

MODUL 6 :

Requirement Analysis

Daftar isi

6.1	Pengantar Requirement Analysis.....	2
6.2	Mengapa Mendefinisikan <i>User Requirement</i>	2
6.3	Use Case Realization	3
6.3.1	Transformasi Use Case ke Communication dan Class Diagram	3
6.3.2	Analisis stereotype dari class.....	5
	Boundary	5
	Control	6
	Entity	7
6.4	Class Diagram	8
6.4.1	Class dan Instance	8
6.4.2	Atribut	9
6.4.3	Link	9
6.4.4	Assosiasi.....	10
6.4.5	Multiplicity.....	11
6.4.6	Operation.....	11
6.5	Langkah-langkah menggabungkan diagram Class.....	12
	Referensi.....	13

6.1 Pengantar Requirement Analysis

Pada bagian ini akan kita lihat bagaimana use case diagram digunakan untuk membuat sebuah model berdasarkan pada *user requirement*. Kita akan mencoba melakukan analisa *requirement* dengan lebih lengkap. Disamping itu akan kita lihat pula bagaimana menganalisa *use case diagram* dengan sudut pandang user untuk mendapatkan sebuah kelas diagram. Kelas diagram ini akan menjadi dasar bagi kelas diagram selanjutnya yang lebih terinci. *Use case* dan kelas diagram tersebut dihubungkan dengan sebuah aktivitas yang disebut dengan *use case realization*. Diagram komunikasi (*Communication diagram*) digunakan untuk membantu transisi dari *use case* ke *Class diagram*.

6.2 Mengapa Mendefinisikan User Requirement

Faktor penting dari kesuksesan proyek SI adalah bagaimana menentukan *user requirement*. Model dibuat untuk melihat sudut pandang analisis dalam mendapatkan *requirement*. *Requirement* yang sebelumnya dibuat membantu memberikan gambaran mengenai kebutuhan sistem yang akan dikembangkan. Untuk merancang sebuah *software*, kita harus melakukan analisa struktur logika dari permasalahan yang dihadapi serta bagaimana interaksi setiap elemen tersebut. Fungsi utama dari model *requirement* antara lain :

- Konfirmasi apa yang diinginkan user terhadap sistem
 - o Harus mudah dipahami user
 - o Harus benar dan lengkap
- Menentukan apa yang harus di rancang oleh perancang sistem
 - o Tidak boleh ambigu

Selain itu *requirement* juga digunakan untuk :

- Menjelaskan apa yang harus dilakukan oleh *software* yang akan di bangun
- Mewakili orang, benda atau konsep untuk memahami apa yang sedang terjadi
- Menunjukkan hubungan serta interaksi antara orang, benda atau konsep tersebut
- Menunjukkan situasi bisnis secara jelas untuk mengevaluasi kemungkinan rancangan yang akan di buat

- *Requirement* disusun kembali sehingga berguna untuk pengembangan *software* berikutnya

Tools yang digunakan untuk proses analisa *requirement* adalah *class diagram*. Ada 2 cara yang digunakan untuk mendapatkan *class diagram*, antara lain:

- Langsung diperoleh berdasarkan pengetahuan pada *application domain*
- Membuat *class diagram* dari setiap *use case* secara terpisah kemudian setiap *class diagram* yang diperoleh disusun menjadi satu *class diagram*.

