BAB 2
TINJAUAN PUSTAKA

2.1 Dinas Koperasi dan UMKM Bandung

2.1.1 OPOP (One Pesantren One Product)

OPOP (One Pesantren One Product) adalah sebuah program dari Pemprov Jabar bersama Dinas KUKM (Dinas Koperasi Usaha Mikro Kecil dan Menengah) Provinsi Jawa Barat, memastikan seluruh Pondok Pesantren di Jawa Barat dapat memperoleh akses atas program pemerintah dalam sektor pemberdayaan ekonomi, teknologi dan produksi yang efisien, tepat serta modern di era digital saat ini. Saat ini, sudah ada 1565 lebih pesantren di Jawa Barat yang terdaftar dalam progam OPOP. Program OPOP bertujuan untuk menciptakan kemandirian umat melalui para santri, masyarakat dan Pondok Pesantren itu sendiri, agar mampu mandiri secara ekonomi, sosial dan juga untuk memacu pengembangan skill, teknologi produksi, distribusi, pemasaran melalui sebuah pendekatan inovatif dan strategis. OPOP juga diadakan agar produk pesantren bisa menyaingi pengusaha-pengusaha lainnya.[1]

2.1.1 Balai Pelatihan Tenaga Koperasi dan Usaha Mikro, Kecil, dan Menengah (BALATKOP) Jawa Barat

Balai Pelatihan Tenaga Koperasi dan Usaha Mikro, Kecil dan Menengah (BALATKOP) adalah Unit Pelaksana Teknis Daerah (UPTD) yang berperan sebagai klinik bisnis dalam menyelesaikan permasalahan Koperasi dan UMKM di Jawa Barat. BALATKOP ini berdiri dibawah naungan Dinas Koperasi dan UMKM. BALATKOP menyediakan sarana pelatihan dan magang untuk pesantren yang memiliki usaha bisnis agar usaha pesantren bisa bersaing dengan pengusaha-pengusaha lainnya. Lokasi BALATKOP berada di Jl. Soekarno Hatta 708, Km.11, Gede Bage, Bandung

2.2 Cloud Computing

Cloud Computing terdiri dari 2 kata, yaitu *cloud* dan *computing*. Yang dimana *cloud* berarti awan, sedangkan *computing* adalah komputasi.

Cloud Computing adalah evolusi selanjutnya dari *internet*. *Cloud Computing* biasanya melibatkan transfer, penyimpanan, dan pemrosesan informasi di infrastruktur 'provider', dimana tidak termasuk dalam aturan control 'pengguna'.

Konsep *Cloud Computing* terhubung dekat dengan Infrastructure as a Service, Platform as a Service (PaaS), dan Software as a Service. Keuntungan *Cloud Computing* salah satunya adalah mengurangi biaya perangkat keras yang digunakan oleh pengguna. Keuntungan lainnya yaitu kelincahannya, biaya yang lebih rendah, perangkat *independency*, lokasi bebas, dan skabilitas.[14] Karakteristik cloud computing ada 5, yaitu:[15]

1. Resource Pooling
2. Broad Access Network
3. Measured Service
4. Rapid Elasticity
5. Self Service

Cloud Computing mempunyai 3 layanan, yaitu:

1. Infrastructure as a Service

IaaS mengatur sumber daya komputasi sebagai layanan, termasuk virtualisasi komputer dan bandwidth untuk penyimpanan dan akses internet [16].

2. Platform as a Service

PaaS sama seperti IaaS tetapi PaaS mengatur juga OS (Operating System) dan membangun suatu aplikasi di suatu platform yang dapat dikustomisasi [17].

3. Software as a Service

SaaS merupakan layanan dimana pengguna tidak perlu mengatur cloud seperti layanan PaaS dan IaaS. Pengguna hanya cukup menggunakan aplikasi yang disediakan saja [18].

2.3 CodeIgniter

CodeIgniter adalah sebuah web *framework* berbasis PHP yang bersifat *open-source*. CodeIgniter menggunakan pola MVC (Model-View-Controller) yang memudahkan pengembang untuk membuat suatu aplikasi. CodeIgniter pertama kali dirilis pada tahun 2006 oleh Rick Ellis. Fungsi CodeIgniter adalah sebagai berikut:

1. Mempermudah pengembangan website.
2. Struktur program tertata dengan rapi, dari segi kode maupun struktur filenya.
3. Memberikan standar coding agar mudah dipahami.[19]

2.4 DigitalOcean

DigitalOcean adalah perusahaan dari Amerika Serikat yang menyediakan layanan infrastruktur cloud. Platform ini menyediakan layanan penyimpanan, pengelolaan, hingga skalasi aplikasi.

