MAKALAH PEMROGRAMAN BERORIENTASI OBJEK

PADA JAVA



Oleh :

DIKI CANDRA

NIM 2022903430010

D4 TEKNOLOGI REKAYASA KOMPUTER JARINGAN

POLITEKNIK NEGERI LHOKSEMAWE

2022/2023

DAFTAR ISI

[BAB I 3](#_Toc131012785)

[PENDAHULUAN 3](#_Toc131012786)

[1.1 LATAR BELAKANG 4](#_Toc131012787)

[1.2 TUJUAN PENULISAN MAKALAH 5](#_Toc131012788)

[1.3 RUANG LINGKUP 6](#_Toc131012789)

[BAB II 7](#_Toc131012790)

[KONSEP DASAR 7](#_Toc131012791)

[PEMROGRAMAN BERORIENTASI OBJEK PADA JAVA 7](#_Toc131012792)

[2.1 DEFINISI PBO PADA JAVA 8](#_Toc131012793)

[2.2 KARAKTERISTIK PBO PADA JAVA 9](#_Toc131012794)

[2.3 KONSEP ABSTRAKSI PADA JAVA 10](#_Toc131012795)

[2.4 KONSEP ENCAPSULUTION PADA JAVA 16](#_Toc131012796)

[2.5 KONSEP INHERITANCE PADA JAVA 18](#_Toc131012797)

[2.6 KONSEP POLYMORPHISM PADA JAVA 20](#_Toc131012798)

[BAB III 21](#_Toc131012799)

[IMPLEMENTASI PBO 21](#_Toc131012800)

[PADA PEMROGRAMAN JAVA 21](#_Toc131012801)

[3.1 KONSTRUKTOR PADA JAVA 22](#_Toc131012802)

[3.1 KELAS DAN OBJEK PADA JAVA 25](#_Toc131012803)

[3.3 METHOD PADA JAVA 26](#_Toc131012804)

[3.4 INHERITANCE PADA JAVA 26](#_Toc131012805)

[3.5 INTERFACES PADA JAVA 26](#_Toc131012806)

[3.6 ABTRAKSI PADA JAVA 26](#_Toc131012807)

[BAB IV 27](#_Toc131012808)

[ANALISA PERBANDINGAN 27](#_Toc131012809)

[PBO PADA PEMROGRAMAN JAVA 27](#_Toc131012810)

[4.1 PERBANDINGAN PBO PADA BAHASA PEMROGRAMAN JAVA DENGAN BAHASA PEMROGRAMAN LAIN 28](#_Toc131012811)

[4.2 KELEBIHAN DAN KEKIRANGAN PBO PADA JAVA 28](#_Toc131012812)

[BAB V 30](#_Toc131012813)

[CONTOH IMPLEMENTASI PBO 30](#_Toc131012814)

[PADA PROYEK NYATA MENGGUNAKAN JAVA 30](#_Toc131012815)

[5.1 STUDI KASUS 1: PEMBUATAN APLIKASI MANEJEMEN TOKO BUKU 30](#_Toc131012816)

[BAB VI 32](#_Toc131012817)

[KESIMPULAN DAN SARAN 32](#_Toc131012818)

[6.1 KESIMPULAN 32](#_Toc131012819)

[6.2 SARAN 32](#_Toc131012820)

[DAFTAR PUSTAKA 34](#_Toc131012821)

# BAB I

# PENDAHULUAN

## 1.1 LATAR BELAKANG

Pemrograman Berorientasi Objek (PBO) adalah paradigma pemrograman yang menggunakan objek sebagai dasar dari pembangunan sebuah sistem atau aplikasi. PBO memiliki banyak keunggulan dibandingkan dengan paradigma pemrograman lainnya, seperti memudahkan pemeliharaan kode, menghasilkan kode yang lebih bersih dan mudah dibaca, dan memungkinkan pengembangan perangkat lunak yang lebih modular dan reusable.

Salah satu bahasa pemrograman yang populer untuk mengimplementasikan konsep PBO adalah Java. Java merupakan bahasa pemrograman yang banyak digunakan di berbagai bidang, mulai dari pengembangan aplikasi desktop, mobile, hingga aplikasi web. Java memiliki dukungan penuh untuk konsep PBO dan menyediakan berbagai fitur yang memudahkan pengembangan aplikasi dengan paradigma PBO.

Dalam makalah PBO dengan fokus pada bahasa pemrograman Java, daftar isi dimulai dengan pendahuluan yang menjelaskan latar belakang penulisan makalah. Latar belakang tersebut bertujuan untuk memberikan pemahaman mengapa makalah ini perlu dibuat dan apa yang akan dibahas dalam makalah ini.

Salah satu alasan utama penulisan makalah ini adalah untuk memberikan pemahaman dasar mengenai konsep PBO pada bahasa pemrograman Java. Dalam daftar isi, konsep dasar PBO pada Java dibahas secara terperinci, seperti definisi PBO pada Java, karakteristik PBO pada Java, dan konsep abstraksi, encapsulation, inheritance, dan polymorphism pada Java. Konsep-konsep tersebut penting untuk dipahami agar pembaca dapat memahami dasar-dasar PBO pada Java.

Setelah itu, daftar isi membahas implementasi PBO pada bahasa pemrograman Java. Bahasa pemrograman Java memiliki berbagai fitur yang mendukung konsep PBO, seperti class, object, konstruktor, method, inheritance, dan interface. Semua fitur tersebut dibahas dalam daftar isi dengan tujuan untuk memberikan pemahaman praktis mengenai implementasi PBO pada Java.

Selanjutnya, daftar isi membahas analisis perbandingan PBO pada bahasa pemrograman Java dengan bahasa pemrograman lainnya. Bahasa pemrograman Java bukan satu-satunya bahasa yang mendukung konsep PBO. Oleh karena itu, penting untuk melakukan perbandingan antara bahasa pemrograman Java dengan bahasa pemrograman lainnya agar pembaca dapat memahami kelebihan dan kekurangan masing-masing bahasa pemrograman.

Terakhir, daftar isi membahas contoh implementasi PBO pada proyek nyata menggunakan bahasa pemrograman Java. Studi kasus tersebut dirancang untuk memberikan gambaran praktis tentang bagaimana PBO dapat diterapkan pada proyek nyata dan memberikan inspirasi bagi pembaca untuk mengembangkan aplikasi mereka sendiri.

Dalam kesimpulannya, daftar isi menyimpulkan keseluruhan isi makalah dan memberikan saran untuk pengembangan PBO pada Java di masa depan. Hal tersebut penting untuk memastikan bahwa pengembangan aplikasi dengan konsep PBO pada Java terus berkembang dan memenuhi kebutuhan pengguna.

## 1.2 TUJUAN PENULISAN MAKALAH

Tujuan utama penulisan makalah PBO dengan fokus pada bahasa pemrograman Java adalah untuk memberikan pemahaman dasar mengenai konsep PBO pada bahasa pemrograman Java dan bagaimana konsep tersebut dapat diterapkan pada proyek nyata.