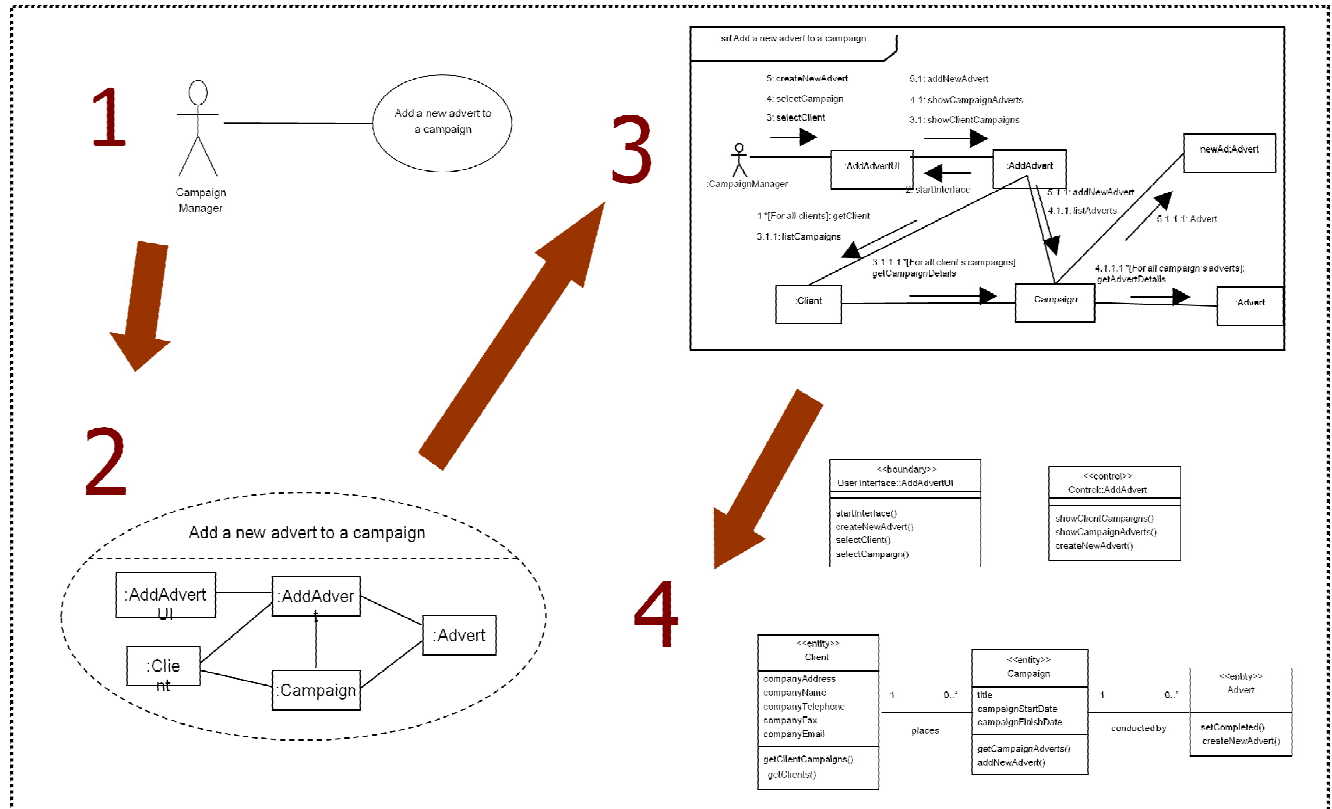
6.3 Use Case Realization

6.3.1 Transformasi Use Case ke Communication dan Class Diagram

Dari model *Use Case* maka tahapan berikutnya adalah melanjutkan ke arah implementasi *software*. Proses pengembangan suatu model abstrak atau elemen yang lebih dekat ke arah implementasi di sebut dengan ***realization***.

Use Case realization dilakukan dengan mencari sekumpulan *class* yang mungkin serta bagaimana interaksi *class* tersebut pada untuk mendapatkan fungsionalitas dari *use case*. Kumpulan dari *class* itu di sebut dengan ***collaboration***. *Collaboration* adalah realisasi dari *use case* tertentu.

From Requirements to Classes



© Bennett, McRobb and Farmer 2005

22

Gambar 4. 1 transformasi Use case ke Diagram Class

Diagram komunikasi sangat berguna untuk melihat secara lebih rinci. Diagram ini juga memperlihatkan interaksi antara objek. Hal terpenting pada diagram komunikasi adalah untuk menunjukkan bagaimana objek berkolaborasi antara satu dengan yang lain. Hasil dari kolaborasi tersebut dapat di representasikan dalam bentuk *Class diagram*. *Class diagram* tidak menunjukkan interaksi antara objek namun lebih memperlihatkan struktur dari objek serta memperlihatkan fasilitas lain yang dimiliki oleh *class*.

6.3.2 Analisis stereotype dari class

Elemen model yang terdapat dalam model analisis disebut kelas analisis (*analysis class*)

- *Boundary*, adalah kelas yang memodelkan interaksi antara satu atau lebih aktor dengan sistem
- *Control*, digunakan untuk memodelkan “perilaku mengatur”, khusus untuk satu atau beberapa use-case
- *Entity*, memodelkan informasi yang harus disimpan oleh sistem