2.4.1 Droplets

DigitalOcean Droplets adalah *virtual machine* (VM) berbasis Linux yang berjalan di virtualisasi perangkat keras. Droplets mempunyai spesifikasi yang sama seperti VPS (*Virtual Private Server*). Droplets di DigitalOcean dapat digunakan untuk:

1. Membuat *virtual host* sehingga bisa diakses lebih dari satu website dalam satu *server*/*droplets*.
2. Menyimpan *database* MySQL.
3. Dapat dijadikan penyimpanan data supaya bisa diakses lebih cepat.

2.5 MySQL

MySQL adalah sebuah *database management system* yang bersifat *open-source* yang diciptakan oleh MySQL AB Swedia dan dikembangkan oleh Oracle Corporation. MySQL mampu mengirim dan menerima data secara cepat menggunakan perintah SQL (*Strucured Query Language*). MySQL tersedia di bawah lisensi GNU *General Public License* (GPL). Salah satu contohnya

sintaksnya adalah INSERT INTO users VALUES('Smith', 'John' , 'jsmith@mysite.com');. Sintaks sebelumnya melakukan *input* data ke tabel users.[20]

2.6 PHP

PHP adalah suatu bahasa pemrograman berbasis *back-end server* yang bertujuan untuk mengolah suatu data dan mengirimkannya kembali ke web menjadi kode HTML. PHP merupakan singkatan dari *Hypertext Preprocessor*. Salah satu keunggulannya adalah PHP bisa didapat dengan gratis dan mendukung OOP (Object Oriented Programming).[21]

2.7 Visual Studio Code

Visual Studio Code adalah program *source-code editor* yang dikembangkan oleh Microsoft. Visual Studio Code mempunyai ekstensi yang berguna untuk mempermudah pengembang dalam mengembangkan suatu aplikasi. Selain itu, Visual Studio Code mempunyai fitur IntelliSense, yaitu fitur yang menyorotkan sintaks dan melengkapi sintaks.[22]

2.8 UML (*Unified Modelling Language*)

UML adalah bahasa pemodelan yang distandarisasi meliputi integrasi diagram, dikembangkan untuk membantu sistem dan *software developer* untuk spesifikasi, visualisasi, konstruksi, dan dokumentasi sistem perangkat lunak.[23] UML digunakan saat membuat suatu sistem atau pemrograman dengan OOP (Object-Oriented Programming). UML memiliki beberapa diagram, yaitu class diagram, activity diagram, sequence diagram, dan use case diagram.

2.8.1 Use Case Diagram

Diagram *use case* adalah sebuah diagram yang mendeskripsikan aktor dan interaksi mereka terhadap sistem. Diagram *use case* memiliki 2 komponen, yaitu:

1. Aktor

Aktor adalah pengguna yang berinteraksi pada sistem. Aktor bisa saja orang, organisasi, atau diluar sistem yang berinteraksi dengan sistem.

2. Sistem

Sistem memiliki urutan aksi yang spesifik antara aktor dan sistem.

Use case memiliki simbol dan notasi. Berikut adalah simbol dan notasi dari diagram *use case*:

1. Associations

Associations adalah sebuah garis yang menghubungkan antara aktor dan *use case*.[24]

2. Generalization

Generalization mendefinisikan relasi antara dua aktor atau dua use case yang dimana mewarisi atau *override* sifat yang lainnya.

