BAB II
TINJAUAN PUSTAKA

2.1 Sistem

Sistem adalah kumpulan elemen-elemen yang saling terkait atau terpadu yang dimaksudkan untuk mencapai suatu tujuan. Sebagai gambaran, jika sebuahsistem terdapat elemen yang tidak memberikan manfaat dalam mencapai tujuan yang sama maka elemen tersebut dapat dipastikan bukan bagian dari sistem. Sehingga dapat disimpulkan bahwa suatu sistem adalah suatu jaringan kerja dari prosedurprosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu. (Dwi Mayasari, 2018)

2.2 Informasi

Informasi berarti data yang telah diproses sehingga meningkatkan pengetahuan seseorang yang menggunakan data tersebut. Sebaliknya data merupakan sekumpulan baris fakta yang mewakili peristiwa yang terjadi pada organisasi atau lingkungan fisik sebelum diolah dalam suatu format yang dapat dipahami dan digunakan orang (Dwi Mayasari, 2018)

2.3 Sistem Informasi

Sistem informasi adalah suatu sistem di dalam organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan (Michael Lukito, 2018)

2.4 Kualitas Informasi

Pernyataan yang dikemukakan oleh Tata Sutabri (2016) Kualitas suatu informasi tergantung dari 3 (tiga) hal, yaitu akurat *(accurate*), tepat waktu (*timeliness*) dan

relevan (*relevance*), penjelasan tentang kualitas informasi tersebut akan dipaparkan di bawah ini:

a. Akurasi (*accuracy*)
   Informasi harus bebas dari kesalahan – kesalahan dan tidak menyesatkan. Akurat juga berarti bahwa informasi harus jelas mencerminkan maksudnya.
b. Tepat waktu (*time lines*)
   Informasi yang datang kepada penerima tidak boleh terlambat. Informasi yang sudah usang tidak mempunyai nilai lagi, karena informasi merupakan suatu landasan dalam mengambil sebuah keputusan dimana bila pengambilan keputusan terlambat maka akan berakibat fatal untuk organisasi.
c. Relevan (*relevance*)
   Informasi tersebut mempunyai manfaat untuk pemakainya. Relevansi informasi untuk setiap orang berbeda. Menyampaikan informasi tentang penyebab kerusakan mesin produksi kepada akuntan perusahaan tentunya kurang relevan. Akan lebih relevan bila ditujukan kepada ahli teknik perusahaan.

2.5 Sekolah

Sekolah adalah kelompok layanan pendidikan yang menyelenggarakan pendidikan pada jalur pendidikan formal, nonformal dan informal daalam setiap jenjang dan jenis pendidikan.

2.6 Metode Rapid Application Development

Metode pengembangan sistem yang digunakan adalah Rapid Application Development (RAD) yaitu suatu strategi siklus hidup yang ditujukan untuk menyediakan pengembangan yang jauh lebih cepat dan mendapatkan hasil dengan kualitas lebih baik dibandingkan dengan hasil yang dicapai melalui siklus tradisional, Rapid Application Development (RAD) adalah suatu pendekatan beriorientasi objek terhadap pengembangan sistem yang mencakup suatu metode pengembangan untuk perangakat – perangkat lunak dimana Rapid Application

Development (RAD) bertujuan mempersingkat waktu yang biasanya diperlukan dalam siklus hidup pengembangan sistem tradisional antara perancangan dan penerapan suatu sistem informasi. Pada akhirnya, Rapid Application Development (RAD) sama-sama berusaha memenuhi syaratsyarat bisnis yang berubah secara cepat (Putrawansyah, 2018). Berikut ini adalah tahapan pengembangan aplikasi dari tiap-tiap fase pengembangan aplikasi :

1). Requirements Planning (Perencanaan Syarat-Syarat)

Dalam fase ini, pengguna dan penganalisis bertemu untuk mengidentifikasikan tujuan-tujuan aplikasi atau sistem serta untuk megidentifikasikan syarat-syarat informasi yang ditimbulkan dari tujuan-tujuan tersebut. Orientasi dalam fase ini adalah menyelesaikan masalah-masalah perusahaan. Meskipun teknologi informasi dan sistem bisa mengarahkan sebagian dari sistem yang diajukan, fokusnya akan selalu tetap pada upaya pencapaian tujuan-tujuan perusahaan.

