BAB II
LANDASAN TEORI

2.1 Sampah

2.1.1 Pengertian Sampah

Sampah adalah sisa kegiatan sehari – hari manusia dan/atau proses alam yang berbentuk padat dan penghasil sampah adalah setiap orang dan/atau akibat proses alam yang menghasilkan timbulan sampah. [1]. Semua buangan yang dihasilkan oleh aktivitas manusia dan hewan yang berbentuk padat, lumpur (sludge), cair maupun gas yang dibuang karena tidak dibutuhkan atau tidak diinginkan lagi. Walaupun dianggap sudah tidak berguna dan tidak dikehendaki, namun bahan tersebut kadang–kadang masih dapat dimanfaatkan kembali dan dijadikan bahan baku [10].

2.1.2 Jenis – Jenis Sampah

Menurut peraturan pemerintah nomor 81 tahun 2012 tentang pengelolaan sampah rumah tangga dan sampah sejenis rumah [11] pada pasal 17 ayat (2) yang berisi:

Pemilahan sebagaimana dimaksud pada ayat (1) dilakukan melalui kegiatan pengelompokan sampah menjadi paling sedikit 5 (lima) jenis sampah yang terdiri atas:

a. Sampah yang mengandung bahan berbahaya dan beracun serta limbah bahan berbahaya dan beracun;
b. Sampah yang mudah terurai;
c. Sampah yang dapat digunakan kembali;
d. Sampah yang dapat didaur ulang; dan
e. Sampah lainnya.

Berikut contoh sampah dari kelima jenis sampah diatas:

a. Sampah yang mengandung bahan berbahaya dan beracun serta limbah bahan berbahaya dan beracun, seperti kemasan obat

serangga, kemasan oli, kemasan obat-obatan, obat-obatan kadaluarsa, peralatan listrik, dan peralatan elektronik rumah tangga.

b. Sampah yang mudah terurai, seperti sampah yang berasal dari tumbuhan, hewan, dan/atau bagian-bagiannya yang dapat terurai oleh makluk hidup lainnya dan/atau mikroorganisme, misalnya sampah makanan dan serasah.
c. Sampah yang dapat digunakan kembali, seperti sampah anorganik yang bisa dipakai kembali sesuai fungsi asalnya ataupun fungsi berbeda tanpa diproses terlebih dulu, misalnya kantong plastik yang masih utuh.
d. Sampah yang dapat didaur ulang, meliputi sampah anorganik yang harus diproses dulu agar dapat dimanfaatkan kembali, seperti botol plastik sekali pakai berbahan PET.
e. Sampah lainnya atau sampah residu adalah sampah selain dari 4 kategori sebelumnya, seperti kemasan saset dan sampah plastik multilayer lainnya.

2.1.3 **Tempat Sampah**

Menurut bapeda provinsi Kalimantan Selatan terdapat 5 warna tempat sampah untuk setiap jenisnya.[12] Berikut warna tempat sampah berserta jenis sampah nya:

1. Warna hijau untuk sampah organik
2. Warna kuning untuk sampah guna ulang
3. Warna merah untuk sampah B3/Bahan Berbahaya & Beracun
4. Warna biru untuk sampah daur ulang
5. Warna abu – abu untuk sampah residu

Gambar 2. 1 Warna Tempat Sampah

2.2 Projek Penguatan Profil Pelajar Pancasila (P5)

Projek Penguatan Profil Pelajar Pancasila (P5) bertujuan untuk mendorong tercapainya profil pelajar Pancasila melalui penerapan model pembelajaran berbasis proyek. Tujuan utama P5 adalah untuk membentuk pelajar Indonesia yang kompeten, berkarakter, dan berperilaku sesuai dengan nilai-nilai Pancasila.[13]

Dalam pembelajaran pengelolaan sampah untuk sekolah dasar terdapat 3 fase. Fase A yaitu kelas 1 – 2, fase B kelas 3- 4 dan fase C kelas 5 – 6. Setiap fase memiliki capaian pembelajaran yang berbeda – beda. Untuk fase A peserta didik mampu memahami jenis sampah organik dan anorganik yang ada di lingkungan rumah atau keluarga, mengidentifikasi makna warna-warna tempat sampah serta mampu memilah sampah yang dihasilkan lingkungan rumah atau keluarga [14]. Untuk fase B peserta didik mampu memahami sampah berbahaya dan sumbernya. Peserta didik mampu menunjukkan perbedaan jenis sampah dan dapat mengelompokkan sampah sesuai dengan jenisnya [15]. Dan untuk fase C peserta didik mampu memahami pengelolaan sampah yang ada di lingkungan sekitar serta menunjukkan cara pengelompokkan sampah dan tempat sampah berdasarkan jenis dan sifat sampah. Peserta didik mampu memahami pemanfaatan sampah organik dan anorganik serta mampu mengaplikasikan cara pengelolaan sampah dalam kehidupan sehari-hari [16].