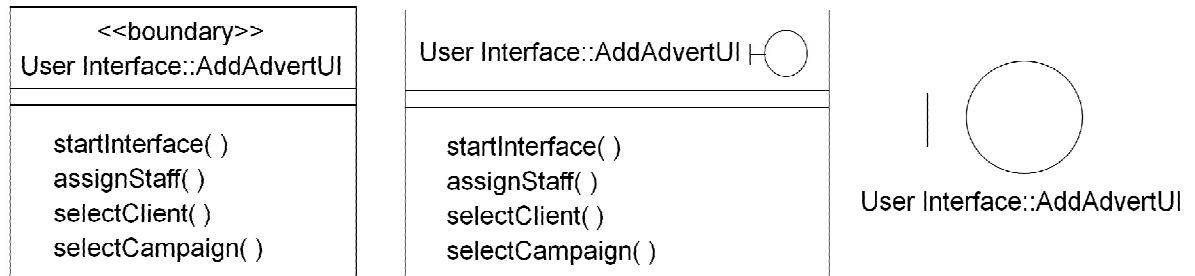
Boundary

Kelas boundary memodelkan bagian dari sistem yang bergantung pada pihak lain disekitarnya dan merupakan pembatas sistem dengan lingkungan luarnya.

Dapat Berupa :

- o **User Interface** : sarana komunikasi sistem dengan user (contoh : *window* dalam GUI)
- o **System Interface** : sarana komunikasi sistem dengan sistem lainnya (*communication protocol*)
- o **Device Interface** : sarana komunikasi sistem dengan device/alat (misal : printer, sensor, dsb)

Alternative notations for boundary class:



© Bennett, McKobb and Farmer 2005

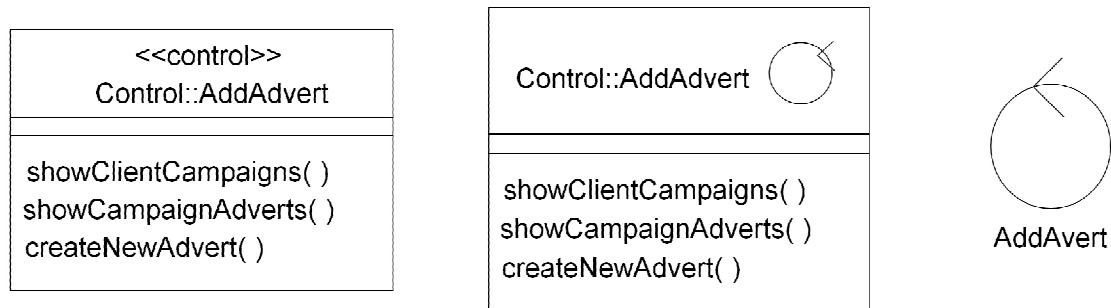
8

Gambar 4. 2 Simbol Class Boundary

Control

Kelas **control** menggunakan atau membuat isi dari kelas entity. Kelas control memasang kelas boundary dengan kelas entity. **Control object** (*instance* dari kelas *Control*) mengkoordinasi perilaku sistem dan menggambarkan dinamika sistem dan mengontrol alur kerja suatu sistem

Alternative notations for control class:



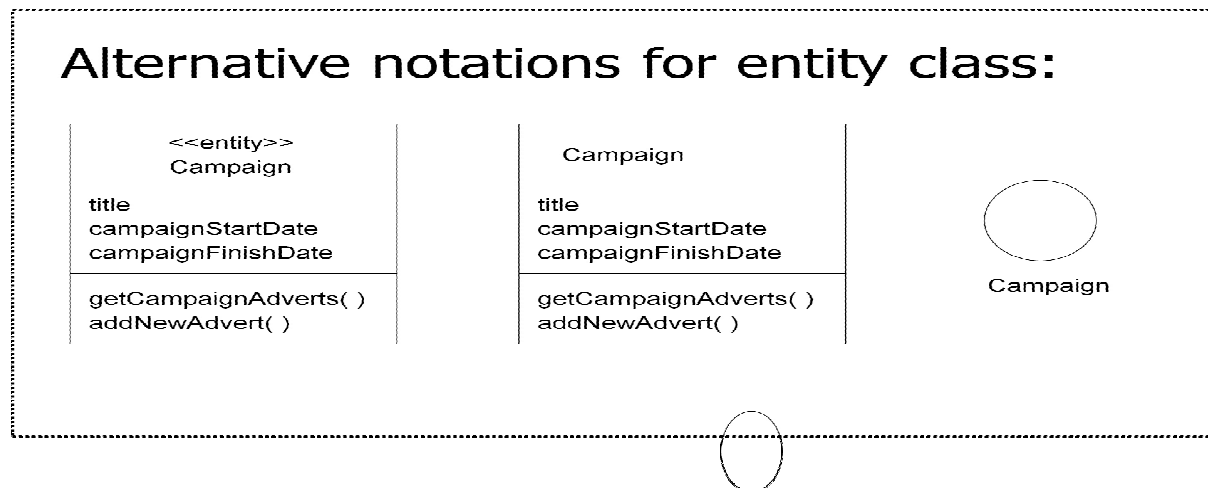
© Bennett, McRobb and Farmer 2005

10

Gambar 4. 3 Simbol Class Control

Entity

Kelas entity memperlihatkan struktur data dari sebuah sistem. Digunakan untuk memahami apa yang kira-kira akan ditawarkan oleh sistem kepada *user*. *Entity object* (*instance* dari kelas *entity*) biasanya pasif dan tetap/tidak berubah-ubah. Tanggungjawabnya adalah menyimpan dan mengatur informasi dalam sistem.

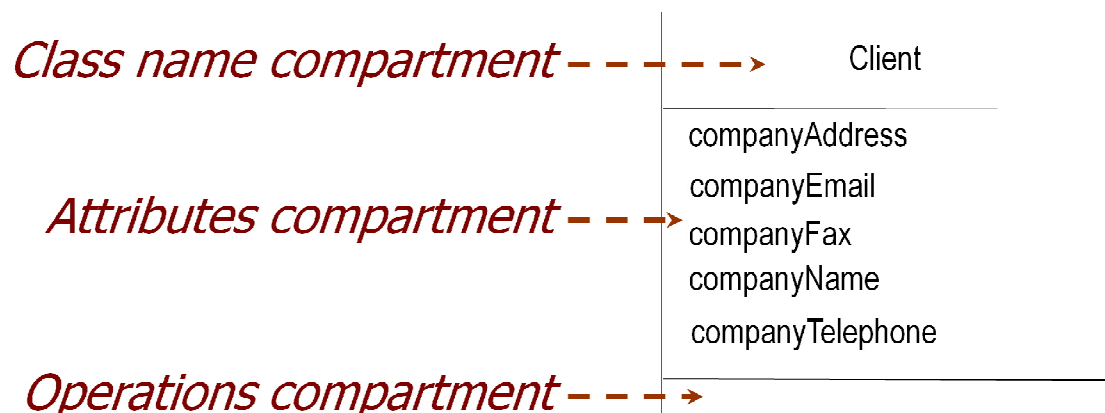


Gambar 4. 4 Simbol Class Entity

6.4 Class Diagram

6.4.1 Class dan Instance

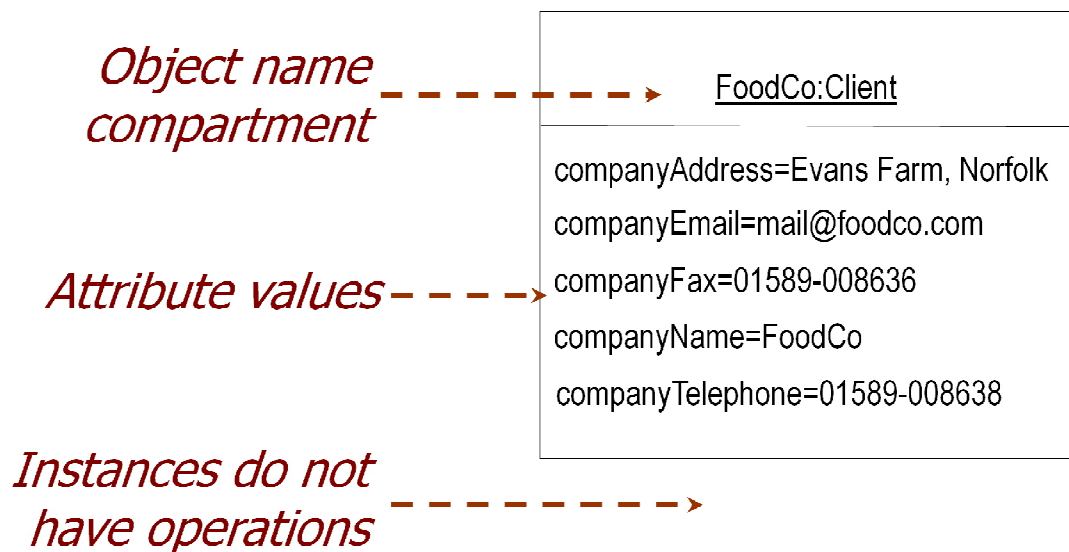
Class diagram berisi kumpulan kelas yang berassosiasi antara satu dengan yang lain. Sebuah *Class diagram* digambarkan dengan kotak yang terdiri atas tiga bagian. Masing-masing bagian berisi tentang nama kelas, atribut serta operation class tersebut. Dibawah ini adalah notasi *class*.



