ELEX MEDIA KOMPUTINDO

Java™ untuk Pemula

Jubilee Enterprise

Java untuk Pemula

Sanksi Pelanggaran Pasal 72:
Undang-Undang Nomor 19 Tahun 2002
Tentang Hak Cipta

1. Barangsiapa dengan sengaja melanggar dan tanpa hak melakukan perbuatan sebagaimana dimaksud dalam Pasal 2 Ayat (1) atau Pasal 49 Ayat (1) dan Ayat (2) dipidana dengan pidana penjara masing-masing paling singkat 1 (satu) bulan dan/atau denda paling sedikit Rp 1.000.000,00 (satu juta rupiah), atau pidana penjara paling lama 7 (tujuh) tahun dan/atau denda paling banyak Rp 5.000.000.000,00 (lima milyar rupiah).

2. Barangsiapa dengan sengaja menyiarkan, memamerkan, mengedarkan, atau menjual kepada umum suatu ciptaan atau barang hasil pelanggaran hak cipta atau hak terkait sebagai dimaksud pada Ayat (1) dipidana dengan pidana penjara paling lama 5 (lima) tahun dan/atau denda paling banyak Rp 500.000.000,00 (lima ratus juta rupiah).

Java untuk Pemula

Jubilee Enterprise

PENERBIT PT ELEX MEDIA KOMPUTINDO

KOMPAS GRAMEDIA

Java untuk Pemula

Jubilee Enterprise

Diterbitkan pertama kali oleh
Penerbit PT Elex Media Komputindo
Kelompok Gramedia, Anggota IKAPI, Jakarta 2014

121142672
ISBN: 978-602-02-5545-3

[eEp]

Dicetak oleh Percetakan PT Gramedia, Jakarta
Isi di luar tanggung jawab percetakan

KATA PENGANTAR

Mengapa Java penting untuk dipelajari? Jika ingin melangkah dari awal untuk masuk ke dalam dunia pemrograman masa kini, maka Java merupakan pijakan yang paling tepat. Saat ini, hampir sebagian besar kampus, perguruan tinggi, dan perusahaan dunia menggunakan Java sebagai platform pemrogramannya.

Jadi, mempelajari Java merupakan investasi masa depan karena di dunia akademisi dan industri, Java sudah menjadi standar dalam dunia pemrograman.

Buku ini ditujukan untuk para pemula. Bagi Anda yang ingin mulai belajar Java, pasti akan merasa cocok membaca buku ini. Diharapkan, setelah membaca buku ini, Anda akan mengenal Java dengan baik.

Yogyakarta, 21 Oktober 2014

Gregorius Agung

Founder Jubilee Enterprise

"*Information Technology is Our Passion and Book is Our Way*"

Do you need top-notch IT Book? Just .... thinkjubilee.com

DAFTAR ISI

1 SEKILAS TENTANG JAVA

Bahasa pemrograman Java dikembangkan oleh Sun Microsystem yang dimulai oleh James Gosling dan dirilis pada tahun 1995. Saat ini Sun Microsystem telah diakuisisi oleh Oracle Corporation.

Java bersifat Write Once, Run Anywhere (program yang ditulis satu kali dan dapat berjalan pada banyak platform).

Berikut ini fitur-fitur Java:

- **Berorientasi Objek**: Dalam Java, semua adalah Objek.
- **Bersifat Platform Independent**: Java di-compile dalam bit kode platform independen dan bukan pada mesin platform spesifik seperti pada C dan C++.
- **Sederhana**: Java didesain untuk dapat dengan mudah dipelajari.
- **Aman**: Dengan fitur keamanan Java, Anda dapat membuat sistem yang bebas virus dan powerful.
- **Bersifat Architectural-neutral**: Compiler Java membuat format file objek yang architectural-neutral, yang membuat kode yang decompile dapat dieksekusi pada berbagai prosesor yang memiliki sistem runtime Java.
- **Portabel**: Java bersifat portable karena adanya fitur platform independent dan architectural-neutral.
- **Kuat dan powerful**: Java mengeliminasi error dengan menjalankan pengecekan pada waktu compile dan runtime.

- **Multithreaded**: Dengan fitur multithread Java, Anda dapat membuat program yang dapat mengerjakan banyak tugas sekaligus.
- **Terinterpretasi**: Kode bit Java ditranslasi secara langsung pada instruksi mesin dan tidak disimpan.
- **Performa tinggi**: Java memiliki performa yang tinggi karena menggunakan compiler langsung.
- **Terdistribusi**: Java didesain untuk lingkungan distribusi internet.
- **Dinamis**: Java lebih dinamis dari C dan C++ karena Java didesain untuk beradaptasi dengan lingkungan pengembangan.

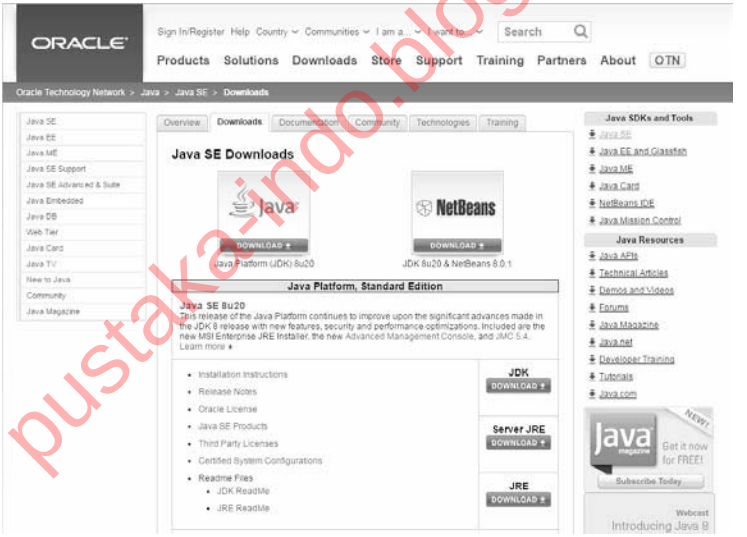
Tools yang Anda Perlukan

Untuk menjalankan contoh-contoh dalam buku ini Anda dapat menggunakan software-software berikut:

- Sistem operasi (Windows, Mac, Linux)
- Java JDK
- Notepad atau editor teks lainnya

2 MEMPERSIAPKAN JAVA

Berikut ini cara mempersiapkan Java agar dapat digunakan pada komputer Anda. Pertama-tama Anda perlu mendownload Java SE pada situs ***http://www.oracle.com/technetwork/java/javase/downloads***.



Gambar 2.1 Mendownload Java SE terlebih dulu

Setelah mendownload Java SE yang sesuai dengan sistem operasi Anda, jalankan file .exe untuk menginstal Java pada komputer Anda.

2.1 Editor Java Populer

Untuk menulis program Java, Anda memerlukan editor teks. Ada beberapa IDE yang tersedia untuk Anda gunakan:

- Notepad atau editor teks sejenis.
- Netbeans (www.netbeans.org)
- Eclipse (www. eclipse.org)

3 SINTAKS DASAR JAVA

Program Java dapat didefinisikan sebagai sebuah kompulan objek-objek yang saling berkomunikasi dengan cara memanggil method-method yang dimiliki masing-masing objek.

Berikut ini penjelasan singkat mengenai kelas, objek, method, dan variabel instance.

- Objek - Objek memiliki state (keadaan) dan behavior (perilaku). Contoh: Seekor anjing memiliki keadaan - warna, nama, jenis dan juga perilaku - menggonggong, makan, tidur. Sebuah objek adalah instance dari kelas.
- Kelas - Sebuah kelas dapat didefinisikan sebagai sebuah template/blue print yang mendeskripsikan perilaku/keadaan yang didukung oleh objek merupakan tipe kelas tersebut.
- Method - Method pada dasarnya adalah sebuah perilaku. Sebuah kelas dapat memiliki banyak method. Penulisan logika, manipulasi data dan eksekusi dari aksi lainnya semuanya dilakukan di dalam method.
- Variabel Instance - Setiap objek memiliki variabel uniknya sendiri. Keadaan sebuah objek dibuat berdasarkan nilai-nilai yang dimasukkan dalam variabel instancenya.

3.1 Program Java pertama

Berikut ini contoh kode sederhana untuk menampilkan Halo Dunia.

```
public class ProgramJavaPertama
{
  /* Ini adalah program java pertama.
   * Ini akan menampilkan output 'Halo Dunia'
   */

  public static void main(String[]args)
  {
    System.out.println("Halo Dunia");
    // menampilkan Halo Dunia
  }
}
```

Untuk menyimpan, compile, dan menjalankan program tersebut, lakukan langkah-langkah berikut:

- Buka editor teks dan tuliskan kode tersebut.
- Simpan file sebagai: ProgramJavaPertama.java.
- Buka jendela command prompt dan pergi ke direktori di mana Anda menyimpan kelas.
- Masukkan ‘ javac ProgramJavaPertama.java ’ dan tekan Enter untuk melakukan compile.
- Masukkan ‘ java ProgramJavaPertama.java ‘ untuk menjalankan program
- ‘ Halo Dunia ‘ akan ditampilkan.

```
C :> javac ProgramJavaPertama.java
C :> java ProgramJavaPertama
HaloDunia
```

3.2 Sintaks Dasar

Berikut ini hal-hal yang harus diperhatikan dalam pemrograman Java:

- Case Sensitive - Java bersifat case sensitive, yang berarti indentifier Halo dan halo memiliki arti yang berbeda.
- Untuk semua nama kelas, huruf pertama harus merupakan huruf kapital. Contoh: KelasJavaPertama.

- Semua nama method harus diawali dengan huruf kecil. Contoh: public void namaMethodSaya().
- Nama file program harus sama dengan nama kelas.
- public static void main(String args[]) - Pemrosesan program Java dimulai dari method main, yang merupakan bagian yang harus ada dalam setiap program Java.

