



PAPER

PEMROGRAMAN BERORIENTASI OBJEK

PENGEMBANGAN BERIORIENTASI OBJEK DAN PENERAPAN DIAGRAM

DI SUSUN OLEH :

1.MOCHAMMAD IQBAL RONI SAPUTRA

2.ARSI DWI RISKY FEBRIAN

KATA PENGANTAR

Puji syukur diucapkan kehadirat Allah SWT atas segala rahmatNya sehingga makalah ini dapat tersusun sampai dengan selesai. Rasa terima kasih tidak juga saya berikan kepada pemilik website dan penulis karya di internet.

Tentu saja paper yang saya buat ini masih jauh dari kata sempurna, dikarenakan hal itu saya ingin meminta kritik dan saran agar kedepannya saya dapat menghasilkan paper yang lebih sempurna dari paper ini. Sebagai gambaran kecil isi dari paper ini, paper ini memuat tentang beberapa unsur yang berkaitan dengan pemrograman berorientasi obyek (PBO) atau yang biasa disebut object oriented programming (OOP).

Sekian kata pengantar paper ini, jika ada kata yang kurang berkenan atau materi yang kurang lengkap saya sebagai penulis memohon maaf yang sebesar-besarnya dan tidak lupa juga saya ingin meminta kritik dan saran yang dapat membantu membangun paper ini kedepannya.

Daftar Isi

KATA PENGANTAR.....	2
BAB I.....	5
METEDOLOGI BERORIENTASI OBYEK.....	5
1.1 Definisi Metodologi Berorientasi Obyek	5
1.2 Karateristik Metodologi Berorientasi Object	5
1.2.1 Encapsulation	5
1.2.2 Inheritance	6
1.2.3 Polymorphism.....	7
1.3 Tema Berorientasi Obyek.....	8
1.3.1 Perkapsulan (informasi hiding)	8
1.3.2 Menghubungkan data dan perilaku	8
1.3.3 Penggunaan Bersama(Reusability).....	9
1.3.4 Definisi Berorientasi Obyek.....	9
BAB II.....	9
STRUKTUR OBYEK	9
2.1 Pengertian Obyek	9
2.2 Attribute	10
2.3 Method.....	10
2.3.1 Return Method.....	10
2.3.2 Void Method	11
2.3.3 Static Method.....	11
BAB III.....	12
ORIENTED ANALYSIS AND DESIGN	12
3.1 Object Oriented Analysis(OOA).....	12
3.2 Object Oriented Design(OOD).....	13
BAB IV.....	14
CLASS DIAGRAM.....	14
4.1 Definisi Class Diagram	14
4.2 Manfaat Class Diagram.....	14
4.3 Komponen Dasar Pada Class Diagram	15
BAB V.....	17
SEQUENCE DIAGRAM.....	17

5.1	Definisi Sequence Diagram	17
5.2	Kegunaan Dan Tujuan Sequence Diagram.....	17
5.3	Komponen Sequence Diagram.....	17
5.4	Contoh Sequence Diagram	18
BAB VI.....		19
PENUTUP		19
6.1	Kesimpulan.....	19
6.2	Saran	19
BAB VII.....		20
DAFTAR PUSTAKA.....		20

BAB I

METEDOLOGI BERORIENTASI OBYEK

1.1 Definisi Metodologi Berorientasi Obyek

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metode berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek.

1.2 Karakteristik Metodologi Berorientasi Object

Metodologi pengembangan system berorientasi object di bagi menjadi 3 bagian antara lain:

1.2.1 Encapsulation

Maksud enkapsulasi disini adalah menyembunyikan detail implementasi sementara memusatkan pada antarmuka. Tujuannya adalah membuat sebuah abstraksi yang memaksa programmer berpikir secara konseptual. Biasanya, anggota – anggota data dari sebuah kelas terlihat oleh penggunaanya. Jika sebuah anggota data harus dibuat agar dapat diakses oleh client kelas, maka kelas tersebut menyediakan sebuah metode yang memeriksanya dan mengembalikan nilainya. Saat sebuah kelas mengekspos anggota data, ia dikatakan memecahkan enkapsulasi.