2.3 *Game*

2.1.4 Pengertian *Game*

Permainan atau *game* terdiri dari aturan – aturan yang membatasi, menentukan jalannya *game* dan membangun situasi bersaing bagi para pemain dengan jumlah pemain dari dua hingga beberapa orang atau kelompok [17]. *Game* adalah sebuah aktivitas yang dilakukan untuk bersenang – senang ataupun belajar. Belajar melalui *game* dapat dimanfaatkan untuk melatih kemampuan siswa dalam menyelesaikan permasalahan, mencari solusi, berfikir cepat dan dapat melatih ke sportifan saat berkompetisi. Tentunya ini tidak hanya dapat meningkatkan pemahaman siswa terhadap materi yang dijadikan *game* tetapi juga dapat meningkatkan kemampuan otak [18], [19].

2.1.5 *Platform Game*

Beberapa klasifikasi *platform game* yang berjalan dapat dilihat pada bagian berikut:

1. *Arcade Game* yaitu *game* yang biasa ditempatkan di mal, restoran, taman hiburan atau tempat – tempat tertentu yang berupa *box* dengan layar dan perangkat kontrol *built-in* yang dapat dimainkan dengan memasukkan koin.
2. *Console Game* yaitu *video game* yang dapat dimainkan menggunakan perangkat *console* tertentu seperti Playstation 1, Playstation 2, Playstation 3, Playstation 4, Playstation 5, XBOX 360, dan Nintendo Wii.
3. *Personal Computer Game* yaitu video *game* yang dapat dimainkan menggunakan perangkat PC atau *Personal Computer*. PC game umumnya dapat memiliki kapasitas masukan, pemrosesan, keluaran baik audio maupun video yang lebih baik jika dibandingkan dengan *arcade game* dan *console game*.
4. *Mobile Game* yaitu *video game* dapat yang dimainkan menggunakan perangkat mobile atau perangkat portable. *Game* ini biasanya dapat dimainkan pada perangkat smartphone, tablet, graphic calculator atau *handheld console*.
5. *Handheld Console Game* yaitu *video game* yang dimainkan di *console* khusus *video game* yang dapat dibawa kemana-mana. Konsol genggam atau

handheld console adalah mesin untuk memainkan game yang bentuknya tidak lebih besar dari genggaman kedua tangan, mudah dibawa dalam genggaman, dan tentunya ringan. Contoh *handheld* adalah Switch dan PSP.

2.1.6 Genre *Game*

Berikut ini beberapa pengelompokan jenis game berdasarkan genre-nya diantaranya adalah [20], [21]:

1. *Adventure*, *game* petualangan menekankan pengalaman cerita melalui dialog dan pemecahan teka-teki. Mekanisme *gameplay* menekankan pada pengambilan keputusan atas tindakan. Pemecahan teka-teki biasanya berkisar pada menggabungkan atau memanipulasi item untuk menjalankan cerita.
2. *Role-Playing (RPG)*, game *Role-Playing (RPG)* termasuk dalam genre game yang bervariasi yang berfokus pada pengembangan karakter.
3. *First Person Shooter (FPS)*, Game FPS biasanyan dicirikan dengan respons motorik yang dipercepat tetapi mengurangi kemampuan untuk membatalkan respons yang kuat.
4. *Platform Game*, game platform atau yang biasa disebut sebagai *platformer* adalah game yang karakter dan latarnya terlihat dari samping. *Gameplay* yang biasa ada di *platformer* ialah pemain harus menggerakkan karakter dari satu titik ke titik lain melewati berbagai rintangan hingga sampai di garis finish.
5. *Strategy, game* strategi berkisar pada penggunaan sumber daya strategis dan/atau taktis sering kali dalam skenario pertempuran atau manajerial.
6. *Racing / driving, game* balap/mengemudi memungkinkan pemain untuk balapan, atau mengendarai kendaraan. Balapan dapat dilakukan di kendaraan, di atas tunggangan, berjalan kaki atau dalam grafik yang sepenuhnya abstrak. *Game* dengan *genre* ini harus memiliki aspek

racing/driving untuk sebagian besar *gameplay game*-nya, bukan hanya sebagai *short sequence*.

