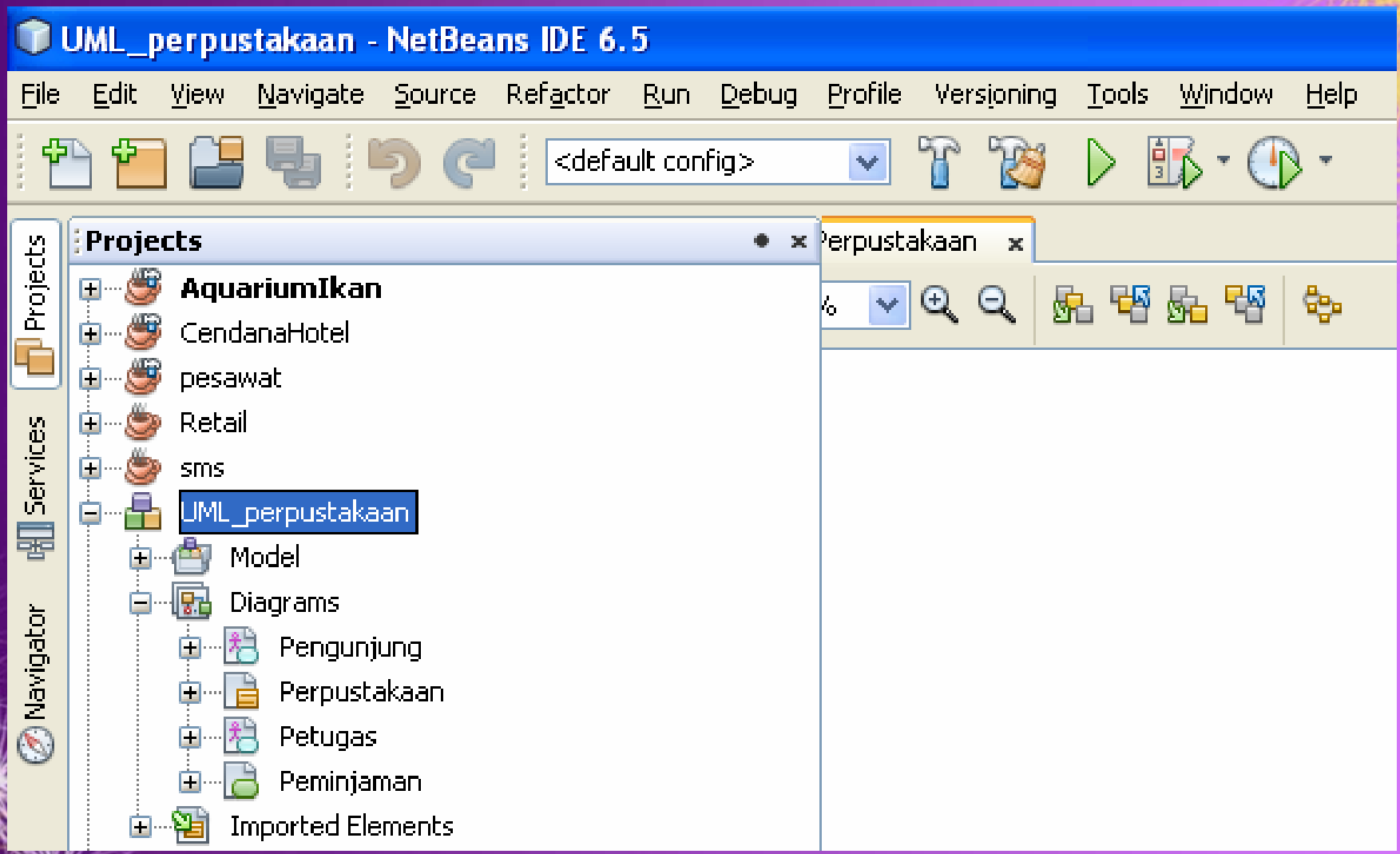
C. Membuat Class Diagram

***Class diagram* mend*e*skripsikan jenis – jenis obyek dalam sistem dan berbagai macam hubungan statis yang terjadi*1*. *Class diagram* juga menunjukkan property dan operasi sebuah *Class* dan batasan yang terdapat dalam hubungan dengan obyek. *Class diagram* merupakan alat terbaik dalam perancangan perangkat lunak. *Class diagram* membantu pengembang m*end*apatkan struktur sistem dan menghasilkan rancangan sistem yang baik**

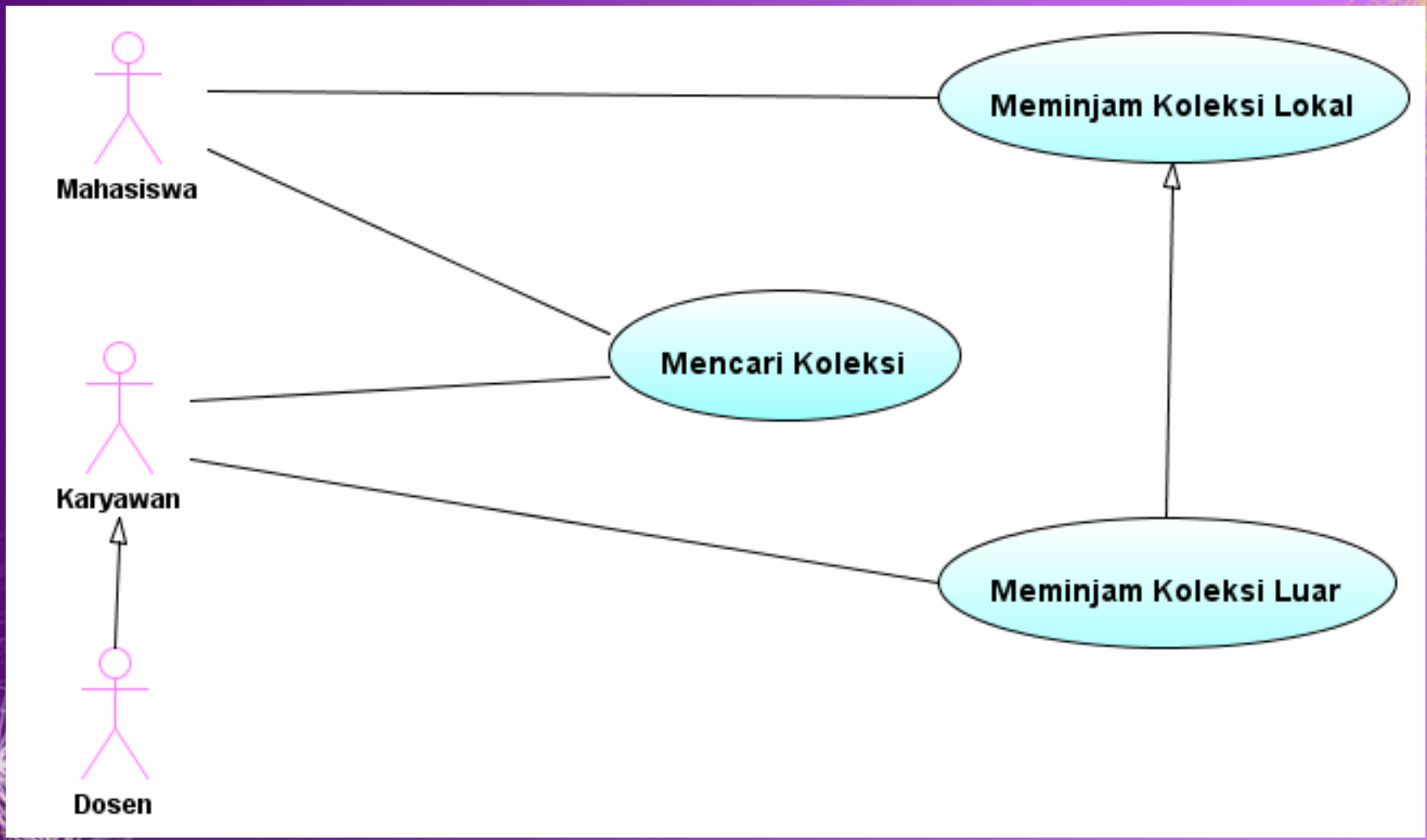
Sebelum kita lanjutkan membuat *Class Diagram*. Pastikan terlebih dahulu untuk UML Project Perpustakaan yang pernah kita buat.