Salah satu tujuan utama adalah untuk menjelaskan konsep dasar PBO pada Java dengan lebih terperinci, seperti definisi PBO pada Java, karakteristik PBO pada Java, dan konsep abstraksi, encapsulation, inheritance, dan polymorphism pada Java. Tujuan ini penting agar pembaca dapat memahami dasar-dasar PBO pada Java dan bagaimana konsep tersebut dapat diterapkan dalam pembangunan aplikasi.

Tujuan lainnya adalah untuk menjelaskan implementasi PBO pada bahasa pemrograman Java dengan menggunakan berbagai fitur yang mendukung konsep PBO, seperti class, object, konstruktor, method, inheritance, dan interface. Tujuan ini bertujuan untuk memberikan pemahaman praktis tentang implementasi PBO pada Java dan bagaimana penggunaan fitur-fitur tersebut dapat membantu dalam pembangunan aplikasi.

Selanjutnya, tujuan penulisan makalah ini adalah untuk melakukan analisis perbandingan antara PBO pada bahasa pemrograman Java dengan bahasa pemrograman lainnya. Hal ini penting agar pembaca dapat memahami kelebihan dan kekurangan masing-masing bahasa pemrograman dan dapat memilih bahasa pemrograman yang sesuai untuk proyek yang sedang dikerjakan.

Tujuan terakhir dari makalah ini adalah untuk memberikan contoh implementasi PBO pada proyek nyata menggunakan bahasa pemrograman Java. Tujuan ini bertujuan untuk memberikan gambaran praktis tentang bagaimana PBO dapat diterapkan pada proyek nyata dan memberikan inspirasi bagi pembaca untuk mengembangkan aplikasi mereka sendiri.

Dengan demikian, tujuan utama penulisan makalah PBO dengan fokus pada bahasa pemrograman Java adalah untuk memberikan pemahaman dasar dan praktis mengenai konsep PBO pada Java dan bagaimana konsep tersebut dapat diterapkan pada proyek nyata. Tujuan lainnya adalah untuk melakukan perbandingan dengan bahasa pemrograman lain dan memberikan contoh implementasi pada proyek nyata agar pembaca dapat memahami konsep PBO pada Java secara komprehensif.

## 1.3 RUANG LINGKUP

Ruang lingkup makalah PBO dengan fokus pada bahasa pemrograman Java mencakup beberapa hal, yaitu:

1. **Pengenalan PBO pada Java**

Pada bagian ini, makalah akan membahas definisi PBO pada Java, karakteristik PBO pada Java, dan konsep-konsep dasar seperti abstraksi, encapsulation, inheritance, dan polymorphism.

1. **Implementasi PBO pada Java**

Bagian ini akan membahas implementasi PBO pada Java menggunakan fitur-fitur seperti class, object, konstruktor, method, inheritance, dan interface. Di sini juga akan dibahas cara penggunaan fitur-fitur tersebut untuk membuat program PBO pada Java.

1. **Analisis perbandingan antara PBO pada Java dengan bahasa pemrograman lain**

Bagian ini akan membahas perbandingan antara PBO pada Java dengan bahasa pemrograman lain, seperti Python atau C++. Perbandingan ini akan mencakup kelebihan dan kekurangan masing-masing bahasa pemrograman dalam mendukung konsep PBO.

1. **Contoh implementasi PBO pada proyek nyata menggunakan bahasapemrograman Java**

Bagian ini akan memberikan contoh konkretnya mengenai implementasi PBO pada proyek nyata menggunakan bahasa pemrograman Java. Contoh ini bertujuan untuk memberikan gambaran tentang bagaimana konsep PBO pada Java dapat diterapkan pada proyek nyata.

1. Kesimpulan

Bagian terakhir dari makalah akan merangkum kesimpulan dari pembahasan sebelumnya dan memberikan penilaian mengenai pentingnya PBO pada bahasa pemrograman Java serta manfaatnya dalam pembangunan aplikasi.

Dengan demikian, ruang lingkup makalah ini meliputi pengenalan PBO pada Java, implementasi PBO pada Java, analisis perbandingan dengan bahasa pemrograman lain, contoh implementasi pada proyek nyata, dan kesimpulan.

# BAB II

# KONSEP DASAR

# PEMROGRAMAN BERORIENTASI OBJEK PADA JAVA

## 2.1 DEFINISI PBO PADA JAVA

Definisi PBO pada Java adalah salah satu konsep pemrograman yang sangat populer dan digunakan secara luas di dalam pembangunan aplikasi. PBO (Object-Oriented Programming) pada Java merupakan salah satu paradigma pemrograman yang menggunakan objek sebagai elemen dasar dari program. Dalam pemrograman PBO, objek yang terdiri dari data dan metode, digunakan untuk merepresentasikan suatu entitas atau objek di dunia nyata.

Pada Java, PBO sering digunakan untuk membangun aplikasi yang kompleks, seperti aplikasi desktop, aplikasi web, dan bahkan game. Definisi PBO pada Java dapat dilihat dari karakteristik yang dimilikinya, diantaranya adalah:

1. Objek:

Pada PBO, setiap objek memiliki identitas, data, dan perilaku. Identitas dari suatu objek ditentukan oleh alamat memori tempat objek tersebut disimpan. Data dari suatu objek dapat berupa variabel dan struktur data. Sementara itu, perilaku dari suatu objek dapat diwakili oleh metode yang ada pada objek tersebut.

1. Encapsulation:

Konsep encapsulation pada PBO bertujuan untuk menyembunyikan data dan perilaku objek agar tidak dapat diakses dari luar objek. Dalam Java, encapsulation dilakukan dengan menggunakan akses modifier (public, private, protected) untuk menentukan tingkat akses suatu data atau metode.

1. Inheritance:

Inheritance pada PBO memungkinkan suatu objek memiliki sifat atau perilaku dari objek lain yang menjadi induknya. Pada Java, inheritance dilakukan dengan menggunakan keyword extends dan superclass-subclass.

1. Polymorphism:

Polymorphism pada PBO memungkinkan suatu objek memiliki banyak bentuk atau perilaku. Pada Java, polymorphism dapat dilakukan dengan menggunakan overloading dan overriding.

## 2.2 KARAKTERISTIK PBO PADA JAVA

Pemrograman Berorientasi Objek (PBO) pada Java memiliki beberapa karakteristik khusus yang membedakannya dari paradigma pemrograman lainnya. Karakteristik tersebut meliputi encapsulation, inheritance, dan polymorphism.

**1. Abstrak**

Abtrak artinya Mengabaikan hal-hal dalam topik yang tidak terkait dengan tujuan saat ini dan fokus pada aspek yang terkait dengan tujuan saat ini. Ini untuk mengekstrak jenis hal tertentu di dunia nyata dan mengekspresikannya dengan kode program. [Abstrak](https://id.wikipedia.org/wiki/Pemrograman_berorientasi_objek) umumnya disebut kelas atau antarmuka. Abstraksi tidak bermaksud untuk memahami semua masalah, tetapi untuk memilih beberapa di antaranya, tanpa beberapa detail untuk saat ini. Abstraksi mencakup dua aspek, abstraksi data, tetapi abstraksi proses.

