BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1. Tinjauan Pustaka

Penelitian sistem informasi akademik berbasis *web* sebelumnya telah dilakukan dengan judul "Sistem Informasi Akademik Berbasis *Web* pada SMK Teknologi Industri Pembangunan Cimahi". Tujuan dari penelitian ini adalah merancang sistem informasi akademik yang mencakup pendaftaran, penjadwalan kelas dan penilaian. Metode yang digunakan pada penelitian ini adalah metode *prototype* untuk pengembangan sistem informasi. Sistem informasi akademik ini menghasilkan kegiatan akademik di SMK Teknologi Industri Pembangunan Cimahi dapat berjalan lancar dan dapat meminimalkan kendala – kendala dalam proses pendaftaran, penjadwalan dan penilaian[4].

Penelitian sebuah sistem informasi akademik berbasis *web* juga telah dilakukan dengan judul "Perancangan Sistem Informasi Akademik Berbasis *Web* di SMP Rahmat Islamiyah". Penelitian ini bertujuan untuk membangun sistem informasi akademik berbasis *web* yang dapat digunakan sebagai salah satu fasilitas di SMP Rahmat Islamiyah untuk penyajian informasi kepada siswa. Dalam membangun sistem ini digunakan alat bantu dengan pengembangan sistem *Data Flow Daigram* (DFD), *context Diagram*, *Entity Relationship Diagram* (ERD) data *flowchart* serta dengan menggunakan bahasa pemograman PHP dan HTML dan MySQL sebagai database nya. Hasil dari sistem informasi akademik ini yaitu

dapat membentuk sistem akademik yang dinamis, sederhana dan mudah digunakan serta dengan adanya sistem ini pencarian data dapat lebih cepat dan akurat[5].

Penelitian lain tentang sistem informasi akademik berbasis *web* sebelumnya juga telah dilakukan dengan judul "Rancang Bangun Sistem Informasi Akademik Sekolah (SIAS) Berbasis *Website*". Penelitian ini bertujuan untuk merancang sistem informasi akademik sekolah berbasis *web* dengan menggunakan metode *waterfall*. Penelitian ini menghasilkan aplikasi siakad berbasis *web* yang dapat membantu dalam pengelolaan data akademik sehingga lebih efektif. Siswa/i dapat lebih mudah dalam mengakses dan melihat nilai, materi pembelajaran ataupun informasi-informasi seputar aktivitas sekolah pada aplikasi akademik berbasis *web* tersebut[1].

Penelitian sebuah sistem informasi akademik berbasis *web* juga telah dilakukan dengan judul "Analisa dan Perancangan Sistem Informasi Akademik Berbasis *Web* pada Mi Al-Mursyidiyyah Al-'Asyirotussyafi'iyyah". Penelitian ini bertujuan untuk mempermudah pengecekan dan pencatatan laporan data nilai siswa yang terkomputerisasi. Sistem yang dibuat adalah Sistem Informasi pengolahan nilai berbasis web dan MySQL sebagai databasenya. Metode yang digunakan menggunakan SDLC (*System Development Life Cycle*) dengan model proses *waterfall*. Dalam Sistem Informasi pengolahan nilai berbasis *web* terdapat sistem validasi yang berguna untuk menghindari kesalahan-kesalahan yang terjadi dengan menggunakan cara ujian dengan *paper base*[6].

Penelitian lain tentang sistem informasi akademik berbasis web sebelumnya juga telah dilakukan dengan judul "Sistem Informasi Akademik Berbasis *Web* Menggunakan Metode *Waterfall* Pada SMA Kemala Bhayangkari I Medan". Penelitian ini bertujuan untuk merancang sebuah sistem informasi akademik berbasis web pada SMA Kemala Bhayangari I Medan sehingga proses informasi akademik dapat lebih mudah dilakukan oleh siswa, guru, kepala sekolah dan masyarakat luas yang ingin mengetahui keberadaan sekolah tersebut. Hasil dari penelitian ini dihasilkan suatu sistem informasi akademik berbasis web yang dibutuhkan guru, siswa, karyawan, orang tua dan masyarakat luar sehingga dapat dengan mudah dan cepat didapatkan tanpa terhalang waktu dan tempat[7].