7. *Action,* Mekanik utama *game action* berkisar pada akurasi, pergerakan, kecepatan mengambil keputusan, reflek dan *timing*.
8. *Educational*, *Game* edukasi mengajar pemain melalui permainan. Biasanya ditujukan untuk anak kecil, *game* edukasi menawarkan cara yang menyenangkan dan tidak langsung melatih mata pelajaran yang "tidak menyenangkan" seperti mengeja, matematika, sejarah, dll. *Game* edukasi sering disebut "*Edutainment*".
9. *Simulation,* Terdapat banyak jenis *game* simulasi. Kesamaan dari semua *Simulasi* adalah *game* simulasi dimodelkan secara realistis dengan situasi Dan/atau variabel kehidupan nyata daripada genre game yang lainnya.
10. *Sports, Game* olahraga adalah *game* di mana pemain mengontrol baik pemain atau manajer olahraga nyata atau fiksi.

2.1.7 *Perspective* dan *Viewpoints Game*

Berikut ini beberapa pengelompokan jenis *game* berdasarkan *perspective* dan *viewpoints*-nya diantaranya adalah [21]:

1. First Person
2. Third Person
3. Isometric
4. Platform
5. Side-Scrolling
6. Top-Down

2.1.8 *Game* Edukasi

Berikut ini beberapa pengelompokan jenis *game* berdasarkan konten dari *game* edukasi yang diangkat diantaranya adalah [21]:

1. Ecology / Nature
2. Foreign Language
3. Geography
4. Graphics / Art
5. Health / Nutrition
6. History
7. Math / Logic
8. Music
9. Pre-school / Toddler
10. Reading / Writing
11. Religion
12. Science
13. Sociology
14. Typing

2.2 *Unifled Modeling Language* (UML)

Unified Modeling Language (UML) adalah bahasa pemodelan untuk menspesifikasi, memvisualisasi, membangun dan mendokumentasikan dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya. UML menyediakan notasi - notasi yang membantu memodelkan sistem dari berbagai prespetktif. UML tidak hanya digunakan dalam pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan [22]. UML terdiri dari berbagai jenis diagram, seperti Use Case Diagram, Activity Diagram, Sequence Diagram, dan Class Diagram, yang masing-masing memiliki tujuan dan kegunaan spesifik dalam pemodelan sistem [23]. Dengan menggunakan UML, pengembang perangkat lunak dapat memvisualisasikan konsep desain, menganalisis arsitektur sistem, dan mengidentifikasi kebutuhan serta interaksi antara komponen sistem.

2.2.1 *Use Case Diagram*

Use case diagram adalah sarana untuk menggambarkan persyaratan sebuah sistem yaitu sistem apa yang seharusnya digunakan. Komponen *use case* yaitu aktor, *use case*, dan subjek (Sistem). Sistem adalah setiap subjek *use case* yang mewakili *system* yang sedang dipertimbangkan dimana *use case* diterapkan. Aktor

adalah pengguna dan sistem lain yang berinteraksi dengan *system* yang akan dibuat. *Use case* adalah spesifikasi prilaku. Sebuah contoh dari use case mengacu pada terjadinya perilaku yang muncul yang sesuai dengan use case [22]. Contoh seperti itu sering dijelaskan oleh interaksi. Komponen dalam *use case* diagram, diantaranya:

1. *Use case* dan Aktor

   Use case adalah spesifikasi sekumpulan perilaku yang dilakukan oleh subjek/sistem, yang memberikan hasil yang dapat diamati yang bernilai bagi satu atau lebih aktor atau pemangku kepentingan lainnya dari subjek/sistem. Aktor adalah memodelkan jenis peran yang dimainkan oleh entitas yang berinteraksi dengan subjek dari *use case* yang terkait.

