

## **BAB 2**

### **TINJAUAN PUSTAKA**

#### **2.1 Coronavirus**

Diawal tahun 2020, dunia digemparkan dengan merebaknya virus baru yaitu coronavirus jenis baru (SARS-CoV-2) dan penyakitnya disebut *Coronavirus disease 2019* (COVID-19). Diketahui asal mula virus ini berasal dari Wuhan, Tiongkok. Ditemukan pada akhir Desember 2019.

Coronavirus merupakan virus RNA strain tunggal positif berkapsul dan tidak bersegmen. Coronavirus tergolong ordo Nidovirales keluarga Coronaviridae. Struktur coronavirus membentuk struktur seperti kubus dengan protein S berlokasi di permukaan virus. [3]

#### **2.2 Global Positioning System (GPS)**

GPS merupakan sistem navigasi dan penentu lokasi berbasis satelit dengan tingkat ketelitian tinggi. GPS juga telah dikembangkan dalam bentuk *smartphone* sehingga penggunaannya lebih mudah yang datanya digunakan untuk mengambil data koordinat dari masing-masing pengguna *smartphone*. [4]

Cara kerja GPS adalah suatu sistem yang memancarkan sebuah sinyal untuk mendeteksi suatu objek yang berada di dalam jangkauan satelit. Satelit tersebut akan mendapatkan sebuah informasi keberadaan objek-objek yang ada di bumi yang akhirnya informasi tersebut dapat diakses oleh semua orang yang mempunyai sistem GPS tersebut.

#### **2.3 Location Based Service (LBS)**

Sistem layanan berbasis lokasi atau yang biasa dikenal dengan *location based services* (LBS), menggabungkan antara proses dari layanan mobile dengan posisi geografis dari penggunaannya. LBS mampu mendeteksi lokasi pengguna berada sehingga dapat memberikan layanan sesuai dengan lokasi pengguna tersebut. Lokasi geografis pengguna ditentukan dengan menggunakan layanan terpisah seperti misalnya *Global Positioning System* (GPS), yang mana GPS adalah

sistem yang berfungsi sebagai sistem navigasi global yang dapat menerima informasi dari sistem satelit. Satelit GPS ini memancarkan sinyal yang memungkinkan penerima sinyal GPS untuk mendapatkan informasi berupa posisi, kecepatan, maupun waktu. [7]

## **2.4 Android**

Android merupakan sistem operasi mobile berbasis linux yang dirancang untuk perangkat yang memiliki kemampuan layar sentuh seperti smartphone maupun tablet. Android dibeli Google pada tahun 2005 dan pada tahun 2008 tepatnya pada tanggal 22 Oktober 2008 diluncurkan smartphone berbasis Android pertama dengan nama HTC Dream. Dikarenakan Android bersifat open source perkembangannya sangat pesat sekali. Sekarang sistem operasi Android tidak hanya dapat berjalan pada smartphone maupun tablet saja tetapi dapat berjalan pada perangkat seperti laptop, TV maupun smart watch. [8]

## **2.5 Android Studio**

Android Studio adalah Lingkungan Pengembangan Terpadu – Integrated Development Environment (IDE) untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA [9]. Selain merupakan editor kode IntelliJ dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas pengembang saat membuat aplikasi Android, misalnya:

1. Sistem versi berbasis Gradle yang fleksibel.
2. Emulator yang cepat dan kaya fitur.
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android.
4. Instant Run untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru.
5. Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh.
6. Alat pengujian dan kerangka kerja yang ekstensif.

7. Alat Lint untuk meningkatkan kinerja, kegunaan, kompatibilitas versi dan masalah-masalah lain.
8. Dukungan C++ dan NDK
9. Dukungan bawaan untuk Google Cloud Platform, mempermudah pengintegrasian Google Cloud Messaging dan App Engine.

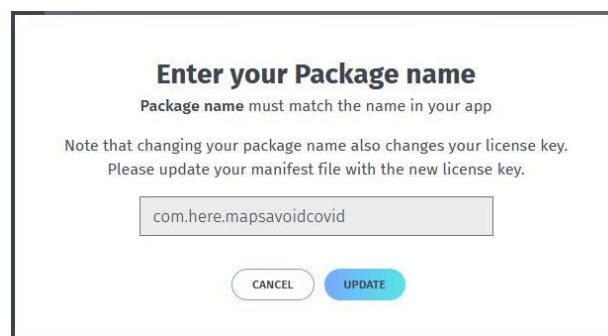
Tujuan dibuatnya Android Studio adalah untuk mempercepat pengembangan dan membantu pengembang membuat aplikasi berkualitas tinggi untuk setiap perangkat Android. Menawarkan alat bantu yang dibuat khusus untuk pengembang Android, meliputi pengeditan kode yang lengkap, debugging, pengujian dan alat pembuatan profil.