**2. Paket**

Enkapsulasi merupakan salah satu ciri dari berorientasi objek, dan ciri utama dari konsep objek dan kelas. Enkapsulasi adalah untuk mengelilingi proses dan data, dan akses ke data hanya dapat melalui antarmuka yang ditentukan. Seperti variabel pribadi, gunakan set dan dapatkan metode untuk mendapatkannya.

Enkapsulasi memastikan bahwa modul memiliki independensi yang baik, membuat pemeliharaan dan modifikasi program lebih mudah. Modifikasi program aplikasi terbatas pada bagian dalam kelas. Sehingga dampak modifikasi program aplikasi dapat diminimalkan.

**3. Warisan**

Model hierarkis untuk menghubungkan kelas, memungkinkan dan mendorong penggunaan kembali kelas, dan menyediakan cara untuk mengekspresikan kesamaan dengan jelas. Kelas baru dari objek dapat diturunkan dari kelas yang sudah ada. Proses ini disebut pewarisan kelas.

Kelas baru mewarisi karakteristik kelas asli, kelas baru disebut kelas turunan (subclass) dari kelas asli, dan kelas asli disebut kelas dasar (kelas induk) dari kelas baru. Kelas turunan dapat mewarisi metode dan variabel instan dari kelas induknya dan kelas dapat memodifikasi atau menambahkan metode baru agar lebih cocok untuk kebutuhan khusus. Oleh karena itu, dapat dikatakan bahwa pewarisan adalah menggunakan kembali kode kelas induk dan sekaligus mempersiapkan realisasi polimorfisme. Baca juga [Dasar Pemrograman Berorientasi Objek Di Pemrograman C++](https://labkom99.com/2021/05/dasar-pemrograman-berorientasi-objek.html)

**4. Polimorfisme**

Polimorfisme mengacu pada memungkinkan berbagai jenis objek untuk menanggapi pesan yang sama. Polimorfisme meliputi polimorfisme parametrik dan polimorfisme inklusi. Bahasa polimorfik memiliki keunggulan fleksibilitas, abstraksi, berbagi perilaku, berbagi kode, yang memecahkan masalah nama fungsi aplikasi yang sama.

Secara umum, penulisan ulang metode, kelebihan beban, dan tautan dinamis merupakan polimorfisme. Salah satu alasan mengapa java memperkenalkan konsep polimorfisme adalah untuk menutupi kekurangan fungsionalitas yang disebabkan oleh pewarisan kelas tunggal.

## 2.3 KONSEP ABSTRAKSI PADA JAVA

Di dalam Pemrograman Berorientasi Obyek (PBO) atau *Object Oriented Programming* (OOP), selain fitur [inheritance](https://blog.rosihanari.net/konsep-pewarisan-inheritance-di-java/), terdapat pula sebuah fitur di mana akan memudahkan kita, sebagai seorang programmer, untuk memikirkan ide dasar dalam mengkonstruksi sebuah class dengan fungsionalnya secara sederhana. Melalui fitur ini, untuk sementara kita bisa mengabaikan fungsionalnya secara detil dengan tidak menghilangkan fungsionalnya secara umum. Fitur ini disebut dengan abstraksi (*abstraction*).

Abstraksi juga merupakan salah satu kemampuan *problem solving* melalui pendekatan [berpikir komputasional (](https://en.wikipedia.org/wiki/Computational_thinking)*[computational thinking](https://en.wikipedia.org/wiki/Computational_thinking)*[)](https://en.wikipedia.org/wiki/Computational_thinking).

Artikel ini akan menjelaskan bagaimana konsep abstraksi secara lebih dalam di PBO, bagaimana karakteristiknya, cara implementasinya di Java, serta studi kasus penerapannya.

Untuk memudahkan pemahaman tentang konsep abstraksi, pertama akan diberikan sebuah contoh kasus. Misalkan kita diminta membuat program untuk menghitung luas dan keliling dari beberapa bangun datar, yaitu: persegi panjang, persegi, dan lingkaran, dengan pendekatan PBO. Untuk memudahkan ide pembuatan program, kita bisa menyederhanakan permasalahannya. Kita bisa mulai dari memikirkan *grand desain*nya, yaitu bagaimana kita mengkonstruksi sebuah class yaitu BangunDatar. Hal ini dikarenakan problem utamanya adalah tentang bangun datar. Selanjutnya, class BangunDatar secara umum memiliki fitur atau fungsional untuk menghitung luas dan keliling berdasarkan kasus yang diberikan. Untuk sementara, kita tidak perlu pedulikan unsur dari BangunDatar tersebut, dan bagaimana cara menghitung luas dan kelilingnya. Hal ini disebabkan unsur dan cara perhitungan luas dan keliling yang berbeda-beda untuk setiap bangun datar. Setelah kita memikirkan *grand desain* dari class BangunDatarnya, selanjutnya barulah kita pikirkan bagaimana mengidentifikasi unsur dari setiap jenis bangun datar, serta cara perhitungan luas dan kelilingnya secara detail.

Cara berpikir seperti di atas, berarti kita menerapkan konsep abstraksi. Class BangunDatar yang secara umum memiliki fungsionalitas untuk menghitung luas dan keliling, namun sifat keduanya masih abstrak. Hal ini disebabkan masih belum jelasnya unsur dari setiap bangun datar, serta bagaimana cara menghitung luas dan kelilingnya. Adapun detil dari penjabaran unsur bangun datar, serta cara perhitungan luas dan kelilingnya baru akan diberikan di setiap jenis bangun datar, yang dalam hal ini nantinya diposisikan sebagai subclass dari BangunDatar tersebut.

**Karakteristik Abstraksi**

Berikut ini beberapa karakteristik dari penerapan konsep abstraksi:

* Class yang masih bersifat abstrak tidak bisa dibuat obyeknya (instansiasi)
* Konsep abstraksi hanya bisa diterapkan melalui penerapan konsep pewarisan ([inheritance](https://blog.rosihanari.net/konsep-pewarisan-inheritance-di-java/))
* Jika ada sebuah class bersifat abstrak (superclass), maka subclassnya WAJIB mengimplementasikan method yang masih abstrak dari superclassnya.
* Jika subclassnya belum mengimplementasikan method superclass yang masih abstrak, maka subclassnya juga bersifat abstrak.

**Implementasi Abstraksi di Java**

Bagaimana implementasi abstraksi di Java? Pada bagian berikut ini akan dipaparkan bagaimana cara membuat class yang abstrak dan bagaimana mengimplementasikannya ke dalam subclassnya.

Untuk membuat class yang abstrak, caranya cukup mudah yaitu hanya menambahkan label ‘abstract’ pada method yang masih abstrak. Selanjutnya di depan nama classnya juga diberikan label ‘abstract‘. Perhatikan contoh di bawah ini.

**public** **abstract** **class** MyClass {

**void** myMethod1(){

        System.out.println("Hello");

    }

**abstract** **void** myMethod2();

}

Dalam contoh di atas, method myMethod2() sifatnya masih abstrak disebabkan belum jelasnya apa yang akan dilakukan dari method tersebut. Sedangkan myMethod1() tidaklah abstrak karena jelas dia akan mencetak “Hello” ketika dipanggil. Dikarenakan ada salah satu method di dalam class MyClass yang abstrak, sehingga secara umum class tersebut juga abstrak. Oleh karena itu perlu ditambahkan label ‘abstract’ di depan nama classnya.

