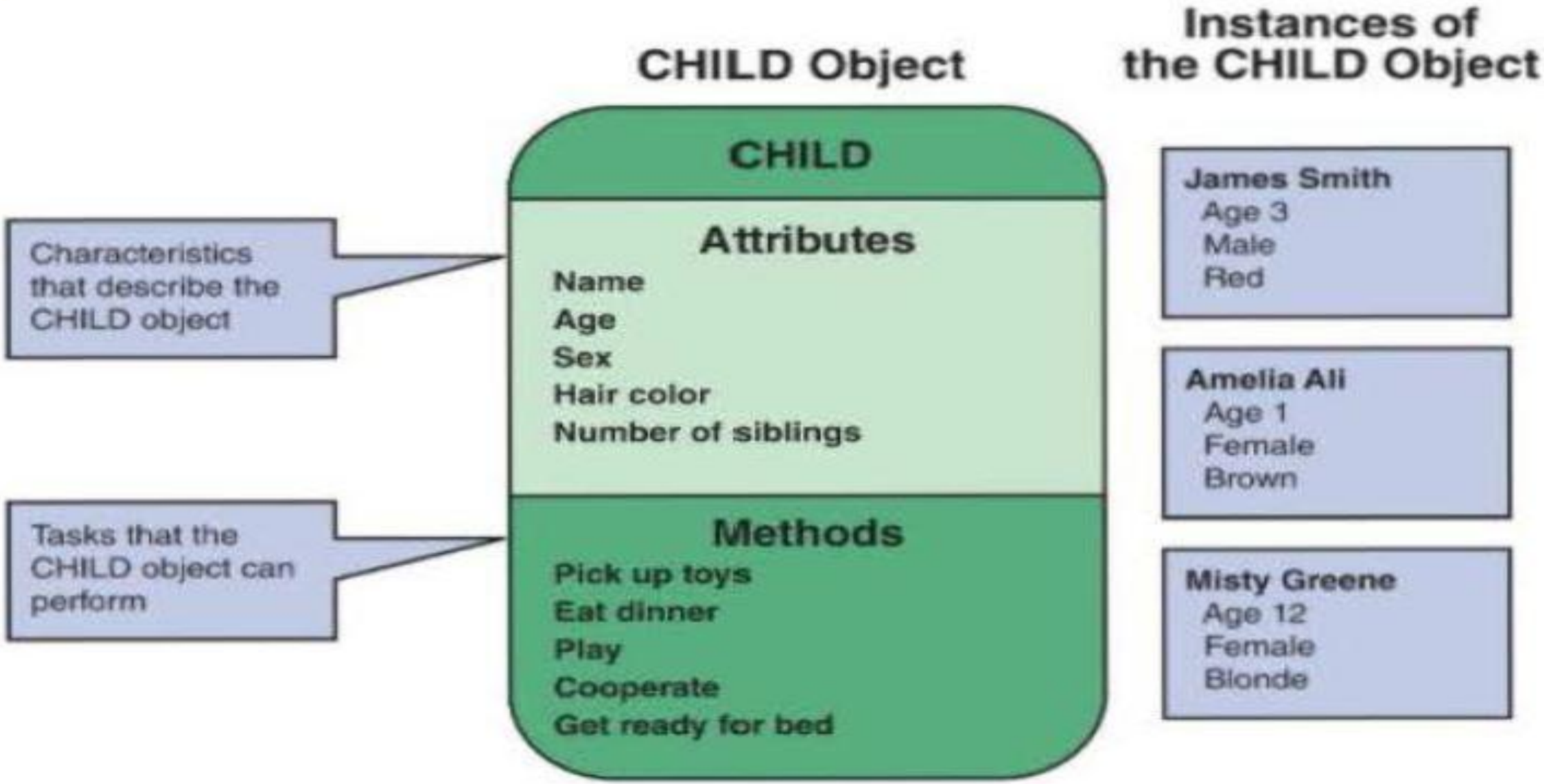



**FIGURE 6-2** The CHILD object has five attributes and five methods. James Smith, Amelia Ali, and Misty Greene are instances of the CHILD object.