3. Depedency

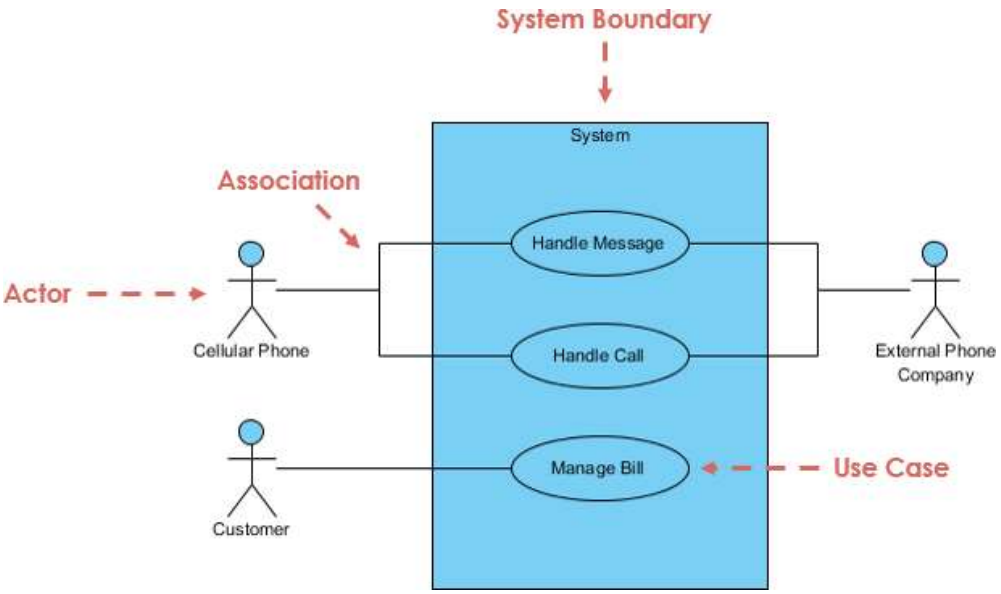
Depedency terbagi menjadi 2, yaitu *include* dan *extend*.

a. Include

Include digunakan untuk mengidentifikasi antara 2 hubungan *use case*, dimana *use case* yang satu akan memanggil *use case* yang lainnya. *Include* digunakan saat diperlukan oleh *use case* lainnya untuk menjalankan fungsinya.

b. Extend

Extend berfungsi untuk apabila ada pemanggilan, akan memerlukan kondisi tertentu maka akan berlaku juga *depedency*.[25]*Extend* juga dapat berdiri sendiri, lain hal dengan *include*.



Gambar 2.1 Use Case Diagram[26]

2.8.2 Class Diagram

Class Diagram merupakan tipe struktural diagram yang menjelaskan struktur sistem dengan menunjukkan struktur sistemnya dengan kelas, atribut, operasi, dan hubungan antar objek. Ada 3 macam notasi kelas, yaitu:

1. Nama Kelas

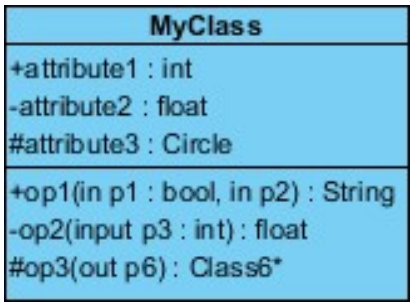
Nama kelas ditunjukkan di kolom pertama

2. Atribut Kelas

Atribut ditunjukkan di kolom kedua. Tipe atribut dibuat setelah titik dua.

3. Operasi Kelas (*Method*)

Operasi ditunjukkan di kolom ketiga.

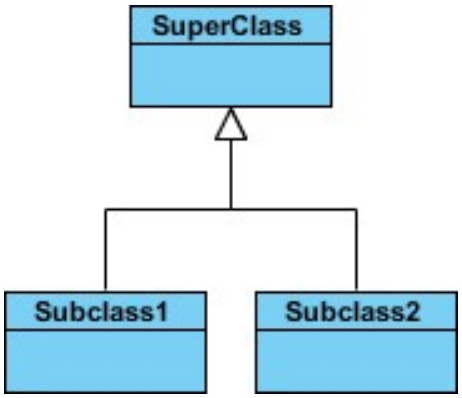


Gambar 2.2 Class Diagram

Diagram kelas mempunyai beberapa relasi, diantaranya:

1. *Inheritance* (Generalisasi)

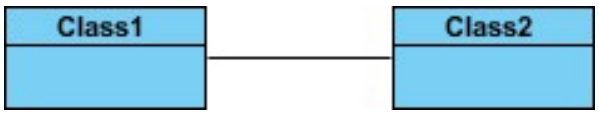
Inheritence adalah hubungan hirarki antar kelas. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metode kelas lainnya. Kebalikan dari pewarisan ialah generalisasi.[27]



Gambar 2.3 Inheritance Class Diagram

2. Asosiasi

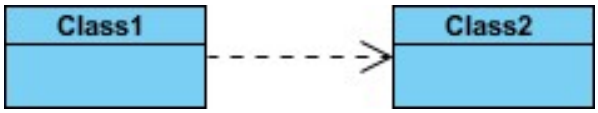
Asosiasi adalah hubungan antar *class*. Pada umumnya menggambarkan *class* yang memiliki atribut *class* yang harus mengetahui eksistensi *class* lain.