Terdapat 2 Use Case Diagram dan 1 Activity Diagram, yaitu :
Use Case Diagram Pengunjung
Use Case Diagram Petugas
Activity Diagram Peminjaman

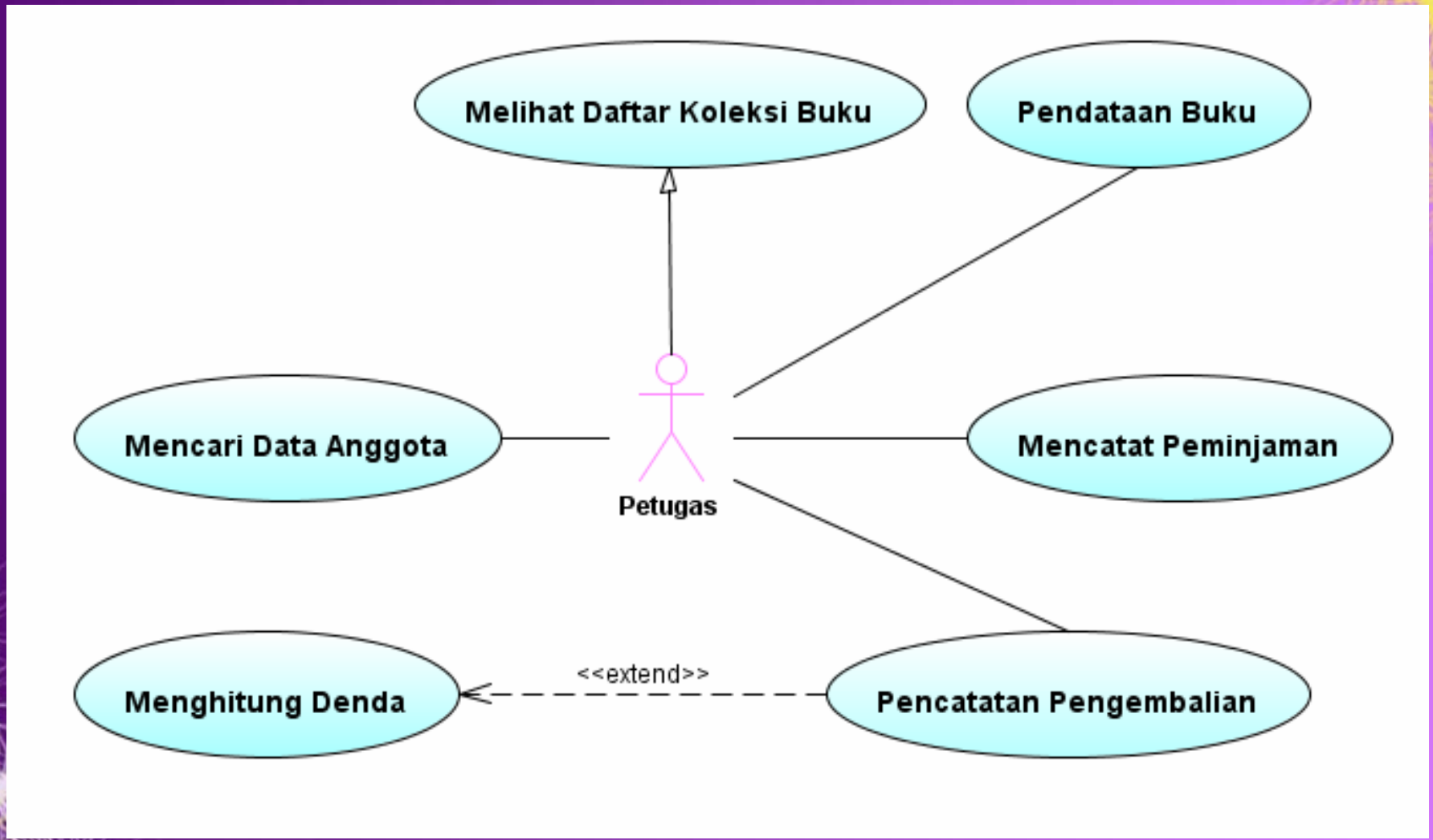
Bentuk Project UML_Perpustakaan



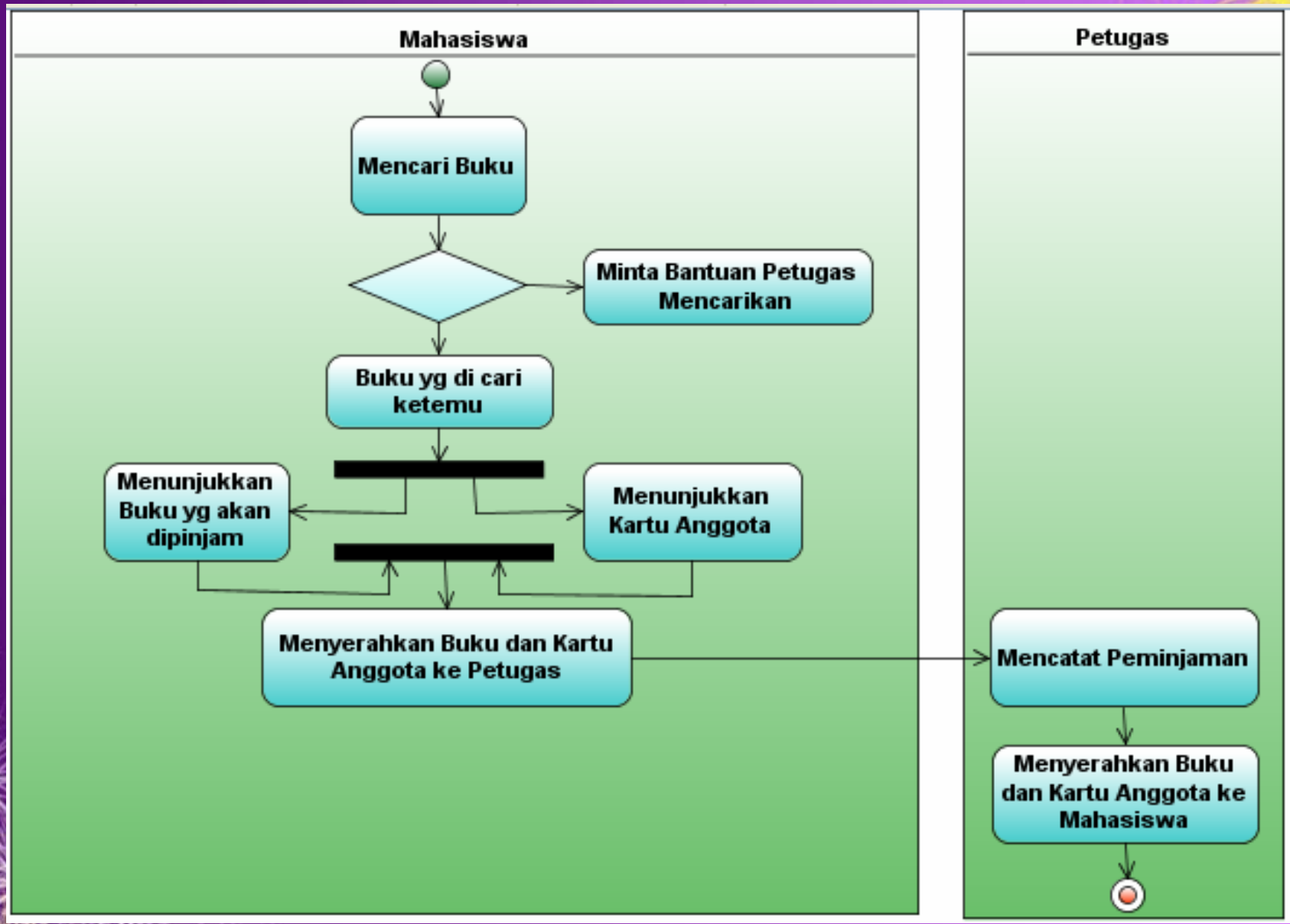
Use Case Diagram Pengunjung



Use Case Diagram Petugas

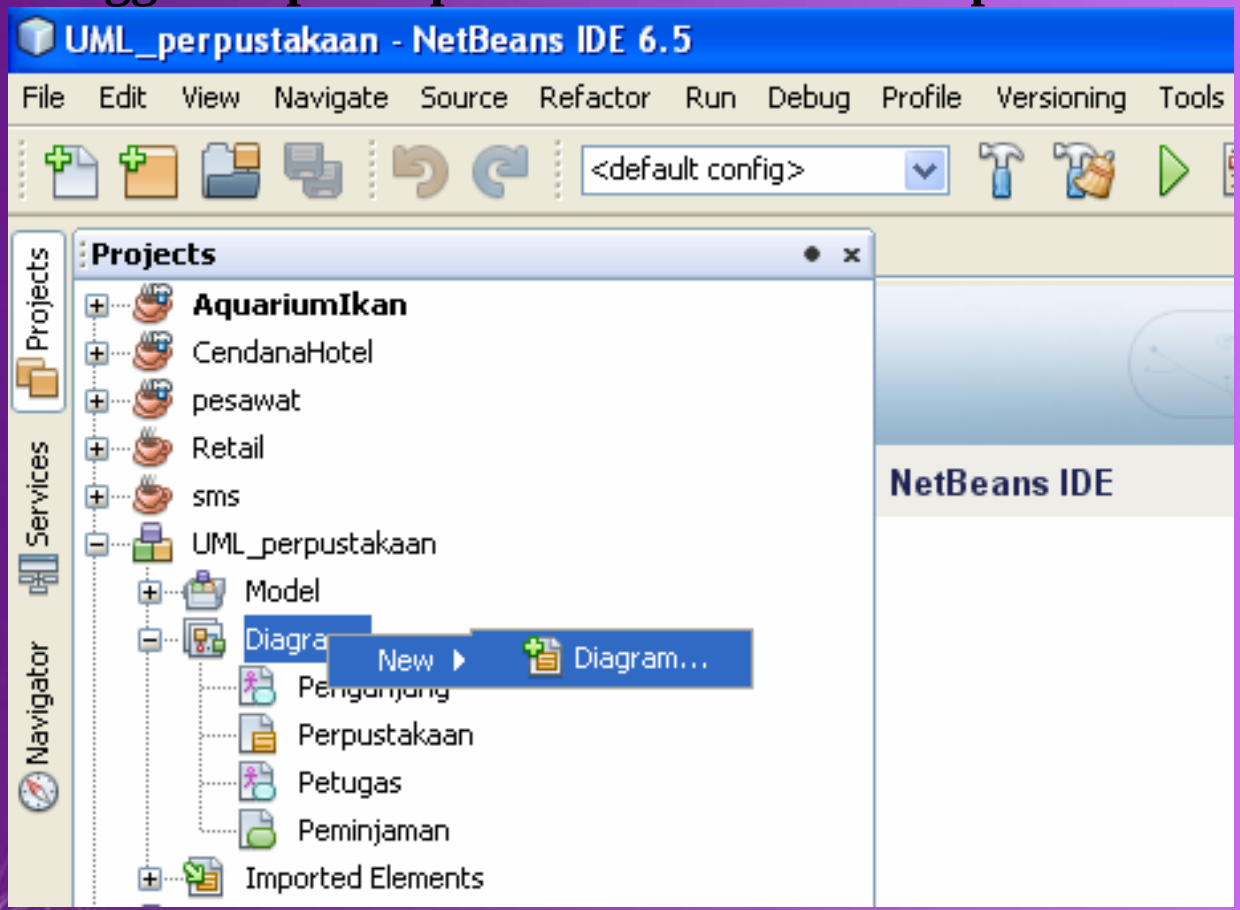


Activity Diagram Peminjaman

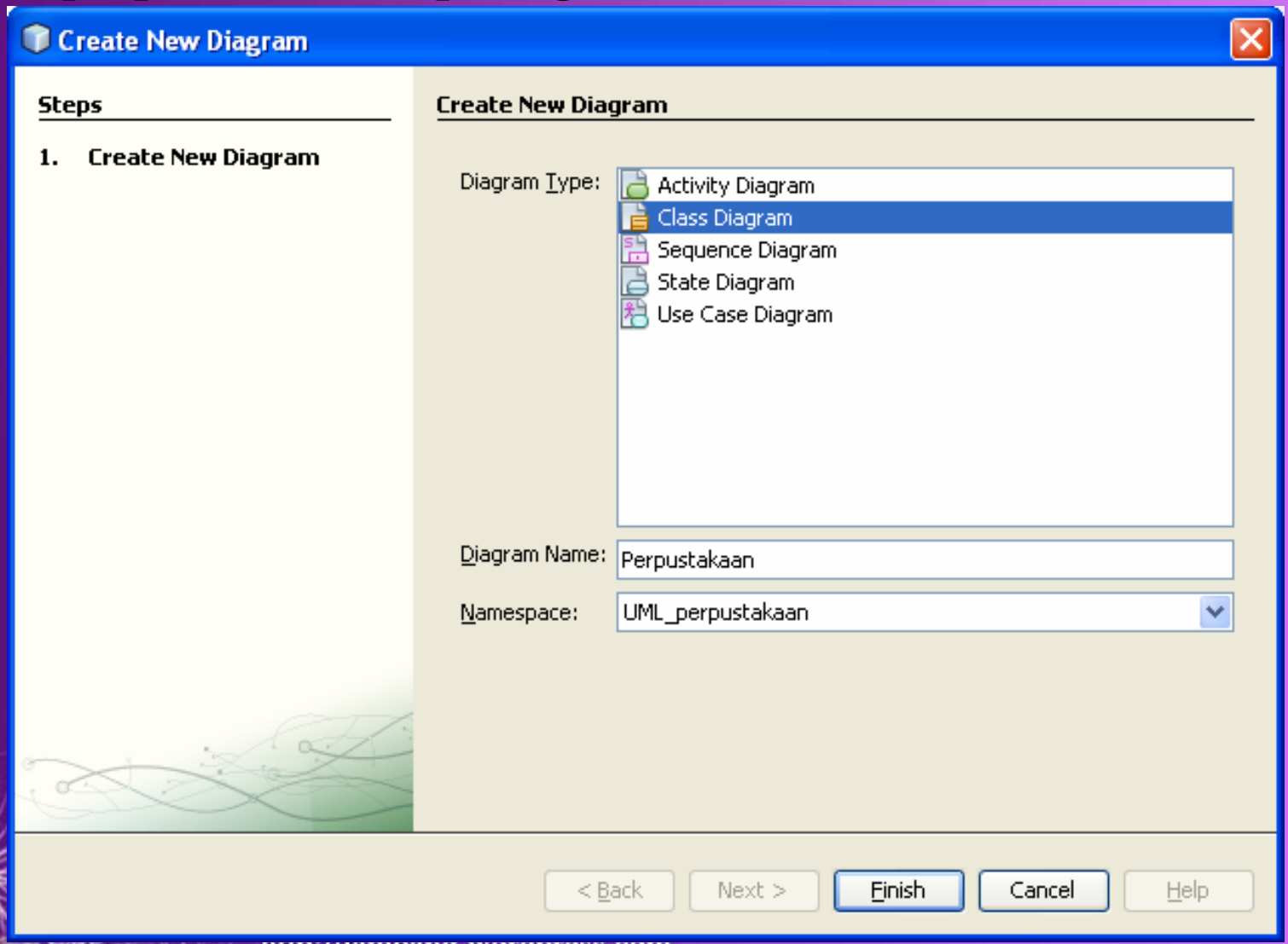