## Pengenalan *Objects*: (lanjutan…)

- **FIGURE 6–2 (Pada *slide* 8)** menunjukkan objek *PARENT* dengan ***attributes*** tertentu seperti nama, umur, jenis kelamin, dan warna rambut.
- Objek *CHILD* memiliki ***attributes*** yang sama dengan objek *PARENT* dan ***attributes*** tambahan yang menunjukkan jumlah saudara kandung (***number of siblings***).
- Sebuah objek *CHILD* melakukan ***methods*** tertentu seperti mengambil mainan, makan malam, bermain, bekerja sama, dan bersiap-siap untuk tidur.
- Dalam memberi sinyal pada objek *CHILD* untuk melakukan tugas tersebut, *PARENT* dapat mengirim pesan tertentu yang akan dipahami oleh objek *CHILD*.
- Contohnya pesan *DINNER'S READY* memberitahu objek *CHILD* untuk datang ke meja, sedangkan pesan *SHARE WITH YOUR BROTHER/SISTER* memberitahu objek *CHILD* untuk bekerja sama dengan objek *CHILD* lainnya.

**Terkait materi topik ini, silahkan telusuri pada buku referensi 1 dimulai dari halaman 180.**

## Pengenalan *Attributes*:

- Sebuah *OBJECT* memiliki ***attributes*** tertentu yang merupakan karakteristik yang menggambarkan *OBJECT*.
- Jika *OBJECT* mirip dengan kata benda, ***attributes*** mirip dengan kata sifat yang menggambarkan karakteristik suatu *OBJECT*.
- Misalnya mobil memiliki ***attributes*** seperti merek, model, dan warna, serta beberapa *OBJECT* mungkin memiliki beberapa ***attributes***, yang lain mungkin memiliki lusinan.
- Analis sistem (*system analyst*) mendefinisikan ***attributes*** *OBJECT* selama proses desain sistem.
- Dalam sistem berorientasi objek (*object–oriented system*), *OBJECT* dapat mewarisi, atau memperoleh ***attributes*** tertentu dari objek lain.
- *OBJECT* dapat memiliki ***attributes*** khusus yang disebut *STATE*.
- *STATE* dari *OBJECT* adalah kata sifat yang menggambarkan status objek saat ini.
- Contohnya tergantung pada *STATE*, siswa dapat menjadi siswa masa depan, siswa saat ini, atau siswa sebelumnya.
- Demikian pula rekening bank dapat aktif, tidak aktif, ditutup, atau dibekukan.

**Terkait materi topik ini, silahkan telusuri pada buku referensi 1 dimulai dari halaman 180.**

## Pengenalan *Methods*:

- *OBJECT* juga memiliki ***methods***, yang merupakan tugas atau fungsi yang dilakukan *OBJECT* saat menerima ***message***, atau perintah untuk melakukannya.
- Sebagai contoh, sebuah mobil melakukan ***methods*** yang disebut *OPERATE WIPER* ketika dikirimi ***message*** pengendalian wiper, dan bisa *APPLY BRAKES* ketika dikirim ***message*** menekan pedal rem.
- Sebuah metode mendefinisikan tugas–tugas tertentu yang *OBJECT* dapat lakukan.

**Terkait materi topik ini, silahkan telusuri pada buku referensi 1 dimulai dari halaman 180.**

| Method: | Steps: |
|---|---|
| MORE FRIES | 1. Heat oil<br>2. Fill fry basket with frozen potato strips<br>3. Lower basket into hot oil<br>4. Check for readiness<br>5. When ready, raise basket and let drain<br>6. Pour fries into warming tray<br>7. Add salt |

**FIGURE 6-3** The MORE FRIES method requires the server to perform seven specific steps.

# Pengenalan *Objects*, *Attributes*, *Methods*, *Messages*, dan *Classes* (lanjutan...)

## Pengenalan *Messages*:

- ***Messages*** adalah suatu perintah yang memberitahu suatu *OBJECT* untuk melakukan ***methods*** tertentu.
- Misalnya, ***messages*** *PICK UP TOYS* mengarahkan kelas *CHILD* untuk melakukan semua langkah yang diperlukan untuk mengambil mainan.
- Kelas *CHILD* memahami **message** dan eksekusi ***methods***.

**Terkait materi topik ini, silahkan telusuri pada buku referensi 1 dimulai dari halaman 180.**


**FIGURE 6-4** In an example of polymorphism, the message GOOD NIGHT produces different results, depending on which object receives it.