Gambar 2.4 Asosiasi

3. Depedency

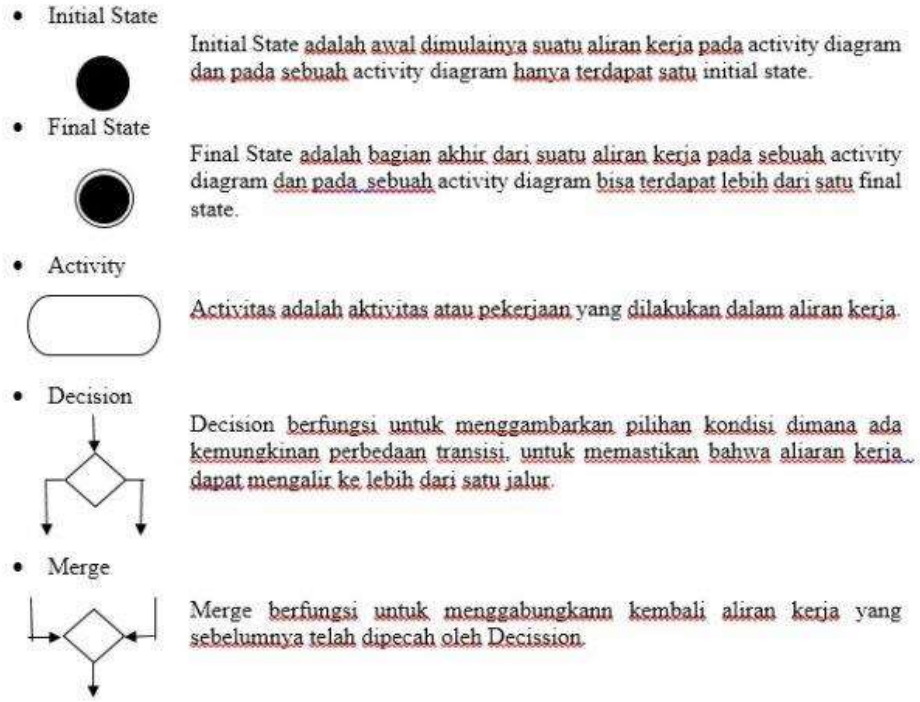
Depedency muncul antara dua *class* jika satu kelas ganti definisi maka *class* lain pun ikut ganti. *Class* satunya selalu bergantung dengan *class* lainnya.[27]