Buka kembali Project UML_perpustakaan dari Menu File – Open Project. Kemudian di dropdown Look in pilih folder/directory dimana project disimpan. Pilih Project "UML_perpustakaan" dan tekan tombol *Open Project*

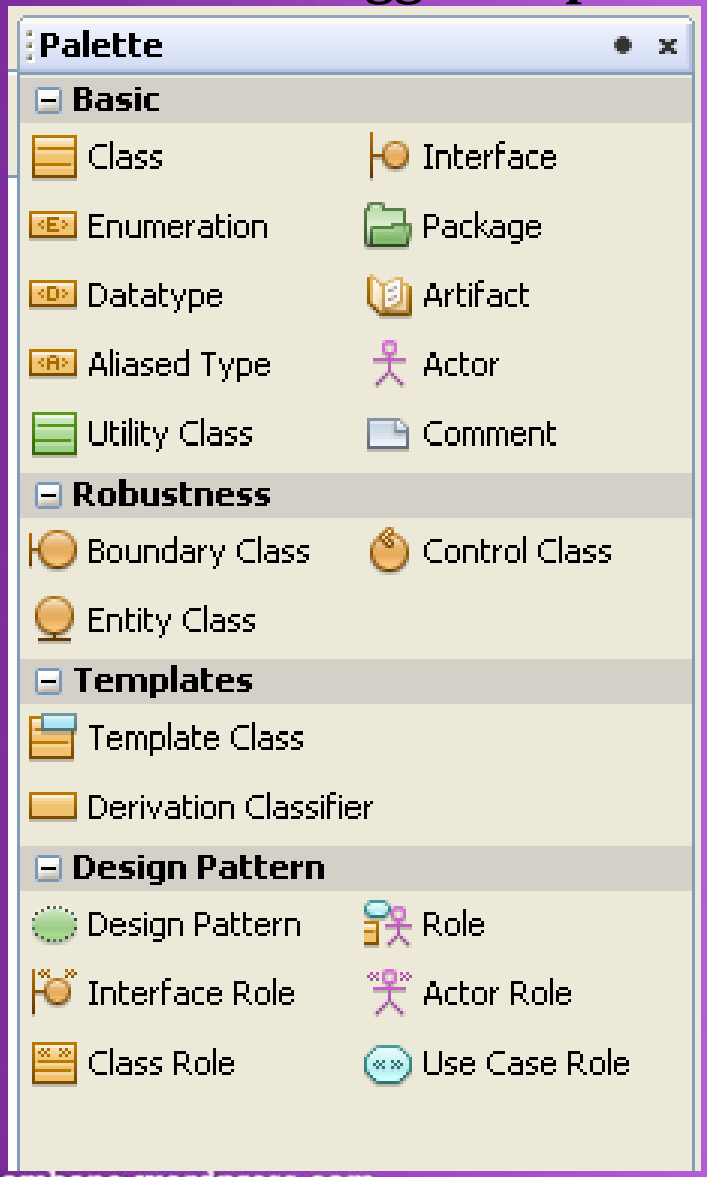
Sorot/Pilih Project UML_perpustakaan – Diagram. Klik kanan pada Diagram sehingga tampak seperti berikut. Dan Klik pada New - Diagram