2). RAD Design Workshop (Workshop Desain RAD)

Fase ini adalah fase untuk merancang dan memperbaiki yang bisa digambarkan sebagai workshop. Penganalisis dan dan pemrogram dapat bekerja membangun dan menunjukkan representasi visual desain dan pola kerja kepada pengguna. Workshop desain ini dapat dilakukan selama beberapa hari tergantung dari ukuran aplikasi yang akan dikembangkan. Selama workshop desain RAD, pengguna merespon prototipe yang ada dan penganalisis memperbaiki modulmodul yang dirancang berdasarkan respon pengguna. Apabila sorang pengembangnya merupakan pengembang atau pengguna yang berpengalaman, Kendall menilai bahwa usaha kreatif ini dapat mendorong pengembangan sampai pada tingkat terakselerasi.

3). Implementation (Implementasi)

Pada fase implementasi ini, penganalisis bekerja dengan para pengguna secara intens selama workshop dan merancang aspek-aspek bisnis dan nonteknis perusahaan. Segera setelah aspek-aspek ini disetujui dan sistem-sistem

dibangun dan disaring, sistem-sistem baru atau bagian dari sistem diujicoba dan kemudian diperkenalkan kepada organisasi.

Gambar 2. 1 Metode RAD (Rapid Application Development)

2.7 Perangkat Lunak Penunjang

Perangkat lunak yang digunakan untuk membangun perancangan sistem informasi ini diantaranya:

2.7.1 PHP (*Hypertext preprocessing*)

Menurut (Dwiki, 2017) singkatan dari *Hypertext Preprocessor* yang merupakan *server-side programming*, yaitu Bahasa pemrograman yang di proses di sisi server. Fungsi utama PHP dalam membangun website adalah untuk melakukan pengolahan data pada database. Data website akan dimasukan ke database, diedit, dihapus dan ditampilkan pada website yang di atur oleh PHP.

Pengembangan demi pengembangan terus berlanjut, ratusan fungsi ditambahkan sebagai fitur bahasa PHP, dan di awal tahun 1999, netcraft mencatat, ditemukan 1.000.000 situs di dunia telah menggunakan PHP. Ini membuktikan bahwa PHP merupakan bahasa yang paling populer digunakan oleh dunia web development. Hal ini mengagetkan para developernya termasuk Rasmus sendiri, dan tentunya sangat diluar dugaan sang pembuatnya. Kemudian Zeev Suraski dan Andi Gutsman selaku *core developer* (programmer inti) mencoba menulis ulang PHP parser, dan di integrasikan dengan menggunakan *Zend scripting engine*, dan mengubah jalan alur operasi PHP. Semua fitur baru tersebut dirilis dalam PHP 4 pada 13 Juli 2004

telah mengalami banyak perbaikan bahasa web populer di dunia, karena tercatat 19 juta domain telah menggunakan PHP sebagai server side scripntingnya.

2.7.2 MySql

Menurut (Rahmad, 2017) adalah nama database *server*. Database *server* adalah server yang berfungsi untuk menangani database. Database adalah suatu pengorganisasian data dengan tujuan memudahkan penyimpanan dan pengakseskan data. Dengan menggunakan MySQL, kita bisa menyimpan data dankemudian data diakses dengan cara yang mudah dan cepat.

2.7.3 HTML

Menurut (Dita, 2017) HTML adalah singkatan dari *HyperText Markup Language*. HTML merpakan *file* teks yang ditulis menggunakan aturan-aturan kode tertentu untuk kemudian disajikan ke user melalui suatu aplikasi web *browser*. Setiap informasi yang ditampil pada web selalu dibuat menggunakan kode HTML. Oleh karena itu, dokumen HTML sering disebut juga sebagai halaman web. Untuk membuat dokumen HTML, kita tidak tergantung pada aplikasi tertentu, karena dokumen HTML dapat dibuat menggunakan aplikasi Text Editor apapun, bisa *Notepad* (untuk lingkungan MS Windows), *Emacs* atau *Vi Editor* (untuk lingkungan *linux*) dan sebagainya.