3.3 Identifier Java

Semua komponen Java memiliki nama. Nama-nama yang digunakan untuk kelas, variabel, dan method disebut identifier.

Berikut ini hal-hal yang perlu diperhatikan mengenai identifier:

- Semua identifier harus diawali dengan huruf (A - Z atau a - z), karakter $ atau karakter _ (underscore).
- Setelah karakter pertama, identifier dapat berisi karakter apa saja.
- Sebuah kata kunci tidak dapat digunakan sebagai identifier.
- Identifier bersifat case sensitive.

3.4 Modifier Java

Seperti pada bahasa pemrograman lainnya, Anda dapat memodifikasi kelas, method dan sebagainya dengan menggunakan modifier.

Ada dua kategori modifier dalam Java:

- Modifier Akses: default, public, protected, private
- Modifier Non-akses: final, abstract, strictfp

3.5 Variabel Java

Terdapat beberapa jenis variabel dalam Java:

- Variabel lokal
- Variabel kelas (variabel static)

- Variabel instance (variabel non-static)

3.6 Array Java

Array adalah objek yang menyimpan beberapa variabel dengan jenis yang sama.

3.7 Enum Java

Enum membatasi sebuah variabel untuk memiliki satu nilai dari nilai-nilai yang telah ditentukan sebelumnya. Nilai-nilai ini kemudian disebut enum.

Berikut ini contoh penggunaan enum untuk membatasi ukuran gelas di tempat penjualan jus segar. Ukuran gelas dibatasi menjadi kecil, sedang, dan besar.

```
Class JusSegar
{
  enum UkuranJusSegar{ KECIL, SEDANG, BESAR }
  UkuranJusSegar ukuran;
}

public class TesJusSegar
{
  public static void main(String args[])
  {
    JusSegar jus = new JusSegar();
    jus.ukuran = JusSegar.UkuranJusSegar.SEDANG ;
  }
}
```

3.8 Kata kunci Java

Berikut ini daftar kata kunci dalam Java. Kata kunci ini tidak dapat digunakan sebagai konstanta, variabel, atau nama identifier.

abstract	assert	boolean	break
byte	case	catch	char
class	const	continue	default
do	double	else	enum
extends	final	finally	float
for	goto	if	implements
import	instanceof	int	interface
long	native	new	package
private	protected	public	return
short	static	strictfp	super
switch	synchronized	this	throw
throws	transient	try	void
volatile	while		

Gambar 3.1 Daftar kata kunci dalam Java

3.9 Komentar dalam Java

Java mendukung penggunaan komentar single-line dan multi-line. Semua karakter di dalam komentar akan diabaikan oleh compiler Java.

```
public class ProgramJavaPertama
{
  /* Ini adalah program java pertama.
   * Ini akan menampilkan output 'Halo Dunia'
   * Ini adalah contoh komentar multi-line
   */

  public static void main(String[]args)
  {
    // Ini adalah contoh komentar single-line
    /* Ini juga adalah contoh komentar single-line */

    System.out.println("Halo Dunia");
    // menampilkan Halo Dunia
  }
}
```

3.10 Menggunakan Baris Kosong

Suatu baris kode yang hanya berisi spasi (baris kosong) akan diabaikan dan tidak akan di-compile oleh Java.

3.11 Pewarisan

Kelas-kelas dalam Java dapat mewarisi kelas-kelas yang lain. Pada dasarnya, jika Anda ingin membuat kelas baru yang membutuhkan kode-kode yang ada pada kelas lain, Anda dapat mewarisi kelas lain itu.

Konsep ini memungkinkan Anda untuk menggunakan field dan method yang sudah ada tanpa harus menulis kembali kode tersebut pada kelas yang baru.

3.12 Interface

Pada Java, interface dapat didefinisikan sebagai sebuah kontrak antara objek-objek tentang bagaimana berkomunikasi satu sama lain.

4 OBJEK DAN KELAS JAVA

4.1 Objek dalam Java

Objek dalam Java dapat diandaikan seperti objek yang biasa Anda temui sehari-hari seperti Mobil, Anjing, Manusia, dan sebagainya. Semua objek yang ada memiliki keadaan dan perilakunya masing-masing.

Misalnya seekor anjing, memiliki keadaan - nama, jenis, warna, dan perilaku - menggonggong, tidur, makan, berlari.

Objek dalam perangkat lunak juga memiliki karakteristik yang sama yaitu memiliki keadaan dan perilaku.

Keadaan objek software disimpan dalam field-field dan perilakunya ditunjukkan melalui method-method.

Dalam pengembangan software, method-method beroperasi terhadap keadaan internal dari sebuah objek dan komunikasi antar objek terjadi melalui method-methodnya.

4.2 Kelas dalam Java

Sebuah kelas merupakan blue print/cetakan untuk membuat tiap-tiap objek.

Berikut ini contoh kelas Anjing:

```
public class Anjing
{
  String jenis;
  int umur;
  String warna;

  void menggonggong()
  {

  }

  void lapar()
  {

  }

  void tidur()
  {

  }
}
```

Sebuah kelas dapat memiliki jenis-jenis variabel berikut:

- **Variabel lokal**: Variabel yang didefinisikan di dalam sebuah method, konstruktor, atau blok. Variabel ini dideklarasikan dan diinisialisasi di dalam method dan kemudian akan dihapus setelah method selesai dijalankan.
- **Variabel instance**: Variabel yang ada di dalam kelas tetapi berada di luar method. Variabel ini diinisialisasi saat kelas dimuat. Variabel ini juga dapat diakses dari dalam method, konstruktor, atau blok dari kelas tersebut.
- **Variabel kelas**: Variabel yang dideklarasikan di dalam sebuah kelas, di luar method dan menggunakan kata kunci static.

4.3 Konstruktor

Setiap kelas memiliki konstruktor. Jika Anda tidak membuat konstruktor untuk sebuah kelas, compiler Java akan membuat konstruktor default secara otomatis.

Setiap kali sebuah objek baru dibuat, satu atau lebih konstruktor akan dijalankan. Aturan utama dari sebuah konstruktor adalah memiliki nama yang sama dengan nama kelas tersebut. Sebuah kelas dapat memiliki lebih dari satu konstruktor.

Berikut ini contoh konstruktor:

```
public class Anjing
{
  public Anjing()
  {

  }

  public Anjing(String nama)
  {
    // Konstruktor ini memiliki satu parameter, yaitu nama.
  }
}
```

4.4 Membuat Objek

Objek dibuat dari sebuah kelas. Pada Java, kata kunci new digunakan untuk membuat objek baru.

Berikut ini tiga langkah untuk membuat sebuah objek:

- **Deklarasi**: Deklarasi variabel dengan nama variabel dan tipe objek.
- **Instansiasi**: Kata kunci 'new' digunakan untuk membuat objek.
- **Inisialisasi**: Kata kunci 'new' diikuti dengan pemanggilan konstruktor. Pemanggilan ini menginisialisasi objek baru.

Berikut ini contoh untuk membuat objek baru:

```
public class Anjing
{
  public Anjing(String nama)
  {
    // Konstruktor ini memiliki satu parameter, yaitu nama.
    System.out.println("Nama yang diberikan:"+ nama );
  }

  public static void main(String[]args)
  {
    // Statemen berikut akan membuat objek anjingKu
    Anjing anjingKu = new Anjing("tommy");
  }
}
```

Jika Anda mengcompile program tersebut, hasilnya adalah sebagai berikut:

```
Namayangdiberikan:tommy
```

4.5 Mengakses Variabel Instance dan Method

Variabel instance dan method dapat diakses melalui objek-objek yang dibuat.

Berikut ini langkah-langkah untuk mengakses variabel instance:

```
/* Pertama-tama membuat objek */
ReferensiObjek = new Konstruktor();


/* Memanggil variabel */
ReferensiObjek.namaVariabel;


/* Sekarang Anda dapat memanggil method dari kelas */
ReferensiObjek.NamaMethod();
```

Berikut ini contoh penggunaannya:

```
public class Anjing
{
  int umurAnjing;

  public Anjing(String nama)
  {
    // Konstruktor ini memiliki satu parameter, yaitu nama.
    System.out.println("Nama yang diberikan:"+ nama );
  }

  public void aturUmur(int umur )
  {
    umurAnjing = umur;
  }


  public int ambilUmur()
  {
    System.out.println("Umur anjing adalah :"+ umurAnjing );
    return umurAnjing;
  }

  public static void main(String[]args)
  {
    /* Membuat objek */
    Anjing anjingKu = new Anjing("tommy");

    /* Memanggil method kelas untuk mengatur umur anjing */
    anjingKu.aturUmur(2);

    /* Memanggil method kelas untuk mengambil umur anjing */
    anjingKu.ambilUmur();

    /* Anda juga dapat mengakses variabel instance
     * sebagai  berikut
```

```
     */
    System.out.println("Nilai Variabel :" +
       anjingKu.umurAnjing );
  }
}
```

Jika program tersebut dijalankan, akan menghasilkan output berikut:

```
Nama yang diberikan:tommy
Umur anjing adalah :2
Nilai Variabel :2
```

4.6 Package Java

Saat Anda membuat suatu aplikasi dengan Java, Anda dapat menggunakan package yang berisi kelas-kelas yang sudah berisi kode-kode yang sesuai dengan kebutuhan Anda.

Untuk menggunakannya Anda perlu menyertakan semua kelas yang ada pada suatu direktori. Berikut ini contoh penggunaannya:

```
import java.io.*;
```

Contoh Kasus

Pada contoh kasus ini akan dibuat dua kelas yaitu Karyawan dan TesKaryawan.