## 2.6 HERE Maps API

HERE adalah unit bisnis Nokia yang menyatukan pemetaan dan posisi suatu lokasi menjadi satu merk. Teknologi HERE didasarkan pada model *cloud-computing* di mana data dan layanan lokasi disimpan pada server internet sehingga pengguna dapat mengakses dengan perangkat yang mereka gunakan.

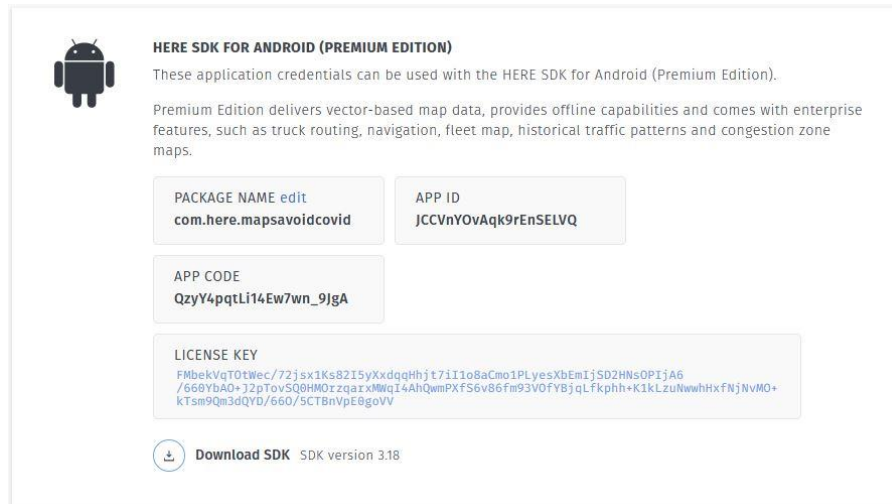
Dalam sarananya menyediakan fasilitas GIS, HERE Maps juga menyediakan API untuk penggunaannya agar bisa membuat peta sendiri di halaman mereka dengan peta dasar HERE Map lengkap dengan perangkatnya. HERE Maps API terdiri dari tiga produk yaitu JavaScript APIs, REST APIs, dan *Mobile SDK*. [5]

Untuk menggunakan API ini, pertama ialah harus memasukkan nama *package* aplikasi yang dibuat, bisa dilihat di Gambar 2.1 Memasukkan Nama Package.



**Gambar 2.1 Memasukkan Nama Package**

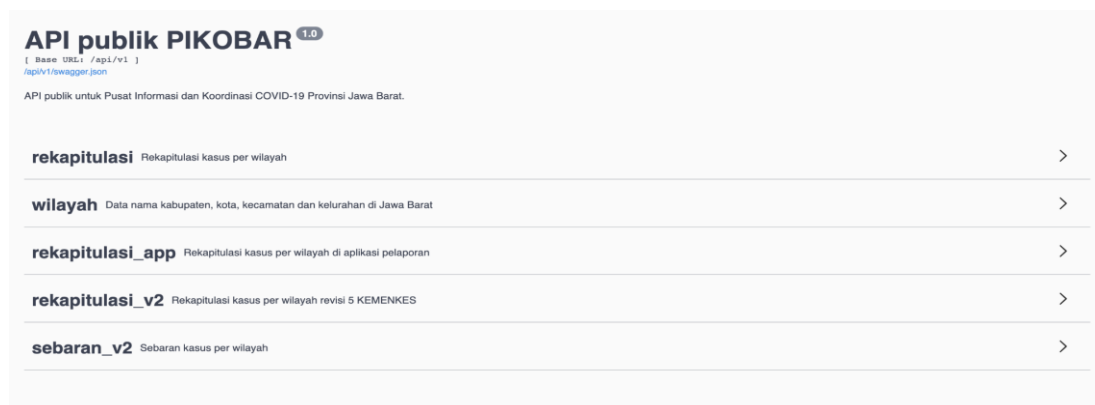
Setelah itu kita akan mendapatkan APP ID, APP CODE, dan LICENSE KEY. Lalu download SDK dari *link* yang telah disediakan, seperti dapat dilihat di Gambar 2.2 Kredensial Aplikasi.



