## Soal Tugas & Evaluasi

## Apa itu acces modifier dan buatlah ilustrasi Batasan akses dari setiap acces modifier

## Jawaban

Acces modifier dalam oop akan menentukan apakah class lain dapat menggunakan file datau meminta izin untuk menngakses data dari suatu class

* Public
* Protected
* Private
* default

## Soal Tugas & Evaluasi

1. Jelaskan perbedaan antara association, aggregation, dan composition dalam hubungan antar class. Berikan contoh kasus setiap jenis relasi

## Jawaban

Association adalah hubungan antara dua kelas di mana satu kelas menggunakan atau berinteraksi dengan kelas lain.

Contoh: Misalkan kita memiliki kelas Pengguna dan kelas Perangkat. Setiap Pengguna dapat menggunakan satu atau lebih Perangkat, dan setiap Perangkat dapat digunakan oleh satu atau lebih Pengguna.

Aggregation adalah jenis khusus dari asosiasi yang menunjukkan hubungan "has-a" di mana satu kelas adalah kumpulan dari beberapa kelas lain

Contoh:

Misalkan kita memiliki kelas Sekolah dan kelas Siswa. Sekolah memiliki banyak Siswa, tetapi Siswa dapat ada tanpa Sekolah.

Composition adalah jenis agregasi yang lebih kuat di mana satu kelas memiliki objek kelas lain dan objek tersebut tidak dapat ada tanpa kelas yang memilikinya

Contoh: Misalkan kita memiliki kelas Rumah dan kelas Kamar. Setiap Rumah memiliki beberapa Kamar, dan Kamar tidak dapat ada tanpa Rumah

## Soal Tugas & Evaluasi

1. Buat sebuah class akunBank yang berisi atribut data-data yang ada di akun seiorang nasabah bank denngan menerapkan encapsulation, kemudian analisis modifier apa saja yang dipakai pada setiap atribut dann tentukan method getter setter nya

## Source Code

*class* AkunBank{  
 *private* String namaNasabah;  
 *private* String Norek;  
 *private double* saldo;  
  
 *public* AkunBank(String namaNasabah, String Norek, *double* saldo){  
 *this*.namaNasabah = namaNasabah;  
 *this*.Norek = Norek;  
 *this*.saldo = saldo;  
 }  
 *public* String getNamaNasabah(){  
 *return* namaNasabah;  
 }  
 *public void* setNamaNasabah(String namaNasabah){  
 *this*.namaNasabah = namaNasabah;  
 }  
 *public* String getNorek(){  
 *return* Norek;  
 }  
 *public void* setNorek(String Norek){  
 *this*.Norek = Norek;  
 }  
 *public double* getSaldo(){  
 *return* saldo;  
 }  
 *public void* setSaldo(*double* saldo){  
 *this*.saldo = saldo;  
 }  
}  
  
*public class* bab5no3 {  
 *public static void* main(String[] args) {  
 AkunBank bank = *new* AkunBank("Nama","4235923",100000);  
 bank.setNamaNasabah("Adam");  
 bank.setNorek("092324");  
 bank.setSaldo(2350000);  
  
 System.out.println("Nama Nasabah : " + bank.getNamaNasabah());  
 System.out.println("Norek : " + bank.getNorek());  
 System.out.println("Saldo : " + bank.getSaldo());  
 }  
}

**Penjelasan**

Membuat akunBank buat data nasabah dan membuat kelas akunBank dengan atribut private string namaNasabah, private string norek, , private double saldo, dan private ini untuk menunjukkan encapsulation nya yang membatasi akses langsung. Lalu membuat konstraknya dan setter getter nya, lalu membaut objek akunbank untuk menginisialisasi dan memanggil set untuk menginisialisasi. Dan menampilkan nama, norek, dan saldo dengan memanggil get dengan nama sesuai nya

**Output**

|  |
| --- |
|  |

## Soal Tugas & Evaluasi

1. Buat dua class mahasiswa dan matakukliah, yang menerapkan asosiasi tidak berarah. Implementasikan method yang memungkinkan seorang mahasiswa untuk mendaftar ke mata kuliah dan mata kuliah menerima pendaftaran mahasiswa

## Source Code

*import* java.util.ArrayList;  
*import* java.util.List;  
  
*class* Mahasiswa {  
 *private* String nama;  
 *private* List<MataKuliah> mataKuliahList;  
  
 *public* Mahasiswa(String nama) {  
 *this*.nama = nama;  
 *this*.mataKuliahList = *new* ArrayList<>();  
 }  
  
 *public* String getNama() {  
 *return* nama;  
 }  
  
 *public void* daftarMataKuliah(MataKuliah mataKuliah) {  
 *if* (!mataKuliahList.contains(mataKuliah)) {  
 mataKuliahList.add(mataKuliah);  
 mataKuliah.tambahMahasiswa(*this*);  
 }  
 }  
  
 *public* List<MataKuliah> getMataKuliahList() {  
 *return* mataKuliahList;  
 }  
}  
  