## Pengenalan *Messages*: (lanjutan…)

- *OBJECT* dapat dilihat sebagai ***black–box encapsulation***, dikarenakan suatu message dapat mengirimkan perintah atau methods yang spesifik (**lihat contoh pada–*slide* sebelumnya**).
- Enkapsulasi menyembunyikan detail suatu *OBJECT* dari bagian lain pada suatu program.
- *OBJECT* hanya dapat digunakan melalui ***methods*** aksesnya, yang ditulis dengan hati–hati untuk menjaga agar objek tetap konsisten dan aman.

**Terkait materi topik ini, silahkan telusuri pada buku referensi 1 dimulai dari halaman 180.**

- Enkapsulasi adalah sebuah peroses pemaketan/penyatu data bersama ***methods–methods*** nya, dimana hal ini bermanfaat untuk menyembunyikan rincian–rincian implementasi dari pemakai/*user*.
- Maksud dari enkapsulasi ini adalah untuk menjaga suatu proses program agar tidak dapat diakses secara sembarangan atau di intervensi oleh program lain.
- Konsep enkapsulasi sangat penting dilakukan untuk menjaga kebutuhan program agar dapat diakses sewaktu–waktu sekaligus menjaga keseluruhan program tersebut.

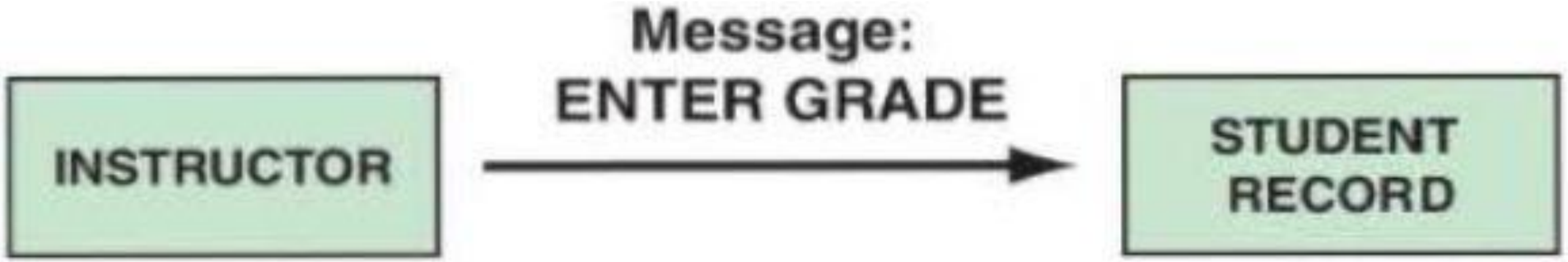



**FIGURE 6-5** In a school information system, an INSTRUCTOR object sends an ENTER GRADE message to an instance of the STUDENT RECORD class.

## Pengenalan *Messages*: (lanjutan…)

- Contoh enkapsulasi dalam program:

```
Belajar.Java

class belajar{
   public String x ="Pintar";
   private String y = "Java";
}

   Pintar.Java
   public class Pintar{
   public static void main(String[]args){
   Coba panggil = new Belajar();
   System.out.println("Panggil X : "+panggil.x);
   System.out.println("Panggil Y : "+panggil.y);

}}
```

**Terkait materi topik ini, silahkan telusuri pada buku referensi 1 dimulai dari halaman 180.**

- Tipe public dan private mempunyai fungsi yang berbeda.
- Fungsi public yang terdapat dalam class Coba pada variable x, memungkinkan nilai dari variable x dapat diakses oleh class Belajar.
- Sedangkan untuk variable y yang menggunakan fungsi private tidak dapat dipanggil didalam class Belajar.

## Pengenalan *Classes*:

- *CLASS* adalah Cetakan yang berisi serangkaian perintah untuk membangun suatu jenis *OBJECT* tertentu.
- *CLASS* juga bisa diartikan sebagai sekelompok *OBJECT* yang memiliki sifat umum.
- Contoh dari *CLASS* pada kehidupan nyata adalah kendaraan, manusia, hewan, dll.
- Contoh membuat *CLASS* dan *OBJECT* pada program:

**Terkait materi topik ini, silahkan telusuri pada buku referensi 1 dimulai dari halaman 180.**

```
public class Kendaraan{

    // Konstruktor Dengan Parameter
    public Kendaraan(String nama){
        System.out.println("Nama Kendaraannya Adalah "+ nama);
    }

    public static void main(String[]args){

        // Perintah untuk membuat objek jenis
        Kendaraan jenis = new Kendaraan("Pesawat Terbang");

    }

}

// Output = Nama Kendaraannya Adalah Pesawat Terbang
```

## Pengenalan *Classes*: (lanjutan…)

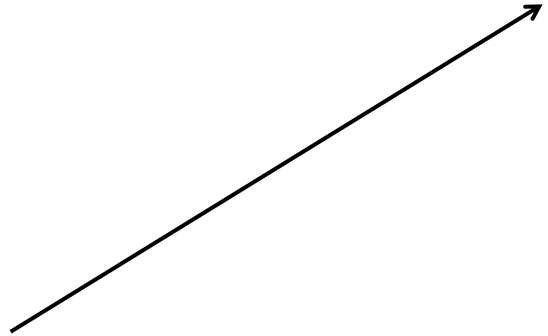
- Contoh kasus sederhana dengan JAVA:

- Pak Kumis adalah pedagang sate, setiap hari dia menjual dan menerima pesanan dari beberapa pembeli. Harga sate yang dijual oleh Pak Kumis adalah Rp 950 per tusuk, lalu ada seorang pembeli memesan sate 115 tusuk, jadi berapakah harga sate 115 tusuk yang harus dibayar oleh pembeli tersebut?

- Berikut program sederhana untuk menampilkan rinciannya:

- Contohnya ada pada *slide* berikutnya.

## Pengenalan *Classes*: (lanjutan…)



- Simpan file yang berisi program JAVA tersebut dengan nama Penjualan.java.
- Jika nama file berbeda, anda perlu mencocokkan nama file dengan nama Class, serta diberi penamaan sama agar program dapat dijalankan.

```
public class Penjualan{

    // Deklarasi Variabel
    private String nama;
    private int harga, jumlah, total;

    // Konstruktor Dengan Parameter
    public Penjualan(String _nama, int _harga, int _jumlah){
        nama  = _nama;
        harga = _harga;
        jumlah= _jumlah;
    }

    // Method Untuk Hitung Harga Total
    public int total(){
        total = harga * jumlah;
        return total;
    }

    // Method Untuk Menampilkan Rincian
    public void tampil(){
        System.out.println("Nama Barang : "+ nama);
        System.out.println("Harga Barang : "+ harga);
        System.out.println("Jumlah yang Dibeli : "+ jumlah);
        System.out.println("_____________________________________");
        System.out.println("Total Harga : "+ total());
    }

    public static void main(String[]args){

        // Perintah untuk membuat objek
        Penjualan resi = new Penjualan("Roti Kumis", 950, 115);
        resi.tampil();
    }
}
```