**Gambar 2.2 Kredensial Aplikasi**

## 2.7 Pikobar

Pikobar adalah sebuah platform aplikasi yang dibuat oleh pemerintah Jawa Barat dengan kepanjangannya adalah Pusat Informasi dan Koordinasi Covid-19 Jawa Barat. Disana pengguna dapat mengakses data terupdate mengenai kasus Covid-19, selain itu ada juga fitur donasi dan fitur cari oksigen. Tersedia dalam versi *website* atau versi aplikasi android maupun iOS, terdapat juga publik api yang bisa digunakan untuk mengambil data kasus Covid-19.



**Gambar 2.3 API Publik Pikobar**

Ada pilihan jenis api; rekapitulasi, wilayah, rekapitulasi\_app, rekapitulasi\_v2, sebaran\_v2, bisa dimanfaatkan untuk mengambil data kota bandung dengan filter id kota nya yang sudah disediakan.

Request nya bisa di filter dengan banyak macam, yaitu berdasarkan tanggal, maksimal data, jenis stage, ataupun grup kan berdasarkan kabupaten/kota atau kelurahan.

## 2.8 Java

Java adalah bahasa pemrograman yang berorientasi objek (OOP) dan dapat dijalankan pada berbagai platform sistem operasi. Perkembangan java tidak hanya terfokus pada satu sistem operasi, tetapi juga dikembangkan untuk berbagai sistem operasi dan bersifat open source [10]. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana. Aplikasi-aplikasi berbasis Java umumnya dikompilasi ke dalam p-code (bytecode) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM).

Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (general purpose), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan.

Aplikasi Java mampu berjalan di beberapa platform sistem operasi yang berbeda, Java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun". Saat ini Java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web.

Java menggunakan model pengamanan tiga lapis (three-layer security model) untuk melindungi sistem dari untrusted Java code. Pertama, bytecode verifier membaca bytecode sebelum dijalankan dan menjamin bytecode memenuhi aturan-aturan dasar bahasa Java. Kedua, class loader menangani pemuatan kelas Java ke runtime interpreter. Ketiga, manajer keamanan menangani keamanan tingkat aplikasi dengan mengendalikan apakah program berhak mengakses sumber daya seperti sistem file, port jaringan, proses eksternal dan sistem window.

Java termasuk bahasa Multithreading. Thread adalah untuk menyatakan program komputer melakukan lebih dari satu tugas di satu waktu yang sama. Java menyediakan kelas untuk menulis program multithreaded, program mempunyai lebih dari satu thread eksekusi pada saat yang sama sehingga memungkinkan program menangani beberapa tugas secara konkuren.

Program Java melakukan garbage collection yang berarti program tidak perlu menghapus sendiri obyek-obyek yang tidak digunakan lagi. Fasilitas ini mengurangi beban pengelolaan memori oleh pemrogram dan mengurangi atau mengeliminasi sumber kesalahan terbesar yang terdapat di bahasa yang memungkinkan alokasi dinamis.

Java mempunyai mekanisme exception-handling yang ampuh. Exception handling menyediakan cara untuk memisahkan antara bagian penanganan kesalahan dengan bagian kode normal sehingga menuntun ke struktur kode program yang lebih bersih dan menjadikan aplikasi lebih tegar. Ketika kesalahan yang serius ditemukan, program Java menciptakan exception. Exception dapat ditangkap dan dikelola program tanpa resiko membuat sistem menjadi down. Program Java mendukung native method yaitu fungsi ditulis di bahasa lain, biasanya C/C++. Dukungan native method memungkinkan pemrogram menulis fungsi yang dapat dieksekusi lebih cepat dibanding fungsi ekuivalen di java. Native method secara dinamis akan di-link ke program java, yaitu diasosiasikan dengan program saat berjalan.

Adapun kelebihan dari Java yaitu:

Java mempunyai beberapa keunggulan dibandingkan dengan Bahasa pemrograman lainnya. Keunggulan bahasa pemrograman Java antara lain:

1. Berorientasi obyek

Java adalah bahasa pemrograman yang berorientasi pada obyek. Java membagi program menjadi obyek - obyek serta memodelkan sifat dan tingkah laku masing-masing dalam menyelesaikan suatu masalah.