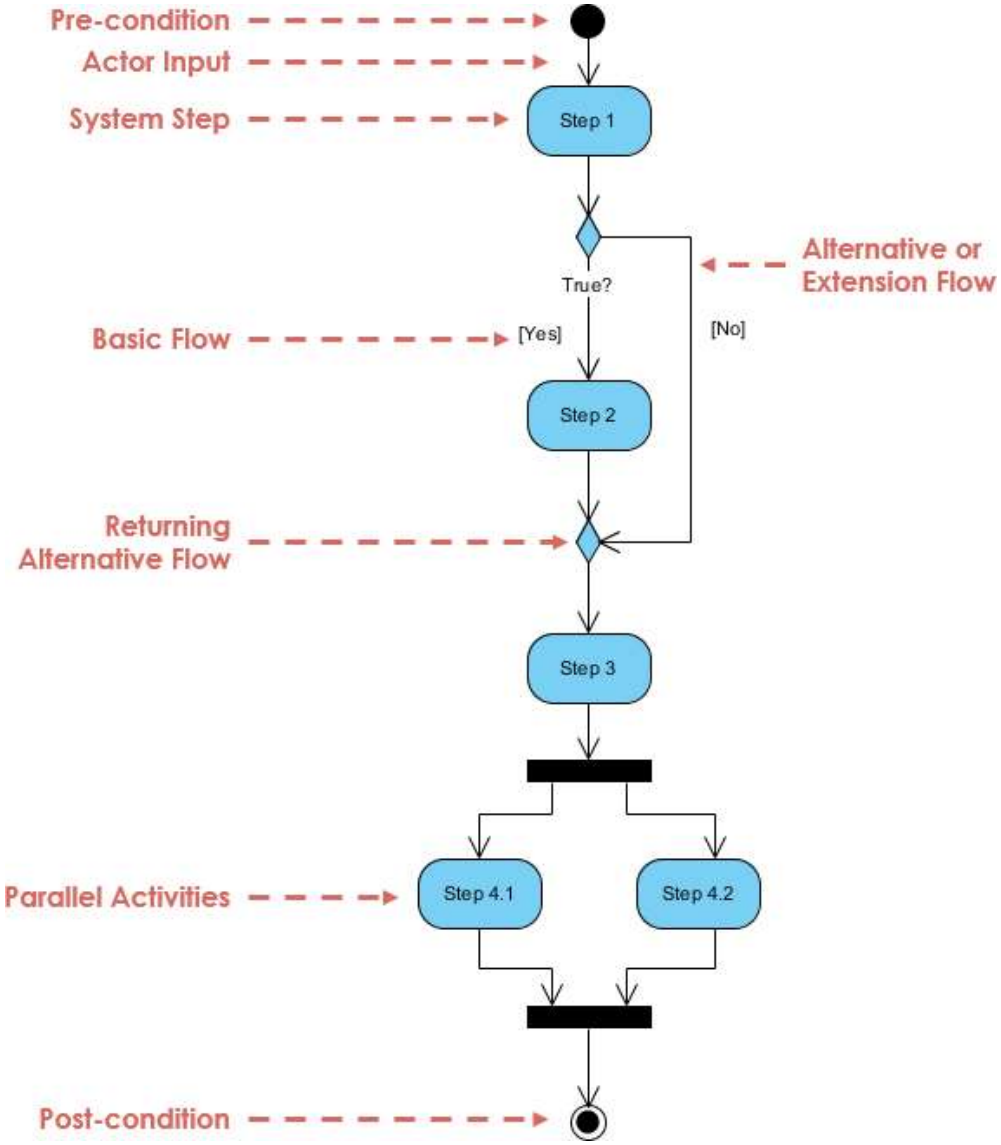
Gambar 2.5 *Depedency*

2.8.3 Activity Diagram

Diagram *activity* adalah diagram yang merancang aliran aktivitas atau alur kerja dalam sebuah sistem yang akan dijalankan. *Activity* diagram memiliki komponen dengan bentuk tertentu yang dihubungkan dengan arah panah. Fungsi *activity* diagram adalah untuk memperlihatkan urutan aliran proses pada sistem. *Activity* diagram terbentuk berdasarkan sebuah *use case*.[28] Komponen dari *activity* diagram adalah sebagai berikut:



Gambar 2.6 Komponen *Activity* Diagram



Gambar 2.7 Contoh *Activity* Diagram[29]

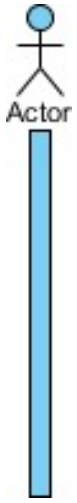
2.8.4 Sequence Diagram

Diagram *sequence* adalah sebuah diagram yang menggambarkan interaksi antara objek secara berurutan.[https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/] Diagram *sequence* berfokus pada waktu dan menunjukkan urutan interaksi secara visual dengan menggunakan sumbu vertikal diagram untuk melihat waktu pesan apa yang dikirim dan kapan. Tujuan dari *sequence* diagram adalah memodelkan interaksi antara objek dalam kolaborasi

yang mewujudkan *use case*. Berikut adalah notasi-notasi dalam diagram sequence:

1. Aktor

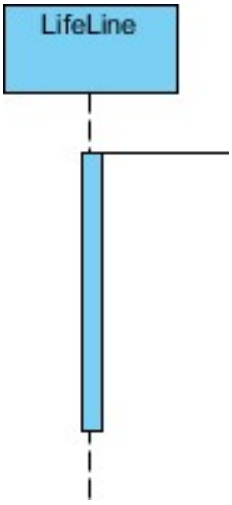
Aktor adalah peran yang dimainkan oleh entitas-entitas yang beinteraksi dengan objek.[https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/]



