RESUME

FUNDAMENTAL OF PROGRAMMING WORKSHOP



DOSEN PENGAJAR

Indrawati, SST.MT

DISUSUN OLEH:

Kelompok I:

Diki Candra

Kelas : TRKJ-1B

POLITEKNIK NEGERI LHOKSEUMAWE

TEKNOLOGI KOMUNIKASI DAN KOMPUTER

D-IV TEKNOLOGI REKAYASA KOMPUTER DAN JARINGAN

2022/2023

Daftar Isi

[BAB 9 2](#_Toc138191474)

[Pemograman Berorientasi Objek 2](#_Toc138191475)

[9.1 konsep pemograman berorientasi objek 2](#_Toc138191476)

[9.2 Menggunakan objek 2](#_Toc138191477)

[9.3 Menciptakan kelas 2](#_Toc138191478)

[9.4 Menggunakan kelas dalam pemograman 3](#_Toc138191479)

[9.5 Kelas dengan metode 3](#_Toc138191480)

[9.6 Kata Kunci this 4](#_Toc138191481)

[9.7 Lingkup Variabel 4](#_Toc138191482)

[9.8 Penetu Akses : private dan public 4](#_Toc138191483)

[9.9 Metode Kelas 5](#_Toc138191484)

[BAB 10 5](#_Toc138191485)

[Konstruktor dan Overloading 5](#_Toc138191486)

[10.1 Konstruktor 5](#_Toc138191487)

[10.2 Overloading terhadap Konstruktor 5](#_Toc138191488)

[10.3 Overloading pada Metode 5](#_Toc138191489)

[10.4 variabel kelas 6](#_Toc138191490)

[BAB 11 6](#_Toc138191491)

[Pewarisan 6](#_Toc138191492)

[11.1 Pengertian Pewarisan 6](#_Toc138191493)

[11.2 Cara Mewariskan Kelas 6](#_Toc138191494)

[11.3 Penentu Akses Protected 7](#_Toc138191495)

[11.4 Memanggil Metode Superkelas 7](#_Toc138191496)

[11.5 Memanggil Konstruktor Superkelas 7](#_Toc138191497)

[11.6 Penentu Akses final 7](#_Toc138191498)

[11.7 Objek dalam kelas 7](#_Toc138191499)

[11.8 Penentu Abstract 7](#_Toc138191500)

[11.9 Ikatan dimanis pada variabel bertipe kelas 7](#_Toc138191501)

[BAB 12 8](#_Toc138191502)

[Paket 8](#_Toc138191503)

[12.1 Pengantar Paket 8](#_Toc138191504)

[12.2 Mendeklarasikan Paket 8](#_Toc138191505)

[12.3 Menggunakan import 8](#_Toc138191506)

[BAB 13 8](#_Toc138191507)

[Interface 8](#_Toc138191508)

[13.1 Pengertian Interface 8](#_Toc138191509)

[13.2 Mendeklarasikan Interface 9](#_Toc138191510)

[13.3 Pewarisan Interface 9](#_Toc138191511)

[13.4 Menggunakan Interface 9](#_Toc138191512)

[BAB 14 9](#_Toc138191513)

[Penanganan Eksepsi 9](#_Toc138191514)

[14.1 Penanganan Eksepsi 9](#_Toc138191515)

[14.1 Menangkap Eksepsi 10](#_Toc138191516)

[14.3 Catch secara Bertingkat 10](#_Toc138191517)

[14.4 Melontarkan Eksepsi 10](#_Toc138191518)

[14.5 Melontarkan kembali Eksepsi 10](#_Toc138191519)

[14.6 Klausa throws 10](#_Toc138191520)

# BAB 9

# Pemograman Berorientasi Objek

## 9.1 konsep pemograman berorientasi objek

Konsep pemrograman berorientasi objek (OOP) adalah cara berpikir tentang aplikasi yang mempelajari untuk berpikir bahwa aplikasi bukan sekedar prosedur melainkan sebagai objek. Objek yang dimaksud disini memiliki pengertian suatu modul yang mengkombinasikan antara data dan kode program yang bekerja sama dalam program dengan melewatkan proses satu sama lain.