Pada penelitian ini, peneliti bermaksud untuk membuat sistem informasi akademik berbasis *web* di SMK KH Ahmad Dahlan yang diharapkan dapat mempermudah dalam pengolahan nilai dan monitoring hasil belajar siswa serta menjadi media untuk siswa agar dapat mengakses informasi mengenai nilai, jadwal pelajaran serta data akademik lainnya. Pada penelitian ini sistem yang akan dikembangkan yaitu proses pengolahan nilai siswa. Sistem yang akan dibuat memiliki fitur pengolahan nilai siswa hingga menjadi nilai akhir siswa dalam bentuk raport. Sistem ini akan dibuat dengan menggunakan metode *waterfall*, bahasa pemrograman PHP dan database yang digunakan yaitu Database MySQL.

2.2. Landasan Teori

2.2.1. Sistem Informasi Akademik

Sistem Informasi Akademik (SIA) adalah perangkat lunak yang digunakan untuk menyajikan informasi dan menata administrasi yang berhubungan dengan kegiatan akademik. Dengan penggunaan perangkat lunak seperti ini diharapkan kegiatan administrasi akademik dapat dikelola dengan baik dan informasi yang diperlukan dapat diperoleh dengan mudah dan cepat.

Sistem Informasi Akademik (SIA) adalah sebuah sistem aplikasi yang dibuat secara khusus untuk mengelola data administrasi akademik dalam dengan penerapan teknologi komputerisasi sehingga pengelolaan administrasi akademik dapat dikelola dengan baik dan dapat menyajikan informasi secara tepat dan cepat.

Sistem informasi akademik adalah suatu sistem yang mengolah data-data akademik pada suatu instansi pendidikan baik formal maupun informal dari tingkat dasar sampai tingkat perguruan tinggi. Secara umum data-data yang diolah dalam sistem informasi akademik meliputi data guru, data siswa, data mata pelajaran dan jadwal mengajar dan data-data lain yang bersifat umum berdasarkan kebutuhan masingmasing lembaga pendidikan.

2.2.2. Rekayasa *Web*

Rekayasa *web* adalah proses yang diunakan untuk menciptakan aplikasi *web* yang berkualitas tinggi. Rekayasa *web* mengadaptasi rekayasa perangkat lunak dalam hal konsep dasar yang

menekankan pada aktifitas teknis dan manajemen. Namun demikian adaptasi tidak secara utuh, tapi dengan perubahan dan penyesuaian. Rekayasa *web* gabungan antara *web publishing* (suatu konsep yang berasal dari *printed publishing*) dan aktifitas rekayasa perangkat lunak. Dikatakan demikian karena desain sebuah aplikasi *web* menekankan pada desain grafis, desain informasi, teori *hypertext*, desain sistem dan pemrograman [1].

Aplikasi *web* berbeda dari software lain karena hal-hal dibawah ini:

1) *Network intensive*. Sifat dasar dari *WebApp* (aplikasi *web*) adalah aplikasi ini ditujukan untuk berada di jaringan dan memenuhi kebutuhan komunitas yang berbeda.
2) *Content-Driven*. Sebagian besar fungsi dari *WebApp* adalah untuk menyajikan informasi dalam bentuk teks, grafik, audio dan video ke end user.
3) *Continuous evolution*. Selalu berkembang secara terus menerus.
4) *Document-oriented*. Halaman-halaman situs yang statis akan tetap ada sekalipun sudah ada pemrograman *web* dengan java atau yang lain.

Selain itu *WebApp* memiliki karakteristik seperti berikut ini :

1) *Immediacy*. Diperlukan segera untuk memenuhi ditayangkan, dipasarkan dalam waktu singkat.
2) *Security*. Untuk melindungi isi yang sensitif dan menyediakan pengiriman data yang aman, keamanan suatu *WebApp* harus diterapkan pada seluruh infrastruktur yang mendukung *WebApp* dan termasuk dalam *WebApp* sendiri.

3) *Aesthetics*. Daya tarik utama *WebApp* adalah tampilan dan keindahan. Jika *WebApp* digunakan untuk memasarkan suatu produk maka sisi aestetika harus diperhatikan sebagaimana sisi teknis

Perancangan dan implementasi *webapp* terkait dengan 3 teknologi yang sangat penting yaitu: *component-based development, security* dan *standart internet*. Seorang *web developer* harus mengenal 3 teknologi untuk membangun*Web*App yang berkualitas:

1) *Component-based development*: CORBA, DCOM/COM dan JavaBeans merupakan standar yang memungkinkan *web developer* menggunakan komponen-komponen yang sudah ada untuk berkomunikasi dengan sistem pada level lain.
2) Keamanan: enkripsi, dan *firewall*
3) standart Internet: HTML, XML

2.2.3. *Waterfall*

Metode air terjun atau yang sering disebut metode *waterfall* sering dinamakan siklus hidup klasik (*classic life cycle*), dimana hal ini menggambarkan pendekatan yang sistematis dan juga berurutan pada pengembangan perangkat lunak[3].