Gambar 2.8 Aktor

2. Lifeline

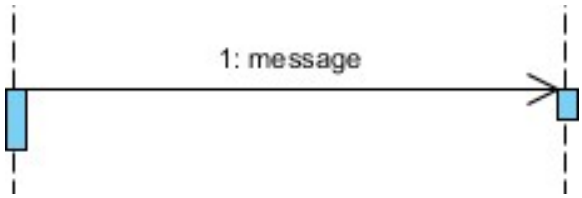
Lifeline adalah elemen yang menggambarkan individu dalam sequence diagram. Pada dasarnya tiap instansi dalam diagram *sequence* digambarkan oleh *lifeline*. Perbedaan antara *lifeline* dengan aktor adalah lifeline menggambarkan objek internal sementara aktor menggambarkan objek secara eksternal dalam sistem.



Gambar 2.9 *Lifeline*

3. Message

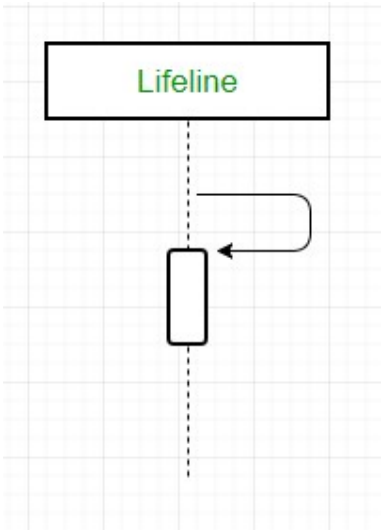
Message adalah sebuah pesan yang menggambarkan komunikasi antara interaksi *lifeline*.



Gambar 2.10 *Message*

4. Self Message

Self Message adalah sebuah pesan dimana objek mengirim pesan ke dirinya sendiri.



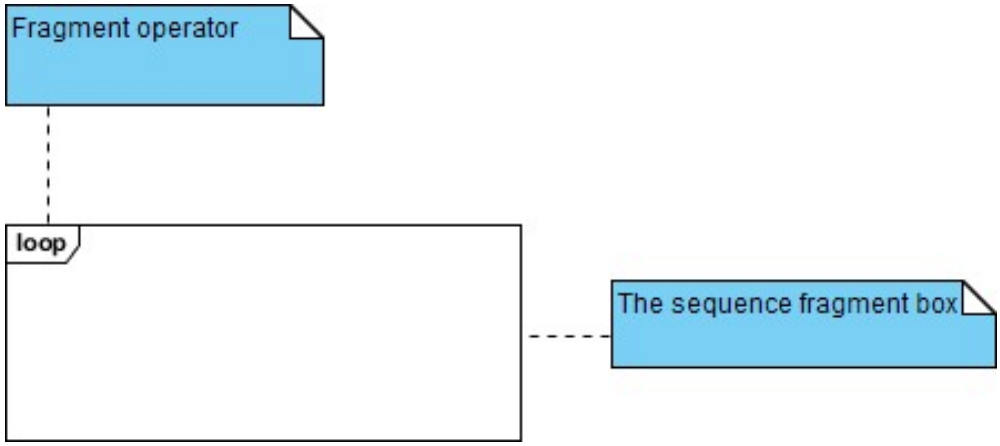
Gambar 2.11 *Self Message*

5. *Sequence Fragments*

Sequence Fragments adalah fragmen yang digambarkan dalam sebuah kotak yang membungkus sebagian interaksi dalam diagram *sequence*. Adapun operator-operator dari *sequence fragments* adalah sebagai berikut:

Tabel 2.1 Operator Sequence Fragment

Operator	Jenis fragmen
alt	Alternatif: Hanya satu kondisi yang bernilai *true* akan dieksekusi.
opt	Opsional: Fragmen dimana hanya mengeksekusi jika kondisi yang disediakan benar.
par	*Parallel*: Tiap fragmen dijalankan dalam paralel.
loop	*Loop*: Fragment akan dieksekusi beberapa kali
region	*Critical Region*: Fragmen hanya mempunyai satu *thread* yang dijalankan sekaligus.
neg	*Negative*: Fragmen yang menujukkan interaksi yang salah
ref	*Reference*: Mengacu pada interaksi yang didefinisikan di diagram yang lain.
sd	*Sequence Diagram*: digunakan untuk mengelilingi seluruh *sequence diagram*



Gambar 2.12 ***Sequence Fragments***