*class* MataKuliah {  
 *private* String namaMataKuliah;  
 *private* List<Mahasiswa> mahasiswaList;  
  
 *public* MataKuliah(String namaMataKuliah) {  
 *this*.namaMataKuliah = namaMataKuliah;  
 *this*.mahasiswaList = *new* ArrayList<>();  
 }  
  
 *public* String getNamaMataKuliah() {  
 *return* namaMataKuliah;  
 }  
  
 *public void* tambahMahasiswa(Mahasiswa mahasiswa) {  
 *if* (!mahasiswaList.contains(mahasiswa)) {  
 mahasiswaList.add(mahasiswa);  
 }  
 }  
 *public* List<Mahasiswa> getMahasiswaList() {  
 *return* mahasiswaList;  
 }  
}  
*public class* Bab5No4 {  
 *public static void* main(String[] args) {  
 Mahasiswa mahasiswa1 = *new* Mahasiswa("Adam");  
 Mahasiswa mahasiswa2 = *new* Mahasiswa("Luthfi");  
  
 MataKuliah mataKuliah1 = *new* MataKuliah("Jaringan Komputer");  
 MataKuliah mataKuliah2 = *new* MataKuliah("Basis Data");  
  
 mahasiswa1.daftarMataKuliah(mataKuliah1);  
 mahasiswa2.daftarMataKuliah(mataKuliah1);  
 mahasiswa1.daftarMataKuliah(mataKuliah2);  
 mahasiswa2.daftarMataKuliah(mataKuliah1);  
  
 System.out.println("Mata Kuliah yang diambil oleh " + mahasiswa1.getNama() + ":");  
 *for* (MataKuliah mk : mahasiswa1.getMataKuliahList()) {  
 System.out.println(mk.getNamaMataKuliah());  
 }  
  
 System.out.println("Mata Kuliah yang diambil oleh " + mahasiswa2.getNama() + ":");  
 *for* (MataKuliah mk : mahasiswa2.getMataKuliahList()) {  
 System.out.println(mk.getNamaMataKuliah());  
 }  
  
 System.out.println("Mahasiswa yang terdaftar di " + mataKuliah1.getNamaMataKuliah() + ":");  
 *for* (Mahasiswa mhs : mataKuliah1.getMahasiswaList()) {  
 System.out.println(mhs.getNama());  
 }  
 }  
}

**Penjelasan**

Program ini menunjukkan relasi asosiasi antara mahasiswa dan mata kuliah, dimana mahasiswa bisa mengambil banyak mata kuliah dan satu mata kuliah dapat diikuti oleh banyak mahasiswa . Metode daftarMataKuliah pada Mahasiswa menambahkan mata kuliah ke daftar mahasiswa dan juga menambahkan mahasiswa tersebut ke daftar di MataKuliah. Hasilnya menampilkan mata kuliah yang diambil oleh setiap mahasiswa serta daftar mahasiswa dalam mata kuliah tertentu.

**Output**

|  |
| --- |
|  |

## Soal Tugas & Evaluasi

1. Buatlah class Mobil dan class Mesin yang menerapkan relasi Aggregation. Dalam program tersebut, Mobil memiliki

## Source Code

*class* Mesin {  
 *private* String jenisMesin;  
  
 *public* Mesin(String jenisMesin) {  
 *this*.jenisMesin = jenisMesin;  
 }  
  
 *public* String getJenisMesin() {  
 *return* jenisMesin;  
 }  
}  
  
*class* Mobil {  
 *private* String merek;  
 *private* Mesin mesin;  
  
 *public* Mobil(String merek) {  
 *this*.merek = merek;  
 }  
  
 *public void* pasangMesin(Mesin mesin) {  
 *this*.mesin = mesin;  
 System.out.println("Mesin " + mesin.getJenisMesin() + " telah dipasang pada mobil " + merek + ".");  
 }  
  
 *public void* nyalakanMesin() {  
 *if* (mesin != *null*) {  
 System.out.println("Mesin " + mesin.getJenisMesin() + " pada mobil " + merek + " dinyalakan.");  
 } *else* {  
 System.out.println("Mesin belum dipasang.");  
 }  
 }  
}  
  
*public class* bab5no5 {  
 *public static void* main(String[] args) {  
 Mesin mesin1 = *new* Mesin("Mesin Bensin");  
 Mesin mesin2 = *new* Mesin("Mesin Diesel");  
  
 Mobil mobil1 = *new* Mobil("hyundai");  
 Mobil mobil2 = *new* Mobil("BMW");  
  
 mobil1.pasangMesin(mesin1);  
 mobil1.nyalakanMesin();  
  
 System.out.println("\n");  
 mobil2.nyalakanMesin();  
 mobil2.pasangMesin(mesin2);  
 mobil2.nyalakanMesin();  
 }  
}