## Pengenalan *Classes*: (lanjutan...)

**Terkait materi topik ini, silahkan telusuri pada buku referensi 1 dimulai dari halaman 180.**

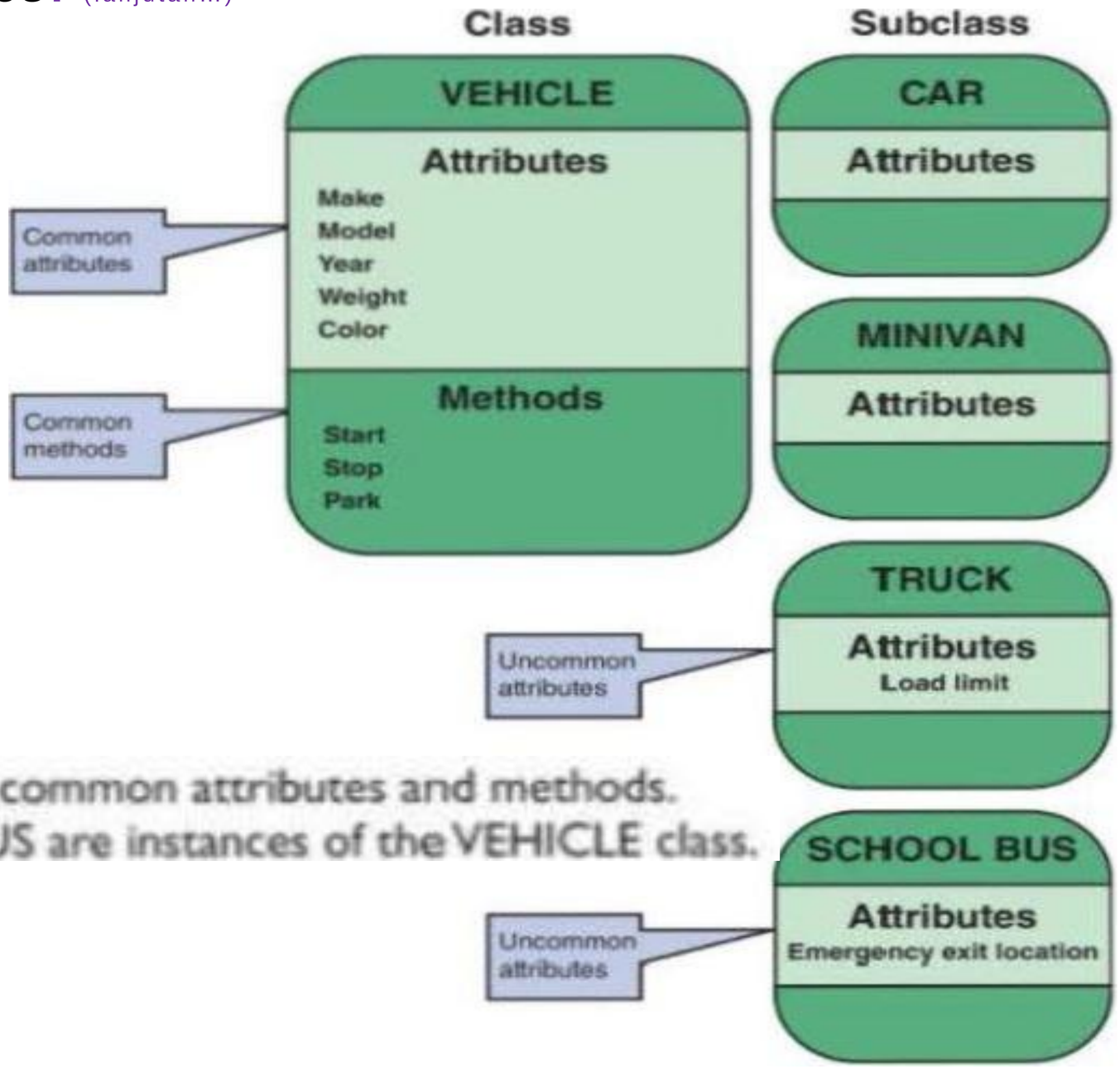



**FIGURE 6-6** The VEHICLE class includes common attributes and methods. CAR, TRUCK, MINIVAN, and SCHOOL BUS are instances of the VEHICLE class.