Berdasarkan karakteristik abstraksi sebelumnya, kita tidak bisa melakukan instansiasi dari class MyClass dikarenakan keabstrakannya. Jika tetap dilakukan, maka akan error.

Selanjutnya, dari class MyClass yang masih abstrak, kita bisa membuat subclassnya. Subclass ini nantinya hukumnya WAJIB mengimplementasikan semua method yang masih abstrak dari superclassnya, dalam hal ini adalah myMethod2(). Perhatikan contoh berikut:

**public** **class** MyClass2 **extends** MyClass {

**void** myMethod2(){

        System.out.println("Hello World");

    }

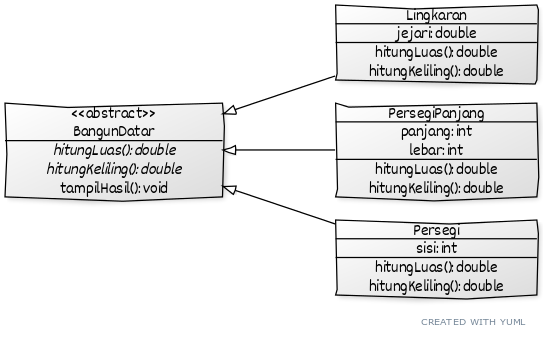
}

Apabila class MyClass2 belum mengimplementasikan atau menjabarkan apa yang akan dilakukan dari myMethod2() yang masih abstrak milik superclassnya, maka MyClass2 nantinya juga akan bersifat abstrak (perlu dilabeli ‘abstract‘ supaya tidak error). Apabila MyClass2 masih abstrak dan dia punya subclass lagi, misalnya MyClass3, maka MyClass3 ini wajib mengimplementasikan myMethod2(), jika tidak maka MyClass3 ini juga bersifat abstrak. Demikian seterusnya.

**Studi Kasus**

Setelah penjabaran tentang teknis bagaimana menerapkan abstraksi di Java, selanjutnya akan diberikan studi kasus yang menerapkan konsep tersebut. Kasus yang akan diambil adalah sama seperti contoh yang diberikan di awal tutorial ini, yaitu tentang membuat program untuk menghitung luas dan keliling bangun datar: persegi panjang, persegi, dan lingkaran.

Dengan menggunakan diagram UML, kita bisa mendesain struktur class-classnya sebagai berikut:



Dari diagram UML di atas, tampak bahwa kita bisa mendesain class BangunDatar yang bersifat abstrak. Secara umum class ini memiliki fitur untuk menghitung luas melalui method hitungLuas(), menghitung keliling dengan method hitungKeliling(), serta untuk menampilkan hasilnya (luas dan keliling) melalui tampilHasil(). Adapun hitungLuas() dan hitungKeliling(), keduanya masih abstrak (ditandai dengan tulisan italic pada UML) dikarenakan formula keduanya berbeda-beda untuk setiap bangun datar. Sedangkan untuk tampilHasil() tidaklah abstrak.

Berikut ini isi kode program Java dari class BangunDatar.

**public** **abstract** **class** BangunDatar {

    // method abstrak

**abstract** **double** hitungLuas();

**abstract** **double** hitungKeliling();

    // method non abstrak

**void** tampilHasil(){

        System.out.println("Luasnya: " + **this**.hitungLuas());

        System.out.println("Kelilingnya: " + **this**.hitungKeliling());

    }

}

Method tampilHasil() dalam hal ini tidak abstrak karena sudah jelas di dalamnya hanya menampilkan hasil hitung luas dan kelilingnya. Namun, proses perhitungan keduanya masih belum jelas (abstrak).

Selanjutnya akan dibuat class untuk setiap bangun datarnya, yaitu class PersegiPanjang, class Persegi, dan class Lingkaran. Apabila dilihat dari desain UML nya, maka ketiga class ini merupakan subclass dari class BangunDatar. Di setiap class ini nanti bisa ditambahkan atribut-atribut yang disesuaikan dengan class masing-masing, sesuai dengan unsur yang ada di setiap jenis bangun datar tersebut. Dikarenakan ketiga class tersebut adalah subclass dari BangunDatar yang abstrak, maka ketiganya wajib untuk mengimplementasikan method hitungLuas() dan hitungKeliling() yang masih abstrak.

Berikut ini adalah kode program untuk masing-masing class tersebut.

public class Lingkaran extends BangunDatar {

    int jejari;

    @Override

    double hitungLuas(){

        return Math.PI \* this.jejari \* this.jejari;

    }

    @Override

    double hitungKeliling(){

        return 2 \* Math.PI \* this.jejari;

    }

}

**public** **class** PersegiPanjang **extends** BangunDatar {

**int** panjang;

**int** lebar;

    @Override

**double** hitungLuas(){

**return** **this**.panjang \* **this**.lebar;

    }

    @Override

**double** hitungKeliling(){

**return** 2 \* (**this**.panjang + **this**.lebar);

    }

}

**public** **class** Persegi **extends** BangunDatar {

**int** sisi;

    @Override

**double** hitungLuas(){

**return** **this**.sisi \* **this**.sisi;

    }

    @Override

**double** hitungKeliling(){

**return** 4 \* **this**.sisi;

    }

}

Secara teknis, untuk mengimplementasikan method yang masih abstrak dari superclassnya adalah dengan melakukan *overriding* terhadap method tersebut, seperti yang ditunjukkan pada kode program Java di atas untuk setiap classnya.

Setelah subclass tiap jenis bangun datar dibuat, selanjutnya kita coba menerapkannya untuk menghitung luas beberapa bangun datar. Berikut ini adalah kode program untuk main class nya.

**public** **class** MainClass {

**public** **static** **void** main(String[] args) {

        // hitung luas & keliling lingkaran r=10

        Lingkaran l1 = **new** Lingkaran();

        l1.jejari = 10;

        l1.tampilHasil();

        // hitung luas & keliling persegi panjang p=10 l=5

        PersegiPanjang pp1 = **new** PersegiPanjang();

        pp1.panjang = 10;

        pp1.lebar = 5;

        pp1.tampilHasil();

        // hitung luas & keliling persegi s=7

        Persegi p1 = **new** Persegi();

        p1.sisi = 7;

        p1.tampilHasil();

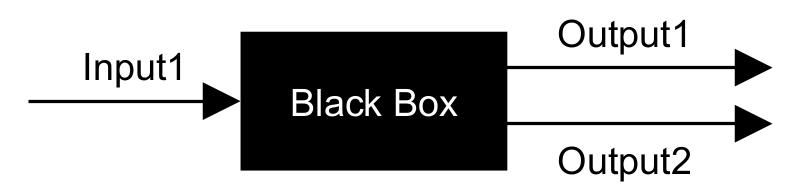
Mudah bukan konsep abstraksi ini? Melalui penerapan abstraksi seperti di atas, nantinya akan memudahkan bagi programmer untuk menambah lagi jenis bangun datar lainnya, misalnya bangun segitiga atau jajaran genjang. Dalam hal ini cukup dengan membuat subclass dari BangunDatar nya, saja dan menjabarkan formula untuk menghitung luas dan keliling masing-masing. Dengan penerapan abstraksi akan lebih memudahkan dalam memahami struktur ide pengembangan program, terutama struktur class-classnya.