Contoh Dari Enkapsulasi :

```

public class Encapsulation {
    private String nama;
    private int nim;

    public String getNama() {
        return this.nama;
    }

    public void ModifNama(String nama) {
        this.nama = nama;
    }
}

public class main {
    public static void main(String[] args) {
        Encapsulation objek = new Encapsulation();
        objek.ModifNama("Muksalmina");
        System.out.println("Nama : " + objek.getNama());
    }
}

```

Gambar 1 Contoh Encapsulation

1.2.2 Inheritance

Pewarisan adalah keuntungan besar dalam pemrograman berbasis object karena suatu sifat atau method didefinisikan dalam superclass, sifat ini secara otomatis diwariskan dari semua subclasses. Jadi, Anda dapat menuliskan kode method hanya sekali dan mereka dapat digunakan oleh semua subclass.

```

package com.latihan;

public class Karyawan {
    String NIP;
    String nama;
    String jenisKelamin;


    public void masukKerja() {
        System.out.println("Masuk kerja");
    }

    public void beriNama(String nama) {
        this.nama = nama;
    }
}

```

Gambar 2 Inheritance

Dibuat class baru yaitu Dosen. Class Karyawan akan diwariskan ke class Dosen, dalam source code untuk membuat pewarisan digunakan keyword `extends` ketika menuliskan deklarasi class (lihat baris ke tiga di source berikut).



```

package com.latihan;

public class Dosen extends Karyawan {
}

```

Gambar 3 Inheritnce

Ketika class Dosen dibuat objectnya, object dosen dapat menggunakan atribut-atribut dan method-method yang dimiliki oleh kelas induk (class induk: Karyawan). Misalnya disini dosen dapat menggunakan method masukKerja() yang dideklarasikan di class Karyawan.



```

package com.latihan;

public class Main {
    public static void main(String[] args) {
        Dosen dosen = new Dosen();
        dosen.masukKerja();
    }
}

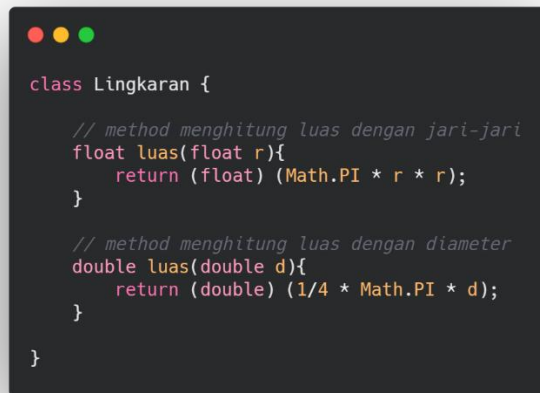
```

Gambar 4 Inheritance

1.2.3 Polymorphism

Polimorfisme dimungkinkan karena adanya mekanisme ikatan dinamis (dynamic binding). Ikatan dinamis adalah ikatan yang terjadi pada saat program dijalankan (run-time). Ikatan yang terjadi pada saat kompilasi disebut ikatan statis. Ikatan dinamis hanya dapat terjadi antara suatu objek dinamis dengan metode yang dinamis juga, dalam hal ini metode virtualnya (maya).

Contohnya misalkan kita memiliki class Lingkaran.java. Pada class ini terdapat method luas(). Nah, si method luas() ini bisa saja memiliki parameter yang berbeda. Misalnya kita ingin menghitung luas berdasarkan jari-jari (radius) atau diameter. Maka kita dapat membuat class-nya seperti ini:



```

class Lingkaran {
    // method menghitung luas dengan jari-jari
    float luas(float r){
        return (float) (Math.PI * r * r);
    }

    // method menghitung luas dengan diameter
    double luas(double d){
        return (double) (1/4 * Math.PI * d);
    }
}

```

Gambar 5 Polymorphism

Class `Lingkaran` memiliki dua method yang namanya sama, yakni `luas()`.