Metode *waterfall* memiliki beberapa tahapan yang berurut. Tahapan-tahapan dari metode *waterfall* adalah sebagai berikut:

1) *Requirement Analysis and Definition*

   Tahap ini pengembang sistem diperlukan komunikasi yang bertujuan untuk menganalisa terhadap kebutuhan sistem. Informasi ini biasanya dapat diperoleh melalui wawancara,

diskusi atau survei langsung. Informasi dianalisis untuk mendapatkan data yang dibutuhkan oleh pengguna.

2) *System and Software Design*

Spesifikasi kebutuhan dari tahap sebelumnya akan dipelajari dalam tahap ini dan desain sistem disiapkan. Desain Sistem membantu dalam menentukan persyaratan dan juga membantu dalam mendefinisikan arsitektur sistem secara keseluruhan. Perancangan perangkat lunak melibatkan identifikasi dan penggambaran abstraksi sistem dasar perangkat lunak dan hubungannya.

3) *Implementation and Unit Testing*

Pada tahap ini, sistem pertama kali dikembangkan di program kecil yang disebut unit, yang terintegrasi dalam tahap selanjutnya. Setiap unit dikembangkan dan diuji untuk fungsionalitas yang disebut sebagai unit testing.

4) *Integration and System Testing*

Seluruh unit yang dikembangkan dalam tahap implementasi diintegrasikan ke dalam sistem setelah pengujian yang dilakukan masing-masing unit. Setelah integrasi seluruh sistem diuji untuk mengecek setiap kegagalan maupun kesalahan.

5) *Operation and Maintenance*

Tahap akhir dalam model *waterfall*. Perangkat lunak yang sudah jadi, dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi unit sistem dan peningkatan jasa sistem sebagai kebutuhan baru.

2.2.4. *BlackBox*

Pengujian sistem merupakan pengujian program perangkat lunak yang lengkap dan terintegrasi. Perangkat lunak atau yang lebih sering dikenal dengan sebutan *software* hanyalah satuan elemen dari sistem berbasis komputer yang lebih besar. Biasanya, perangkat lunak dihubungkan dengan perangkat lunak dan perangkat keras lainnya[8].

Black Box Testing atau yang sering dikenal dengan sebutan pengujian fungsional merupakan metode pengujian Perangkat Lunak yang digunakan untuk menguji perangkat lunak tanpa mengetahui struktur internal kode atau Program. Dalam pengujian ini, *tester* menyadari apa yang harus dilakukan oleh program tetapi tidak memiliki pengetahuan tentang bagaimana melakukannya[8].

Pengujian *Black box testing* didasarkan pada detail aplikasi seperti tampilan aplikasi, fungsi-fungsi yang ada pada aplikasi, dan kesesuaian alur fungsi dengan bisnis proses yang diinginkan oleh customer. Pengujian ini tidak melihat dan menguji *source code* program. Metode *black box testing* ini bertujuan untuk memeriksa, setelah tahap akhir proyek, apakah perangkat lunak atau aplikasi berfungsi dengan baik, dan melayani penggunanya secara efisien. Biasanya, penguji mencari fungsi yang hilang atau salah; antarmuka, kinerja, inisialisasi program dan kesalahan keluar; struktur data atau kesalahan akses basis data eksternal [9].

2.2.5. *Flowchart*

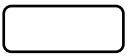
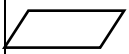
Flowchart adalah bagan-bagan yang mempunyai arus yang menggambarkan langkah-langkah penyelesaian suatu masalah. *Flowchart* merupakan cara penyajian dari suatu algoritma.