Gambar 4. 5 Notasi Class

Class Instance adalah representasi dari *class*. Pada *class instance* menunjukkan masing-masing objek dari setiap *class*. *Class instance* tidak memiliki operation.

Class Diagram: Instance Symbol



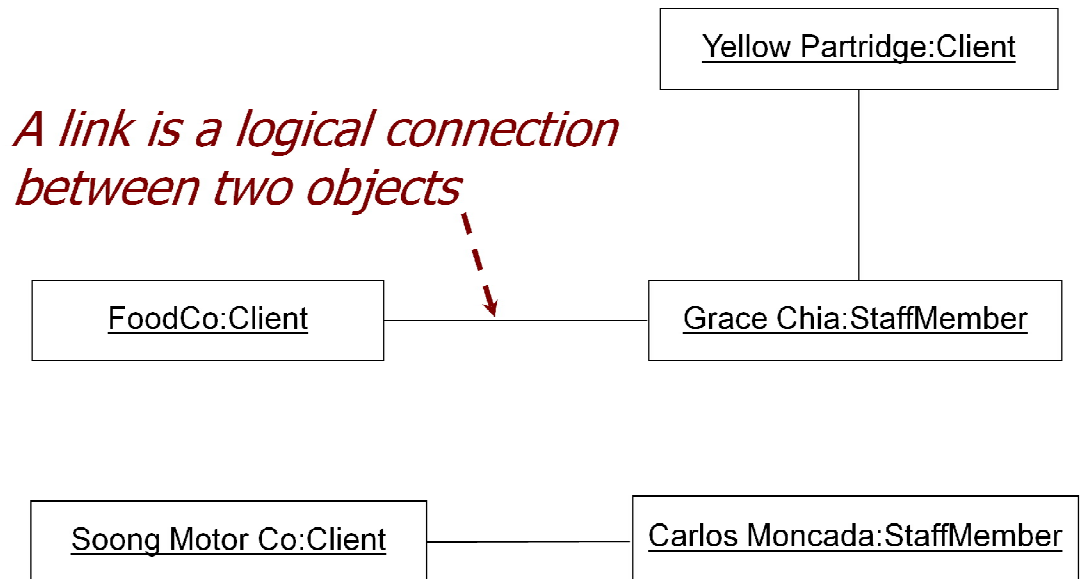
Gambar 4. 6 Notasi Instance

6.4.2 Atribut

Atribut merupakan bagian paling penting dari pendefinisian *class*. Atribut merupakan struktur umum yang dimiliki oleh kelas. Setiap objek memiliki nilai atribut.

6.4.3 Link

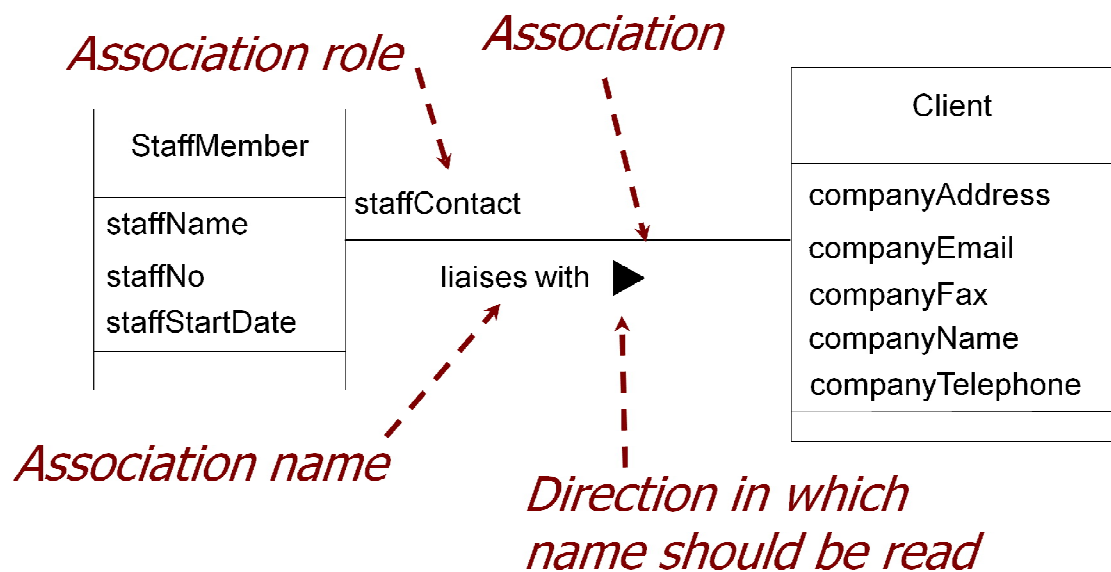
Link merupakan koneksi logic antara dua buah objek