## 2.4 KONSEP ENCAPSULUTION PADA JAVA

Enkapsulasi *(encapsulation)* adalah pembungkusan data atau penyembunyian data-data privat dari suatu obyek sehingga tidak dapat diakses dari obyek lain.

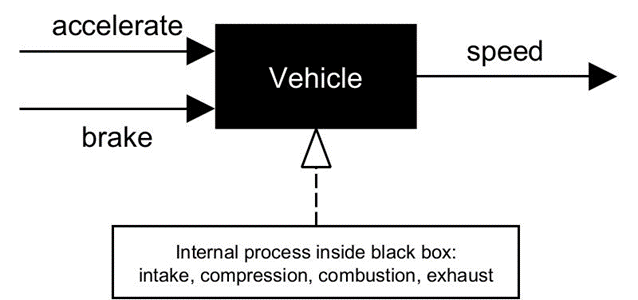
Dalam suatu obyek seringkali terdapat data-data privat yang hanya digunakan untuk operasi internal di dalam obyek tersebut. Data-data privat tersebut sebaiknya dilindungi untuk menghindari modifikasi dari obyek lain yang berpotensi mengganggu operasi internal dalam obyek. Perlindungan data-data privat juga dapat mengurangi kompleksitas kode karena pemrogram tidak perlu dipusingkan dengan data dan proses mikro. Sebaliknya, pemrogram hanya perlu fokus pada fungsi-fungsi publik yang memang disediakan sebagai sarana interaksi antar obyek.

Enkapsulasi dapat merubah cara pandang kita terhadap obyek dimana suatu obyek dapat diibaratkan sebagai black box. Dengan paradigma obyek sebagai black box, pemrogram tidak perlu mengetahui proses apa saja yang terjadi di dalam obyek (black box). Pemrogram hanya perlu memberikan input terhadap black box dan mengambil output dari proses yang dilakukan dalam black box.



Sebagai contoh pada kelas Vehicle (kendaraan) adalah sbb:

* Input dapat berupa fungsi-fungsi accelerate (akselerasi), brake (rem), dsb.
* Output dapat berupa getSpeed (mendapatkan kecepatan), dsb.
* Operasi internal kendaraan yang berada di dalam black box dapat berupa segala mekanisme teknis yang terjadi di dalam mesin seperti intake (pengambilan bahan bakar), compression (kompresibahanbakar), combustion (pembakaran), exhaust (pembuangan), dsb.



Untuk mengatur hak akses dari masing-masing method dan atribut digunakan modifier akses. Modifier akses yang paling umum adalah public yang artinya dapat diakses dari obyek manapun, dan private yang artinya hanya bisa diakses dalam obyek yang bersangkutan. Disamping public dan private ada modifier akses lainnya yang kita bahas pada artikel tentang modifier.

Method Setter

Method setter *(setter method)* adalah method publik yang berfungsi untuk memberikan nilai dari atribut yang bersifat privat. Pada enkapsulasi setiap atribut dapat dibuat privat agar data didalamnya terlindungi. Sebagai gantinya, kita dapat membuat sebuah method publik sebagai jembatan agar obyek lain dapat memberikan nilai dari atribut.

class Vehicle {

private int speed;

//setter method to set the value of speed

public void setSpeed(int speed) {

this.speed = speed;

}

}

Penjelasan program:

Atribut speed adalah atribut privat ditandai dengan kata kunci private sehingga speed tidak dapat diakses dari luar kelas Vehicle. Sebagai gantinya dibuat method setter yang bersifat publik yaitu setSpeed berfungsi agar obyek lain dapat memberi nilai pada speed. Method publik ditandai dengan kata kunci public yang artinya dapat diakses dari kelas lain.

Method Getter

Berkebalikan dengan method setter, method getter *(getter method)* adalah method yang berfungsi untuk mendapatkan nilai dari atribut privat.

class Vehicle {

private int speed;

//getter method to set the value of speed

public int getSpeed() {

return speed;

}

}

Penjelasan program:

Atribut speed adalah atribut privat ditandai dengan kata kunci private sehingga speed tidak dapat diakses dari luar kelas Vehicle. Sebagai gantinya dibuat method setter yang bersifat publik yaitu etgSpeed berfungsi agar obyek lain dapat mengambil nilai pada speed. Method publik ditandai dengan kata kunci public yang artinya dapat diakses dari kelas lain.

## 2.5 KONSEP INHERITANCE PADA JAVA

Di Java memungkinkan untuk mewariskan/menurunkan atribut dan method dari satu class ke class lainnya. Atau yang lebih dikenal dengan istilah pewarisan atau inheritance. Kita mengelompokkan konsep inheritance ke dalam dua kategori:

* **subclass** (anak) – class yang diturunkan dari class lainnya
* **superclass**(parent/orang tua) – class yang menurunkan/mewariskan

Untuk melakukan inheritance dari sebuah class, gunakan keyword extends.  
  
Pada contoh di bawah ini, class Mobil (subclass) mewarisi atribut dan method dari class Kendaraan (superclass). Contoh :

*class* Kendaraan {

protected String merk = "Ford"; // Atribut class kendaraan

public void klakson() {

System.out.println("Tiin, tiin!");

}

}

*class* Mobil extends Kendaraan {

private String namaModel = "Mustang"; // Atribut class mobil

public static void main(String[] args) {

// Membuat object mobil

Mobil mobilSaya = new Mobil();

// Panggil method klakson() (dari class kendaraan) pada object mobilSaya

mobilSaya.klakson();

// Menampilkan nilai dari atribut merk (dari class kendaraan) dan nilai dari nama model dari class Mobil

System.out.println(mobilSaya.merk + " " + mobilSaya.namaModel);

}

}

Output :

Tiin, tiin!

Ford Mustang

Pada contoh di atas kita menggunakan modifier protected dalam class Kendaraan. Kita menetapkan atribut **merk** pada class **Kendaraan**sebagai modifier protected, jika kita menetapkan sebagai private, class Mobil tidak dapat mengakses itu.

**MENGAPA MENGGUNAKAN INHERITANCE?**

Konsep inheritance atau pewarisan akan sangat berguna ketika suatu kode ingin digunakan kembali pada sebuah class baru atau menggunakan kembali atribut dan method dari sebuah class yang telah ada ketika Anda membuat sebuah class baru.

**KEYWORD FINAL**

Jika Anda tidak ingin class lain mewarisi dari sebuah class lainnya, gunakan keyword final. Contoh :

final *class* Kendaraan {

protected String merk = "Toyota";

public void klakson() {

System.out.println("Tiin, tiin!");

}

}

*class* Main extends Kendaraan {

private String namaModel = "Avanza";

public static void main(String[] args) {

Main mobilSaya = new Main();

mobilSaya.klakson();

System.out.println(mobilSaya.merk+ " " + mobilSaya.namaModel);

}

}

Maka akan menghasilkan seperti ini :

Main.java:8: error: cannot inherit from final Kendaraan

class Main extends Kendaraan {

^

1 error

## 2.6 KONSEP POLYMORPHISM PADA JAVA

*Polymorphism*adalah konsep pemrograman yang berorientasi pada objek yang mengacu pada kemampuan variabel, fungsi atau objek untuk mengambil beberapa bentuk. *Polymorphism*adalah penggunaan salah satu *item*seperti fungsi, atribut, atau *interface*pada berbagai jenis objek yang berbeda dalam bahasa pemrograman. Dalam bahasa pemrograman yang menunjukkan polimorfisme, objek kelas miliki hierarki yang sama yang diwariskan dari kelas induk yang sama, mungkin memiliki fungsi dengan nama yang sama, tetapi dengan perilaku berbeda.