Tujuan Membuat *flowchart*:

1) Menggambarkan suatu tahapan penyelesaian masalah
2) Secara sederhana, terurai, rapi dan jelas
3) Menggunakan simbol-simbol standar

Berikut adalah simbol – simbol *flowchart*:

Tabel 2. 1 Simbol – simbol *flowchart*

No.	Simbol	Nama	Keterangan
1.		Terminal	Menyatakan permulaan atau akhir suatu program
2.		*Input/ Output*	Menyatakan proses input atau output tanpa tergantung jenis peralatannya
3		*Flow*	Menyatakan jalannya arus suatu proses
4		*Decision*	Menujukkan suatu kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban : ya / tidak

5		*Document*	Mencetak keluaran dalam bentuk dokumen (melalui printer)
6		*Manual Operation*	Menyatakan suatu tindakan (proses) yang tidak dilakukan oleh komputer
7		*Manual Input*	Memasukkan data secara manual dengan menggunakan online keyboard

2.2.6. *Unified Modelling Language* (UML)

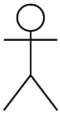
Unified Modelling Language (UML) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek. Abstraksi konsep dasar UML terdiri dari *structural classification, dynamic behavior,* dan model management dapat kita pahami *main concepts* sebagai *term* yang akan muncul pada saat membuat diagram dan *view* adalah kategori dari diagram tersebut. UML mendefinisikan diagram-diagram sebagai *Use case diagram, Class diagram, Statechart diagram, Activity diagram, Sequence diagram, Collaboration diagram, Component diagram* dan *Deployment diagram*[10].

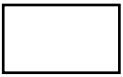
1) *Use Case Diagram*

Use case diagram merupakan pemodelan untuk kelakuan sistem informasi yang akan dibangun. *Use case* mendeskripsikan sebuah interaksi anatra satu atau lebih aktor

dengan sistem informasi yang akan dibangun. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada pada sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut [11]. Berikut ini adalah simbol-simbol *use case diagram*, seperti yang terlihat pada tabel dibawah ini:

Tabel 2. 2 Simbol – simbol *use case diagram*

No.	Simbol	Nama	Keterangan
1.		*Use case*	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.
2.		*Actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
3.		*Association*	Komunikasi antara aktor dan *use case* yang berpatisipasi pada *use*

			case atau *use case* memiliki interaksi dengan actor.
4.		*Include*	Relasi *use case* tambahan ke sebuah *use case* di mana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.
5.		*Extend*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu.
6.		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.

2) *Sequence Diagram*

Sequence diagram memiliki fokus pada perilaku didalam sistem, mengilustrasikan bagaimana objek berinteraksi dengan objek lainnya. Didalam *sequence diagram* terdapat objek dan pesan yang dikirim antar objek. Biasanya *sequence diagram*

digunakan untuk menggambarkan interaksi objek yang terjadi dalam suatu *use case.* Untuk satu *use case* hanya diperlukan satu *sequence diagram*, jika terdapat beberapa skenario dalam *use case* maka bisa ilustrasikan sebagai fragmen dalam *sequence diagram*[12]. Berikut ini adalah simbol-simbol *sequence diagram*, seperti yang terlihat pada tabel dibawah ini:

Tabel 2. 3 Simbol – simbol *sequence diagram*

No.	Simbol	Nama	Keterangan
1.		*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.
2.		*Actor*	Menggambarkan user atau pengguna.
3.		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi - informasi tentang aktivitas yang terjadi.
4.		*Boundary*	Menggambarkan sebuah *form.*
5.		*Control*	Menghubungkan *boundary* dengan Tabel.
6.		*Entity*	Menggambarkan hubungan kegiatan yang akan dilakukan.

2.2.7. Basis Data

Basis adalah Gudang / markas / tempat berkumpul / tempat bersarang. Data adalah Representasi fakta dunia nyata yang mewakili suatu obyek (manusia, benda, kejadian, dll) yang disimpan dalam bentuk teks, angka, gambar, bunyi, simbol, atau kombinasinya

Basis Data adalah Kumpulan dari item data yang saling berhubungan satu dengan lainnya yang diorganisasikan berdasar sebuah skema atau struktur tertentu, tersimpan di hardware komputer dan dengan software digunakan untuk melakukan manipulasi data (diperbaharui, dicari, diolah dengan perhitungan - perhitungan tertentu, dan dihapus) dengan tujuan tertentu.

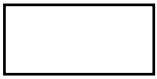
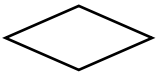
DBMS (*Database Management System*) adalah sistem pengorganisasian dan pengolahan database pada komputer. Sistem ini dirancang untuk mampu melakukan berbagai data dengan beberapa referensi data yang sama. DBMS ini mampu diakses oleh berbagai aplikasi. DBMS merupakan perangkat lunak yang bersifat generalpurpose yang memiliki fasilitas proses *define, construct* dan *manipulate* basis data untuk aplikasi yang bervariasi.