## ➢ Pengenalan *Classes*: (lanjutan...)

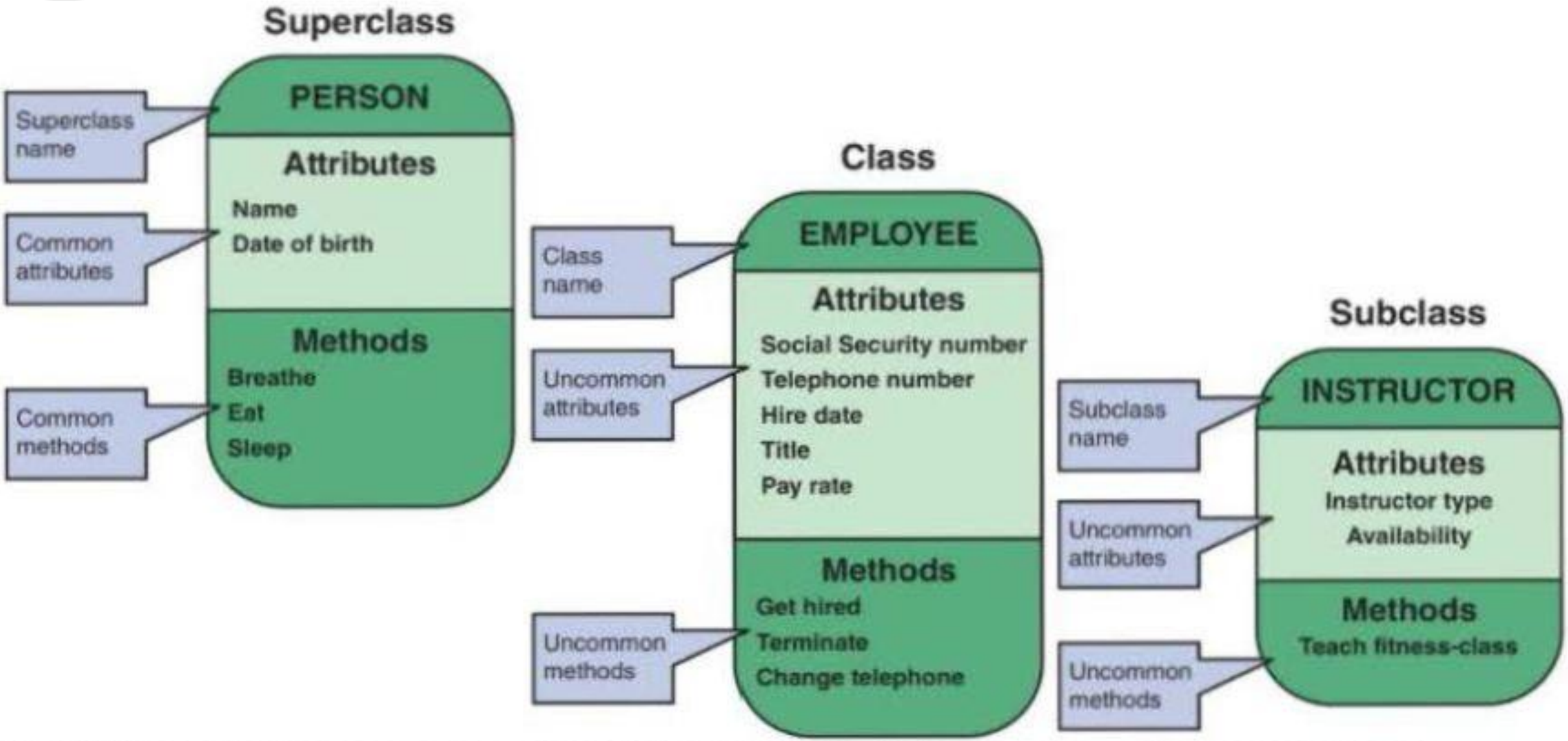



**FIGURE 6-8** At the fitness center, the PERSON subclass includes common attributes and methods. EMPLOYEE is a class within the PERSON superclass. INSTRUCTOR is a subclass within the EMPLOYEE class.

# Relasi/Hubungan Antara *Objects dan Class*

## Pengenalan Relasi/*Relationships*:

- Relasi memungkinkan OBYEK untuk berkomunikasi dan berinteraksi saat mereka melakukan fungsi bisnis serta transaksi yang diperlukan oleh sistem.

- Hubungan menggambarkan apa yang perlu diketahui OBYEK tentang satu sama lain, bagaimana OBYEK menanggapi perubahan OBYEK lain, dan efek dari keanggotaan di CLASSES, SUPERCLASSES, dan SUBCLASSES.