Tapi parameter dan tipe datanya berbeda dan juga isi atau rumus di dalamnya berbeda

1.3 Tema Berorientasi Obyek

Terdapat beberapa tema sebagai dasar teknologi berorientasi objek yang sangat menunjang dalam sistem berorientasi objek, antara lain:

1.3.1 Perkapsulan (informasi hiding)

Pengkapsulan terdiri dari pemisahan aspek eksternal dari suatu objek, dimana dapat diakses oleh objek lain. Pengkapsulan melindungi program dari saling keterkaitan sehingga kesempatan kecil mempunyai akibat penyimpangan.

1.3.2 Menghubungkan data dan perilaku

Kode yang bukan berorientasi objek menampilkan isi dari window harus membedakan tipe dari setiap bentuk, seperti poligon, lingkaran atau teks dan memanggil prosedur yang tepat untuk menampilkannya. Program berorientasi objek lebih sederhana meminta operasi draw pada setiap bentuk, dimana prosedur yang digunakan dibuat implisit oleh setiap objek berdasarkan kelasnya dan tidak perlu selalu mengulang pilihan prosedur setiap saat operasi dilakukan dalam program aplikasi. Pemeliharaan lebih mudah, karena prosedur yang dipanggil tidak perlu dimodifikasi setiap kelas baru ditambahkan dan dalam sistem berorientasi objek hirarki struktur data identik dengan pewarisan hirarki operasi.

1.3.3 Penggunaan Bersama(Reusability)

Teknik berorientasi objek menawarkan *reusability* yaitu penggunaan bersama untuk beberapa tingkat yang berbeda. Pewarisan struktur data dan perilaku memungkinkan penggunaan bersama antara beberapa subkelas yang sama tanpa *redundancy*.

1.3.4 Definisi Berorientasi Obyek

Secara spesifik, pengertian “berorientasi obyek” berarti bahwa kita mengorganisasikan perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. Hal ini yang membedakan dengan pemrograman konvensional dimana struktur data dan perilaku hanya berhubungan secara terpisah. Terdapat beberapa cara untuk menentukan karakteristik dalam pendekatan berorientasi objek, tetapi secara umum mencakup empat hal, yaitu:

1. Identifikasi
2. Klasifikasi
3. *Polymorphism*(polimorfisme)
4. *Inheritance*(pewarisan)

BAB II

STRUKTUR OBYEK

2.1 Pengertian Obyek

Object atau obyek adalah hasil cetak dari *class*, atau bisa disebut juga hasil konkret dari *class*. Sedangkan untuk pengertian *class* sendiri adalah *blueprint* atau cetak biru dari sebuah obyek(*object*), *class* sendiri hanya digunakan hanya untuk membuat kerangka dasar dari sebuah obyek.

```

Class Employee {
    private String Nama = "Akbar";

    public String getNama(){
        return Nama;
    }

    void sayNama(String SayNama){
        System.out.println("Hello My Name is " +SayNama);
    }
}

```

Gambar 6 Contoh class Employee

2.2 Atribute

Attribute atau biasa disebut juga sebagai properti adalah data yang terdapat dalam sebuah *class*. Atribut sebenarnya hanyalah variabel yang terletak di dalam sebuah *class*. Seluruh aturan dan tipe data yang biasa diinput ke dalam variabel, bisa juga diinput kedalam atribut. Aturan tata cara penamaan atribut sama dengan aturan penamaan variabel

```

Class Employee {
    String nama = "Akbar";
    String kelas = "12";
    String Jurusan = "RPL"
}

```

Gambar 7 Contoh Atribut

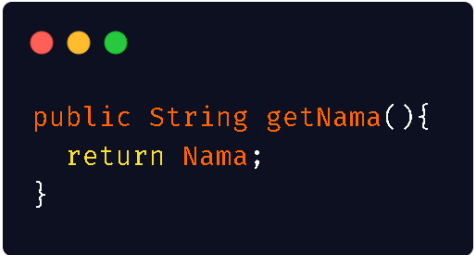
2.3 Method

Method(metode) merupakan Tindakan yang dilakukan di dalam *class*. *Method* pada dasarnya adalah *function* yang berada di dalam *class*. Seluruh fungsi dan sifat *function* bisa diterapkan ke dalam *method*, seperti argument/parameter, mengembalikan nilai(*return*), dll.