Basis Data Relasional merupakan suatu cara untuk mengelola data secara fisik kedalam memori. Basis Data relasional menggunakan tabel dua dimensi yang terdiri atas baris dan kolom untuk memberi gambaran sebuah berkas data. Untuk menerapkan sebuah basis data (yang terdiri atas sejumlah tabel yang saling berhubungan), dibutuhkan perangkat lunak (software) khusus. Perangkat lunak ini disebut Sistem Pengelola Basis Data.

Keunggulan basis data relasional yaitu bentuknya yang sederhana dan mudah untuk melakukan berbagai operasi data.

Entity Relationship diagram (ERD) [13] merupakan teknik yang digunakan untuk memodelkan kebutuhan data dari suatu organisasi, biasanya oleh *System Analys* dalam tahap analisis persyaratan proyek pengembangan system. Sementara seolah-olah teknik diagram atau alat peraga memberikan dasar untuk desain database relasional yang mendasari sistem informasi yang dikembangkan. ERD bersama-sama dengan detail pendukung merupakan model data yang pada gilirannya digunakan sebagai spesifikasi untuk database.

Tabel 2. 4 Simbol – simbol ERD

No.	**Simbol**	**Nama**
1.		Entitas
2.		Atribut
3.		Hubungan
4.		Garis

1) Entitas

   Entity (entitas) yaitu suatu obyek yang dapat dibedakan dari lainnya yang dapat diwujudkan dalam basis data.

2) Hubungan (relasi/*relationship*)

Suatu hubungan adalah hubungan antara dua jenis entitas dan direpresentasikan sebagai garis lurus yang menghubungkan dua entitas.

3) Atribut

Atribut memberikan informasi lebih rinci tentang jenis entitas. Atribut memiliki struktur internal berupa tipe data.

Jenis-jenis atribut:

a. Atribut *key*

Atribut *key* adalah satu atau gabungan dari beberapa atribut yang dapat membedakan semua baris data (*Row/Record*) dalam tabel secara unik. Dikatakan unik jika pada atribut yang dijadikan *key* tidak boleh ada baris data dengan nilai yang sama.

b. Atribut *simple*

Atribut yang bernilai atomic, tidak dapat dipecah/ dipilah lagi.

c. Atribut *multivalue*

Nilai dari suatu attribute yang mempunyai lebih dari satu (*multivalue*) nilai dari atrribute yang bersangkutan.

d. Atribut *composite*

Atribut *composite* adalah suatu atribut yang terdiri dari beberapa atribut yang lebih kecil yang mempunyai arti tertentu yang masih bisah dipecah lagi atau mempunyai sub attribute.

e. Atribut *derivatif*

Atribut yang tidak harus disimpan dalam database Ex. Total. atau atribut yang dihasilkan dari atribut lain atau dari suatu *relationship*. Atribut ini dilambangkan dengan bentuk oval yang bergaris putus-putus.

Derjat relasi atau kardinalitas rasio :

1) *One to one* (1:1)

   Setiap anggota entitas A hanya boleh berhubungan dengan satu anggota entitas B, begitu pula sebaliknya.

2) *One to many* (1:M/*Many*)

   Setiap anggota entitas A dapat berhubungan dengan lebih dari satu anggota entitas B tetapi tidak sebaliknya. Setiap anggota entitas A dapat berhubungan dengan lebih dari satu anggota entitas B tetapi tidak sebaliknya.

3) *Many to many* (M:M)

   Setiap entitas A dapat berhubungan dengan banyak entitas himpunan entitas B dan demikian pula sebaliknya.

Data Definition Language atau DDL merupakan sub bahasa SQL yang berfungsi untuk membangun kerangka database. Terdapat tiga perintah yang termasuk kedalam perintah DDL, yaitu:

1) CREATE

   Perintah CREATE ini digunakan untuk membuat database baru, tabel baru, *view* baru, dan membuat kolom.

2) RENAME

   Perintah RENAME ini digunakan untuk mengubah nama suatu tabel.

3) ALTER

   Perintah ALTER ini digunakan untuk membuat struktur dari tabel yang sudah dibuat. Hal yang dapat dilakukan yaitu mengganti nama tabel, menambah kolom, mengubah kolom, menghapus kolom, serta memberi atribut pada kolom.

4) DROP

   Perintah DROP ini digunakan untuk menghapus database ataupun menghapus tabel.