## 9.2 Menggunakan objek

Menggunakan objek dalam pemrograman berorientasi objek memungkinkan kita untuk membuat objek baru dari suatu kelas, mengakses atribut yang telah didefinisikan dalam kelas tersebut, dan memanggil metode yang terkait dengan kelas. Dengan menggunakan objek, kita dapat mengorganisasi dan mengelompokkan kode program ke dalam struktur yang lebih teratur dan terorganisir.

## 9.3 Menciptakan kelas

class Mahasiswa:

def \_\_init\_\_(self, nama, nim):

self.nama = nama

self.nim = nim

def perkenalan(self):

print("Halo, nama saya", self.nama)

print("NIM saya adalah", self.nim)

# Membuat objek mahasiswa1

mahasiswa1 = Mahasiswa("John Doe", "123456")

# Mengakses atribut dan memanggil metode objek

print(mahasiswa1.nama) # Output: John Doe

print(mahasiswa1.nim) # Output: 123456

mahasiswa1.perkenalan() # Output: Halo, nama saya John Doe \n NIM saya adalah 123456

Dalam contoh tersebut, kita menciptakan kelas "Mahasiswa" dengan atribut "nama" dan "nim", serta metode "perkenalan". Metode \_\_init\_\_ merupakan metode khusus yang digunakan untuk menginisialisasi objek ketika dibuat. Metode perkenalan digunakan untuk mencetak pesan perkenalan objek Mahasiswa.

## 9.4 Menggunakan kelas dalam pemograman

Menggunakan kelas dalam pemrograman berorientasi objek memungkinkan kita untuk mengorganisasi kode program menjadi unit-unit yang lebih terstruktur dan terorganisir. Setiap objek yang dibuat dari kelas akan memiliki atribut-atribut yang sesuai dengan kelas tersebut, serta dapat memanggil metode-metode yang telah didefinisikan dalam kelas. Dengan menggunakan kelas, kita dapat menciptakan program yang lebih modular, mudah dipahami, dan dapat digunakan kembali.

## 9.5 Kelas dengan metode

Kelas dalam Java adalah blueprint untuk objek. Kelas dapat memiliki metode yang mendefinisikan perilaku objek tersebut. Metode adalah tindakan yang dapat dilakukan oleh objek. Contoh kelas dengan metode adalah MyClass dengan metode printHello() yang mencetak pesan dan metode sum() yang menghitung penjumlahan.

## 9.6 Kata Kunci this

Kata kunci "this" merujuk pada objek saat ini dalam kode Java. Digunakan untuk mengakses variabel instans, memanggil konstruktor lain, mengirim objek saat ini, dan menggunakan metode dari kelas lain.

## 9.7 Lingkup Variabel

Lingkup variabel menentukan di mana variabel dapat diakses dan digunakan dalam program. Ada beberapa jenis lingkup variabel yang dapat digunakan dalam Java:

1. Lingkup metode: Variabel hanya dapat diakses di dalam metode di mana mereka dideklarasikan.

2. Lingkup blok: Variabel hanya dapat diakses di dalam blok kode di mana mereka dideklarasikan, seperti blok if atau blok for.

3. Lingkup kelas: Variabel dapat diakses oleh semua metode dan blok kode dalam kelas.

4. Lingkup parameter: Variabel hanya dapat diakses di dalam metode di mana mereka menjadi parameter.

5. Lingkup instans: Variabel terkait dengan objek yang dibuat dari kelas dan dapat diakses oleh semua metode dan blok kode dalam kelas.

Pemahaman tentang lingkup variabel membantu dalam mengorganisir dan mengelola data dengan baik dalam program Java.

## 9.8 Penetu Akses : private dan public

`private` digunakan untuk mengakses anggota hanya di dalam kelas itu sendiri, sedangkan `public` digunakan untuk mengakses anggota dari mana saja.

## 9.9 Metode Kelas

Metode kelas adalah blok kode dalam sebuah kelas yang dapat melakukan tindakan atau mengembalikan nilai.

# BAB 10

# Konstruktor dan Overloading

## 10.1 Konstruktor

Konstruktor adalah metode khusus dalam sebuah kelas yang digunakan untuk membuat objek dan menginisialisasi nilainya.