Pada Window Create New Diagram - Diagram Type, Pilih Class Diagram, Diagram Name ketikkan "Perpustakaan", Namespace pilih "UML_perpustakaan" seperti gb berikut. Tekan Tombol Finish

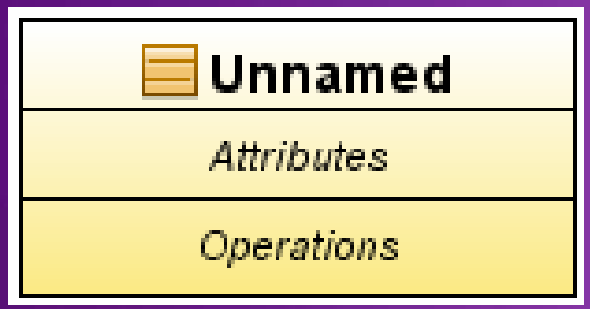


Munculkan Window Palette dari menu window – pallete atau dengan menekan tombol Ctrl+Shift+8 sehingga tampak seperti gb berikut :



Class

Adalah sebuah kategori yang akan membungkus informasi dan perilaku2 atau dengan kata lain kelas adalah rancangan dari obyek. Dimana obyek yang diciptakan dari suatu kelas akan memiliki semua yang dimiliki oleh kelasnya. Secara umum kelas dalam UML dinotasikan sebagai berikut



Nama Class

Daftar Atribut

Daftar Operasi

Nama

Nama kelas haruslah unik, karena ini adalah identitas yang dimiliki oleh setiap *Class*.

Atribut

Atribut disini menunjukkan informasi yang dimiliki oleh suatu class, bisa juga disebut informasi yang berhubungan dengan class.

Operasi

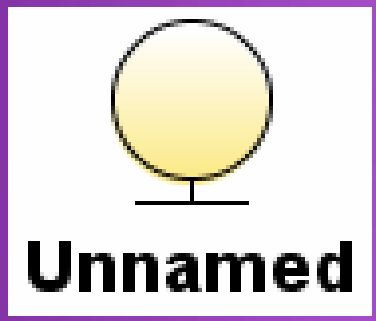
Operasi digunakan untuk menunjukkan apa yang suatu class bisa lakukan atau apa yang bisa dilakukan pada suatu class

Boundary Class

Boundary atau Kelas pembatas merupakan class yang menyalurkan interaksi antara sistem dengan dunia sekitarnya. Seperti form, laporan, obyek – obyek pada form. Setiap kelas pembatas biasanya akan mewakili interaksi antara seorang actor dengan use case. Boundary Class atau Kelas pembatas digambarkan sebagai berikut

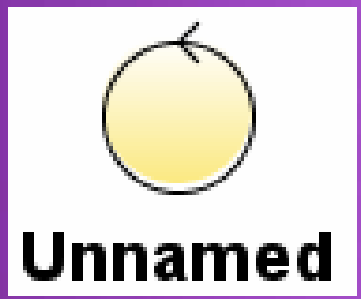
Entity Class atau Kelas entitas

Kelas ini biasanya digunakan untuk menangani informasi yang mungkin akan selalu disimpan dalam proses bisnis. Cara melakukan identifikasi kelas entitas adalah dengan memperhatikan kata b*end*a. Seperti kalimat “petugas mencatat peminjaman buku”, buku bisa menjadi *Class* entitas. Kelas entitas juga dapat digunakan untuk mewakili table – table yang terdapat dalam database. Digambarkan sebagai berikut



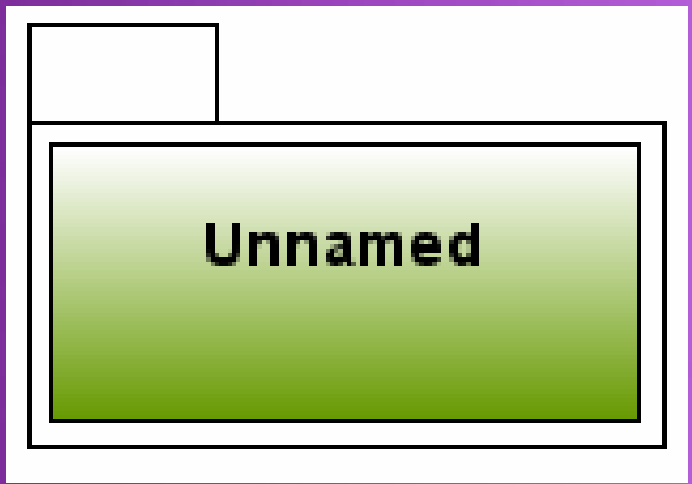
Control Class atau Kelas control

Kelas ini bersifat opsional, apabila kelas ini digunakan maka satu kelas *control* untuk satu *use case* yang digunakan mengatur kejadian dalam use case tersebut. Contohnya kelas transaksi yang bertanggung jawab dalam transaksi baik peminjaman maupun pengembalian buku dalam suatu perpustakaan. Notasi Control Class seperti berikut ;



Package atau Paket

Paket, merupakan sebuah notasi yang sangat berguna. Biasanya digunakan untuk mengelompokan kelas – kelas yang memiliki kesamaan. Selain itu juga dapat digunakan membedakan antara 2 kelas atau lebih yang memiliki nama sama namun memiliki fungsionalitas yang berbeda. Dinotasikan sebagai berikut :



Membuat Class Diagram Berdasarkan Use Case Diagram

Cara untuk menentukan sebuah class adalah dengan melihat mana saja elemen dari Use Case Diagram yang merupakan Kata Benda, menunjukkan Orang dan yang menunjukkan Proses.

Dari Use Case Diagram Pengunjung :

Mahasiswa Karyawan Dosen	Masing –masing 1 class, sehingga menjadi 3 class baru
Koleksi Koleksi Lokal Koleksi Luar	Masing –masing 1 class, sehingga menjadi 3 class baru

Dari Use Case Diagram Petugas :

Petugas Denda	Masing –masing 1 class, sehingga menjadi 2 class baru
Peminjaman Pengembalian	Dijadikan 1 class dinamakan dengan Transaksi, sehingga hanya menjadi 1 class baru

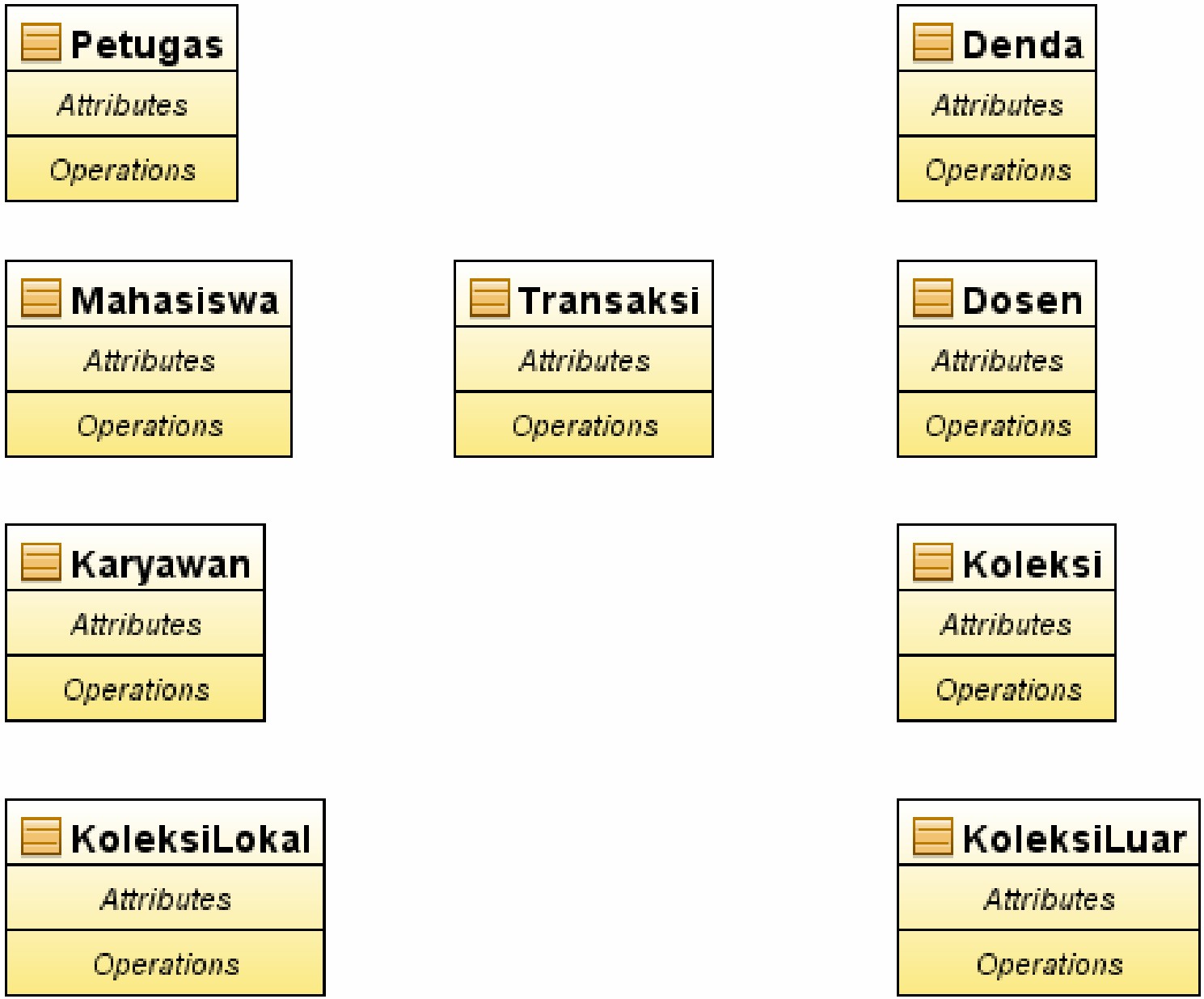
Sehingga Dari kedua Use Case Diagram diperoleh 9 class