2. Java bersifat multiplatform

Java dirancang untuk mendukung aplikasi yang dapat beroperasi di lingkungan jaringan berbeda. Untuk mengakomodasi hal tersebut, Java compiler membangkitkan bytecodes (sebuah format yang tidak tergantung pada arsitektur

tertentu yang didesain untuk mengirimkan kode ke banyak platform perangkat keras dan perangkat lunak secara efisien). Java dapat dijalankan oleh banyak platform seperti Linux, Unix, Windows, Solaris, maupun Mac.

### 3. Java bersifat multithread

Multithreading adalah kemampuan suatu program komputer untuk mengerjakan beberapa proses dalam suatu waktu. Thread dalam Java memiliki kemampuan untuk memanfaatkan kelebihan multi prosessor apabila sistem operasi yang digunakan mendukung multi prosessor.

### 4. Dapat didistribusi dengan mudah

Java memiliki library rutin yang lengkap untuk dirangkai pada protocol TCP/IP (seperti HTTP dan FTP) dengan mudah. Kemampuan networking Java lebih kuat dan lebih mudah digunakan. Java memudahkan tugas pemrograman jaringan yang sulit seperti membuka dan mengakses sebuah socket koneksi. Java juga memudahkan pembuatan CGI (Common Gateway Interface).

### 5. Bersifat dinamis

Java dirancang untuk beradaptasi dengan lingkungan yang sedang berkembang. Java bersifat dinamis dalam tahap linking. Class yang ada dapat di-link sebatas yang diperlukan, apabila diperlukan modul kode yang baru dapat di-link dari beberapa sumber, bahkan dari sumber dalam jaringan Internet.

Kekurangan Java yaitu:

#### 1. Tulis sekali

Masih ada beberapa hal yang tidak kompatibel antara platform satu dengan platform lain. Untuk J2SE, misalnya SWT-AWT bridge yang sampai sekarang tidak berfungsi pada Mac OS X.

#### 2. Mudah didekompilasi

Dekompilasi adalah proses membalikkan dari kode jadi menjadi kode sumber. Ini dimungkinkan karena kode jadi Java merupakan bytecode yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, metode, dan tipe data. Hal yang sama juga terjadi pada Microsoft .NET Platform. Dengan demikian, algoritma yang digunakan program akan lebih sulit disembunyikan dan mudah dibajak/di-reverse-engineer.

#### 3. Penggunaan memori yang banyak

Penggunaan memori untuk program berbasis Java jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan Pascal (lebih spesifik lagi, Delphi dan Object Pascal). Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru (karena trend memori terpasang makin murah), tetapi menjadi masalah bagi mereka yang masih harus berurusan dengan mesin komputer berumur lebih dari 4 tahun.

## **2.9 Object Oriented Programming**

*Object oriented programming* adalah metode yang diimplementasikan dengan pengorganisasian atas kerjasama dari sekumpulan objek, dimana masing-masing objek tersebut adalah sebuah *instance* dari kelas dan semua kelas adalah anggota dari hirarki yang direlasikan melalui pewarisan dari *class* utamanya (Booch *et al*, 2005). Dari pernyataan tersebut dapat ditarik kesimpulan bahwa object oriented programming mempunyai ciri sebagai berikut [6]

1. *Object oriented programming* menggunakan objek bukan algoritma sebagai dasar logika.
2. Masing-masing objek adalah *instance* dari sebuah kelas
3. Kelas bisa direlasikan dengan kelas lainnya dengan menggunakan relasi pewarisan.

### **2.10 Object Oriented Design**

*Object Oriented Design* adalah metode desain yang menggunakan pemecahan proses berdasarkan object oriented dan notasi untuk menggambarkan *logical* dan *physical* baik *dynamic* maupun *static* model (Booch *et al*, 2005). *Object oriented design* menggunakan abstraksi kelas dan objek sebagai komponen pembentuk sistem dan menggambarkan pemodelan dari abstraksi tersebut dengan model yang mendukung *object oriented*.

### **2.11 Object Oriented Analysis**

*Object oriented analysis* adalah metode analisis yang menggunakan perspektif kelas dan objek dalam menentukan kebutuhan sistem (Booch *et al*, 2005). Pemodelan berorientasi objek mempunyai beberapa karakteristik yaitu



1. *Encapsulation*

Informasi suatu objek tidak bisa diakses sepenuhnya oleh objek lain. Ada informasi-informasi dan data tertentu yang tidak boleh diakses (*private*).