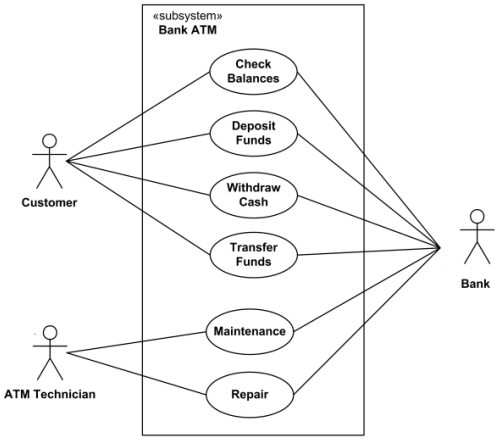
2. Extends

   Extends adalah relasi *use case* tambahan ke sebuah *use case* tambahan lain yang menentukan bagaimana dan kapan prilaku di definisikan dalam *use case* tambahan dapat dimasukkan ke dalam prilaku yang ditentukan di *use case* tambahan. *Extends* digunakan ketika ada beberapa prilaku tambahan yang harus ditambahkan, mungkin secara bersyarat, ke prilaku yang ditentukan dalam satu atau lebih *use case.*

3. Includes

   Includes adalah relasi terarah antara dua *use case*, yang menunjukkan bahwa perilaku *use case* tambahan dimasukkan ke dalam perilaku *use case* lainnya untuk menjalankan fungsinya. Relasi *include* digunakan ketika ada bagian umum dari kasus (perilaku) dua atau lebih *use case*. Bagian umum kemudian diektraksi ke *use case* secara terpisah, untuk dimasukkan oleh semua *use case* dasar yang memiliki bagian yang sama. Sebagai pengguna utama sebuah relasi dalam *include* adalah penggunaan kembali bagian umum, apa yang tersisa di *use case* dasar yang biasanya tidak lengkap tetapi bergantung pada bagian dari

include agar bermakna. Digambarkan dalam arah relasi, menunjukan bahwa *use case* dasar bergantung pada *use case* tambahan.

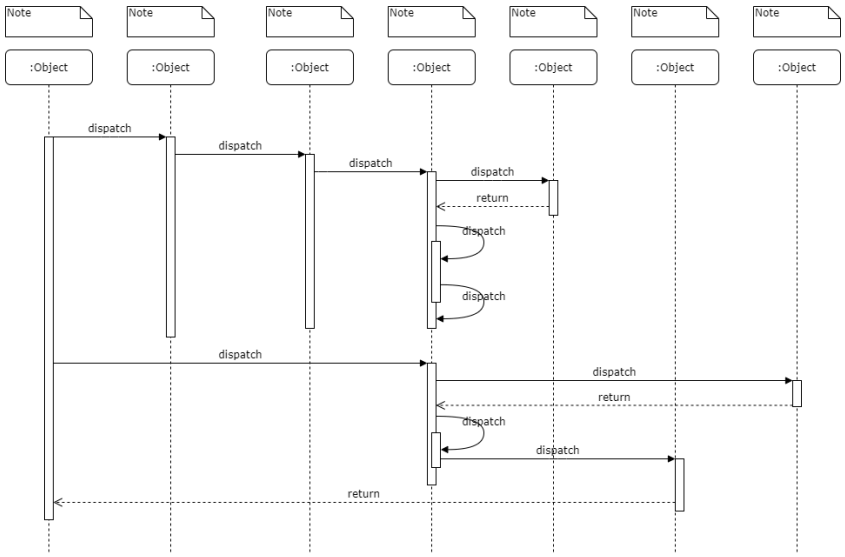


Gambar 2. 2 Contoh Use Case Diagram

2.2.2 ***Sequence Diagram***

Sequence diagram adalah sebuah diagram yang menggambarkan kolaborasi dari objek – objek yang saling berinteraksi antar elemen dari suatu *class*. Berikut ini merupakan komponen dalam *sequence diagram*:

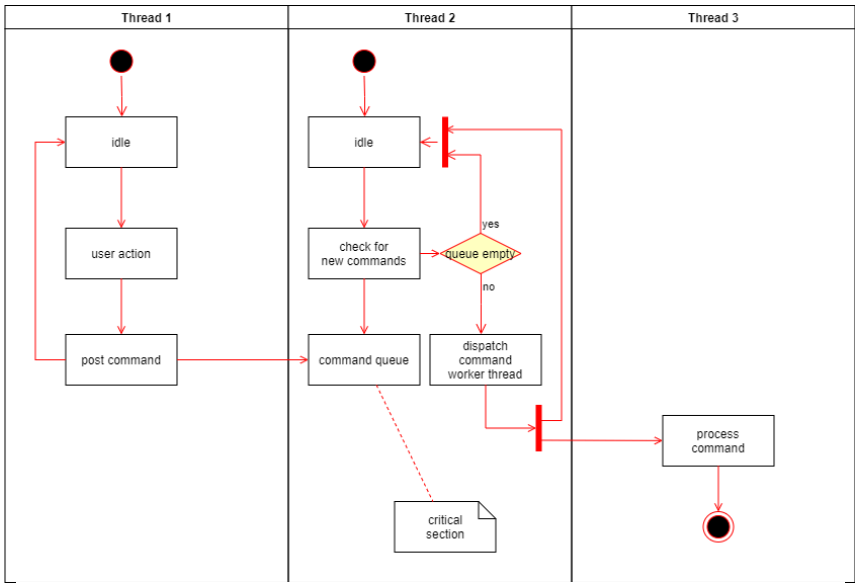
1. *Activations*, menjelaskan tentang eksekusi dari fungsi yang dimiliki oleh suatu objek.
2. *Actor*, menjelaskan tentang peran yang melakukan serangkaian aksi dalam suatu proses.
3. *Collaboration boundary*, menjelaskan tentang tempat untuk lingkungan percobaan dan digunakan untuk memonitor objek.
4. *Parallel vertical lines*, menjelaskan tentang suatu garis proses yang menunjuk pada suatu state.
5. *Processes*, menjelaskan tentang tindakan/aksi yang dilakukan oleh aktor dalam suatu waktu.
6. *Window*, menjelaskan tentang halaman yang sedang ditampilkan dalam suatu proses.
7. *Loop*, menjelaskan tentang model logika yang berpotensi untuk diulang beberapa kali.



Gambar 2. 3 Contoh Sequence *D*iagram

2.2.3 *Activity Diagram*

Activity diagram adalah suatu diagram yang menggambarkan konsep aliran aktivitas, aksi terstruktur serta dirancang dengan baik dalam suatu sistem. Diagram ini menggunakan notasi untuk merepresentasikan aktivitas, keputusan, kondisi, dan garis alur untuk menggambarkan urutan aktivitasb[24].



Gambar 2. 4 Contoh Activity Diagram

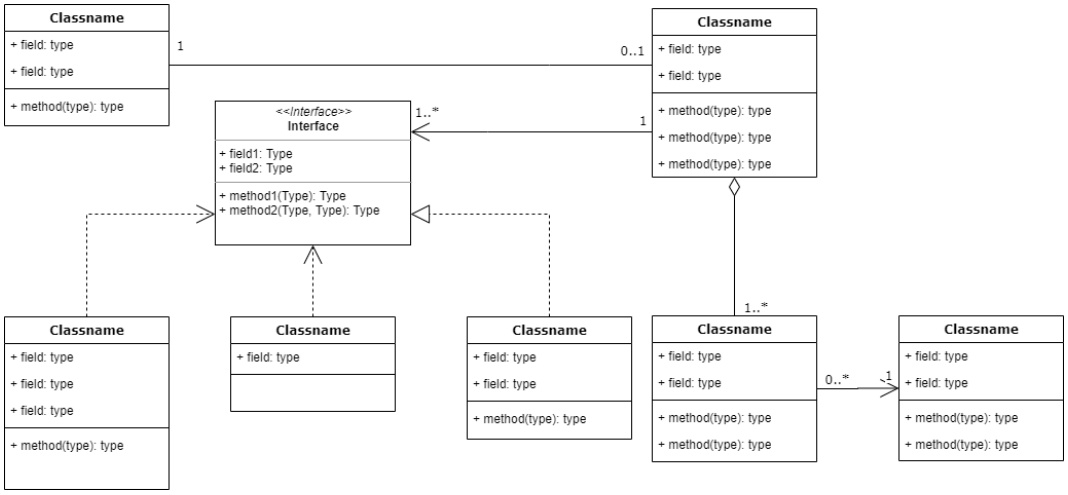
2.2.4 *Class Diagram*

Class diagram adalah sebuah diagram yang menggambarkan struktur dan menunjukkan hubungan antar *class* yang didalamnya terdapat atribut dan fungsi dari suatu objek. *Class diagram* menjelaskan model yang digunakan dalam perancangan atribut dan fungsi – fungsi yang akan digunakan untuk membangun sistem baru [24], [25]. *Class diagram* mempunyai 3 relasi dalam penggunaannya, yaitu:

1. Asosiasi (*Assosiation*) adalah sebuah hubungan yang menunjukkan adanya interaksi antar class. Hubungan ini dapat ditunjukkan dengan garis dengan

mata panah terbuka di ujungnya yang mengindikasikan adanya aliran pesan dalam satu arah.