- Mahasiswa
- Karyawan
- Dosen
- Koleksi
- Koleksi Lokal
- Koleksi Luar
- Petugas
- Denda
- Transaksi

Aturan Pembuatan Class Diagram dari ke 9 class diatas :

- Untuk nama kelas yang hanya 1 suku kata, diawali dengan huruf besar. Contoh : Mahasiswa, Dosen, Karyawan, Petugas, Denda, Transaksi.
- Untuk nama kelas yang lebih dari 1 suku kata, setiap huruf pertama suku kata diawali dengan huruf besar, dan jangan menggunakan tanda spasi, underscode (_), dash (-), pipeline (|) atau tanda baca yang lain Contoh : KoleksiLuar, KoleksiLokal, KaryawanDanDosen

Dari ke 9 Class tersebut buat dan susun seperti tampak pada gambar berikut


Petugas
Attributes
Operations
Denda
Attributes
Operations
Mahasiswa
Attributes
Operations
Transaksi
Attributes
Operations
Dosen
Attributes
Operations
Karyawan
Attributes
Operations
Koleksi
Attributes
Operations
KoleksiLokal
Attributes
Operations
KoleksiLuar
Attributes
Operations

Relasi Pada Class Diagram

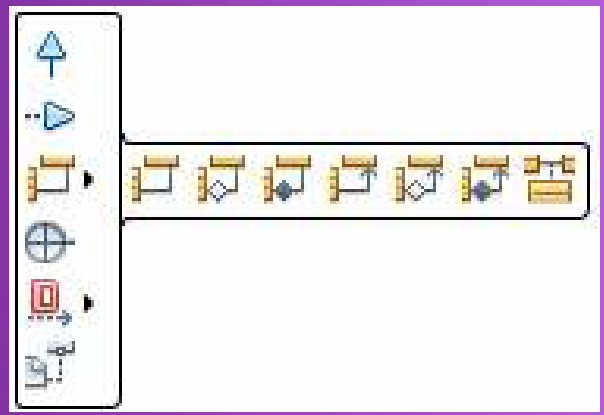
Multiplicity

Pada relasi terdapat suatu penanda yang disebut multiplicity. Multiplicity ini akan mengindikasikan berapa banyak obyek dari suatu kelas terelasi ke obyek lain. Notasi UML untuk multiplicity ini adalah sebagai berikut:

Multiplicity	Arti
*	Banyak
0	Nol
1	Satu, bisa ditulis bisa tidak
0..*	Antara Nol sampai banyak
1..*	Antara Satu sampai banyak
0..1	Nol atau Satu
1..1	Tepat Satu

Relasi

Selain kelas – kelas yang nantinya akan mengisi sebuah kelas diagram, tentunya ada hubungan antara satu kelas dengan kelas lainnya yang disebut relasi. Relasi digunakan oleh suatu kelas untuk berkomunikasi dengan kelas lainnya.



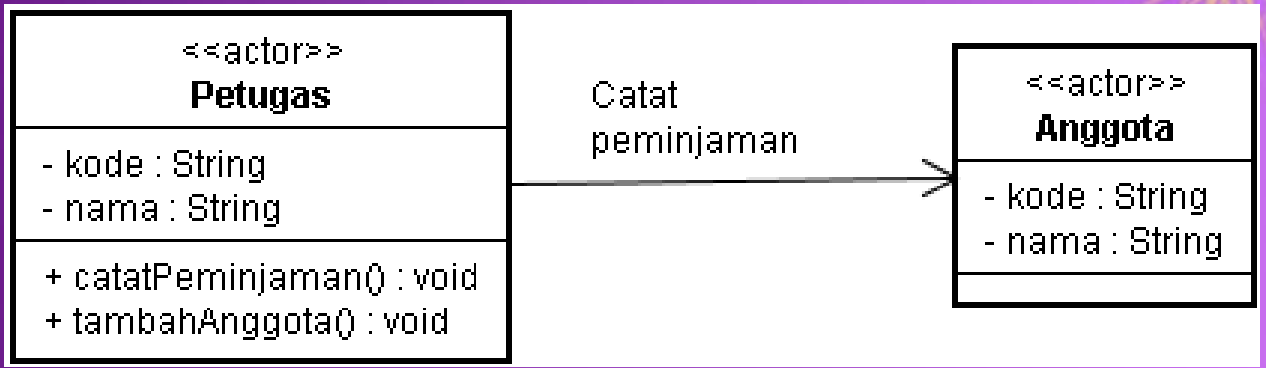
Untuk kelas diagram terdapat beberapa relasi yaitu

1. *Association* atau Asosiasi

Asosiasi adalah hubungan yang terjadi antara kelas yang ada. Asosiasi memungkinkan suatu kelas untuk menggunakan atau mengetahui atribut atau operasi yang dimiliki oleh kelas lain. Asosiasi juga menggambarkan interaksi yang mungkin terjadi antara satu kelas dengan kelas yang lain. Asosiasi ada beberapa jenis, antara lain

i. ***Directional Association* atau Asosiasi 1 arah**

Asosiasi ini menggambar bahwa pesan atau urutan kejadian terjadi dari hanya salah satu kelas sedangkan kelas yang lain pasif. Contohnya pada saat seorang petugas perpustakaan melakukan pencatatan peminjaman terhadap seorang anggota, maka pesan dikirimkan oleh petugas dan diterima oleh anggota. Dimana petugas akan mencatat identitas anggota peminjam dan anggota peminjam berlaku pasif bukannya malah gantian mencatat identitas penjual.



Directional Association atau Relasi 1 arah antara *Class* Petugas dan Anggota

ii. Asosiasi 2 arah (*Bidirectional Association*)

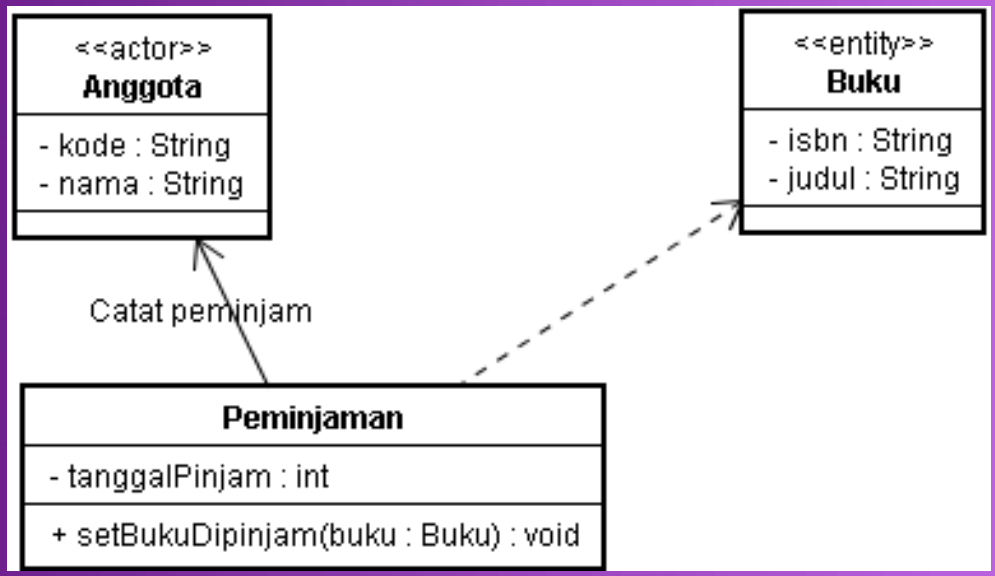
Asosiasi ini terjadi ketika salah satu kelas mengirimkan pesan kepada kelas yang lain kemudian kelas yang lain mengirimkan pesan kepada kelas yang mengirimnya pesan. Contohnya pada saat seorang calon anggota m*end*aftar menjadi anggota perpustakaan maka yang terjadi adalah anggota menyerahkan identitas untuk diproses oleh petugas dan beberapa saat kemudian petugas akan memberikan kartu keanggotaan perpusatakaan.