Kelas Karyawan memiliki empat variabel instance yaitu nama, umur, posisi, dan gaji. Kelas ini memiliki satu konstruktor yang menggunakan parameter.

```
import java.io.*;

public class Karyawan
{
  String nama;
  int umur;
  String posisi;
  double gaji;

  // Ini adalah konstruktor kelas Karyawan
  public  Karyawan(String nama)
  {
    this.nama = nama;
  }

  // Memasukkan umur Karyawan ke dalam variabel umur.
  public void umurKar(int umurKar)
```

```
  {
    umur = umurKar;
  }

  /* Memasukkan posisi Karyawan ke dalam variabel posisi.*/
  public void posKar(String posKar)
  {
    posisi = posKar;
  }

  /* Memasukkan gaji Karyawan ke dalam variabel gaji.*/
  public void gajiKar(double gajiKar)
  {
    gaji = gajiKar;
  }

  /* Menampilkan informasi karyawan */
  public void tampilKar()
  {
    System.out.println("Nama:"+ nama );
    System.out.println("Umur:"+ umur );
    System.out.println("Posisi:"+ posisi );
    System.out.println("Gaji:"+ gaji);
  }
}
```

Untuk menjalankan kelas tersebut, Anda perlu membuat method main dan objek. Pada contoh ini method main dan objek akan dibuat pada kelas yang lain.

Berikut ini kelas TesKaryawan yang akan membuat dua instance dari kelas Karyawan dan menggunakan method untuk setiap objek untuk memasukkan nilai pada variabel-variabelnya.

```
import java.io.*;

public class TesKaryawan
{
  public static void main(String args[])
  {
    /* Membuat dua objek menggunakan konstruktor */
    Karyawan karSatu = new Karyawan("James Smith");
    Karyawan karDua = new Karyawan("Mary Anne");

    // Menggunakan method untuk setiap objek
    karSatu.umurKar(26);
    karSatu.posKar("Senior Software Engineer");
    karSatu.gajiKar(1000);
    karSatu.tampilKar();

    karDua.umurKar(21);
    karDua.posKar("Software Engineer");
    karDua.gajiKar(500);
    karDua.tampilKar();
  }
}
```

Compile kedua kelas dan jalankan TesKaryawan untuk menampilkan hasilnya:

```
C :> javac Karyawan.java
C :> vi TesKaryawan.java
C :> javac TesKaryawan.java
C :> java TesKaryawan


Nama:JamesSmith
Umur:26
Posisi:SeniorSoftwareEngineer
Gaji:1000.0

Nama:MaryAnne
Umur:21
Posisi:SoftwareEngineer
Gaji:500.0
```

5 TIPE DATA DALAM JAVA

Variabel adalah lokasi-lokasi dalam memori untuk menampung nilai-nilai. Ini berarti saat Anda membuat variabel, Anda menyiapkan ruang penampungan dalam memori.

Berdasarkan tipe data dari sebuah variabel, sistem operasi mengalokasikan memori dan menentukan tipe nilai apa yang dapat ditampung dalam memori tersebut.

Ada dua tipe data dalam Java:

- Tipe Data Primitif
- Tipe Data Referensi/Objek

5.1 Tipe Data Primitif

Terdapat delapan tipe data primitif yang didukung dalam Java. Tipe data primitif didefinisikan oleh bahasa dan diberi nama dengan sebutah kata kunci.

Berikut ini tipe data primitif dalam Java.

byte

- Tipe data byte merupakan integer 8-bit yang bersifat signed.
- Nilai minimum adalah -128.

- Nilai maksimum adalah 127.
- Nilai default adalah 0.
- Tipe data byte digunakan untuk menghemat ruang pada array yang besar.
- Contoh: byte a = 100, byte b = -50.

short

- Tipe data short merupakan integer 16-bit yang bersifat signed.
- Nilai minimum adalah -32,768.
- Nilai maksimum adalah 32,767.
- Nilai default adalah 0.
- Tipe data short juga dapat menghemat ruang seperti pada tipe data byte.
- Contoh: short s = 10000, short r = -20000.

int

- Tipe data int merupakan integer 32-bit yang bersifat signed.
- Nilai minimum adalah - 2,147,483,648.
- Nilai maksimum adalah 2,147,483,647.
- Nilai default adalah 0.
- Tipe data int digunakan secara umum untuk menampung nilai-nilai integral.
- Contoh: int a = 100000, int b = -200000.

long

- Tipe data long merupakan integer 64-bit yang bersifat signed.
- Nilai minimum adalah - -9,223,372,036,854,775,808.
- Nilai maksimum adalah 9,223,372,036,854,775,807.
- Nilai default adalah 0L.

- Tipe data long digunakan untuk menampung nilai-nilai dengan jangkauan nilai yang lebih dari tipe data int.
- Contoh: int a = 100000L, int b = -200000L.

float

- Tipe data float merupakan nilai desimal 32-bit presisi tunggal.
- Float biasanya digunakan untuk menghemat ruang pada array besar yang berisi nilai-nilai desimal.
- Nilai default adalah 0.0f.
- Tipe data float tidak pernah digunakan untuk nilai-nilai seperti nilai mata uang.
- Contoh: float f1 = 234.5f

double

- Tipe data double merupakan nilai desimal 64-bit presisi ganda.
- Double biasanya digunakan sebagai tipe data default untuk nilai desimal.
- Nilai default adalah 0.0d.
- Tipe data double tidak pernah digunakan untuk nilai-nilai presisi seperti nilai mata uang.
- Contoh: double d1 = 123.4

boolean

- Tipe data Boolean merepresentasikan satu bit informasi.
- Hanya ada dua nilai yang mungkin: true (benar) dan false (salah).
- Tipe data ini digunakan sebagai penanda kondisi benar/salah.
- Nilai default adalah false.
- Contoh: boolean satu = true.

char

- Tipe data char merupakan karakter Unicode 16-bit tunggal.
- Nilai minimum adalah '\u0000'.
- Nilai maksimum adalah '\uffff'.
- Tipe data char digunakan untuk menampung karakter apa saja.
- Contoh: char hurufA ='A'.

5.2 Tipe Data Referensi

- Variabel referensi dibuat dengan menggunakan konstruktor yang didefinisikan dalam kelas. Variabel ini digunakan untuk mengakses objek-objek. Variabel ini dideklarasikan sebagai tipe tertentu yang tidak dapat diubah. Sebagai contoh Karyawan, Anjing, dsb.
- Objek kelas dan berbagai tipe variabel array merupakan tipe data referensi.
- Nilai default adalah null.
- Variabel referensi dapat digunakan untuk mengacu pada objek dengan tipe yang sama atau yang kompatibel.
- Contoh: Hewan hewan = new Hewan("jerapah");

5.3 Literal dalam Java

Bahasa Java mendukung beberapa escape sequence untuk literal String dan char.

Notasi	Karakter yang direpresentasikan
\n	Baris baru
\r	Awal baris
\f	Halaman baru
\b	Backspace

\s	Spasi
\t	Tab
\"	Tanda petik dua
\'	Tanda petik
\\	Garis miring
\ddd	Karakter octal
\uxxxx	Karakter UNICODE hexadesimal

6 TIPE VARIABEL DALAM JAVA

Setiap variabel dalam Java memiliki tipe spesifik, yang menentukan ukuran dan layout memori; jangkauan nilai yang dapat disimpan; dan operasi-operasi yang dapat dijalankan terhadap variabel.

Anda harus mendeklarasikan semua variabel sebelum digunakan. Berikut ini bentuk dasar deklarasi variabel:

```
tipe data variabel [ = nilai][, variabel [= nilai] ...] ;
```

Untuk mendeklarasikan lebih dari satu variabel dengan tipe tertentu, Anda dapat menggunakan tanda koma sebagai pemisah.

Berikut ini adalah contoh deklarasi dan inisialisasi variabel dalam Java:

```
int a, b, c; // Mendeklarasikan tiga int, a, b, dan c.

int a = 10, b = 10; // Contoh inisialisasi.

byte B = 22; // Inisialisasi variabel B dengan tipe byte.

double pi = 3.14159; // Deklarasi dan inisialisasi nilai pi.

char a = 'a'; // variabel char a diinisialisasi dengan
              // nilai 'a'.
```

6.1 Variabel Lokal

- Variabel lokal dideklarasikan di dalam method, konstruktor, atau blok.

- Variabel lokal dibuat saat method, konstruktor atau blok mulai dijalankan dan akan dihapus saat selesai dijalankan.
- Modifier akses tidak dapat digunakan untuk variabel lokal.
- Variabel lokal hanya dapat digunakan di dalam method, constructor, atau blok tempat pendeklarasiannya.
- Tidak ada nilai default untuk variabel lokal sehinggal variabel lokal harus dideklarasikan dan diinisialisasi sebelum digunakan.

Berikut ini contoh variabel lokal umur yang didefinisikan di dalam method umurAnj() dan lingkup penggunaannya hanya di dalam method tersebut.

```
public class Tes
{
  public void umurAnj()
  {
    int umur = 0;
    umur = umur + 7;
    System.out.println("Umur anjing adalah : " + umur);
  }

  public static void main(String args[])
  {
    Tes tes = new Tes(); tes.umurAnj();
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Umur anjing adalah: 7
```

Jika Anda tidak menginisialisasikan umur, akan terjadi error:

```
....
    int umur;
    umur = umur + 7;
....

Tes.java:4:variable number might not have been initialized
umur = umur + 7;
          ^
1 error
```

6.2 Variabel Instance

- Variabel instance dideklarasikan di dalam kelas, tetapi di luar method, konstruktor, atau blok.

- Saat sebuah objek dibuat, tempat untuk setiap variabel instance disiapkan dalam memori.
- Variabel instance dibuat saat sebuah objek dibuat dengan menggunakan kata kunci 'new' dan dihapus saat objek tersebut dihapus.
- Variabel instance dapat dideklarasikan dalam kelas sebelum atau sesudah penggunaan.
- Modifier akses dapat digunakan pada variabel instance.
- Variabel instance dapat digunakan oleh semua method, konstruktor, dan blok di dalam kelas.
- Variabel instance mempunyai nilai default: untuk tipe numerik 0, untuk Boolean false dan untuk referensi objek adalah null.