## 10.2 Overloading terhadap Konstruktor

Overloading konstruktor adalah konsep di mana sebuah kelas memiliki beberapa konstruktor dengan nama yang sama, tetapi dengan parameter yang berbeda. Hal ini memungkinkan pembuatan objek dengan variasi parameter yang berbeda. Contoh penggunaan overloading konstruktor adalah saat kita memiliki konstruktor tanpa parameter, konstruktor dengan satu parameter, dan konstruktor dengan beberapa parameter yang berbeda. Dengan overloading konstruktor, kita dapat membuat objek dengan parameter yang sesuai dengan kebutuhan aplikasi.

## 10.3 Overloading pada Metode

Overloading metode adalah konsep di mana sebuah kelas memiliki beberapa metode dengan nama yang sama, tetapi dengan parameter yang berbeda. Hal ini memungkinkan pemanggilan metode yang sama dengan variasi parameter yang berbeda. Contohnya, kita dapat memiliki metode "add" dengan parameter int dan metode "add" dengan parameter double. Dengan overloading metode, kita dapat menggunakan metode yang sesuai dengan tipe data parameter yang diberikan.

## 10.4 variabel kelas

Variabel kelas adalah variabel yang dideklarasikan di dalam kelas dan dapat diakses oleh semua metode dalam kelas tersebut. Setiap objek yang dibuat dari kelas memiliki salinan independen dari variabel kelas. Variabel kelas digunakan untuk menyimpan data atau keadaan yang berkaitan dengan objek tersebut.

# BAB 11

# Pewarisan

## 11.1 Pengertian Pewarisan

Pewarisan adalah konsep dalam pemrograman berorientasi objek di mana sebuah kelas dapat mewarisi properti dan metode dari kelas lain yang disebut kelas induk atau superclass. Dengan pewarisan, kelas anak atau subclass dapat memperluas dan mengubah perilaku kelas induk serta mewarisi fitur-fiturnya. Hal ini memungkinkan untuk membuat hierarki kelas yang lebih terorganisir dan memungkinkan penggunaan kembali kode yang ada.

## 11.2 Cara Mewariskan Kelas

Pewarisan kelas adalah konsep dalam pemrograman berorientasi objek di mana kelas turunan dapat mewarisi semua anggota dari kelas super yang telah didefinisikan sebelumnya.

## 11.3 Penentu Akses Protected

Penentu akses `protected` digunakan untuk memberikan akses terbatas kepada kelas-kelas turunan dan kelas-kelas dalam paket yang sama.

## 11.4 Memanggil Metode Superkelas

Dalam pemrograman berorientasi objek, kita dapat menggunakan kata kunci `super` untuk memanggil metode dari kelas super (superclass). Ini memungkinkan kita untuk menjalankan implementasi metode dari kelas super sebelum menambahkan logika tambahan di dalam metode turunan.

## 11.5 Memanggil Konstruktor Superkelas

Untuk memanggil konstruktor superkelas, gunakan kata kunci **super** diikuti oleh argumen yang sesuai dalam konstruktor kelas anak. Ini memungkinkan eksekusi logika inisialisasi superkelas sebelum melanjutkan dengan logika inisialisasi kelas anak.

## 11.6 Penentu Akses final

Penentu akses **final** digunakan untuk membuat entitas yang tidak dapat diubah setelah inisialisasi.

## 11.7 Objek dalam kelas

Objek adalah representasi konkret dari sebuah kelas dengan atribut dan metode yang dapat digunakan dalam program.

## 11.8 Penentu Abstract

Penentu akses **abstract** digunakan untuk mendeklarasikan kelas atau metode yang tidak dapat digunakan secara langsung, melainkan harus diwarisi dan diimplementasikan oleh kelas turunannya.

## 11.9 Ikatan dimanis pada variabel bertipe kelas

Ikatan dinamis memungkinkan pemilihan metode berdasarkan objek aktual saat program berjalan, bukan tipe variabel referensi.

# BAB 12

# Paket

## 12.1 Pengantar Paket

Paket digunakan untuk mengorganisasi kelas-kelas terkait dan menghindari konflik nama dalam program Java.