Bidirectional Association atau Relasi 2 arah antara *Class* Petugas dan Anggota

2. *Depedency* atau Dependensi

Relasi jenis ini menunjukkan bahwa sebuah kelas mengacu kepada kelas lainnya. Oleh sebab itu perubahan pada kelas yang diacu akan sangat berpengaruh pada kelas yang mengacu.

Contohnya apabila seorang anggota h*end*ak meminjam buku maka ada sebuah kelas yang bertanggung jawab melakukan pencatatan peminjam. Kelas ini akan mengetahui anggota yang meminjam dan membuat daftar buku apa saja yang dipinjam oleh anggota tersebut.



Relasi Dependency antara *Class* Peminjaman dan Buku

3. *Aggregation* atau Agregasi

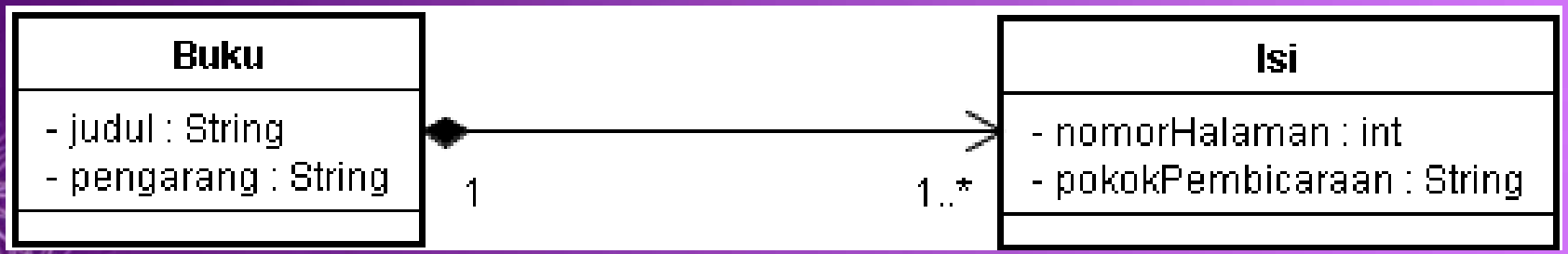
Relasi aggregasi adalah suatu bentuk relasi yang jauh lebih kuat dari pada asosiasi. Aggregasi dapat diartikan bahwa suatu kelas merupakan bagian dari kelas yang lain namun bersifat tidak wajib. Contohnya sebuah buku memiliki pengarang, daftar pustaka, namun bisa saja suatu buku tidak memiliki daftar pustaka. Dari contoh kasus dapat diartikan bahwa daftar pustaka merupakan bagian dari buku namun buku tetap disebut sebagai buku meskipun tidak memiliki daftar pustaka.

Relasi *Aggregation* antara buku dan daftar pustaka

4. *Composition* atau Komposisi

Relasi ini merupakan relasi yang paling kuat dibandingkan dengan asosiasi dan aggregasi. Pada komposisi diartikan bahwa suatu kelas merupakan bagian yang wajib dari kelas yang lain.

Contoh kasus yaitu pada sebuah buku, sudah pasti terdapat halaman isi sekurang kurangnya satu

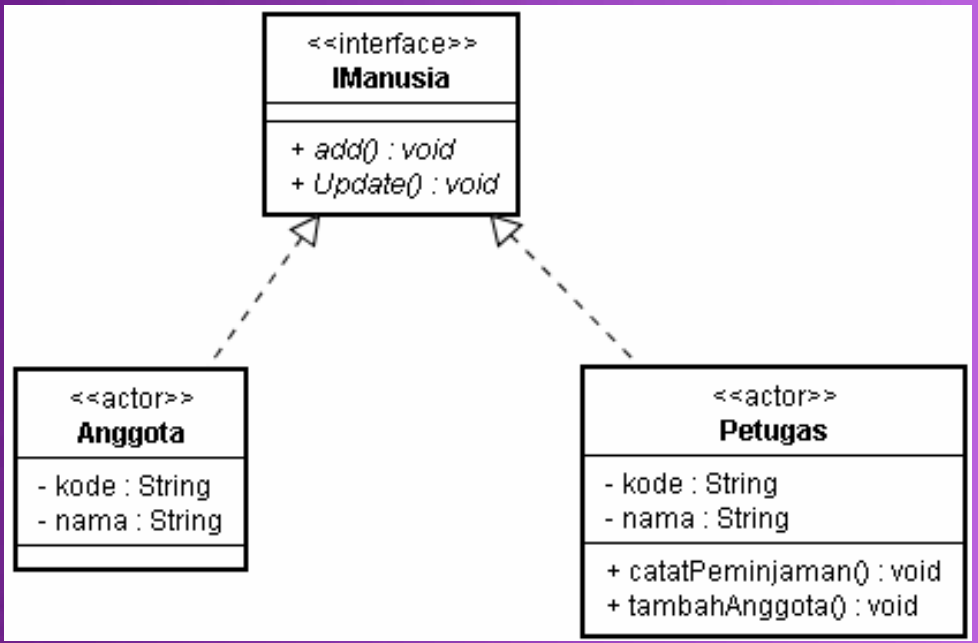


Relasi *Composition* antara buku dan isi

5. *Realization* atau Realisasi

Realisasi, bisa disebut juga implementasi merupakan suatu relasi yang menunjukkan penerapan terhadap suatu interface kepada sebuah *Class*. Relasi realisasi biasanya digunakan untuk mewajibkan suatu kelas memiliki suatu *Method* yang sudah didefinisikan bentuk kerangkanya dalam suatu interface.

Contohnya pada kelas petugas dan anggota, kedua kelas ini tentunya memiliki *Method* yang wajib dimiliki namun melakukan kegiatan yang berbeda seperti add dan update

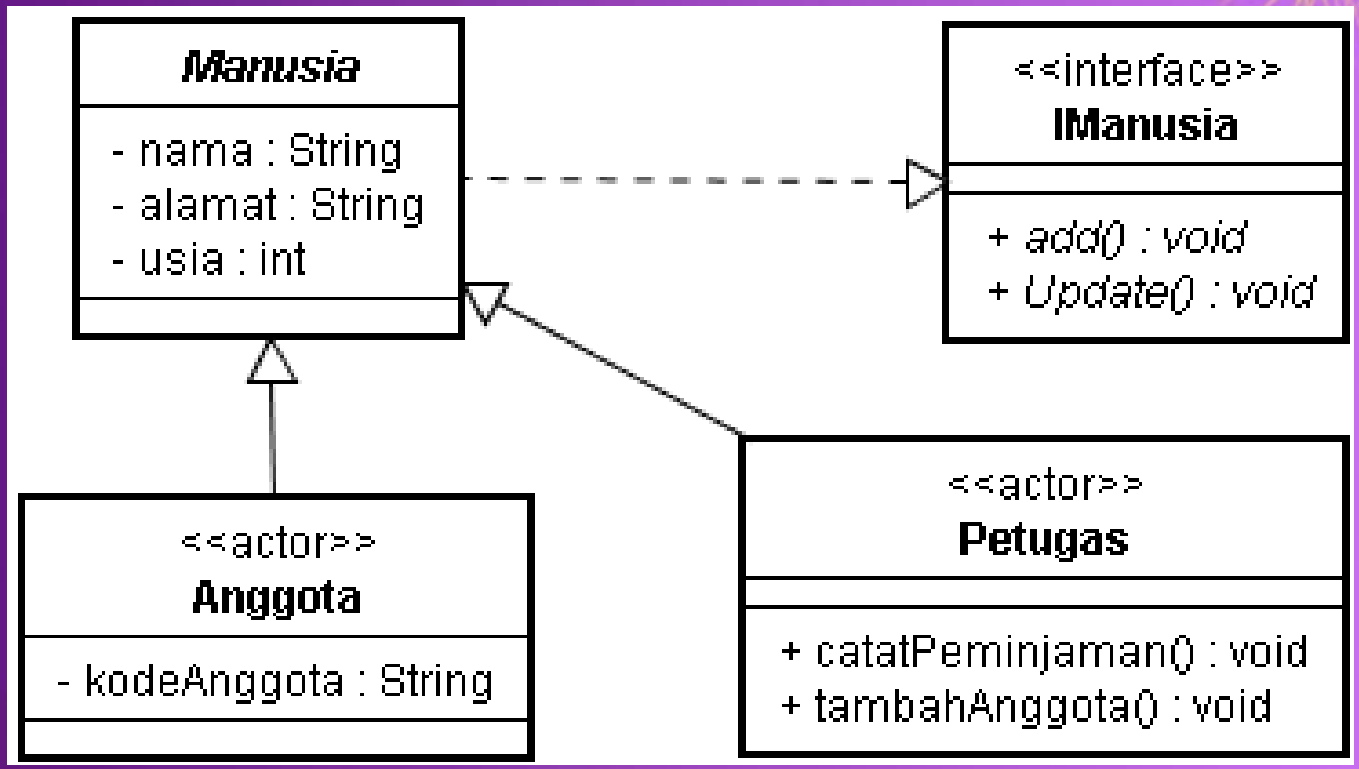


Relasi *Realization* antara IManusia dengan Anggota dan Petugas

6. *Generalization* atau Generalisasi

Adalah relasi pewarisan antara dua *Class*. Relasi jenis ini memungkinkan suatu kelas mewarisi attribute dan operasi yang dimiliki oleh base *Class*3. Attribute dan operasi yang bisa diwarisi oleh suatu kelas adalah yang memiliki access modifier public, protected dan default.