*Polymorphism*memiliki beberapa keuntungan seperti:

* Membantu programmer menggunakan kembali kode dan kelas setelah ditulis, diuji dan diimplementasikan.
* Nama variabel tunggal dapat digunakan untuk menyimpan variabel dari beberapa tipe data (*float*, *double*, *long*, *int*, dll).
* Membantu menyusun abstraksi yang kuat dan kompleks dari yang lebih sederhana.

Ada dua aspek implementasi polymorphism, yaitu compile time polymorphism (contohnya metode overloading) dan runtime polymorphism (contohnya metode overriding).

1. Static polymorphism

Static polymorphism adalah tipe yang dilakukan pada waktu compile, yang dapat bekerja lebih cepat namun membutuhkan bantuan compiler tambahan. Contoh penggunaan tipe static polymorphism adalah static binding, compile-time binding, early binding, dan metode overloading pada sebuah class.

2. Dynamic polymorphism

Dynamic polymorphism adalah tipe yang dilakukan pada waktu program berjalan atau runtime. Implementasi dari dynamic polymorphism dianggap lebih fleksibel, namun bekerja lebih lambat dari tipe static polymorphism. Contoh penggunaan tipe dynamic polymorphism adalah dynamic binding, runtime binding, late binding, metode overriding pada class yang berbeda.

# BAB III

# IMPLEMENTASI PBO

# PADA PEMROGRAMAN JAVA

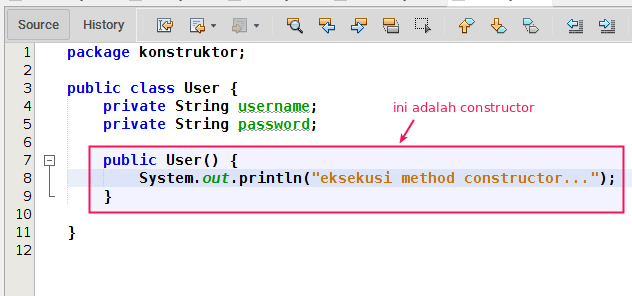
## 3.1 KONSTRUKTOR PADA JAVA

Constructor adalah method khusus yang akan dieksekusi pada saat pembuatan objek *(instance)*.

Biasanya method ini digunakan untuk inisialisasi atau mempersiapkan data untuk objek.

**Contoh Constructor dan Cara Membuatnya**

Berikut ini contoh Constructor:



Cara membuat constructor adalah dengan menuliskan nama method constructor sama seperti nama class.

Pada contoh di atas constructor ditulis seperti ini:

*public* User() {

System.out.println("eksekusi method constructor...");

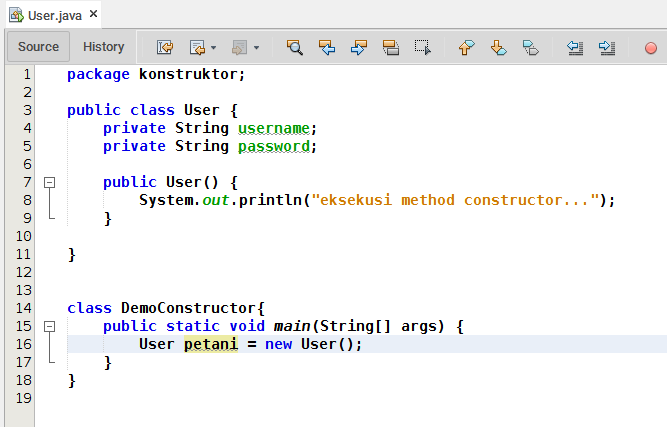
}

Pastikan memberikan modifier **public** kepada Constructor, karena ia akan dieksekusi saat pembuatan objek *(instance)*.

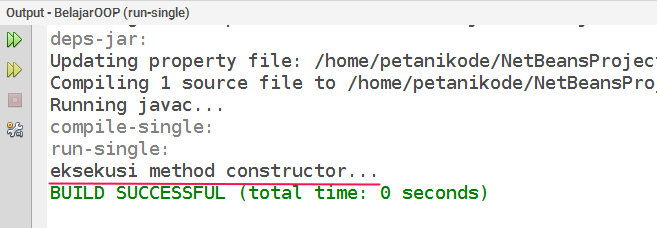
Mari kita coba membuat objek baru dari class **User**:

User petani = new User();

Sehingga sekarang kita punya kode lengkap seperti ini:



Hasilnya saat dieksekusi:



**Constructor dengan Parameter**

Constructor biasanya digunakan untuk *initialize* (menyiapkan) data untuk class.

Untuk melakukan ini, kita harus membuat parameter sebagai inputan untuk constructor.

Contoh:

*public* *class* User {

*public* String username;

*public* String password;

*public* User(String username, String password){

this.username = username;

this.password = password;

}

}

Pada kode class **User** di atas, kita menambahkan parameter **username** dan **password** ke dalam constructor.

Berarti nanti saat kita membuat objek, kita harus menambahkan nilai parameter seperti ini:

User petani = new User("petanikode", "kopi");

Contoh Kode lengkapnya:

package konstruktor;

*public* *class* User {

*public* String username;

*public* String password;

*public* User(String username, String password){

this.username = username;

this.password = password;

}

}

*class* DemoConstructor{

*public* *static* void main(String[] args) {

User petani = new User("petanikode", "kopi");

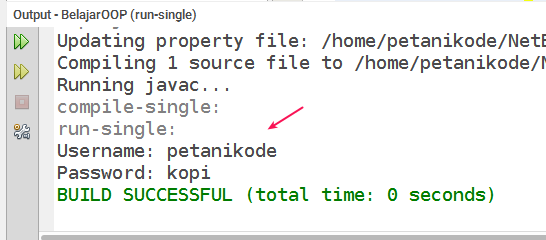
System.out.println("Username: " + petani.username);

System.out.println("Password: " + petani.password);

}

}

Hasil outputnya:



## 3.1 KELAS DAN OBJEK PADA JAVA

1. Objek.

Yang dimaksud dengan objek pada java adalah sekumpulan software yang terdiri dari variable dan method-method yang terkait. Objek juga merupakan benda nyata yang di buat berdasarkan rancangan yang di definisikan di dalam class

Object adalah instance dari class. Jika class secara umum mepresentasikan (template) sebuah object, sebuah instance adalah representasi nyata dari class itu sendiri. Contoh : Dari class Fruit kita dapat membuat object Mangga, Pisang, Apel, dan lain-lain.