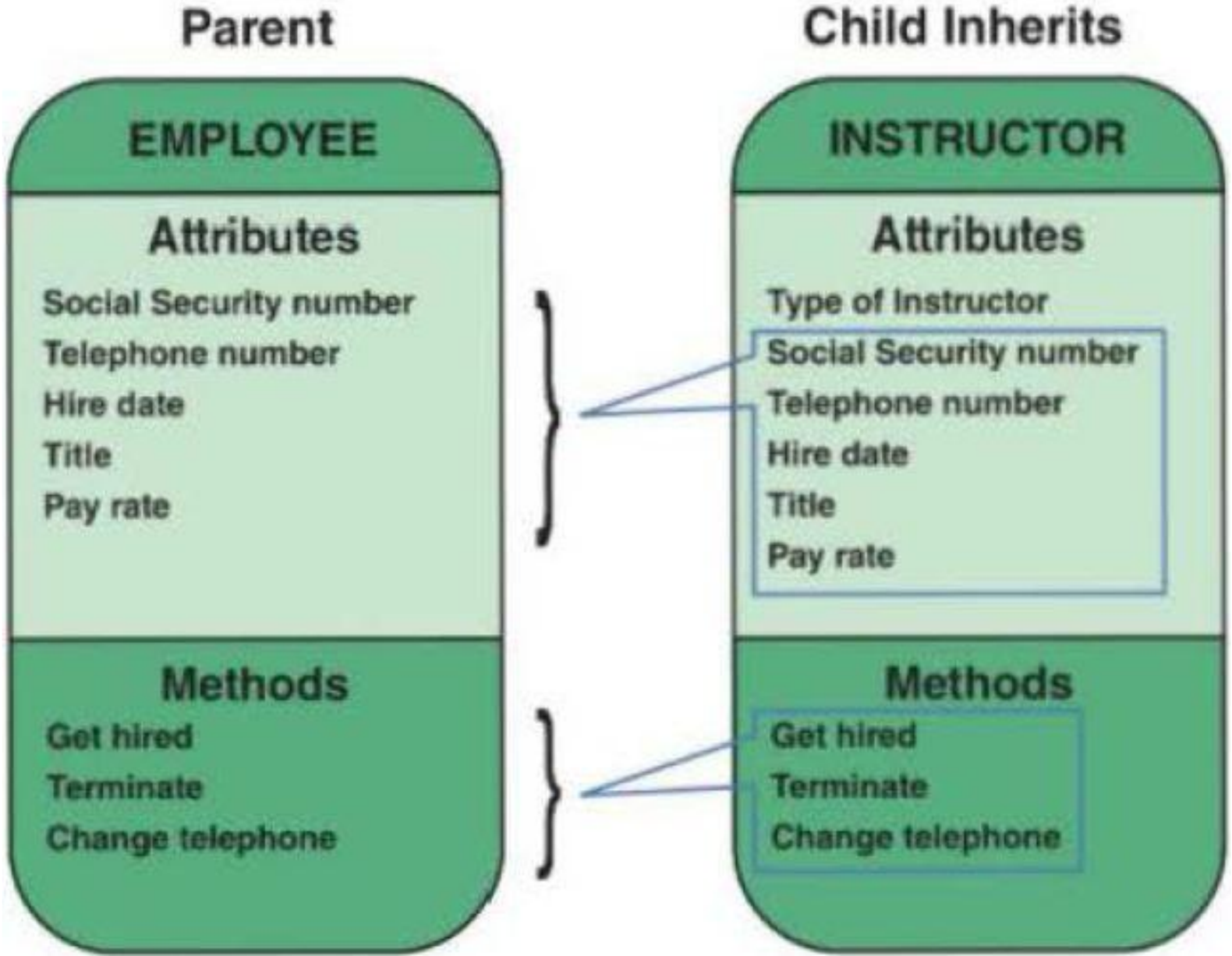
- Beberapa hubungan lebih kuat dari yang lain dimana hubungan terkuat disebut ***inheritance***.

- ***Inheritance*** memungkinkan OBJECT yang disebut ***child*** untuk memperoleh satu atau lebih ***attributes*** dari OBJECT lain (disebut sebagai ***parent***).

# Relasi/Hubungan Antara *Objects dan Class* (lanjutan…)

## ➢ Pengenalan Relasi/*Relationships*: (lanjutan…)

- Contoh relasi ***inheritance*** sebagai berikut:




- Relasi ***Inherintance*** terbentuk antara *OBJECT* INSTRUCTOR dengan EMPLOYEE.
- *Child OBJECT* (INSTRUCTOR) mewarisi karakteristik ***atrributes*** *Parent OBJECT* (EMPLOYEE).
- Namun *Child OBJECT* dapat memilki ***attributes*** tambahan miliknya sendiri.

# Relasi/Hubungan Antara *Objects dan Class* (lanjutan...)

## ➢ Pengenalan Relasi/*Relationships*: (lanjutan...)

- Contoh *object relatinship diagram* sebagai berikut:

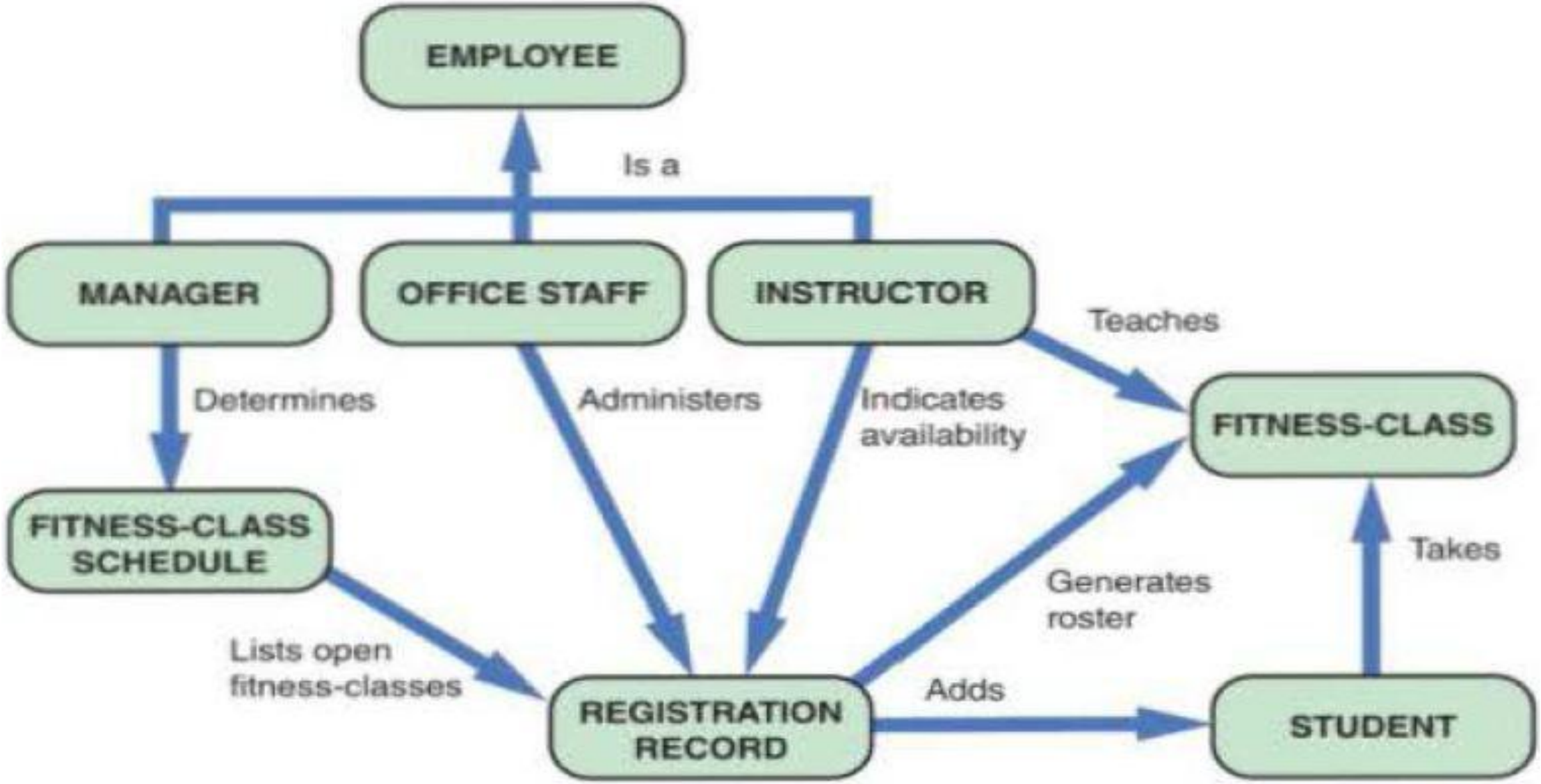



**FIGURE 6-10** An object relationship diagram for a fitness center.

# REFERENSI


- Tilley, Scott, System Analysis and Design, CENGAGE, USA, 2020.
- Dennis, Alan, System Analysis and Design: An Object Oriented Approach with UML, Wiley, USA, 2015.
- Satzinger, Jackson, Burd, System Analysis and Design in A Changing World, CENGANE, USA, 2012.
- Langer, Arthur, Analysis and Design of Information Systems, Springer, USA, 2008.
- Bentlet, Whitten, System Analysis and Design Methods, McGraw–Hill Irwin, USA, 2007.
- Wasson, Charles, System Analysis, Design, and Development, Concepts, Principles, and Practices, Wiley–Interscience, Canada, 2006.
- researchgate.com
- salaryexplorer.com.
- https://www.gotutorid.com/java/class-dan-object/


# TERIMA KASIH