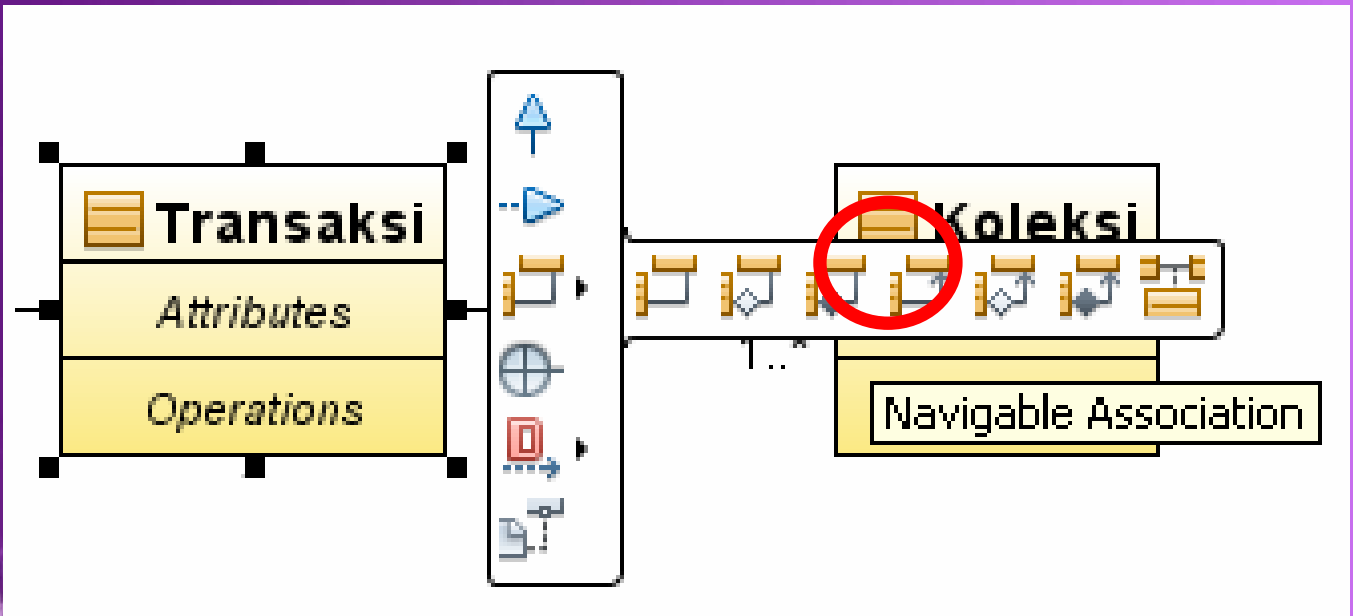
Contohnya bahwa Anggota dan Petugas adalah dua buah kelas yang mewarisi sifat yang dimiliki oleh kelas manusia. Disini kelas manusia berupa kelas abstract yang berarti kelas ini baru bisa digunakan ketika sudah diwariskan kepada suatu kelas atau bila digunakan langsung, *Method* – *Method* wajib yang terdapat didalamnya harus difungsionalitaskan terlebih dahulu



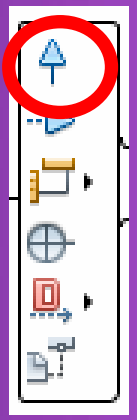
Generalization dari kelas manusia

Kita lanjutkan pembuatan Class Diagram Perpustakaan untuk menambahkan relasi-realasi.

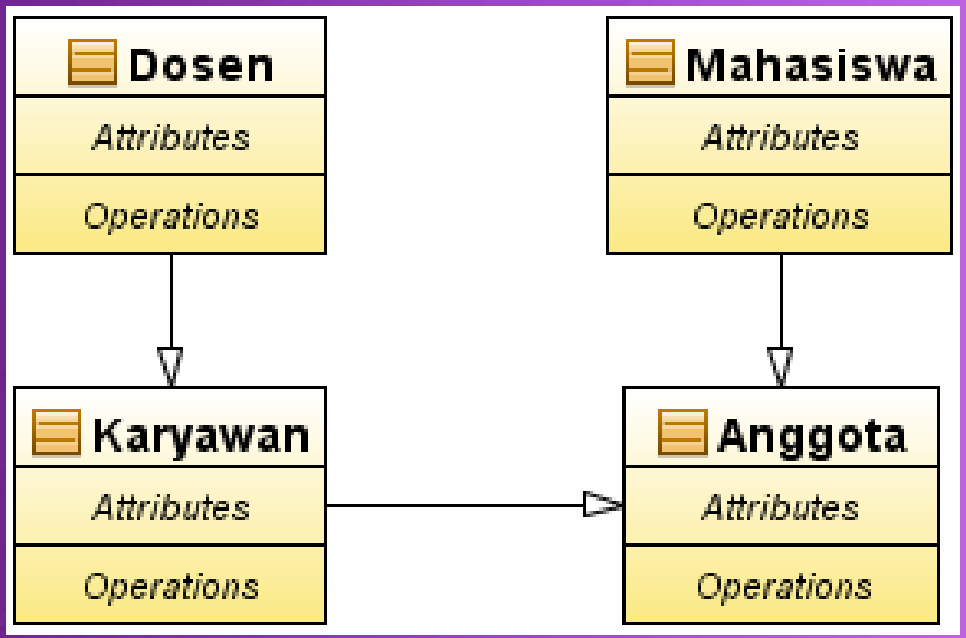
- Tambahkan 1 Class dan berikan nama "Anggota"
- Buat salah satu relasi, misalnya relasi antara Class Transaksi dengan Class Koleksi dengan cara klik pada class Transaksi, Kemudian pilih/sorot Association, klik pada tanda panah disebelahnya, kemudian pilih *Navigable Association*, klik dan drag menuju ke class Koleksi. Navigable Association digunakan untuk menandakan bahwa 1 buah Transaksi terdiri dari 1 sampai banyak transaksi.



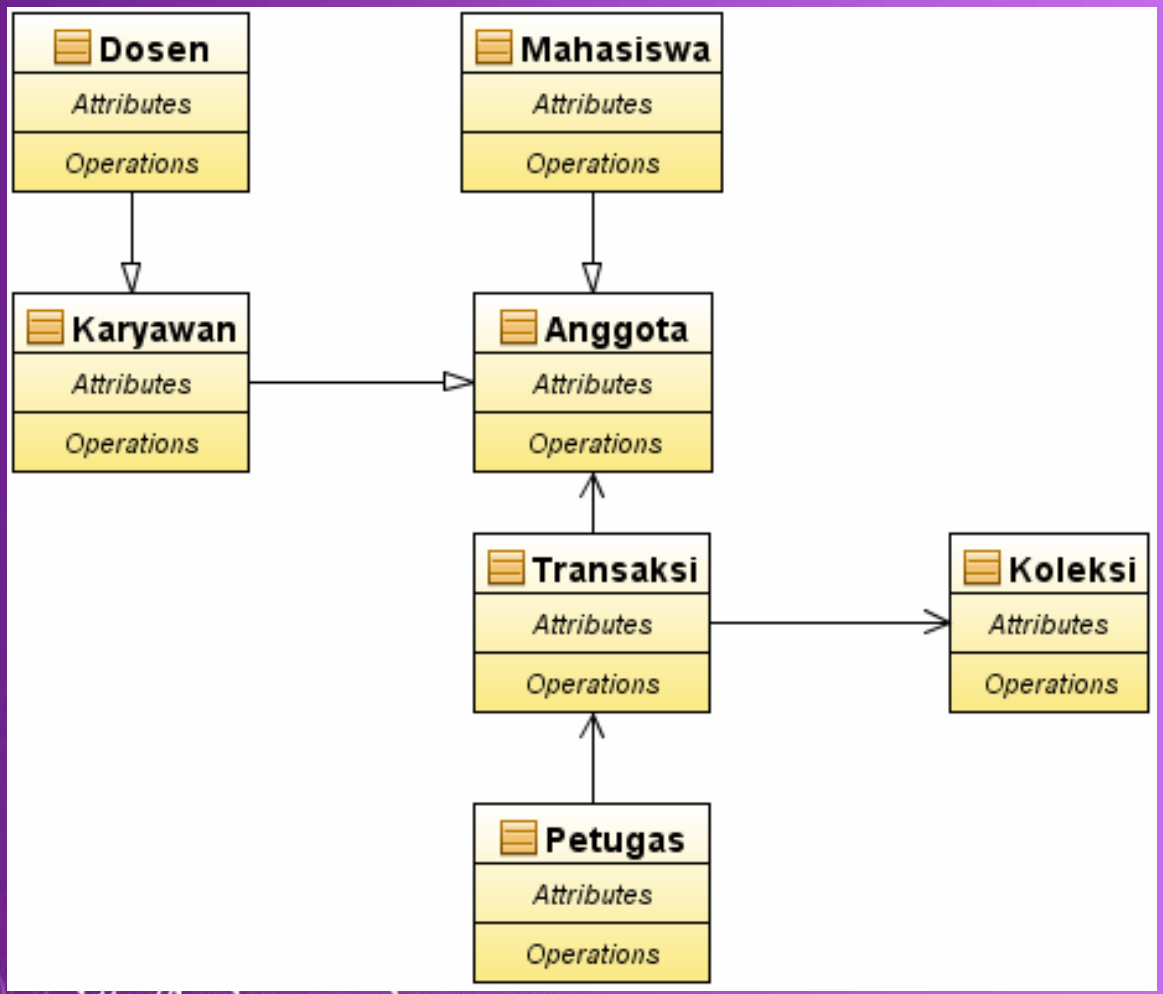
Mahasiswa, Karyawan adalah turunan dari Anggota, dan Dosen adalah turunan dari Karyawan, maka buat relasi generalization.