2. *Generalization* adalah sebuah hubungan antar *class* yang bersifat dari khusus ke umum.
3. *Constraint* adalah sebuah hubungan yang digunakan dalam sistem untuk memberi batasan pada sistem sehingga didapat aspek yang tidak fungsional.



Gambar 2. 5 Contoh Class Diagram

2.3 *Android*

Android adalah sistem operasi berbasis *Linux* bagi telepon seluler seperti telepon pintar dan komputer *tablet*. *Android* juga menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri yang akan digunakan untuk berbagai macam piranti gerak. Awalnya, *Google Inc*. membeli *Android Inc*., pendatang baru yang membuat piranti lunak untuk ponsel. kemudian dalam pengembangan *Android*, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan piranti keras, piranti lunak, dan telekomunikasi, termasuk *Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia* [26].

2.4 *Aplication Programming Interface (API)*

Application Programming Interface (API) adalah konsep fungsi antarmuka pemrograman aplikasi, yang menjadi salah satu cara agar suatu aplikasi dapat diakses dan dimanfaatkan oleh pihak lain tanpa mengubah struktur kode utama maupun database sistem, serta memudahkan komunikasi antar sistem meskipun berbeda platform. *Web Service* adalah API yang berperan dalam memberikan akses

pengguna dalam proses pengambilan data. Melalui arsitektur *Representational State Transfer* (ReST) yang dioperasikan melalui *Hypertext Transfer Protocol* (HTTP), berisikan sebuah *file Javascript Object Notation* (JSON), filetersebut yang akan disajikan kepada para pengguna saat mengakses API [27].

2.5 *Object Oriented Programming (OOP)*

Object Oriented Programming (OOP) atau pemrograman berorientasi objek (PBO) merupakan pemrograman yang berorientasikan kepada objek, dimana semua data dan fungsi dibungkus dalam *class – class* atau *object – object*. Setiap *object* dapat menerima pesan, memproses data, mengirim, menyimpan dan memanipulasi data. Beberapa *object* berinteraksi dengan saling memberikan informasi satu terhadap yang lainnya. Masing – masing *object* harus berisikan informasi mengenai dirinya sendiri dan dapat dihubungkan dengan *object* yang lain. Pemrograman berorientasi objek berbeda dengan pemrograman prosedural yang hanya menggunakan satu halaman kebawah untuk mengerjakan banyak perintah atau statement. Penggunaan pemrograman berorientasi objek sangat benyak sekali, contoh: java, php, perl, c#, cobol, dan lainnya [28].

Dalam konsep pemrograman berorientasi objek dikenal beberapa istilah umum, yaitu: *object* dan *class, encapsulation, inheritance, polymorphism*:

1. Object dan Class

Dalam pemrograman berbasis OOP, kelas (*class*) adalah konsep yang digunakan untuk mendefinisikan struktur dan perilaku objek yang dibuat. Kelas merupakan *blueprint* atau template yang menjelaskan properti dan perilaku untuk membuat objek. Sebuah kelas dapat dianggap sebagai template yang digunakan untuk membuat objek yang serupa dengan kelas tersebut. Kelas OOP terdiri dari beberapa komponen yaitu, variabel yang digunakan untuk menyimpan nilai objek. Metode berfungsi untuk melakukan tindakan pada objek. *Construktor* berfungsi untuk membuat objek baru dengan nilai awal tertentu. Inheritance (warisan) berfungsi untuk memungkinkan kelas mewarisi properti dan perilaku dari kelas lain.

Kelas OOP digunakan untuk membuat objek yang mirip dengan kelas tersebut. Objek yang dibuat dari kelas-kelas ini disebut instantiasi dari kelas. Kelas

memungkinkan pemrogram untuk membuat objek yang memiliki properti dan perilaku yang sama tetapi nilainya berbeda. Hal ini memungkinkan programer untuk membuat program yang lebih modular dan lebih mudah untuk dikelola.