1. Class

Class adalah prototype, atau blueprint, atau rancangan yang mendefinisikan variable dan method-methode pada seluruh objek tertentu. Class berfungsi untuk menampung isi dari program yang akan di jalankan, di dalamnya berisi atribut / type data dan method untuk menjalankan suatu program.

Class merupakan suatu blueprint atau cetakan untuk menciptakan suatu instant dari  object. class juga merupakan grup suatu object dengan kemiripan attributes/properties, behaviour dan relasi ke object lain. Contoh : Class Person, Vehicle, Tree, Fruit dan lain-lain.

## 3.3 METHOD PADA JAVA

Method adalah kumpulan program yang mempunyai nama. Method merupakan sarana bagi programmer untuk memecah program menjadi bagian-bagian yang kecil agar jadi lebih kompleks sehingga dapat di gunakan berulang-ulang.

Method merupakan suatu operasi berupa fungsi-fungsi yang dapat dikerjakan oleh suatu object. Method didefinisikan pada class akan tetapi dipanggil melalui object. Contoh, pada object pear : terdapat method ambilRasa , kupasKulit dan lainlain

## 3.4 INHERITANCE PADA JAVA

Inheritance merupakan proses pewarisan data dan method dari suatu class yang telah ada kepada suatu class baru. Class yang mewariskan disebut dengan superclass / parent class / base class, sedangkan class yang mewarisi (class yang baru) disebut dengan subclass / child class / derived class.

## 3.5 INTERFACES PADA JAVA

Interface merupakan suatu mekanisme yang disediakan oleh java yang memungkinkan berbagi konstanta atau menentukan bentuk metode yang dapat digunakan oleh sejumlah kelas bentuk metode yang dapat digunakan oleh sejumlah kelas.

## 3.6 ABTRAKSI PADA JAVA

Abstraksi adalah konsep penting dalam Pemrograman Berorientasi Objek (PBO) pada bahasa pemrograman Java. Abstraksi adalah teknik pemrograman yang digunakan untuk menyembunyikan detail yang tidak relevan dalam suatu objek dan menampilkan hanya informasi yang relevan untuk pemakai objek tersebut.

Dalam bahasa pemrograman Java, abstraksi dapat dicapai dengan menggunakan kelas abstrak dan interface. Kelas abstrak adalah kelas yang tidak dapat di-instantiate, artinya tidak dapat dibuat objeknya, namun digunakan sebagai kerangka untuk membuat kelas turunan yang dapat di-instantiate. Kelas turunan ini akan mengimplementasikan semua method yang diwarisi dari kelas abstrak tersebut.

# BAB IV

# ANALISA PERBANDINGAN

# PBO PADA PEMROGRAMAN JAVA

## 4.1 PERBANDINGAN PBO PADA BAHASA PEMROGRAMAN JAVA DENGAN BAHASA PEMROGRAMAN LAIN

Pemrograman Berorientasi Objek (PBO) adalah paradigma pemrograman yang sangat populer dan telah digunakan dalam banyak bahasa pemrograman, termasuk bahasa pemrograman Java. Namun, selain Java, terdapat juga bahasa pemrograman lain yang mendukung konsep PBO, seperti C++, Python, Ruby, dan PHP. Oleh karena itu, perbandingan antara PBO pada bahasa pemrograman Java dengan bahasa pemrograman lain menjadi penting untuk dipahami.

Pertama, perbandingan antara Java dan C++ dalam hal PBO dapat dilakukan. Meskipun Java dan C++ memiliki banyak kesamaan dalam hal konsep PBO, Java memiliki beberapa kelebihan yang tidak dimiliki oleh C++. Salah satu kelebihan Java adalah memungkinkan pengembang untuk membuat program yang lebih aman dan stabil, karena Java memiliki mekanisme garbage collection yang membantu mengelola memori secara otomatis dan mengurangi kemungkinan terjadinya kesalahan memori.

Selain itu, Java memiliki kemampuan untuk melakukan pengembangan aplikasi berbasis web dan mobile yang kuat, sehingga Java menjadi salah satu bahasa pemrograman yang paling populer untuk pengembangan aplikasi di bidang enterprise dan web. Di sisi lain, C++ biasanya digunakan untuk pengembangan sistem operasi dan perangkat lunak yang membutuhkan performa tinggi.

Kedua, perbandingan antara Java dan Python juga dapat dilakukan dalam hal PBO. Python adalah bahasa pemrograman yang mudah dipelajari dan mudah digunakan, dengan sintaks yang sederhana dan mudah dibaca. Namun, Java memiliki kelebihan dalam hal kinerja dan skalabilitas, yang menjadikannya lebih cocok untuk pengembangan aplikasi besar dan kompleks.

Selain itu, Python memiliki kelemahan dalam hal abstraksi, sehingga sulit untuk mengimplementasikan konsep PBO yang kompleks dalam bahasa pemrograman ini. Di sisi lain, Java menawarkan dukungan yang kuat untuk konsep PBO, seperti class, object, inheritance, dan polymorphism, yang memungkinkan pengembang untuk membuat program yang lebih mudah dipelihara dan diperbarui.

## 4.2 KELEBIHAN DAN KEKIRANGAN PBO PADA JAVA

1. Alami (Natural)

Pendekatan terhadap object yang menggambarkan segala sesuatu yang nyata, seperti sifat suatu benda maupun kegunaan dari benda tersebut. Jadi suatu objek mempunyai sifat benda berupa data yang logis dan memiliki kegunaan.

2. Modularity

Setiap object yang dibentuk dikelola secara terpisah dari object lainnya meskipun berasal dari sebuah

kelas yang sama. Modifikasi terhadap sebuah object bisa dilakukan tanpa mempengaruhi fungsionalitas dari object yang lainnya. Jadi ketika kita membuat suatu objek dan ditampilkan kita bisa memisah dan menampilkannya.

3. Reusable

Class yang telah dibuat dalam pemrograman berorientasi object bisa digunakan lagi pada program yang lain. Jadi ketika membuat objek untuk bahasa pemrograman dan objek tersebut terdiri dari class-class yang terpisah. Class tersebut bisa digunakan lagi walapun sifatnya dinamis.

4. Informasi Hiding

Menggunakan object tanpa harus tahu teknis yang ada didalamnya serta memiliki kemampuan untuk menyembunyikan ataupun mengatur jangkauan akses sebuah data. Jadi setiap objek class ter enkapsulasi terbungkus dan ter enkripsi serta memiliki hak akses.

**Kekurangan PBO/OOP (Object Oriented Programing) :**

1. Memiliki memori lebih besar dibandingkan dengan program terstruktur.

2. Mudah mengembalikan kode jadi menjadi kode sumber sehingga algoritma yang digunakan pun sulit untuk disembunyikan dan mudah dibajak.

# BAB V

# CONTOH IMPLEMENTASI PBO

# PADA PROYEK NYATA MENGGUNAKAN JAVA

## 5.1 STUDI KASUS 1: PEMBUATAN APLIKASI MANEJEMEN TOKO BUKU

Aplikasi manajemen toko buku merupakan salah satu aplikasi yang memanfaatkan konsep PBO, seperti class, object, inheritance, dan polymorphism, untuk membantu manajemen toko buku dalam mengelola stok buku, pemesanan buku, dan penjualan buku. Berikut adalah beberapa langkah dalam mengimplementasikan PBO pada proyek aplikasi manajemen toko buku menggunakan bahasa pemrograman Java.