Data Manipulation Language atau DML merupakan sub bahasa SQL yang berfungsi untuk memanipulasi data yang terdapat pada database yang telah dibuat. Terdapat empat perintah yang termasuk kedalam perintah DML, yaitu:

1) INSERT

   Perintah INSERT dapat digunakan setelah pembuatan database dan tabel. Perintah ini berfungsi untuk memanipulasi data dengan memasukan ataupun menyisipkan data baru kedalam tabel.

2) SELECT

Perintah SELECT ini berfungsi untuk mengambil atau menampilkan data dari suatu tabel ataupun beberapa tabel yang saling berelasi.

3) UPDATE

Perintah UPDATE dapat digunakan jika terdapat data yang salah atau perlu diperbaiki. Perintah ini berfungsi untuk memperbarui data yang sudah ada.

4) DELETE

Perintah DELETE dapat digunakan jika ingin mengahapus data yang ada pada tabel. Perintah ini tidak dapat digagalkan apabila perintah di jalankan data tidak dapat dikembalikan.

Data Control Language atau DCL merupakan sub bahasa SQL yang berfungsi untuk mengontrol data dan server databasenya. Terdapat dua perintah yang termasuk kedalam perintah DCL, yaitu:

1) GRAND

Perintah GRAND dapat digunakan oleh administrator server untuk memberikan hak akses kepada user. Hak akses yang dapat diberikan berupa CREATE, SELECT, DELETE, UPDATE, dan hah-hak lainnya yang berkaitan dengan database.

2) REVOKE

Perintah REVOKE dapat digunakan oleh administrator server untuk mencabut hak akses user. Perintah ini merupakan kebalikan dari perintah GRAND. Hak akses

yang dapat dicabutpun sama yaitu CREATE, SELECT, DELETE, UPDATE, dan hah-hak lainnya yang berkaitan dengan database.

2.2.8. Pemrograman Berbasis Objek

Pemrograman berbasis objek atau biasa disebut OOP merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Dalam pemrograman berbasis objek, kita dituntut untuk memahami dan memetakan masalah kedalam class serta memecah masalah kedalam *class-class* yang lebih kecil dan simpel agar solusi yang dibuat lebih spesifik. Selanjutnya, *class-class* tersebut akan saling berkomunikasi dan berkolaborasi untuk memecahkan masalah yang kompleks. *Class-class* ini nantinya akan dirubah menjadi objek-objek pada saat runtime.

Setiap *class* dalam OOP mempunyai *method* atau fungsi serta *property* atau atribut. *Method* dalam *class* secara mudah diartikan sebagai segala kemampuan dari *class* atau apa saja yang dapat dilakukan oleh sebuah *class.* Sedangkan *property* adalah segala sesuatu yang dimiliki oleh *class.* Dalam OOP, *property* dan *method* dalam *class* saling bekerjasama membangun sebuah solusi dari suatu masalah.

Enkapsulasi adalah sebuah fitur yang menggabungkan antara fungsionalitas dan data untuk menyembunyikan informasi. Enkapsulasi memungkinkan kita menggunakan fungsi dari sebuah objek tanpa perlu mengetahui detail dari apa yang terjadi dalam

fungsi tersebut. Ekapsulasi mengatur bagaimana kita menggunakan objek, fungsi atau atribut apa saja yang dapat digunakan dan yang tidak dapat digunakan.

Pewarisan adalah sebuah fitur yang memungkinkan kita menggunakan fitur dari suatu *class* tanpa perlu mendefinisikan ulang semua *method* dan *property* yang ada pada *class* tersebut. Pewarisan ini sangat bermanfaat jika kita ingin mempunyai sebuah *class* yang secara fitur mirip dengan *class* lain namun ada sebuah spesifikasi khusus yang spesifik dari *class* tersebut. Dalam OOP, *class* yang mewariskan sifat atau fitur disebut sebagai parent class sedangkan *class* yang diwarisi sifat atau fitur disebut *child class*. *Child class* memiliki semua fitur yang ada pada *parent class* ditambah dengan fitur spesifik miliknya sendiri.

Polimorfisme adalah sebuah fitur yang erat kaitannya dengan fitur sebelumnya yaitu pewarisan. Fitur ini secara mudah adalah sebuah kemampuan dari objek untuk merespon atau mengolah secara berbeda terhadap input yang sama. Polimorfisme adalah salah satu fitur yang harus ada dalam OOP dimana sebuah *method* yang sama pada *class* yang berbeda akan memberikan *output* yang berbeda pula jika digunakan.

Halaman Sengaja Dikosongkan