2. *Inheritance*

Teknik yang menyatakan bahwa anak dari objek akan mewarisi data/*property* dan metode dari induknya langsung. Sehingga objek anak mempunyai seluruh *property* dan *method* dari induknya.

3. *Polymorphism*

Konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku yang sama. Operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda

## **2.12 Unified Modelling Language (UML)**

Unified Modelling Language (UML) adalah sebuah bahasa pemodelan visual yang digunakan untuk memvisualisasikan, menspesifikasikan, membangun dan mendokumentasikan artefak sistem perangkat lunak [11]. UML digunakan untuk memahami, mendesain, mengeksplorasi, mengkonfigurasi, memelihara, dan mengontrol informasi tentang sistem. Hal Ini dimaksudkan untuk digunakan dengan semua metode pengembangan, tahapan siklus hidup, domain aplikasi, dan media.


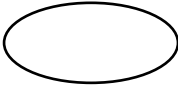


UML bukanlah bahasa pemrograman, melainkan alat yang dapat menyediakan generator kode dari UML ke dalam berbagai bahasa pemrograman serta membangun model rekayasa balik dari program yang ada. UML adalah bahasa pemodelan diskrit, yang dimaksudkan untuk menjadi bahasa pemodelan umum untuk sistem diskrit seperti yang terbuat dari perangkat lunak, firmware, atau logika digital.




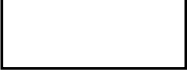
UML diadopsi dengan suara bulat oleh keanggotaan Object Management Group (OMG) sebagai standar pada November 1997. OMG memiliki tanggung jawab untuk pengembangan lebih lanjut dari standar UML. Bahkan sebelum adopsi terakhir, sejumlah buku diterbitkan yang menguraikan hal-hal penting dari UML.

### 2.13 Use case Diagram

*Use case diagram* dapat digunakan selama proses analisis untuk menangkap requirements sistem dan untuk memahami bagaimana sistem seharusnya bekerja. Selama tahap desain, usecase diagram menetapkan perilaku (*behavior*) sistem saat diimplementasikan. Dalam sebuah model mungkin terdapat satu atau beberapa usecase diagram. Tabel simbol usecase diagram dapat dilihat pada Tabel 2.1. Simbol *Use case Diagram*.

**Tabel 2.1 Simbol Use Case Diagram**

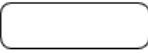

No.	Simbol	Nama	Keterangan
1		<i>Actor</i>	Merupakan orang, proses atau <i>system</i> lain yang berinteraksi dengan <i>system</i> yang akan dibuat. Jadi walaupun simbol aktor dalam diagram <i>usecase</i> berbentuk orang, namun aktor belum tentu orang.
2		<i>Use case</i>	Merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling berinteraksi atau bertukar pesan antar unit maupun aktor.
3		<i>Association</i>	Merupakan relasi yang digunakan untuk menggambarkan interaksi antara <i>usecase</i> dan aktor. Asosiasi juga menggambarkan berapa banyak objek lain yang bisa berinteraksi dengan suatu objek atau disebut <i>multiplicity</i> .
4		<i>Extend</i>	Menghubungkan antara satu objek dengan objek lain.






5		<i>Generalization</i>	Hubungan dimana objek berbagai perilaku dan struktur data dari objek yang ada di atasnya objek induk.
6		<i>Include</i>	Menspesifikasi bahwa <i>usecase</i> sumber secara eksplisit.
7		<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
8		<i>System Boundary</i>	Menspesifikasikan paket yang menampilkan sistem secara terbatas.

#### 2.14 Activity Diagram

Activity diagram memodelkan alur kerja (work flow) sebuah proses bisnis dan urutan aktivitas dalam suatu proses. Diagram ini sangat mirip dengan sebuah flowchart, karena user dapat memodelkan sebuah alur kerja dari satu aktivitas ke aktivitas lainnya atau dari satu aktivitas ke dalam keadaan sesaat (state). Diagram Activity berfokus pada aktifitas-aktifitas yang terjadi yang terkait dalam suatu proses tunggal. Jadi dengan kata lain, diagram ini menunjukkan bagaimana aktifitas-aktifitas tersebut bergantung satu sama lain. Tabel simbol *activity diagram* dapat dilihat pada Tabel 2.2 Simbol *Activity Diagram*.