**Penjelasan**

Program di atas menggambarkan relasi **agregasi** antara kelas Mobil dan Mesin, di mana mesin dapat dipasang atau dilepas dari mobil tanpa keduanya bergantung penuh satu sama lain. Kelas Mesin memiliki atribut jenisMesin, yang menentukan tipe mesin, sementara kelas Mobil memiliki atribut merek dan referensi ke objek Mesin. Metode pasangMesin pada Mobil menghubungkan mesin dengan mobil tertentu, dan menampilkan pesan bahwa mesin berhasil dipasang. Metode nyalakanMesin pada Mobil mengecek apakah mesin telah dipasang; jika ya, mesin dinyalakan, tetapi jika belum, pesan peringatan akan muncul. Dalam main, mesin berbeda dipasang dan dinyalakan pada objek Mobil yang berbeda, memperlihatkan fleksibilitas agregasi antara Mobil dan Mesin.

**Output**

|  |
| --- |
|  |

## Soal Tugas & Evaluasi

6. Vincent diminta dosennya untuk membuat sebuah sistrm manaejeme perpustakaan sederhana. System ini harus mampu mengelola data buku, anggota perpustakaan , dan transaksi peminjaman buku, tidak hany satu Vincent diwajibkan menggunakan berbagai jenis data collection seperti arraylist, hashmap, dan queue. Bualah program untuk vincent

## Source Code

*import* java.util.ArrayList;  
*import* java.util.HashMap;  
*import* java.util.List;  
*import* java.util.Map;  
  
*class* Buku {  
 String judul;  
 String penulis;  
  
 *public* Buku(String judul, String penulis) {  
 *this*.judul = judul;  
 *this*.penulis = penulis;  
 }  
}  
  
*class* Anggota {  
 String nama;  
 String idAnggota;  
  
 *public* Anggota(String nama, String idAnggota) {  
 *this*.nama = nama;  
 *this*.idAnggota = idAnggota;  
 }  
}  
  
*class* Perpustakaan {  
 *private* List<Buku> daftarBuku;  
 *private* Map<String, Anggota> daftarAnggota;  
 *private* List<String> antrianPeminjaman;  
  
 *public* Perpustakaan() {  
 daftarBuku = *new* ArrayList<>();  
 daftarAnggota = *new* HashMap<>();  
 antrianPeminjaman = *new* ArrayList<>();  
 }  
  
 *public void* tambahBuku(Buku buku) {  
 daftarBuku.add(buku);  
 System.out.println("Buku " + buku.judul + " telah ditambahkan.");  
 }  
  
 *public void* tambahAnggota(Anggota anggota) {  
 daftarAnggota.put(anggota.idAnggota, anggota);  
 System.out.println("Anggota " + anggota.nama + " telah ditambahkan.");  
 }  
  
 *public void* pinjamBuku(String idAnggota, String judulBuku) {  
 *if* (!daftarAnggota.containsKey(idAnggota)) {  
 System.out.println("Anggota tidak ditemukan.");  
 *return*;  
 }  
  
 *for* (Buku buku : daftarBuku) {  
 *if* (buku.judul.equalsIgnoreCase(judulBuku)) {  
 antrianPeminjaman.add(idAnggota);  
 System.out.println("Anggota " + idAnggota + " telah meminjam buku " + judulBuku + ".");  
 *return*;  
 }  
 }  
 System.out.println("Buku tidak ditemukan.");  
 }  
  
  
 *public void* tampilkanDaftarBuku() {  
 System.out.println("Daftar Buku:");  
 *for* (Buku buku : daftarBuku) {  
 System.out.println("- " + buku.judul + " oleh " + buku.penulis);  
 }  
 }  
}  
  
*public class* bab4no6 {  
 *public static void* main(String[] args) {  
 Perpustakaan perpustakaan = *new* Perpustakaan();  
  
 perpustakaan.tambahBuku(*new* Buku("War Machine", "Sulistiawan"));  
 perpustakaan.tambahBuku(*new* Buku("Magic Of Snake", "Tolkien"));  
  
 perpustakaan.tambahAnggota(*new* Anggota("Adam", "1"));  
 perpustakaan.tambahAnggota(*new* Anggota("Nana", "2"));  
  
 perpustakaan.pinjamBuku("1", "War Machine");  
 perpustakaan.pinjamBuku("2", "Magic of snake");  
  
  
 perpustakaan.tampilkanDaftarBuku();  
 }  
}

**Penjelasan**

Membuat class buku dengan attribute judul dan penulis lalu membuat constructor nya

Lalu membuat class anggota nama dan id anggota beserta constructornya

Lalu membuat class peprpustakaan dan menginisialisasi dengan lust buku untuk daftar buku,map untuk anggota list string untuk antrian peminjaman dan membuat method tambah buku untuk menambahkan buku,lalu membuat tambah anggota unttuk menambahkan anggota dan membuat pijambuku untuk meminjam buku lalu membuat tampilkan daftarbuku untuk menampilkan daftar bukunya,dan membuat objek perpustakaan untuk menginisialisasi untuk di tampilkan

**Output**

|  |
| --- |
|  |