1. Analisis kebutuhan dan perancangan sistem

Pada tahap awal, dilakukan analisis kebutuhan dan perancangan sistem untuk memastikan bahwa aplikasi manajemen toko buku sesuai dengan kebutuhan pengguna. Selain itu, juga dilakukan perancangan class dan object yang dibutuhkan dalam aplikasi, seperti class Book, Customer, Order, dan Payment.

1. Pembuatan class dan object

Setelah perancangan sistem dan class selesai, langkah selanjutnya adalah membuat class dan object yang diperlukan untuk menjalankan aplikasi manajemen toko buku. Dalam hal ini, setiap class akan memiliki atribut dan method yang sesuai dengan fungsinya masing-masing, seperti class Book memiliki atribut judul, pengarang, dan harga, serta method untuk menambah dan mengubah data buku.

1. Implementasi inheritance

Selanjutnya, implementasi inheritance dilakukan untuk mempermudah pengelolaan class dan object pada aplikasi manajemen toko buku. Misalnya, class Order dapat diwarisi dari class Book dan Customer untuk memudahkan pengelolaan pesanan buku oleh pelanggan.

1. Implementasi polymorphism

Polymorphism juga diterapkan pada aplikasi manajemen toko buku untuk memungkinkan objek yang sama dapat digunakan dalam berbagai situasi. Misalnya, objek Order dapat digunakan sebagai objek pembelian oleh pelanggan, dan juga sebagai objek stok buku oleh toko.

1. Pengujian dan debugging

Setelah aplikasi selesai dibuat, langkah terakhir adalah melakukan pengujian dan debugging untuk memastikan bahwa aplikasi manajemen toko buku berjalan dengan baik dan sesuai dengan kebutuhan pengguna.

Dalam implementasi PBO pada proyek aplikasi manajemen toko buku menggunakan bahasa pemrograman Java, dapat terlihat bahwa konsep PBO dapat membantu dalam memudahkan pengembangan aplikasi yang kompleks dan membantu meningkatkan efisiensi dalam pengelolaan data. Selain itu, dengan mengimplementasikan konsep PBO, aplikasi menjadi lebih mudah dipelihara dan diperbarui di masa depan.

# BAB VI

# KESIMPULAN DAN SARAN

## 6.1 KESIMPULAN

dapat disimpulkan bahwa makalah ini membahas tentang konsep Pemrograman Berorientasi Objek (PBO) pada bahasa pemrograman Java. PBO merupakan paradigma pemrograman yang fokus pada objek sebagai unit dasar dalam pemrograman. PBO pada Java memiliki beberapa karakteristik, seperti encapsulation, inheritance, dan polymorphism, yang dapat membantu memudahkan pengembangan aplikasi.

Selain itu, makalah ini juga membahas tentang konsep abstraksi pada Java, yaitu cara untuk menyembunyikan detail implementasi dari sebuah objek dan hanya menampilkan fungsionalitas yang dibutuhkan. Implementasi PBO pada proyek nyata juga dicontohkan melalui studi kasus pembuatan aplikasi manajemen toko buku dan aplikasi e-commerce sederhana.

Dari kedua studi kasus tersebut, dapat dilihat bahwa PBO dapat membantu memudahkan pengembangan aplikasi yang kompleks dan meningkatkan efisiensi dalam pengelolaan data. Selain itu, dengan mengimplementasikan konsep PBO, aplikasi menjadi lebih mudah dipelihara dan diperbarui di masa depan.

dapat disimpulkan bahwa pemahaman tentang konsep PBO pada Java sangat penting dalam pengembangan aplikasi modern. Oleh karena itu, para pengembang perlu mempelajari konsep PBO secara mendalam untuk menghasilkan aplikasi yang berkualitas dan mudah dipelihara di masa depan.

## 6.2 SARAN

Berikut adalah beberapa saran untuk pengembangan Pemrograman Berorientasi Objek (PBO) pada bahasa pemrograman Java di masa depan:

1. Mengembangkan library PBO yang lebih kaya dan beragam: Java memiliki library PBO yang sudah cukup lengkap, namun masih ada potensi untuk mengembangkan library PBO yang lebih kaya dan beragam, terutama untuk aplikasi yang lebih spesifik. Library PBO yang lebih lengkap dan beragam akan memudahkan para pengembang dalam mengembangkan aplikasi yang lebih kompleks.
2. Meningkatkan performa aplikasi: Dalam mengembangkan aplikasi PBO pada Java, performa aplikasi dapat menjadi masalah jika tidak dikelola dengan baik. Oleh karena itu, pengembang perlu terus meningkatkan performa aplikasi agar dapat menjalankan proses dengan lebih efisien.
3. Memperbarui fitur-fitur PBO: Seiring berkembangnya teknologi, fitur-fitur PBO pada Java juga perlu diperbarui agar sesuai dengan kebutuhan pengembangan aplikasi yang lebih modern dan kompleks. Pembaruan fitur-fitur PBO pada Java dapat membantu para pengembang dalam mengembangkan aplikasi dengan lebih efisien dan menghasilkan aplikasi yang lebih berkualitas.
4. Mengintegrasikan PBO dengan teknologi lain: PBO pada Java dapat diintegrasikan dengan teknologi lain seperti database, jaringan, dan mobile development untuk menghasilkan aplikasi yang lebih komprehensif. Oleh karena itu, pengembang perlu terus mengintegrasikan PBO dengan teknologi lain agar dapat menghasilkan aplikasi yang lebih canggih dan efisien.
5. Meningkatkan penggunaan PBO pada Java di masa depan: Pemrograman Berorientasi Objek merupakan paradigma pemrograman yang penting dan sering digunakan dalam pengembangan aplikasi modern. Oleh karena itu, penggunaan PBO pada Java perlu terus ditingkatkan agar pengembang dapat menghasilkan aplikasi yang berkualitas dan efisien.

# DAFTAR PUSTAKA

* <https://ocw.upj.ac.id/files/GBPP-TIF301-Handout-DPBO.pdf>
* <http://duniakomputerdanpemrograman.blogspot.com/2012/04/cara-pengambilan-data-dari-combo-box.html>
* <https://dosenit.com/java/encapsulation-di-java-object-oriented-4>
* <https://www.gamelab.id/news/1171-java--menggunakan-konsep-inheritance-agar-kode-pemrograman-menjadi-lebih-terstruktur>
* <https://nextgen.co.id/inheritance-pada-java/>
* <https://blog.rosihanari.net/konsep-abstraksi-di-java/>
* <https://www.dicoding.com/blog/pengertian-polimorfisme-dalam-pemrograman-java/>
* <https://repository.dinus.ac.id/docs/ajar/Karakteristik_PBO.pdf>
* <https://www.academia.edu/31820055/penjelasan_3_karakteristik_metodologi_berorientasi_objek_dan_contohnya>
* <https://labkom99.com/2021/12/karakteristik-berorientasi-objek.html>
* <https://www.petanikode.com/java-oop-constructor/>