2.7.4 XAMPP

Menurut (Aulia, 2017) XAMPP adalah perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan komplikasi dari beberapa program. Fungsinya adalah sebagai server yang berdiri sendiri (*localhost*), yang terdiri atas program *Apache* HTML *server*, MySQL database, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Namun XAMPP merupakan singkatan dai X(empat sistem operasi apapun), Apache, MySQL, PHP dan Perl. Program ini tersedia dalam General Public License dan bebas, merupakan web server yang mudah digunakan yang dapat melayani tampilan halaman web yang dinamis.

XAMPP dikembangkan dari sebuah tim proyek bernama *Apache Friends*, yang terdiri dari tim inti (*Core Team*), Tim Pengembang (*Development Team*) & Tim Dukungan(*Support Team*). Bagian XAMPP yang biasa digunakan pada umumnya:

1. HTDOC adalah folder tempat meletakkan berkas-berkas yang akan dijalankan, seperti berkas PHP, HTML dan skrip lainnya
2. PhpMyAdmin merupakan bagian untuk mengelola basis data MySQL yang ada di komputer. Hal pertama yang harus dilakukan untuk mengoperasikannya yaitu, buka browser lalu ketik alamat htttp://localhost/phpMyAdmin, maka akan muncul halaman phpMyAdmin.
3. Kontrol Panel berfungsi untuk mengelola layanan seperti, memulai atau menghentikan layanan XAMPP.

2.8 Flowchart

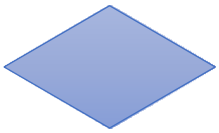
Menurut Rodlo (2017) *Flowchart* adalah penggambaran secara grafik dari langkah-langkah dan urutan-urutan prosedur dari suatu program. *flowchart* menolong analis dalam untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam pengoprasiannya.

Menurut Tague (2005) *flowchart* adalah alat pemetaan sederhana yang menunjukan urutan tundakan dalam proses dalam bentuk yang mudah dibaca dan dikomunikasikan.

Dari pengertian di atas dapat didefinisikan *flowchart* sebagai skema penggambaran dari algoritma atau proses. Tabel berikut menampilkan simbol-simbol yang digunakan dalam penyusunan *flowchart.* Simbol-simbol dalam *flowchart* dalam dilihat pada Tabel 2.1

Tabel 2.1 Simbol-Simbol dalam *Flowchart*

Simbol	**Keterangan**
	**Terminator,** Sebagai simbol '*START*' atau *End*' untuk memulai atau mengakhiri *flowchart.*

	**Input/output,** Digunakan untuk menuliskan proses menerima data atau mengeluarkan.
	**Porses,** Digunakan untuk menuliskan proses yang di perlukan, misalnya operasi aritatika.
	**Predefinded,** adalah simbol yang digunakan untuk menunjukan pelaksanaan suatu bagian prosedur (sub-proses). Dengan kata lain, prosedur yang terinformasi di sini belum detail dan akan dirinci di tempat lain**.**
	**Conditional / Decision,** Digunakan untuk menyatakan proses yang membutuhkan keputusan.
	**Preparation,** Digunakan untuk membeikan nilai awal.
	**Arrow,** Sebagai penunjuk arah dan alur proses.
	**Connector *(On-page)*,** Digunakan untuk menyatukan beberapa arrow.
	**Connector (Off-page)** Digunakan untuk menghubungkan flowchart yang harus digambarkan pada halaman yang berbeda. Biasanya pada simbol ini

	diberi nomor sebagai penanda, misalnya angka 1.

2.9 Unified Modeling Language (UML)

2.9.1 Pengenalan *Unified Modeling Language* (UML)

Menurut (Adyanto, 2019) *Unified Modeling Language (UML)* adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek.

UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. *UML* muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak.

UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan *UML* tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya *UML* paling banyak digunakan pada metodologi berorientasi objek.

2.9.2 Sejarah UML

Menurut (Adyanto, 2019) Bahasa pemrograman berorientasi objek yang pertama dikembangkan dikenal dengan nama Simula-67 yang dikembangkan pada tahun 1967. Perkembangan aktif dari pemrograman berorientasi objek mulai menggeliat ketika berkembangnya bahasa pemrograman Smalltalk pada awal 1980-an yang kemudian diikuti dengan perkembangan bahasa pemrograman berorientasi objek yang lainnya seperti C objek, C++, Eiffel, dan CLOS. Sekitar lima tahun setelah *Small talk* berkembang, maka berkembang pula metode pengembangan berorientasi objek. Karena banyaknya metodologi yang berkembang pesat pada saat itu, maka muncul ide untuk membuat sebuah bahasa yang dapat dimengerti semua orang. Oleh sebab itu dibuat bahasa yang merupakan gabungan dari beberapa konsep, seperti konsep *Object Modeling Technique (OMT)* dari Rumbaugh dan Booch (1991), konsep *The Classes, Responsibilities, Collaborators (CRC)* dari Rebecca Wirfs-Brock (1990), konsep pemikiran Ivar Jacobson, dan beberapa konsep lainnya dimana James R. Rumbaigh, Grady Booch, dan Ivar Jacobson bergabung dalam

sebuah perusahaan yang bernama Rational Software Corporation menghasilkan bahasa yang disebut dengan *Unified Modeling Language (UML).*

Pada tahun 1996, *Object Management Group (OMG)* mengajukan proposal agar adanya standarisasi pemodelan berorientasi objek dan pada bulan September 1997 *UML* diakomodasi oleh *OMG* sehingga sampai saat ini *UML* telah memberikan kontribusinya yang cukup besar di dalam metodologi berorientasi objek dan hal-hal yang terkait di dalamnya.

2.9.3 Diagram UML

Pada *UML* terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Berikut ini penjelasan singkat dari pembagian kategori tersebut.

1. *Structure diagram*, yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan. *Structure diagram* terdiri dari *class diagram, object diagram, component diagram, composite structure diagram, package diagram dan deployment diagram.*
2. *Behavior diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem. Behavior diagram terdiri dari *Use case diagram, Aktivity diagram, State Machine System.*
3. *Interaction diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem. *Interaction diagram* terdiri dari *Sequence Diagram, Communication Diagram, Timing Diagram, Interaction Overview Diagram.*

2.9.4 Use Case Diagram

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih *actor* dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Simbol-simbol yang terdapat pada diagram *use case* dapat dilihat pada Tabel 2.2

Tabel 2.2 Simbol-Simbol *Use Case* Diagram

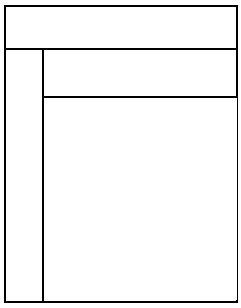
Simbol	Keterangan Fungsi
Aktor 	Aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.
Use Case 	*Use Case* adalah deskripsi dari urutan aksi- aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
Asosiasi 	Asosiasi adalah apa yang menghubungkan antara objek satu dengan objek yang lainnya.
Generalisasi 	Generalisasi adalah hubungan dimana objek anak (*descendent*) berbagi prilaku dan struktur data dari objek yang ada diatasnya atau sebaliknya dari bawah ke atas.
Defendency	*Defendency* (ketergantungan) adalah hubungan dimana perubahan yang terjadi pada suatu elemen defenden (mandiri) akan mempengaruhi elemen yang bergantung padanya (independen).