**Tabel 2.2 Simbol Activity Diagram**


No.	Simbol	Nama	Keterangan
1.		<i>Action</i>	<i>State</i> dari sistem yang mencerminkan suatu aksi.
2.		<i>Decision</i>	Digunakan untuk menggambarkan suatu pengambilan keputusan / tindakan pada kondisi tertentu.


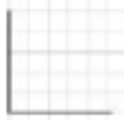


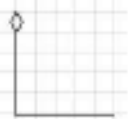

3.		<i>Initial Node</i>	Menunjukkan dimulainya suatu aktifitas.
4.		<i>Final Node</i>	Menunjukkan akhir dari aktifitas.
5.		<i>Fork Node</i>	Digunakan untuk menunjukkan aktifitas yang dilakukan secara paralel.
6.		<i>Join Node</i>	Digunakan untuk menunjukkan aktifitas yang digabungkan.
7.		<i>Swimlane</i>	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktifitas yang terjadi.

### 2.15 Class Diagram

Class diagram memberikan pandangan secara luas dari suatu sistem dengan menunjukkan kelas-kelasnya dan hubungan mereka. Class diagram bersifat statis, menggambarkan hubungan apa yang terjadi bukan apa yang terjadi jika mereka berhubungan. Class menggambarkan keadaan (atribut/properti). Class diagram menggambarkan struktur dan deskripsi class, package, atau objek beserta hubungan satu sama lain seperti pewarisan, asosiasi, dan lain-lain. Tabel simbol class diagram dapat dilihat pada Tabel 2.3 Simbol *Class Diagram*.

**Tabel 2.3 Simbol *Class Diagram***

No	Simbol	Nama	Keterangan
1.		<i>Class</i>	Blok pembangunan pada pemrograman berorientasi objek. Bagian atas adalah bagian dari class. Bagian tengah


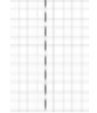
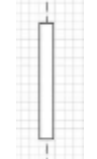



			mendefinisikan <i>property/atribut class</i> . Bagian akhir mendefinisikan <i>method - method</i> dari sebuah class.
2.		<i>Association</i>	<i>Relationship</i> paling umum antara 2 <i>class</i> , dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 <i>class</i> . Garis ini bisa melambangkan tipe-tipe <i>relationship</i> dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah relasi.
3.		<i>Compisition</i>	Jika sebuah <i>class</i> tidak bisa berdiri sendiri dan harus merupakan bagian dari <i>class</i> yang lain, maka <i>class</i> tersebut memiliki relasi <i>composition</i> terhadap <i>class</i> tempat bergantung tersebut.
4.		<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri ( <i>independent</i> ) akan mempengaruhi elemen yang bergantung pada elemen yang tidak mandiri ( <i>independent</i> ).
5.		<i>Aggregation</i>	Mengindikasikan keseluruhan bagian <i>relationship</i> dan biasanya disebut sebagai relasi.
6.		<i>Directed Association</i>	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya disertai <i>multiplicity</i>

### 2.16 Sequence Diagram

Sequence Diagram digunakan untuk menunjukkan aliran fungsionalitas dalam use case. Sequence Diagram merupakan salah satu diagram Interaction yang

menjelaskan bagaimana suatu operasi itu dilakukan; message (pesan) apa yang dikirim dan kapan pelaksanaannya. Diagram ini diatur berdasarkan waktu. Obyek obyek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut. Tabel simbol sequenece diagram dapat dilihat pada Tabel 2.4 Simbol *Sequence Diagram*.

**Tabel 2.4 Simbol *Sequence Diagram***

No.	Simbol	Nama	Keterangan
2.		partisipan	Merupakan instance dari sebuah class dan dituliskan tersusun secara horizontal.
3.		<i>Lifeline</i>	Mengindikasikan keberadaan sebuah objek dalam basis waktu.
4.		<i>Activation</i>	Mengindikasikan sebuah objek yang akan melakukan sebuah aksi.
5.		<i>Message</i>	Mengindikasikan komunikasi antar <i>obejct</i> .
6.		<i>Self-Message</i>	Mengindikasikan komunikasi kembali kedalam sebuah objek itu sendiri.
7.		<i>Sequence fragment</i>	Membungkus sebagian interaksi dengan diagram urutan yang memiliki beberapa tipe seperti <i>ref</i> , <i>assert</i> , <i>loop</i> , <i>break</i> , <i>alt</i> , <i>opt</i> , <i>neg</i> , <i>par</i> , <i>region</i>