Pada dasarnya *method* dibagi menjadi tiga jenis yaitu *method* yang mengembalikan nilai, tanpa mengembalikan nilai(*void*), dan *static method*.

2.3.1 Return Method

Return adalah *method* yang mengembalikan nilai secara langsung atau sebuah nilai dari *variable*




```
public String getNama(){
    return Nama;
}
```

Gambar 8 Contoh Retrun Method

2.3.2 Void Method

Void adalah *method* yang tidak memiliki nilai kembali/*return*, biasanya digunakan tidak untuk mencari nilai dalam suatu operasi, untuk mendeklarasikannya kita harus menambahkan kata kunci *void*. Agar *method* tersebut dapat berjalan, kita perlu memanggilnya pada *method main*, kita harus membuat objek dari *class* yang kita gunakan terlebih dahulu, lalu panggil pada *method main*



```
class Employee{
    String Name;

    Employee(String Name){this.Name = name;}

    void sayHello(String name){
        System.out.println("Hallo " + name "Aku Employee" +this.Name);
    }
}
```

Gambar 9 Contoh Void Method

2.3.3 Static Method

Static merupakan suatu sifat yang bisa kita gunakan pada *variable* atau *method*, jika kita menggunakan *static* pada sebuah variabel ataupun *method*, untuk memanggilnya kita tidak perlu menginisialisasi suatu *class* maksudnya kita tidak perlu membuat sebuah objek dari *class*, berbeda dengan jenis *method* sebelumnya, pada jenis *method* sebelumnya seperti *Void* dan *Return* kita harus membuat objek dari *class* terlebih dahulu untuk memanggil *method* tersebut.



```

Void sayNama(String sayNama){
    System.out.println("Halo Namaku " +sayNama + "dan Umurku "+vUmur());
}

static int vUmur(){
    umur = tahun_s - tahun_l;
    return umur;
}

```

Gambar 10 Contoh Static Method

BAB III

ORIENTED ANALYSIS AND DESIGN

3.1 Object Oriented Analysis(OOA)

Object oriented analysis (OOA) merupakan metode analisis yang memeriksa requirements (syarat/ keperluan yang harus dipenuhi oleh sistem) dari sudut pandang kelas – kelas dan objek – objek yang ditemui dalam ruang lingkup permasalahan. OOA mempelajari permasalahan dengan menspesifikasikannya atau mengobservasi permasalahan tersebut dengan menggunakan metode berorientasi objek. Biasanya analisa sistem dimulai dengan adanya dokumen permintaan yang diperoleh dari semua pihak yang berkepentingan. Analisa ini sebaiknya dilakukan oleh orang-orang yang benar-benar memahami implementasi sistem yang berbasis atau berorientasi objek, karena tanpa pemahaman itu maka sistem yang dihasilkan bisa jadi tidak realistis jika di implementasikan dengan berbasis objek.

```

public class Siswa{
    String nama,jurusan;
    int kelas;

    Siswa(String vNama, String vJurusan, int vkelas){
        this.nama = vNama;
        this.kelas = vKelas;
        this.jurusan = vJurusan;
    }

    void sayHello(){
        System.out.println("Perkenalkan nama saya" +this.nama+"saya duduk di kelas"
            +this.kelas+"dan saya juga masuk jurusan "+this.jurusan);
    }
}

class SiswaApp{
    public static void main (String args){
        Siswa siswa = new Siswa(vnama"Arsi", vJurusan"RPL", vKelas"12");

        siswa.sayHello();
    }
}

```

Gambar 11 Contoh object dan blueprint(class)

Pada gambar 11 terdapat object 'Siswa' yang merupakan cetak biru dari class 'Siswa'. Class 'Siswa' memiliki atribut nama dan jurusan yang bertipe data string, dan ada juga atribut kelas yang bertipe data integer (*int*). Di dalam object Siswa juga terdapat sebuah method yang bernama 'sayHello'.