Berikut ini contoh penggunaannya:

```
import java.io.*;

public class Karyawan
{
  // variabel instance ini dapat digunakan oleh semua subkelas
  // (kelas anak).
  public String nama;

  // variabel gaji hanya dapat digunakan dalam kelas Karyawan.
  private double gaji;

  // Variabel nama diinisialisasi dalam konstruktor.
  public Karyawan (String namaKar)
  {
    nama = namaKar;
  }

  // Memasukkan nilai dalam variabel gaji.
  public void aturGaji(double gajiKar)
  {
    gaji = gajiKar;
  }

  // Method ini menampilkan informasi karyawan.
  public void  tampilKar()
  {
    System.out.println("nama : " + nama );
    System.out.println("gaji :" + gaji);
  }

  public static void main(String args[])
  {
    Karyawan karSatu = new Karyawan("Ransika");
    karSatu.aturGaji(1000);
```

```
    karSatu.tampilKar();
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
nama : Ransika
gaji :1000.0
```

6.3 Variabel Kelas/Static

- Variabel kelas/static dideklarasikan dengan kata kunci static di dalam kelas, tetapi di luar method, konstruktor, atau blok.
- Hanya ada satu salinan dari variabel kelas per kelas.
- Variabel static biasanya dideklarasikan sebagai konstanta. Variabel konstanta nilainya tetap dan tidak berubah.
- Variabel static dibuat ketika program dimulai dan dihapus saat program berhenti.
- Nilai defaultnya sama dengan variabel instance.
- Variabel static dapat diakses dengan memanggil nama kelas . NamaKelas.NamaVariabel.
- Saat mendeklarasi variabel kelas dengan public static final, maka nama variabel (konstanta) dituliskan semua dalam hurum kapital. Jika variabel static tidak dideklarasikan dengan public final, syntax penamaannya sama dengan variabel instance dan lokal.

Berikut ini contoh penggunaan variabel kelas:

```
import java.io.*;

public class Karyawan
{
  // variabel gaji adalah variabel private static
  private static double gaji;

  // DEPARTMENT adalah konstanta
  public static final String DEPARTMENT = "Pengembangan ";

  public static void main(String args[])
  {
    gaji = 1000;
    System.out.println(DEPARTMENT+"gaji rata-ratanya:"+gaji);
  }
}
```

Berikut adalah hasil output dari kode di atas:

```
Pengembangan gaji rata-ratanya:1000
```

7 TIPE MODIFIER DALAM JAVA

Modifier adalah kata kunci yang ditambahkan pada definisi variabel untuk menentukan artinya. Berikut ini daftar modifier dalam bahas pemrograman Java.

7.1 Modifier Akses

Java menyediakan beberapa modifier akses untuk mengatur level akses untuk kelas, variabel, method dan konstruktor.

Berikut ini empat level akses:

- Dapat diakses oleh package, yang merupakan level akses default. Tidak memerlukan modifier.
- Dapat diakses hanya oleh kelas (private).
- Dapat diakses oleh semua (public).
- Dapat diakses oleh package dan semua subkelas (protected).

Untuk menggunakan modifier, Anda menambahkan kata kunci pada definisi kelas, method, atau variabel. Modifier berada di bagian awal statemen.

Berikut ini contohnya:

```
public class namaKelas
{
  // ...
```

```
}

private boolean penanda;
static final double minggu = 9.5;
protected static final int LEBARKOTAK = 42;

public static void main(String[] arguments)
{
  // isi method
}
```

Modifier Akses Default - Tanpa Kata Kunci

Variabel atau method yang dideklarasikan tanpa kata kunci dapat diakses oleh semua kelas pada package yang sama.

Berikut ini contohnya:

```
String version ="1.5.1";

boolean processOrder()
{
  return true;
}
```

Modifier Akses Privat - Private

Method, variabel dan konstruktor yang dideklarasikan dengan kata kunci private hanya dapat diakses di dalam kelas tempat pendeklarasiannya.

Berikut ini contohnya:

```
public class Logger
{
  private String format;


  public String getFormat()
  {
    return this.format;
  }

  public void setFormat(String format)
  {
    this.format = format;
  }
}
```

Pada contoh di atas, variabel format dalam kelas Logger adalah privat, jadi kelas lain tidak dapat mengakses atau memodifikasi nilainya secara langsung.

Agar variabel ini dapat diakses oleh semua, Dibuatkan dua method public yaitu getFormat() yang mengembalikan nilai format, dan setFormat(String) yang mengatur nilai format.

Modifier Akses Publik - Public

Sebuah kelas, method, konstruktor, interface dll yang dideklarasikan public dapat diakses oleh kelas-kelas lainnya.

Berikut ini contohnya:

```
public static void main(String[] arguments)
{
  // ...
}
```

Modifier Akses Protected - Protected

Variabel, method dan konstruktor yang dideklarasikan protected di dalam superclass dapat diakses hanya oleh subclass pada package yang berbeda atau kelas yang ada dalam package yang sama.

Berikut ini contoh kelas yang menggunakan protected untuk memungkinkan kelas turunannya melakukan override method openSpeaker():

```
class AudioPlayer
{

protected boolean openSpeaker(Speaker sp)
  {
    // implementasi
  }
}

class StreamingAudioPlayer
{
  boolean openSpeaker(Speaker sp)
  {
    // implementasi
  }
}
```

Pada contoh di atas, jika Anda mendefinisikan method openSpeaker() sebagai private, maka method itu tidak dapat diakses oleh kelas lain selain AudioPlayer.

Jika Anda mendefinisikan sebagai public, maka dapat diakses oleh semua. Tapi tujuan dari program itu bahwa method tersebut hanya dapat diakses oleh subclass, sehingga menggunakan modifier protected.

Kontrol Akses dan Pewarisan

Berikut ini aturan untuk method yang diwariskan:

- Method yang dideklarasikan public pada superclass juga harus public dalam semua subclass.
- Method yang dideklarasikan protected pada superclass harus protected atau public dalam subclass; tidak boleh private.
- Method yang dideklarasikan tanpa kontrol akses (tanpa modifier) dapat dideklarasikan lebih private dalam subclass.
- Method yang dideklarasikan private tidak diwariskan. Sehingga tidak ada aturannya.

7.2 Modifier Non Akses

Java menyediakan beberapa modifier non-akses.

- Modifier *static* untuk membuat kelas, method dan variabel.
- Modifier *final* untuk menetapkan implementasi dari kelas, method, dan variabel.
- Modifier *abstract* untuk membuat kelas dan method abstract.
- Modifier *synchronized* dan *volatile*, yang digunakan untuk thread.

8 OPERATOR DASAR DALAM JAVA

8.1 Operator Aritmatika

Operator aritmatika digunakan pada ekspresi matematik seperti pada operasi aljabar. Berikut ini daftar operator aritmatika:

Diasumsikan bahwa variabel A bernilai 10 dan B bernilai 20, maka:

Operator	Deskripsi	Contoh
+	Penjumlahan - Menambahkan nilai-nilai yang ada di kedua sisi operator	A + B hasilnya 30
-	Pengurangan - Mengurangkan nilai operan di sebelah kiri dengan nilai operan di sebelah kanan	A - B hasilnya -10
*	Perkalian - Mengalikan nilai-nilai yang ada di kedua sisi operator	A * B hasilnya 200
/	Pembagian - Membagi nilai operan di sebelah kiri dengan nilai operan di sebelah kanan	B / A hasilnya 2
%	Modulus - Mengurangkan nilai operan di sebelah kiri dengan nilai operan di sebelah kanan dan mengembalikan sisa nilainya	B % A hasilnya 0

++	Peningkatan - Menambahkan 1 pada nilai operan	B++ hasilnya 21
--	Penurunan - Mengurangkan 1 dari nilai operan	B-- hasilnya 19

Berikut ini contoh program yang menggunakan operator aritmatika.

```
public class Tes
{
  public static void main(String args[])
  {
    int a =10;
    int b =20;
    int c =25;
    int d =25;

    System.out.println("a + b = "+(a + b));
    System.out.println("a - b = "+(a - b));
    System.out.println("a * b = "+(a * b));
    System.out.println("b / a = "+(b / a));
    System.out.println("b % a = "+(b % a));
    System.out.println("c % a = "+(c % a));
    System.out.println("a++ = "+(a++));
    System.out.println("b-- = "+(a--));

    // Melihat perbedaan d++ dan ++d
    System.out.println("d++ = "+(d++));
    System.out.println("++d = "+(++d));
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
a + b =30
a - b =-10
a * b =200
b / a =2
b % a =0
c % a =5
a++=10
b--=11
d++=25
++d =27
```

8.2 Operator Relasional

Berikut ini operator relasional yang didukung dalam Java. Diasumsikan bahwa variabel A bernilai 10 dan B bernilai 20, maka:

Operator	Deskripsi	Contoh
==	Memeriksa apakah nilai kedua operan sama atau tidak, jika sama maka kondisi bernilai benar.	(A == B) adalah tidak benar.
!=	Memeriksa apakah nilai kedua operan sama atau tidak, jika tidak sama maka kondisi bernilai benar.	(A != B) adalah benar.
>	Memeriksa apakah nilai operan di sebelah kiri lebih dari nilai operan di sebelah kanan, jika ya maka kondisi bernilai benar.	(A > B) adalah tidak benar.
<	Memeriksa apakah nilai operan di sebelah kiri kurang dari nilai operan di sebelah kanan, jika ya maka kondisi bernilai benar.	(A < B) adalah benar.
>=	Memeriksa apakah nilai operan di sebelah kiri lebih dari atau sama dengan nilai operan di sebelah kanan, jika ya maka kondisi bernilai benar.	(A >= B) adalah tidak benar.
<=	Memeriksa apakah nilai operan di sebelah kiri kurang dari atau sama dengan nilai operan di sebelah kanan, jika ya maka kondisi bernilai benar.	(A <= B) adalah benar.