## 12.2 Mendeklarasikan Paket

Untuk mendeklarasikan paket dalam Java, buat direktori dengan nama paket yang diinginkan, buat file Java dengan deklarasi paket pada baris pertama, dan simpan file tersebut dalam direktori paket yang sesuai.

## 12.3 Menggunakan import

Dalam Java, `import` digunakan untuk mengimpor kelas-kelas dari paket lain. Dengan mengimpor kelas, kita dapat menggunakan kelas tersebut tanpa menuliskan nama paket secara lengkap setiap kali.

# BAB 13

# Interface

## 13.1 Pengertian Interface

Interface adalah kontrak yang mendefinisikan metode-metode yang harus diimplementasikan oleh kelas-kelas yang menggunakannya. Ini mendorong polimorfisme dan memungkinkan desain modular dalam pemrograman.

## 13.2 Mendeklarasikan Interface

Untuk mendeklarasikan interface dalam Java, gunakan kata kunci `interface` diikuti dengan nama interface dan deklarasikan metode-metode yang harus diimplementasikan oleh kelas-kelas yang menggunakannya.

## 13.3 Pewarisan Interface

Pewarisan Interface adalah konsep di mana sebuah interface dapat mewarisi metode-metode dari interface lainnya. Ini memungkinkan interface untuk memiliki metode-metode dari interface lain dan menambahkan metode-metode baru jika diperlukan.

## 13.4 Menggunakan Interface

Untuk menggunakan interface dalam Java:

1. Buat kelas yang mengimplementasikan interface dengan kata kunci `implements`.

2. Implementasikan semua metode yang dideklarasikan dalam interface.

3. Gunakan objek dari kelas tersebut untuk memanggil metode-metode dalam interface tersebut.

# BAB 14

# Penanganan Eksepsi

## 14.1 Penanganan Eksepsi

Penanganan eksepsi adalah mekanisme untuk menangani kesalahan atau kondisi yang tidak terduga saat program berjalan. Dalam Java, kita menggunakan blok `try-catch` untuk menangkap dan mengelola eksepsi. Kode yang berpotensi menyebabkan eksepsi ditempatkan di dalam blok `try`, dan jika eksepsi terjadi, program akan melompat ke blok `catch` yang sesuai untuk menanganinya. Dalam blok `catch`, kita dapat menentukan tindakan yang harus dilakukan untuk menangani eksepsi tersebut.

## 14.1 Menangkap Eksepsi

Untuk menangkap eksepsi dalam Java, gunakan blok `try-catch`. Dalam blok `try`, letakkan kode yang mungkin menghasilkan eksepsi. Jika eksepsi terjadi, program akan melompat langsung ke blok `catch` yang sesuai untuk menanganinya. Di dalam blok `catch`, tentukan tindakan yang akan diambil terkait eksepsi yang ditangkap.

## 14.3 Catch secara Bertingkat

Catch secara bertingkat adalah teknik penanganan eksepsi di Java di mana kita menggunakan beberapa blok `catch` yang bersarang untuk menangani berbagai jenis eksepsi. Setiap blok `catch` menangani jenis eksepsi yang spesifik. Dengan menggunakan catch secara bertingkat, kita dapat mengelola dan merespons eksepsi secara terperinci sesuai dengan kebutuhan program.

## 14.4 Melontarkan Eksepsi

Melontarkan eksepsi adalah tindakan melempar sebuah kesalahan atau situasi yang tidak diharapkan dalam program. Dalam Java, menggunakan pernyataan `throw` untuk melontarkan eksepsi. Melontarkan eksepsi memungkinkan kita memberikan informasi tentang kesalahan kepada pengguna atau kode pemanggil, dan memungkinkan penanganan yang sesuai terhadap kesalahan tersebut.

## 14.5 Melontarkan kembali Eksepsi

Melontarkan kembali eksepsi adalah tindakan menangkap dan melempar kembali eksepsi ke blok pemanggil tanpa menanganinya secara penuh. Ini dilakukan dengan menggunakan pernyataan `throw` di dalam blok `catch`.

## 14.6 Klausa throws

Klausa `throws` digunakan dalam deklarasi metode untuk menunjukkan bahwa metode tersebut dapat melemparkan eksepsi tertentu kepada pemanggil metode.