3.2 Object Oriented Design(OOD)

Object-oriented design adalah metoda untuk mengarahkan arsitektur perangkat lunak yang didasarkan pada manipulasi objek-objek sistem atau subsistem. Model kebutuhan-kebutuhan yang dibuat pada fase analisis diperkaya dalam fase perancangan. Kadang-kadang ditambahkan lebih banyak lagi atribut dan pelayanan dan ditambahkan antarmuka obyek-obyek.

Object Oriented Design bertujuan untuk mengoptimalkan *maintainability*, *reusability*, *enhancebility* dan *reliability*.

BAB IV

CLASS DIAGRAM

4.1 Definisi Class Diagram

Class Diagram adalah salah satu jenis diagram yang paling berguna di UML, hal ini karena dapat dengan jelas memetakan struktur sistem tertentu dengan memodelkan kelas, atribut, operasi serta hubungan antar objek.

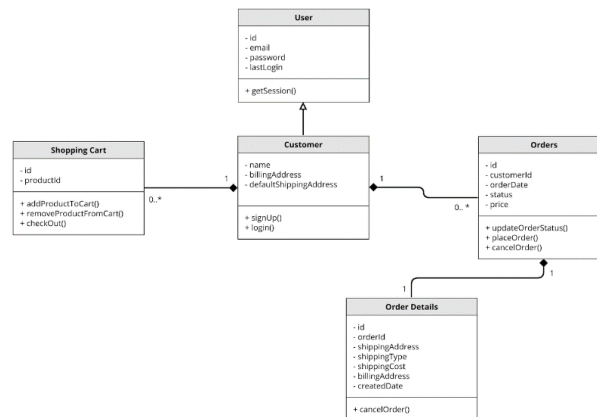
Class Diagram menggambarkan serta deskripsi atau penggambaran dari class, atribut, dan objek disamping itu juga hubungan satu sama lain seperti pewarisan, containmet, asosiasi dan lainnya.

4.2 Manfaat Class Diagram

1. Menggambarkan suatu model data untuk sistem informasi, tidak peduli apakah model data tersebut sederhana maupun kompleks.
2. Dengan mempelajari class diagram maka akan meningkatkan pemahaman mengenai gambaran umum skema dari suatu aplikasi.
3. Mampu menyatakan secara visual akan kebutuhan spesifik suatu informasi serta dapat menyebarkan informasi tersebut ke seluruh bisnis.

4. Dengan Class Diagram dapat dibuat bagan secara jelas dan terperinci dengan cara memperhatikan kode spesifik apa saja yang dibutuhkan suatu program sehingga mampu mengimplementasikannya ke struktur yang digambarkan.
5. Class Diagram mampu memberikan penggambaran implementasi-independen dari suatu jenis sistem yang digunakan, kemudian dilewatkan di antara berbagai komponen-komponennya.

4.3 Komponen Dasar Pada Class Diagram



Gambar 12 Contoh Class Diagram

- Bagian atas Berisi Nama kelas, bagian ini selalu diperlukan, baik ketika berbicara tentang classifier atau objek.
- Bagian tengah Bagian tengah berisi Atribut-attribut kelas, Atribut dapat menjelaskan rentang nilai sifat tersebut.
- Bagian ini adalah komponen *class diagram* yang menyertakan operasi *class* yang ditampilkan dalam format daftar. Sehingga, setiap operasi mengambil barisnya sendiri. Komponen ini juga menggambarkan bagaimana masing-masing class berinteraksi dengan data.
- Komponen Tambahan
 - *Class*, yang merepresentasikan obyek atau sekumpulan obyek yang memiliki persamaan struktur.
 - *Signals*, yaitu simbol yang mewakili komunikasi asinkron satu arah antara objek aktif.
 - Tipe data yang terbentuk dengan melakukan pengklasifikasi yang menentukan nilai data. Tipe data ini dapat menghasilkan tipe primitif dan pencacahan.
 - *Packages* yang dirancang untuk mengatur pengklasifikasi terkait dalam diagram. Komponen ini dilambangkan dengan bentuk persegi panjang.