Berikut ini contoh program yang menggunakan operator relasional.

```
public class Tes
{
  public static void main(String args[])
  {
    int a =10;
    int b =20;

    System.out.println("a == b = "+(a == b));
    System.out.println("a != b = "+(a != b));
    System.out.println("a > b = "+(a > b));
    System.out.println("a < b = "+(a < b));
    System.out.println("b >= a = "+(b >= a));
    System.out.println("b <= a = "+(b <= a));
  }
}
```

Kode di atas akan menghasilkan output berikut:

```
a == b =false
a != b =true
a > b =false
a < b =true
b >= a =true
b <= a =false
```

8.3 Operator Logika

Berikut ini daftar operator logika. Diasumsikan bahwa nilai Boolean variabel A adalah true dan B adalah false, maka:

Operator	Deskripsi	Contoh
&&	Operator logika AND. Jika kedua operan bukan nol, maka kondisi bernilai benar.	(A && B) adalah tidak benar.
||	Operator logika OR. Jika ada operan yang bukan nol, maka kondisi bernilai benar.	(A || B) adalah benar.
!	Operator logika NOT. Digunakan untuk membalik keadaan logika dari operan. Jika kondisi bernilai benar maka operator NOT akan membuatnya menjadi tidak benar.	!(A && B) adalah benar.

Berikut ini contoh program yang menggunakan operator logika.

```
public class Tes
{
  public static void main(String args[])
  {
    boolean a =true;
    boolean b =false;

    System.out.println("a && b = "+(a&&b));
    System.out.println("a || b = "+(a||b));
    System.out.println("!(a && b) = "+!(a && b));
  }
}
```

Kode di atas akan menghasilkan output berikut:

```
a && b =false
a || b =true
!(a && b)=true
```

8.4 Operator Assignment

Berikut ini daftar operator assignment dalam Java:

Operator	Deskripsi	Contoh
=	Memasukkan nilai pada operan di sebelah kanan ke dalam operan di sebelah kiri	C = A + B akan memasukkan nilai A + B ke dalam C
+=	Menjumlahkan nilai operan di sebelah kiri dengan nilai operan di sebelah kanan dan memasukkan hasilnya ke dalam operan di sebelah kiri	C += A hasilnya sama dengan C = C + A
-=	Mengurangkan nilai operan di sebelah kiri dengan nilai operan di sebelah kanan dan memasukkan hasilnya ke dalam operan di sebelah kiri	C -= A hasilnya sama dengan C = C - A
*=	Mengalikan nilai operan di sebelah kiri dengan nilai operan di sebelah kanan dan memasukkan hasilnya ke dalam operan di sebelah kiri	C *= A hasilnya sama dengan C = C * A
/=	Membagi nilai operan di sebelah kiri dengan nilai operan di sebelah kanan dan memasukkan hasilnya ke dalam operan di sebelah kiri	C /= A hasilnya sama dengan C = C / A
%=	Menggunakan operasi modulus terhadap kedua operan dan memasukkan hasilnya ke dalam operan di sebelah kiri	C %= A hasilnya sama dengan C = C % A

Berikut ini contoh program yang menggunakan operator assignment:

```
public class Tes
{
  public static void main(String args[])
  {
    int a =10;
    int b =20;
    int c =0;

      c = a + b;
    System.out.println("c = a + b = "+ c );

      c += a ;
    System.out.println("c += a = "+ c );

      c -= a ;
    System.out.println("c -= a = "+ c );

      c *= a ;
    System.out.println("c *= a = "+ c );


      a =10;
      c =15;
      c /= a;
    System.out.println("c /= a = "+ c );

      a =10;
      c =15;
      c %= a ;
    System.out.println("c %= a = "+ c );
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
c = a + b =30
c += a =40
c -= a =30
c *= a =300
c /= a =1
c %= a =5
```

8.5 Operator Kondisional (?:)

Operator kondisional juga disebut operator ternary. Operator ini terdiri dari tiga operand dan digunakan untuk mengevaluasi ekspresi Boolean. Tujuan dari operator ini adalah untuk menentukan nilai mana yang akan dimasukkan ke dalam variabel.

Berikut ini syntax penulisan operator ternary:

```
variabel x =(ekspresi)? nilai jikabenar: nilai jikasalah
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    int a , b;

      a =10;
      b =(a ==1)?20:30;
    System.out.println("Nilai b adalah : "+ b );

      b =(a ==10)?20:30;
    System.out.println("Nilai b adalah : "+ b );
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Nilai b adalah : 30
Nilai b adalah : 20
```

8.6 Operator instanceof

Operator ini digunakan hanya untuk variabel referensi objek. Operator memeriksa apakah objek merupakan tipe tertentu.

```
(Objek variabel referensi) instanceof (tipe kelas/interface)
```

Jika objek yang yang direferensikan oleh variabel di sebelah kiri operator merupakan tipe kelas/interface di sebelah kanan, maka hasilnya adalah benar. Berikut ini contohnya:

```
String nama = “James”;
boolean hasil = nama instanceof String;
// Ini akan bernilai benar karena nama bertipe String
```

Pada contoh berikut tetap bernilai benar karena objek yang dibandingkan kompatibel dengan tipe di sebelah kanan:

```
class Kendaraan{}

public class Mobil extends Kendaraan
{
  public static void main(String args[])
  {
    Kendaraan a = new Mobil();
    boolean hasil = a instanceof Mobil;
    System.out.println(hasil);
```

```
    }
}
```

Kode tersebut akan menghasilkan output berikut:

```
true
```

8.7 Preseden Operator Java

Berikut ini urutan preseden operator-operator Java:

Kategori	Operator	Asosiasi
Postfix	() [] . (operator titik)	Kiri ke kanan
Unary	++ - - ! ~	Kanan ke kiri
Multiplicative	* / %	Kiri ke kanan
Additive	+ -	Kiri ke kanan
Shift	>>>>>><	Kiri ke kanan
Relasional	>>= <<=	Kiri ke kanan
Persamaan	== !=	Kiri ke kanan
Bitwise AND	&	Kiri ke kanan
Bitwise XOR	^	Kiri ke kanan
Bitwise OR	|	Kiri ke kanan
Logical AND	&&	Kiri ke kanan
Logical OR	||	Kiri ke kanan
Kondisional	?:	Kanan ke kiri
Assignment	= += -= *= /= %= >>= <<= &= ^ = |=	Kanan ke kiri
Koma	,	Kiri ke kanan

9 KONTROL PERULANGAN DALAM JAVA

9.1 Perulangan While

Perulangan while adalah struktur kontrol yang memungkinkan Anda mengulangi suatu proses dengan jumlah perulangan tertentu.

Berikut ini sintaks dari perulangan while:

```
while(ekspresi_Boolean)
{
  // Statemen
}
```

Selama ekspresi_Boolean bernilai benar, maka statemen dalam tubuh while akan terus dieksekusi.

Berikut ini contoh penggunaannya:

```
public class Test
{
  public static void main(String args[])
  {
    int x =10;

    while( x <20)
    {
      System.out.print("nilai x : "+ x );
      x++;

      System.out.print("\n");
    }
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
nilai x :10
nilai x :11
nilai x :12
nilai x :13
nilai x :14
nilai x :15
nilai x :16
nilai x :17
nilai x :18
nilai x :19
```

9.2 Perulangan do…while

Perulangan do…while sama seperti pengulangan while, tetapi perulangan do…while pasti akan dieksekusi minimal satu kali.

Berikut sintaksnya:

```
do
{
  // Statemen
}
while(ekspresi_boolean);
```

Berikut ini contoh penggunaannya:

```
public class Test
{
  public static void main(String args[])
  {
    int x =10;

    do
    {
      System.out.print("nilai x : "+ x );
      x++;

      System.out.print("\n");
    }
    while( x <20);
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
nilai x :10
nilai x :11
nilai x :12
nilai x :13
nilai x :14
```

```
nilai x :15
nilai x :16
nilai x :17
nilai x :18
nilai x :19
```

9.3 Perulangan for

Perulangan for adalah struktur kontrol repetitif yang memungkinkan Anda untuk mejalankan proses dengan jumlah perulangan tertentu (jumlah perulangan sudah diketahui sebelumnya).

Berikut ini sintaksnya:

```
for(inisialisasi;ekspresi_boolean; update)
{
  // Statemen
}
```

Berikut ini aliran proses dalam perulangan for:

- Bagian inisialisasi dieksekusi pertama kali dan hanya sekali. Bagian ini memungkinkan Anda untuk mendeklarasikan dan menginisialisasi variabel kontrol perulangan.
- Setelah itu, ekspresi Boolean dievaluasi. Jika bernilai benar, statemen dalam tubuh for dieksekusi. Jika tidak, statemen tidak dieksekusi dan proses berlanjut pada bagian setelah perulangan for.
- Setelah statemen dalam for dieksekusi, aliran proses kembali pada bagian update. Statemen update ini memungkinkan Anda untuk mengupdate variabel kontrol dalam for.
- Ekspresi Boolean kemudian dievaluasi lagi. Jika benar, statemen dalam perulangan for kembali dieksekusi, dan dilanjutkan ke bagian update. Jika tidak, perulangan for selesai dan proses berlanjut pada bagian setelah perulangan for.

Berikut ini contoh penggunaannya:

```
public class Tes

{
  public static void main(String args[])
  {
    for(int x =10; x <20; x = x+1)
    {
      System.out.print("nilai x : "+ x );
```

```
      System.out.print("\n");
    }
  }
}
```

Berikut ini hasil outputnya:

```
nilai x :10
nilai x :11
nilai x :12
nilai x :13
nilai x :14
nilai x :15
nilai x :16
nilai x :17
nilai x :18
nilai x :19
```

9.4 Perulangan for Baru dalam Java

Perulangan for baru disediakan pada Java versi 5 dan biasanya digunakan untuk array.