Gambar 4. 7 Link

6.4.4 Assosiasi

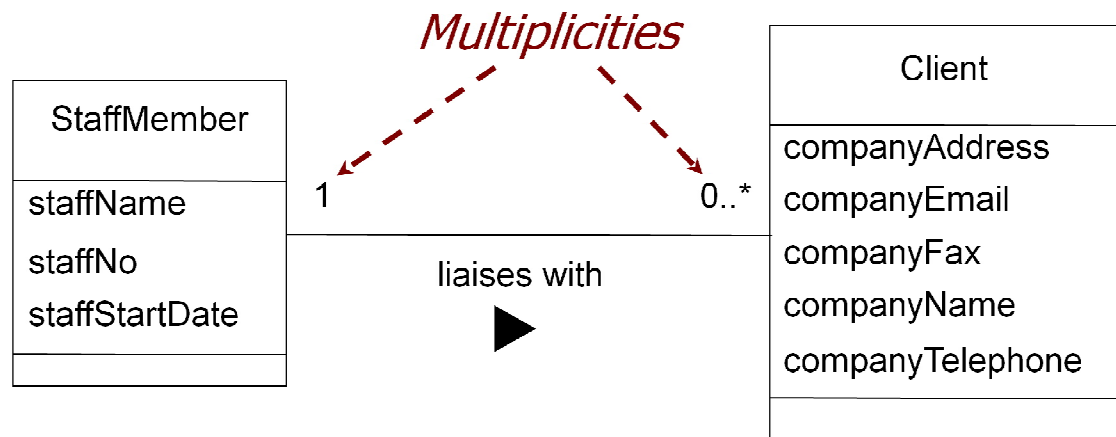
Assosiasi menunjukkan Kemungkinan hubungan logis atau hubungan antara objek dari satu kelas dengan objek lain. Jika dua Objek bisa dihubungkan maka ada assosiasi antar kelas.



Gambar 4. 8 Assosiasi

6.4.5 Multiplicity

Merupakan *range* kardinalitas dari suatu asosiasi. Sebagai contoh: Setiap nasabah bank mungkin memiliki satu atau lebih rekening. Setiap *account* adalah untuk satu, dan hanya satu, pelanggan

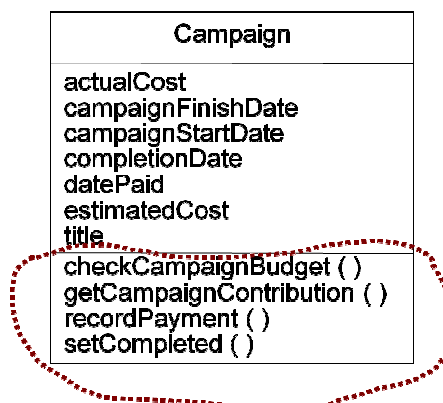


- Exactly one staff member liaises with each client
- A staff member may liaise with zero, one or more clients

Gambar 4. 9 Multiplicity

6.4.6 Operation

Operation merupakan bagian yang penting dari class. Operation biasanya merupakan *behavior* (perilaku) dari objek sebuah *class*. *Operation* menggambarkan apa yang bisa dilakukan oleh *instance* dari sebuah class.



Gambar 4. 10 Operation

6.5 Langkah-langkah menggabarkan diagram Class

Untuk menggambarkan diagram class dapat dilakukan hal sebagai berikut:

1. Mulai dari satu buah *Use Case*
2. Identifikasi semua *Class* yang terlibat (Gunakan *use case collaboration*)
3. Gambar *collaboration diagram* yang memenuhi kebutuhan *use case*
4. Terjemahkan *collaboration* ke *class diagram*
5. Ulangi untu *use case* lainnya

Referensi

1. Simon Bennet, Steve McRobb and Ray Farmer, *Object Oriented Systems Analysis and Design Using UML*, Edisi 3. ; McGraw Hill, 2006. (SB)