- *Interface*, yaitu sekumpulan atribut yang mendefinisikan sekumpulan perilaku yang kohesif. Komponen ini mirip dengan *class*, namun harus memiliki setidaknya satu *class* untuk mengimplementasikannya.
- *Enumerations*, yang menggambarkan tipe data yang ditentukan pengguna. Komponen *enumerations* mencakup kelompok pengenalan yang mewakili nilai-nilai pencacahan.
- Objek, adalah item yang dapat ditambahkan ke dalam *class diagram* untuk mewakili contoh konkret atau prototipe.

BAB V

SEQUENCE DIAGRAM

5.1 Definisi Sequence Diagram

Diagram sequence merupakan salah satu yang menjelaskan bagaimana suatu operasi itu dilakukan, *message* (pesan) apa yang dikirim dan kapan pelaksanaannya. Diagram ini diatur berdasarkan waktu. Objek-objek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut.

Diagram sequence menampilkan interaksi antar objek dalam dua dimensi. Dimensi vertikal adalah poros waktu, dimana waktu berjalan ke arah bawah. Sedangkan dimei horizontal merepresentasikan objek-objek individual. Tiap objek (termasuk *actor*) tersebut mempunyai waktu aktif yang direpresentasikan dengan kolom vertikal yang disebut dengan *lifeline*. Pesan (*message*) direpresentasikan sebagai panah dari satu *lifeline* ke *lifeline* yang lain. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, messagnse akan dipetakan menjadi operasi/metoda dari *class*.

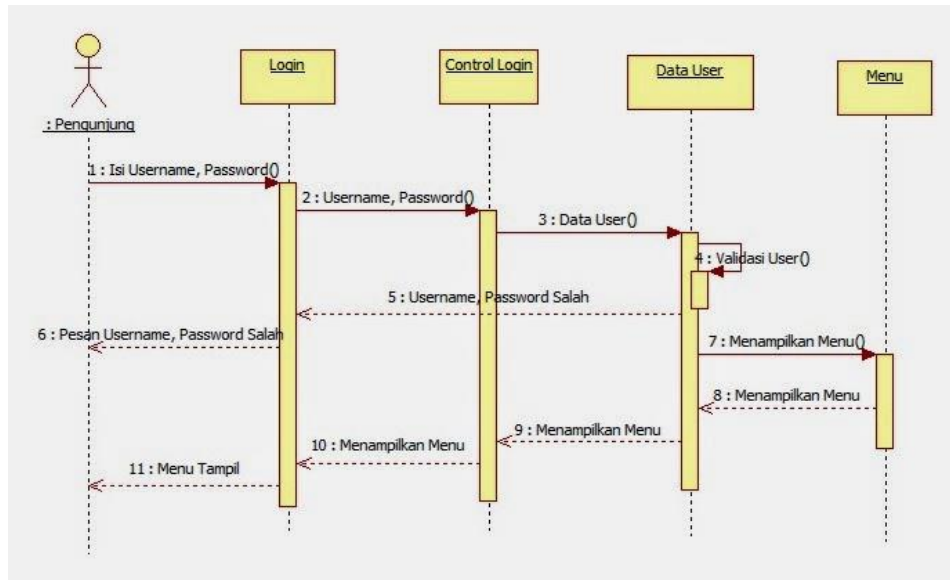
5.2 Kegunaan Dan Tujuan Sequence Diagram

1. Menganalisa, mendesain dan memfokuskan pada identifikasi sebuah metode yang digunakan sistem.
2. Sebagai alat untuk mengomunikasikan kebutuhan requerment kepada bagian teknis, sebab diagram ini lebih muda untuk dibaca dan diimplementasikan.
3. Salah satu jenis diagram yang sangat cocok digunakan untuk mengembangkan model deskripsi use case menjadi sebuah spesifikasi *design*.
4. Sequence diagram ini digunakan untuk menggambarkan dan memodelkan use case.
5. Berguna untuk memodelkan sebuah logika dari sebuah *method* operasi, *function* atauoun prosedur.
6. Dugunakan untuk memodelkan logika dari service.