Berikut ini sintaksnya:

```
for(deklarasi : ekspresi)
{
  // Statemen
}
```

- **Deklarasi**: berisi variabel blok yang baru dideklarasi, yang kompatibel dengan jenis elemen array yang diakses.
- **Ekspresi**: ekspresi ini dievaluasi berdasarkan array yang diakses.

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    int[] angka ={10,20,30,40,50};

    for(int x : angka )
    {
      System.out.print(x);
      System.out.print(",");
    }

    System.out.print("\n");
```

```
    String[] namanama ={"James","Larry","Tom","Lacy"};

    for(String nama : namanama )
    {
      System.out.print( nama );
      System.out.print(",");
    }
  }
}
```

Berikut ini hasil output dari kode di atas:

```
10,20,30,40,50,
James,Larry,Tom,Lacy,
```

9.5 Kata Kunci break

Kata kunci break digunakan untuk menghentikan eksekusi perulangan.

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    int[] angka ={10,20,30,40,50};

    for(int x : angka)
    {
      if(x ==30)
      {
        break;
      }

      System.out.print( x );
      System.out.print("\n");
    }
  }
}
```

Berikut ini output dari kode tersebut:

```
10
20
```

9.6 Kata Kunci Continue

Kata kunci continue digunakan untuk melompat pada iterasi selanjutnya pada perulangan.

- Pada perulangan for, kata kunci menyebabkan aliran proses melompat langsung pada bagian update.
- Pada perulangan while, aliran proses melompat langsung pada bagian ekspresi Boolean.

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    int[] angka ={10,20,30,40,50};

    for(int x : angka)
    {
      if( x ==30)
      {
        continue;
      }

      System.out.print( x );
      System.out.print("\n");
    }
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
10
20
40
50
```

10 PENGAMBILAN KEPUTUSAN

10.1 Statemen if

Stemen if terdiri dari sebuah ekspresi Boolean yang diikuti dengan satu statemen atau lebih.

Berikut ini sintaks untuk statemen if:

```
if(ekspresi_boolean)
{
  // Statemen akan dieksekusi jika ekspresi Boolean
  // bernilai benar
}
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    int x =10;

    if( x <20)
    {
      System.out.print("Ini adalah statemen if");
    }
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Ini adalah statemen if
```

10.2 Statemen if…else

Statemen if dapat diikuti dengan statemen else opsional, yang dieksekusi jika ekspresi Boolean bernilai salah.

Berikut ini sintaksnya:

```
if(ekspresi_boolean)
{
  // Dieksekusi jika ekspresi Boolean bernilai benar
}
else
{
  // Dieksekusi jika ekspresi Boolean bernilai salah
}
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    int x =30;

    if(x <20)
    {
      System.out.print("Ini adalah statemen if");
    }
    else
    {
      System.out.print("Ini adalah statemen else");
    }
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Ini adalah statemen else
```

10.3 Statemen if…else if…else

Statemen if dapat diikuti dengan statemen else if…else opsional.

Saat menggunakan statemen if, else if, else ada beberapa hal yang harus Anda perhatikan:

- Sebuah statemen if dapat memiliki nol atau satu statemen else dan harus digunakan setelah statemen else if.
- Sebuah statemen if dapat memiliki nol atau banyak statemen else if dan harus digunakan sebelum statemen else.
- Setelah sebuah statemen else if dieksekusi, statemen else if atau else yang lain tidak akan diperiksa (akan langsung melompat pada kode setelah else terakhir).

Berikut ini sintaksnya:

```
if(ekspresi_boolean1)
{
  // Dieksekusi jika ekspresi boolean 1 bernilai benar
}
elseif(ekspresi_boolean2)
{
  // Dieksekusi jika ekspresi boolean 2 bernilai benar
}
elseif(ekspresi_boolean3)
{
  // Dieksekusi jika ekspresi boolean 3 bernilai benar
}
else
{
  // Dieksekusi jika tidak ada kondisi di atas yang
  // bernilai benar.
}
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    int x =30;

    if( x ==10)
    {
      System.out.print("Nilai X adalah 10");
    }
    elseif( x ==20)
    {
      System.out.print("Nilai X adalah 20");
    }
    elseif( x ==30)
    {
      System.out.print("Nilai X adalah 30");
    }
    else
    {
      System.out.print("Ini adalah statemen else");
    }
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Nilai X adalah 30
```

10.4 Statemen if…else Bertumpuk

Anda dapat memasukkan statemen if else ke dalam statemen if else lainnya.

Berikut ini sintaks dari statemen if else bertumpuk:

```
if(ekspresi_boolean1)
{
  // Dieksekusi jika ekspresi boolean 1 bernilai benar
  if(ekspresi_boolean2)
  {
    // Dieksekusi jika ekspresi boolean 2 bernilai benar
  }
}
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    int x =30;
    int y =10;

    if( x ==30)
    {
      if( y ==10)
      {
        System.out.print("X = 30 dan Y = 10");
      }
    }
  }
}
```

Hasil dari kode tersebut adalah sebagai berikut:

```
X =30 dan Y =10
```

10.5 Statemen Switch

Statemen switch memungkinkan sebuah variabel diperiksa kesamaannya dengan daftar nilai-nilai yang ada. Setiap nilai dalam daftar disebut case.

Berikut ini sintaks statemen switch:

```
switch(expression)
{
  case nilai :
    // Statemen
    break; // opsional
  case nilai :
    // Statemen
    break; // opsional
  // Anda dapat menggunakan beberapa statemen case.
  default: // optional
    // Statements
}
```

Berikut ini aturan penggunaan statemen switch:

- Variabel yang digunakan dalam switch hanya boleh berupa byte, short, int, atau char.
- Anda dapat menggunakan beberapa case di dalam switch. Setiap case diikuti dengan nilai yang akan dibandingkan dan tanda titik dua (:).
- Nilai case harus sama tipe datanya dengan variabel dalam switch dan harus berupa konstanta atau literal.
- Ketika variabel yang dibandingkan sama dengan dengan case, statemen pada case tersebut akan dieksekusi sampai pada kata kunci break.
- Ketika sampai pada kata kunci break, switch berhenti dieksekusi, dan aliran proses melompat pada kode sesudah statemen switch.
- Tidak semua case memerlukan break. Jika tidak ada break, aliran proses akan terus dilanjutkan pada case selanjutnya sampai pada break.
- Statemen switch dapat memiliki case default, yang diletakkan pada akhir statemen. Default dijalankan jika tidak ada case yang dijalankan.

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    char nilai = args[0].charAt(0);

    switch(nilai)
    {
```

```
        case'A':
          System.out.println("Sempurna!");
          break;
        case'B':
        case'C':
          System.out.println("Bagus");
          break;
        case'D':
          System.out.println("Lumayan");
        case'F':
          System.out.println("Coba lagi");
          break;
        default:
          System.out.println("Nilai tidak valid");
      }

      System.out.println("Nilai Anda "+ nilai);
    }
}
```

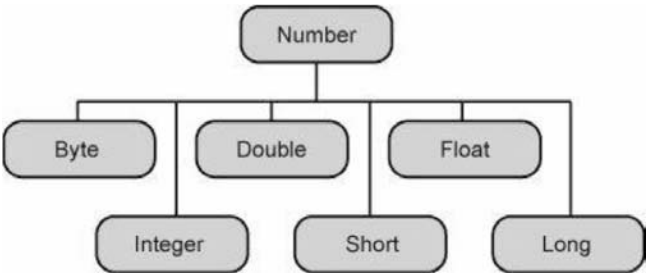
Kode tersebut menghasilkan output sebagai berikut:

```
$ java Tes a
Nilai tidak valid
Nilai Anda a

$ java Tes A
Sempurna!
Nilai Anda A

$ java Tes C
Bagus
Nilai Anda C
```

11 ANGKA DALAM JAVA

Tipe angka dalam Java memiliki kelas pembungkus (wrapper class) sesuai dengan tipenya masing-masing.

Berikut ini gambar diagram kelas Number dalam Java:



Gambar 11.1 Diagram class dan subclass dari Number

Semua kelas pembungkus (Byte, Integer, Double, Short, Float, Long) merupakan subkelas dari kelas abstrak Number.

11.1 Method-Method dalam Number

Berikut ini beberapa method-method instance penting yang diimplementasikan dalam subkelas dari kelas Number:

equals()

Method ini memeriksa apakah objek Number yang menggunakan method sama dengan argumen pada method.

Berikut sintaksnya:

```
public Boolean equals(Objek o)
```

- Argumen o dapat berupa objek apa saja.
- Method mengembalikan nilai true jika argumen tidak null dan merupakan objek yang sama tipenya dan sama nilainya.

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    Integer x =5;
    Integer y =10;
    Integer z =5;
    Short a =5;

    System.out.println(x.equals(y));
    System.out.println(x.equals(z));
    System.out.println(x.equals(a));
  }
}
```

Kode tersebut akan menghasilkan output sebagai berikut:

```
false
true
false
```

valueOf()

Method ini mengembalikan objek Number relevan yang menampung nilai dari argumen. Argumen dapat memiliki tipe data primitif, String, dsb.

Berikut sintaksnya:

```
staticInteger valueOf(int i)
staticInteger valueOf(String s)
staticInteger valueOf(String s,int radix)
```

- Argumen i adalah int yang merupakan representasi Integer yang akan dikembalikan.
- Argumen s adalah string yang merupakan representasi Integer yang akan dikembalikan.
- Argumen radix digunakan untuk menentukan nilai dari Integer yang dikembalikan berdasarkan nilai String yang digunakan.
- valueOf(int i) mengembalikan objek Integer yang mengandung nilai dari primitif tersebut.
- valueOf(String s) mengembalikan objek Integer yang mengandung nilai dari String tersebut.
- valueOf(String s, int radix) mengembalikan objek Integer yang mengandung nilai dari String, yang diuraikan dengan nilai radix.

Berikut contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    Integer x = Integer.valueOf(9);
    Double c = Double.valueOf(5);
    Float a = Float.valueOf("80");

    Integer b = Integer.valueOf("444",16);

    System.out.println(x);
    System.out.println(c);
    System.out.println(a);
    System.out.println(b);
  }
}
```

Berikut ini hasil dari kode tersebut:

```
9
5.0
80.0
1092
```

toString()

Method ini digunakan untuk mendapatkan objek String yang merepresentasikan nilai dari objek Number.