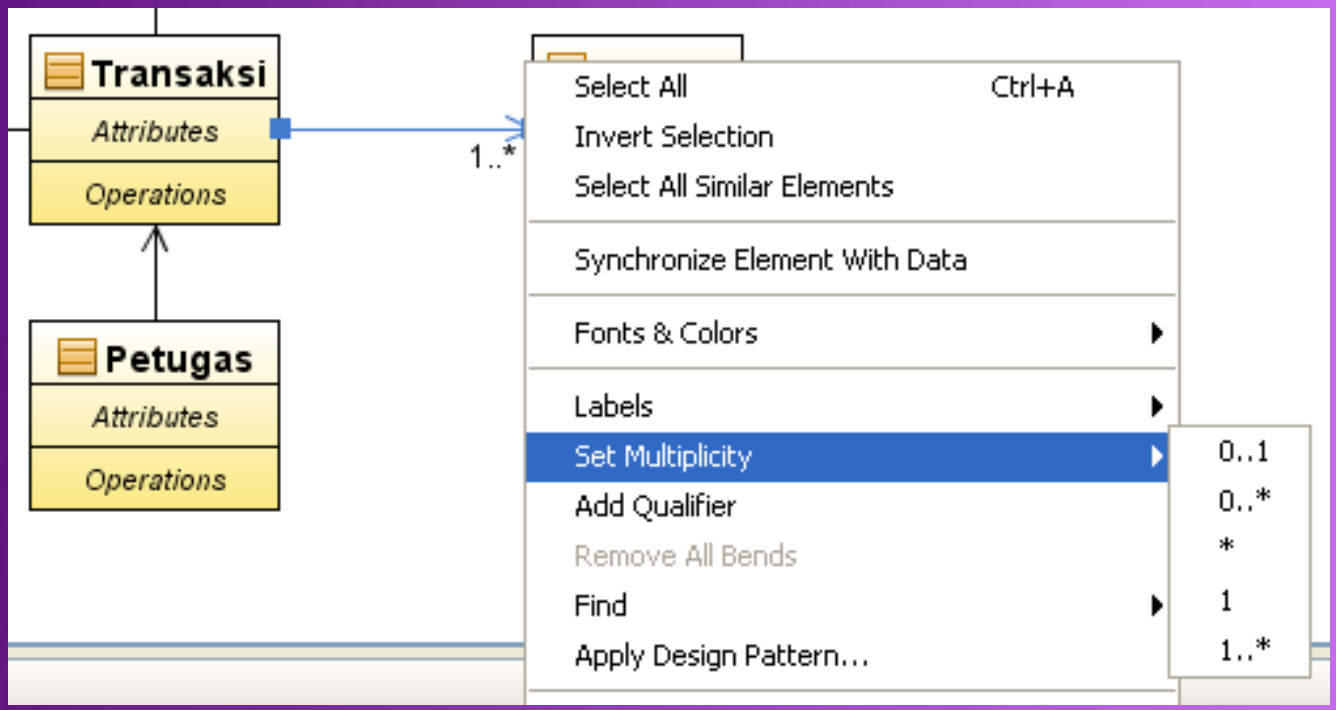
Sehingga relasi yang terbentuk dari keempat Class seperti terlihat berikut.



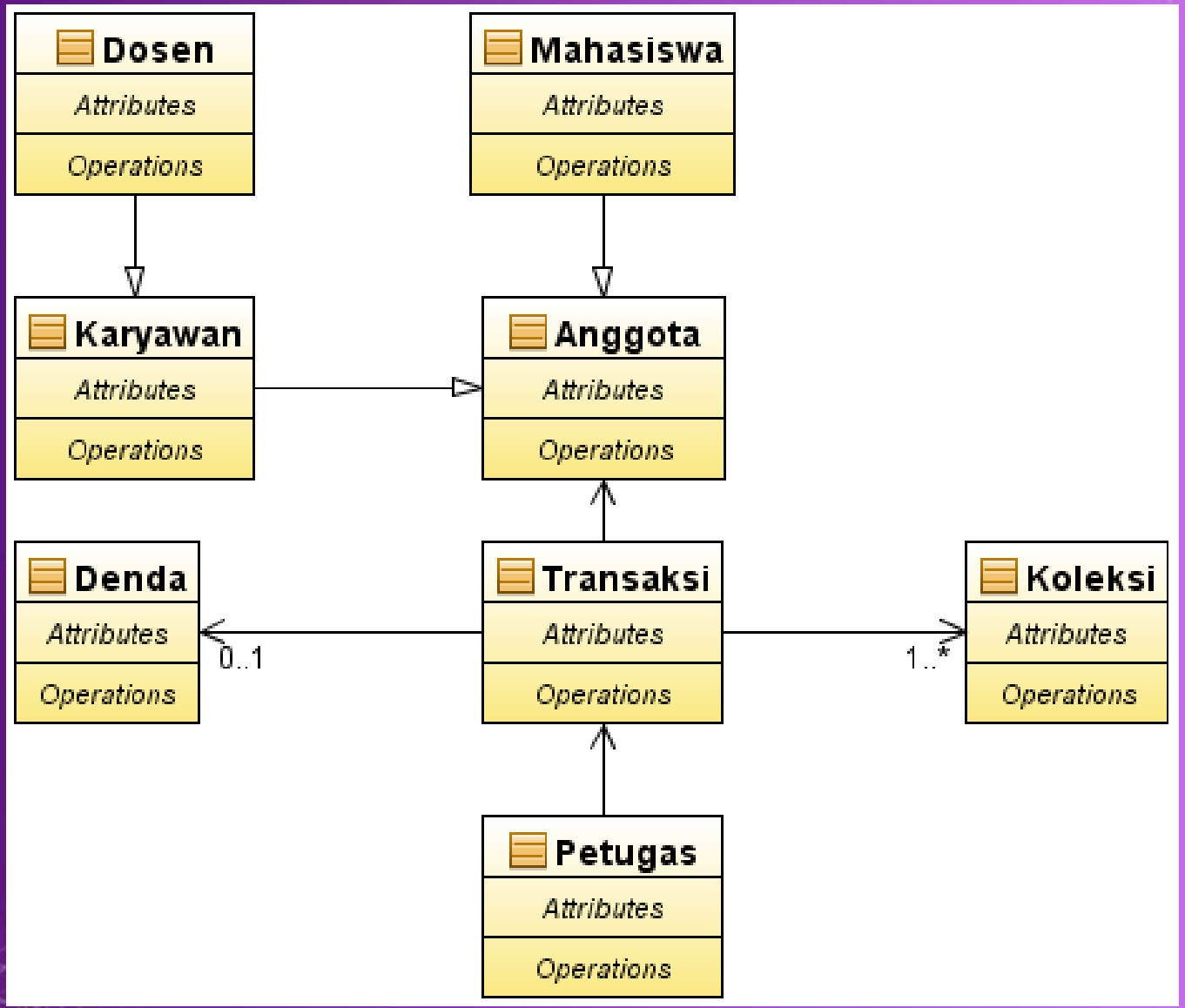
Seorang Petugas perpustakaan akan melayani 1 buah transaksi peminjaman dalam satu waktu, dimana 1 transaksi terdiri dari 1 buah koleksi yang dilakukan oleh seorang anggota, maka gambarnya sebagai berikut



- **Sebuah transaksi pengembalian, bisa memiliki denda atau tidak memiliki denda**
- **Untuk menentukan multiplicity, klik kanan di dekat panah atau pangkal**



Tentukan agar multiplicity yang ada seperti tampak gambar berikut



Apabila tidak terlihat angka 1, maka dianggap bernilai 1