5.3 Komponen Sequence Diagram

1. Object
2. Object adalah komponen berbentuk kotak yang mewakili sebuah class atau object. Mereka mendemonstrasikan bagaimana sebuah object berperilaku pada sebuah system.
3. Activation boxes
4. Activation boxes adalah komponen yang berbentuk persegi panjang yang menggambarkan waktu yang diperlukan sebuah object untuk menyelesaikan tugas. Lebih lama waktu yang diperlukan, maka activation boxes akan lebih panjang.
5. Actors
6. Actors adalah komponen yang berbentuk stick figure. Komponen yang mewakili seorang pengguna yang berinteraksi dengan system.
7. Lifeline
8. Lifeline adalah komponen yang berbentuk garis putus-putus. Lifeline biasanya memuat kotak yang berisi nama dari sebuah object. Berfungsi menggambarkan aktifitas dari object.

5.4 Contoh Sequence Diagram



Gambar 13 Contoh Sequence Diagram

Penjelasan:

Pada Sequence Diagram diatas,bias dilihat bahwa yang menjadi actor adalah pengunjung .Activation boxes biasanya memiliki garis yang memberitahu aktifitas yang terjadi ketika actor atau object berinteraksi ke object lain.

BAB VI

PENUTUP

6.1 Kesimpulan

Dari hasil pembuatan paper diatas saya dapat menyimpulkan beberapa hal

Metodologi berorientasi obyek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya.

Struktur obyek, obyek dapat dikatakan sebagai class apabila sudah memenuhi semua kriteria dari sebuah class, dari hasil pencarian yang sudah saya lakukan terdapat 2 struktur obyek yakni attributes(properties) dan method.

Object oriented analysis and design merupakan metode analisis yang memeriksa requirements dari sudut pandang kelas-kelas dan objek yang ditemui dalam ruang lingkup permasalahan yang mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem.

Class diagram adalah salah satu jenis diagram struktur statis dalam UML yang menggambarkan struktur sistem dengan menunjukkan sistem class, atributnya, metode, dan hubungan antar objek.

Sequence diagram merupakan salah satu yang menjelaskan bagaimana suatu operasi itu dilakukan, message (pesan) apa yang dikirim dan kapan pelaksanaannya. Diagram ini diatur berdasarkan waktu. Objek-objek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut.

6.2 Saran

Untuk pembelajaran ini kita tidak hanya belajar teori saja namun praktik/percobaan langsung kita pelajari. Tentunya terhadap penulis sudah menyadari jika dalam penyusun paper di atas masih banyak ada kesalahan serta jauh dari kata sempurna. Semoga para pendengar sekalian bisa memahami dengan benar

BAB VII

DAFTAR PUSTAKA

[https://repository.nusamandiri.ac.id/index.php/unduh/item/70550/15,-BAB-II-\(7-20.pdf](https://repository.nusamandiri.ac.id/index.php/unduh/item/70550/15,-BAB-II-(7-20.pdf)

<https://www.wildantechnoart.net/2017/10/apa-itu-method-void-return-static-pada-java.html>

<https://www.wildantechnoart.net/2017/10/apa-itu-method-void-return-static-pada-java.html>

<https://www.duniailkom.com/tutorial-belajar-oop-php-pengertian-class-object-property-dan-method/>

<https://docplayer.info/40944132-Object-oriented-analysis-ooa-dan-object-oriented-design-ood.html>

<http://arizkism.blogspot.com/2016/12/definisi-object-oriented-design-ood.html>

<https://www.webstudi.site/2019/02/Class-Diagram.html>

<https://binus.ac.id/malang/2020/12/diagram-sequence-dalam-analisa-desain-sistem-informasi/>

<https://www.webstudi.site/2019/11/Sequence-Diagram-adalah.html>

<http://www.tutorialkampus.com/2014/06/perancangan-diagram-dalam-sistem-e.html>