Berikut sintaksnya:

```
String toString()
static String toString(int i)
```

- Argumen i merupakan int yang akan dikembalikan representasi stringnya.
- toString() mengembalikan objek String yang merepresentasikan nilai dari Integer ini.
- toString(int i) mengembalikan objek String yang merepresentasikan integer tersebut.

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    Integer x =5;

    System.out.println(x.toString());
    System.out.println(Integer.toString(12));
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
5
12
```

parseInt()

Method ini digunakan untuk mendapatkan tipe data primitif dari String tertentu.

Berikut sintaksnya:

```
static int parseInt(String s)
static int parseInt(String s,int radix)
```

- Argumen s merupakan representasi string dari nilai desimal.
- Argumen radix akan digunkan untuk mengubah String s menjadi int.
- parseInt(String s) mengembalikan integer (desimal).
- parseInt(int i) mengembalikan integer, dengan representasi string untuk desimal, biner, octal, atau hexadecimal.

Berikut ini contoh penggunaannya:

```
public class Tes
```

```
{
  public static void main(String args[])
  {
    int x = Integer.parseInt("9");
    double c = Double.parseDouble("5");
    int b = Integer.parseInt("444",16);

    System.out.println(x);
    System.out.println(c);
    System.out.println(b);
  }
}
```

Kode tersebut akan menampilkan hasil berikut:

```
9
5.0
1092
```

abs()

Method ini memberikan nilai absolut dari argumen.

Berikut sintaksnya:

```
double abs(double d)
float abs(float f)
int abs(int i)
long abs(long lng)
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    Integer a =-8;
    double d =-100;
    float f =-90;

    System.out.println(Math.abs(a));
    System.out.println(Math.abs(d));
    System.out.println(Math.abs(f));
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
8
100.0
90.0
```

ceil()

Method ini memberikan nilai integer terkecil yang lebih dari atau sama dengan argumen yang diberikan.

Berikut sintaksnya:

```
double ceil(double d)
double ceil(float f)
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    double d =-100.675;
    float f =-90;

    System.out.println(Math.ceil(d));
    System.out.println(Math.ceil(f));

    System.out.println(Math.floor(d));
    System.out.println(Math.floor(f));
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
-100.0
-90.0

-101.0
-90.0
```

floor()

Method ini memberikan integer terbesar yang kurang dari atau sama dengan argumen yang diberikan.

Berikut sintaksnya:

```
double floor(double d)
double floor(float f)
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
```

```
    double d =-100.675;
    float f =-90;

    System.out.println(Math.floor(d));
    System.out.println(Math.floor(f));

    System.out.println(Math.ceil(d));
    System.out.println(Math.ceil(f));
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
-101.0
-90.0

-100.0
-90.0
```

round()

Method ini membulatkan nilai argumen menjadi nilai long atau int terdekat.

Berikut sintaksnya:

```
long round(double d)
int round(float f)
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    double d =100.675;
    double e =100.500;
    float f =100;
    float g =90f;

    System.out.println(Math.round(d));
    System.out.println(Math.round(e));
    System.out.println(Math.round(f));
    System.out.println(Math.round(g));
  }
}
```

Kode tersebut akan menghasilkan output sebagai berikut:

```
101
101
```

```
100
90
```

min()

Method ini mengembalikan nilai terkecil di antara dua argumen.

Berikut sintaksnya:

```
double min(double arg1,double arg2)
float min(float arg1,float arg2)
int min(int arg1,int arg2)
long min(long arg1,long arg2)
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    System.out.println(Math.min(12.123,12.456));
    System.out.println(Math.min(23.12,23.0));
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
12.123
23.0
```

max()

Method ini mengembalikan nilai terbesar di antara dua argumen.

Berikut sintaksnya:

```
double max(double arg1,double arg2)
float max(float arg1,float arg2)
int max(int arg1,int arg2)
long max(long arg1,long arg2)
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    System.out.println(Math.max(12.123,12.456));
    System.out.println(Math.max(23.12,23.0));
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
12.456
23.12
```

pow()

Method ini mengembalikan nilai argumen pertama pangkat argumen kedua.

Berikut sintaksnya:

```
double pow(double bil, double pangkat)
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    double x =11.635;
    double y =2.76;

    System.out.printf("Nilai e adalah %.4f%n",Math.E);
    System.out.printf("pow(%.3f, %.3f) adalah %.3f%n",x,
                      y,Math.pow(x, y));
  }
}
```

Berikut ini output dari kode tersebut:

```
Nilai e adalah 2.7183
pow(11.635, 2.760) adalah 874.008
```

sqrt()

Method ini mengembalikan nilai akar pangkat dua dari argumen.

Berikut sintaksnya:

```
double sqrt(double d)
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    double x =11.635;
    double y =2.76;

    System.out.printf("Nilai e adalah %.4f%n",Math.E);
    System.out.printf("sqrt(%.3f) adalah %.3f%n",
                      x,Math.sqrt(x));
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Nilai e adalah 2.7183
sqrt(11.635) adalah 3.411
```

random()

Method ini digunakan untuk mendapatkan nilai acak di antara 0.0 sampai 1.0.

Berikut sintaksnya:

```
static double random()
```

Berikut ini contoh pengunaannya:

```
public class Test
{
  public static void main(String args[])
  {
    System.out.println(Math.random());
    System.out.println(Math.random());
  }
}
```

Berikut ini output dari kode tersebut (output akan berbeda untuk setiap eksekusi):

```
0.16763945061451657
0.400551253762343
```

12 KARAKTER DALAM JAVA

12.1 Escape Sequence

Sebuah karakter yang diawali dengan tanda garis miring (\) merupakan escape sequence dan memiliki arti khusus bagi compiler.

Berikut ini daftar escape sequence dalam Java:

Escape Sequence	Deskripsi
\t	Menambahkan tab pada teks.
\b	Menambahkan backspace pada teks.
\n	Menambahkan baris baru pada teks.
\r	Mengarahkan kursor pada awal baris.
\f	Menambahkan halaman baru pada teks.
\'	Menambahkan tanda petik tunggal pada teks.
\"	Menambahkan tanda petik ganda pada teks.
\\	Menambahkan karakter garis miring (/) pada teks.

Berikut ini contoh penggunaan escape sequence:

```
public class Tes
{
  public static void main(String args[])
  {
    System.out.println("Dia mengatakan \"Halo!\" pada saya.");
  }
}
```

Kode tersebut akan menghasilkan output:

```
Dia mengatakan "Halo!" pada saya.
```

12.2 Method-Method Karakter

Berikut ini beberapa method-method penting yang dimplementasikan oleh semua subkelas dari kelas Character.

isUpperCase()

Method ini menentukan apakah karakter merupakan huruf kapital.

Berikut sintaksnya:

```
boolean isUpperCase(char ch)
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    System.out.println(Character.isUpperCase('c'));
    System.out.println(Character.isUpperCase('C'));
    System.out.println(Character.isUpperCase('\n'));
    System.out.println(Character.isUpperCase('\t'));
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
false
true
false
false
```

isLowerCase()

Method ini menentukan apakah karakter merupakan huruf kecil.

Berikut sintaksnya:

```
boolean isLowerCase(char ch)
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    System.out.println(Character.isLowerCase('c'));
    System.out.println(Character.isLowerCase('C'));
    System.out.println(Character.isLowerCase('\n'));
    System.out.println(Character.isLowerCase('\t'));
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
true
false
false
false
```

toUpperCase()

Method ini mengembalikan bentuk kapital dari nilai char yang dispesifikasikan.

Berikut sintaksnya:

```
char toUpperCase(char ch)
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    System.out.println(Character.toUpperCase('c'));
    System.out.println(Character.toUpperCase('C'));
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
C
C
```

toLowerCase()

Method ini mengembalikan bentuk kecil/biasa dari nilai char yang dispesifikasikan.

Berikut sintaksnya:

```
char toLowerCase(char ch)
```

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    System.out.println(Character.toLowerCase('c'));
    System.out.println(Character.toLowerCase('C'));
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
c
c
```

13 STRING DALAM JAVA

13.1 Membuat String

Berikut ini cara membuat string secara langsung:

```
String salam = "Halo dunia!";
```

Anda juga dapat membuat objeks String dengan menggunakan kata kunci new dan sebuah konstruktor.

Berikut ini contoh membuat String:

```
public class StringDemo
{
  public static void main(String args[])
  {
    char[] haloArray ={'h','a','l','o','.'};
    String haloString = new String(haloArray);
    System.out.println(haloString);
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
halo.
```

13.2 Panjang String

Method yang digunakan untuk memperoleh informasi mengenai suatu objek disebut method aksesor. Salah satu method aksesor adalah length() yang mengembalikan jumlah karakter yang ada dalam objek string.

Berikut ini contoh penggunaannya:

```
public class StringDemo
{
  public static void main(String args[])
  {
    String contoh ="Kalimat ini adalah contoh";
    int pan = contoh.length();
    System.out.println("Panjang String adalah : "+ pan );
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Panjang String adalah : 25
```

13.3 Menyambung String

Kelas String menyediakan method untuk menyambung string.

```
string1.concat(string2);
```

Method tersebut mengembalikan string baru yaitu string1 yang ditambahkan dengan string2.