2.9.5 Activity Diagram

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu di perhatikan disini adalah bahwa diagram aktivitas

menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Simbol-simbol yang terdapat pada diagram aktivitas dapat dilihat pada Tabel 2.3

Tabel 2.3 Simbol-Simbol *Activity Diagram.*

Simbol	**Keterangan**
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
Percabangan/*decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
Penggabungan/join 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
Status akhir 	Status akhir yang dilakukan oleh sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

2.9.6 Class Diagram

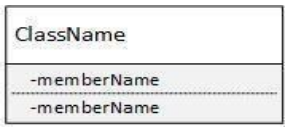
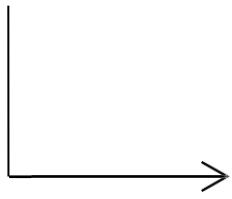
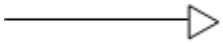
Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas

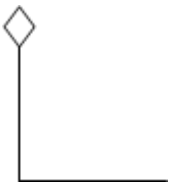
memiliki apa yang disebut atribut dan *method* atau operasi. Simbol-simbol yang terdapat pada *class diagram* dapat dilihat pada Tabel 2.4.

Berikut penjelasan atribut dan *method* :

1. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau *method* adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Tabel 2.4 Simbol-Simbol *Class Diagram*

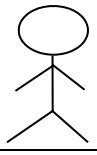
Simbol	**Keterangan**
Kelas 	Kelas pada struktur sistem
Antarmuka/*interface* 	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi/*association* 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah/ *directed association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi 	Relasi antar kelas dengan makna generalisasi - spesialisasi (umum-khusus)

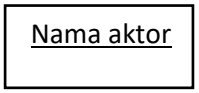
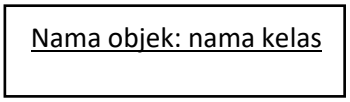
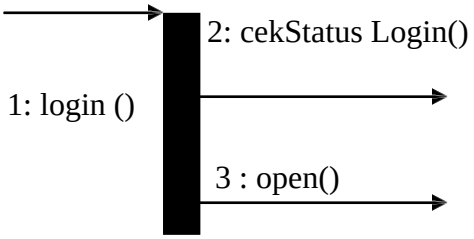
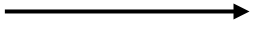
Kebergantungan/*dependensi* 	Relasi antar kelas dengan makna kebergantungan antar kelas
Agrgasi/*aggregation* 	Relasi antar kelas dengan makna semua-bagian (*whole-part*)

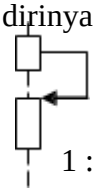
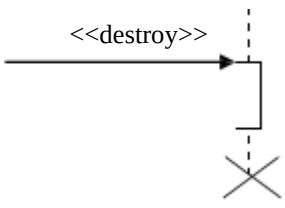
2.9.7 *Squence Diagram*

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dengan *massage* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*. Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup dalam diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak. Simbol-simbol yang terdapat pada diagram sekuen dapat dilihat pada Tabel 2.5.

Tabel 2.5 Simbol-Simbol *Squence Diagram*

Simbol	**Keterangan**
Aktor 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi

Atau  Tanpa waktu aktif	yang akan dibuat itu sendiri, jadi walaupun simbol dari akor adalah gambar orang, tapi akor belum tentu merupakan orang, biasanya dinyatakan dalam menggunakan kata benda di awal frase nama aktor.
Garis hidup /*lifeline* 	Menyataan kehidupan suatu objek
Objek  	Menyetakan objek yang berinteraksi peran
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semuanya yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya  Maka cek status login() dan open() dilakukan didalam metode login(). Akor tidak memiliki waktu aktif.
Pesan tipe *create*  	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat

Pesan tipe *call* 1 : nama_metode()	Menyatakan suatu objek memanggil oprasi/metode yang ada pada objek lain atau dirinya sendiri,  1 : nama_metode() Arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/ metode, maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.
Pesan tipe *send* 1 : masukan	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi/ ke objek lainnnya, arah panah mengarah pada objek yang menerima kembalian.
Pesan tipe *return* 1: keluaran	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalianke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.
Pesan tipe *destroy*  	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaliknya jika ada *create* maka ada D*estroy*