2. Encapsulations

Encapsulation merupakan suatu mekanisme untuk menyembunyikan atau memproteksi suatu proses dari kemungkinan interferensi atau penyalahgunaan dari luar sistem dan sekaligus menyederhanakan penggunaan sistem tersebut. *Encapsulaition* dapat disimpulkan sebagai mekanisme unutk menyembunyikan data internal suatu objek dari akses luar secara langsung, sehingga memungkinkan manipulasi hanya melalui metode yang disediakan. Hal ini membantu melindungi integritas data dan mencegah akses yang tidak diperlukan.

3. Inheritance

Inheritance merupakan mekanisme yang memungkinkan suatu kelas untuk mewarisi atribut dan metode dari kelas lain. Kelas yang mewarisi disebut *subclass* (kelas turunan), dan kelas yang diwarisi disebut superclass (kelas induk). Hal ini memungkinkan penggunaan mengulang kode dan membuat hierarki kelas.

Dengan konsep ini class yang dibuat cukup mendefinisikan *attribute* dan *method* yang spesifik didalamnya, sedangkan *attribute* dan *method* yang lebih umum akan didapatkan dari class yang menjadi induknya.

4. Polimorphism

Polimorphism berfungsi sebagai suatu objek yang dapat digunakan untuk tujuan berbeda dengan nama yang sama. Dalam OOP, *polimorphism* dapat dianggap sebagai teknik pemrograman yang memungkinkan banyak tipe berbeda yang direpresentasikan menggunakan satu simbol. *Polimorfisme* dapat dibagi menjadi dua jenis yaitu *Polimorphism* statis dan *Polimorphism* dinamis. *Polimorphism* statis menggunakan metode *overloading*, sedangkan *polimorphism* dinamis menggunakan metode penggantian. *Polimorphism* dinamis juga dapat diartikan sebagai teknik pewarisan (*inheritance*) yang memungkinkan objek dapat mewarisi sifat dan perilaku dari objek lain.

2.6 C#

Common Intermediate Language (CIL) adalah semacam kode biner yang Membutuhkan untuk ditafsirkan. CIL adalah platform dan bahasa yang independen, Seperti C# open standard. Keuntungan menggunakan interpretasi Common Language Runtime (CLR) alih-alih kode mesin adalah bahwa ada kerangka kerja Yang jelas dan tidak ada masalah file .dll yang tidak kompatibel dan dengan Demikian menghindari masalah seperti yang biasa disebut “DL Hell” pada buku “Avancerad Pocket Visual Basic NET”. CLR mengkompilasi kode CIL ke kode Mesin ketika bagian dari kode tersebut digunakan, disebut Just In Time Compilation. Ini memungkinkan pengoptimalan platform dan keamanan jenis runtime.

2.7 Unity

Unity (umumnya dikenal sebagai Unity3D) adalah game engine dan Integrated development environment (IDE) untuk membuat media interaktif, Biasanya video game. Unity dapat digunakan untuk membantun game bergrafis 2D, 3D, Augmented Reality serta Virtual Reality. Unity terkenal dengan kemampuannya Melakukan prototyping yang cepat dan dengan target penerbitan yang berjumlah Besar. Unity menggunakan bahasa C# untuk bagian scripting. Unity memiliki Beberapa build target yang dapat melakukan compilation kode menjadi bahasa Native pada platform yang dituju. Berikut beberapa target build yang disediakan:

1. StandaloneOSX
2. StandaloneWindows
3. iOS
4. Android
5. StandaloneWindows64
6. WebGL
7. WSAPlayer
8. StandaloneLinux64
9. PS4
10. XboxOne
11. tvOS
12. Switch
13. Stadia
14. CloudRendering
15. PS5

2.8 *Visual Studio Code*

Visual Studio Code adalah sebuah IDE ringan yang dapat digunakan untuk mengembangkan berbagai jenis aplikasi, termasuk aplikasi web, aplikasi desktop, dan pengembangan berbasis cloud.VS Code menyediakan berbagai fitur yang kuat untuk mendukung pengembangan perangkat lunak[29]. Seperti fitur debugging yang terintegrasi untuk berbagai bahasa pemrograman. Pengembang aplikasi juga dapat menginspeksi variabel, dan melacak kode dengan mudah menggunakan alat debugging yang disediaka