Anda juga dapat menggunakan method concat() dengan string literal seperti ini:

```
“Nama saya adalah “.concat(“Zara”);
```

String biasanya disambung dengan operator + seperti ini:

```
“Halo, ”+”dunia”+”!”
```

Kode tersebut akan menghasilkan:

```
“Halo, dunia!”
```

13.4 Method-Method dalam String

Berikut ini beberapa method penting yang dapat Anda gunakan dalam kelas String.

char charAt(int indeks)

Method ini mengembalikan karakter yang ada pada posisi indeks tertentu. Indeks string dimulai dari nol.

Berikut sintaksnya:

```
public char charAt(int indeks)
```

Contoh kode berikut akan menghasilkan output a:

```
public class Tes
{
  public static void main(String args[])
  {
    String s = "String adalah";
    char hasil = s.charAt(7);
    System.out.println(hasil);
  }
}
```

static String copyValueOf(char[] data)

Method ini memiliki dua bentuk:

- **public static String copyValueOf(char[] data):** Mengembalikan String yang merepresentasikan urutan karakter dalam array yang digunakan.
- **public static String copyValueOf(char[] data, int offset, int panjang):** Mengembalikan String yang merepresentasikan urutan karakter dalam array yang digunakan dengan offset dan panjang tertentu.

Berikut ini contoh penggunaannya:

```
public class Tes
{
  public static void main(String args[])
  {
    char[]Str1={'h','a','l','o',' ','d','u','n','i','a'};
    String Str2="";

    Str2=Str2.copyValueOf(Str1);
```

```
    System.out.println("String hasil: "+ Str2);

    Str2=Str2.copyValueOf(Str1,2,6);
    System.out.println("String hasil: "+ Str2);
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
String hasil: halo dunia
String hasil: lo dun
```

void getChars(int indeksAwal, int indeksAkhir, char[] tuj, int tujAwal)

Method ini menyalin karakter-karakter dari string ke array karakter tujuan.

- indeksAwal adalah indeks dari karakter pertama yang akan disalin.
- indeksAkhir adalah indeks dari karakter terakhir yang akan disalin.
- tuj adalah array tujuan penyalinan.
- tujAwal adalah posisi awal tempat menyalin array.

Berikut ini contoh penggunaannya:

```
import java.io.*;
public class Tes
{
  public static void main(String args[])
  {
    String Str1 = new String("Selamat datang di
                              thinkjubilee.com");
    char[]Str2 = new char[7];

    try
    {
      Str1.getChars(2,9,Str2,0);
      System.out.print("Nilai yang disalin = ");
      System.out.println(Str2);
    }
    catch(Exception ex)
    {
      System.out.println("Gunakan exception...");
    }
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Nilai yang disalin = lamat d
```

int indexOf(int ch)

Method ini mengembalikan posisi indeks dari karakter yang dispesifikasikan.

Method ini memiliki beberapa varian:

- **public int indexOf(int ch):** Mengembalikan posisi indeks string dari karakter yang dispesifikasikan atau -1 jika karakter tidak terdapat dalam string.
- **public int indexOf(int ch, int indeksAwal):** Mengembalikan posisi indeks string dari karakter yang dispesifikasikan, pencarian dimulai pada posisi indeksAwal.
- **int indexOf(String str):** Mengembalikan posisi indeks string dari substring yang dispesifikasikan atau -1 jika substring tidak terdapat dalam string.
- **int indexOf(String str, int indeksAwal):** Mengembalikan posisi indeks string dari substring yang dispesifikasikan, pencarian dimulai pada posisi indeksAwal.

Berikut ini contoh penggunaannya:

```
import java.io.*;

public class Test
{
  public static void main(String args[])
  {
    String Str=new String("Welcome to Tutorialspoint.com");
    StringSubStr1=new String("Tutorials");
    StringSubStr2=new String("Sutorials");

    System.out.print("Indeks yang ditemukan :");
    System.out.println(Str.indexOf('o'));
    System.out.print("Indeks yang ditemukan :");
    System.out.println(Str.indexOf('o',5));
    System.out.print("Indeks yang ditemukan :");
    System.out.println(Str.indexOf(SubStr1));
    System.out.print("Indeks yang ditemukan :");
    System.out.println(Str.indexOf(SubStr1,15));
    System.out.print("Indeks yang ditemukan :");
    System.out.println(Str.indexOf(SubStr2));
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Indeks yang ditemukan :4
Indeks yang ditemukan :9
Indeks yang ditemukan :11
```

```
Indeks yang ditemukan :-1
Indeks yang ditemukan :-1
```

String Replace(char charLama, char charBaru)

Method ini mengembalikan string baru yang merupakan hasil dari mengganti semua charLama dalam string dengan charBaru.

Berikut ini contoh penggunannya:

```
import java.io.*;
public class Tes
{
  public static void main(String args[])
  {
    String Str = new String("Selamat datang di
                            thinkjubilee.com");
    System.out.print("Nilai balikan :");
    System.out.println(Str.replace('o','T'));
    System.out.print("Nilai balikan :");
    System.out.println(Str.replace('l','D'));
  }
}
```

Berikut ini hasil dari kode tersebut:

```
Nilai balikan : Selamat datang di thinkjubilee.cTm
Nilai balikan : SeDamat datang di thinkjubiDee.com
```

Boolean Starts With(String Awalan)

Method ini memeriksa apakah string dimulai dengan awalan tertentu dan yang pencariannya dimulai pada indeks tertentu.

Berikut sintaksnya:

```
public boolean startsWith(String awalan, int indeksMulai)
atau
public boolean startsWith(String awalan)
```

Berikut ini contoh penggunaannya:

```
import java.io.*;

public class Tes
{
  public static void main(String args[])
  {
    String Str=new String("Selamat datang di
                          thinkjubilee.com");

    System.out.print("Nilai balikan :");
    System.out.println(Str.startsWith("Selamat"));

    System.out.print("Nilai balikan :");
    System.out.println(Str.startsWith("thinkjubilee"));
```

```
    System.out.print("Nilai balikan :");
    System.out.println(Str.startsWith("thinkjubilee ",18));
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Nilai balikan :true
Nilai balikan :false
Nilai balikan :true
```

String Substring(int indeksMulai)

Method ini mengembalikan string baru yang merupakan substring dari string tersebut.

Berikut sintaksnya:

```
public String substring(int indeksMulai)
or
public String substring(int indeksMulai,int indeksAkhir)
```

Berikut ini contoh penggunaannya:

```
import java.io.*;

public class Tes
{
  public static void main(String args[])
  {
    String Str=new String("Selamat datang di
                          thinkjubilee.com ");

    System.out.print("Nilai balikan :");
    System.out.println(Str.substring(10));

    System.out.print("Nilai balikan :");
    System.out.println(Str.substring(10,15));
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Nilai balikan :tang di thinkjubilee.com
Nilai balikan :tang
```

String trim()

Method ini mengembalikan salinan string yang spasi di awal dan akhir string dihapus.

Berikut ini contoh penggunaannya:

```
import java.io.*;
```

```
public class Tes
{
  public static void main(String args[])
  {
    String Str=new String("  Selamat datang di
                          thinkjubilee.com  ");
    System.out.print("Nilai balikan :");
    System.out.println(Str.trim());
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Nilai balikan :Selamat datang di thinkjubilee.com
```

static String valueOf(tipe data primitif x)

Method ini mengembalikan representasi string dari argumen yang digunakan.

Berikut sintaksnya:

```
static String valueOf(boolean b)
atau
static String valueOf(char c)
atau
static String valueOf(char[] data)
atau
static String valueOf(char[] data,int offset,int jumlah)
atau
static String valueOf(double d)
atau
static String valueOf(float f)
atau
static String valueOf(int i)
atau
static String valueOf(long l)
atau
static String valueOf(Object obj)
```

Berikut ini contoh penggunaannya:

```
import java.io.*;

public class Tes
{
  public static void main(String args[])
  {
    double d =102939939.939;
    boolean b =true;
    long l =1232874;
    char[] arr ={'a','b','c','d','e','f','g'};

    System.out.println("Nilai balikan : "+String.valueOf(d));
    System.out.println("Nilai balikan : "+String.valueOf(b));
    System.out.println("Nilai balikan : "+String.valueOf(l));
```

```
        System.out.println("Nilai balikan : "
                                        +String.valueOf(arr));
    }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Nilai balikan : 1.02939939939E8
Nilai balikan : true
Nilai balikan : 1232874
Nilai balikan : abcdefg
```

14 ARRAY DALAM JAVA

Java menyediakan sebuah struktur data, array, yang menampung koleksi elemen-elemen dengan tipe yang sama. Array dapat digambarkan sebagai koleksi variabel-variabel dengan tipe yang sama.

14.1 Deklarasi Variabel Array

Untuk menggunakan array, pertama-tama Anda perlu mendeklarasi sebuah variabel sebagai referensi dari array, dan Anda perlu menentukan tipe referensi dari array tersebut.

Berikut ini sintaks untuk mendeklarasi variabel array:

```
tipeData[] varRefArr; // cara yang dianjurkan.
atau
tipeData varRefArr[]; // cara yang tidak dianjurkan.
```

Berikut ini contoh penggunaannya:

```
double[] daftarKu; // cara yang dianjurkan.
atau
double daftarKu[]; // cara yang tidak dianjurkan.
```

14.2 Membuat Array

Anda dapat membuat array dengan menggunakan operator new seperti pada sintaks berikut:

```
varRefArr = new tipeData[ukuranArray];
```

Deklarasi variabel array, membuat array, dan memasukkan referensi array ke dalam variabel, dapat dilakukan dalam satu statemen sebagai berikut:

```
tipeData[] varRefArr = new tipeData[ukuranArray];
atau
tipeData[] varRefArr = {nilai0, nilai1, ...., nilaik};
```

14.3 Memroses Array

Untuk memroses array, Anda dapat menggunakan perulangan for atau foreach.

Berikut ini contoh pembuatan, inisialisasi, dan pemrosesan array:

```
public class TesArray
{
  public static void main(String[] args)
  {
    double[] daftarKu ={1.9,2.9,3.4,3.5};

    // Menampilkan semua elemen array
    for(int i =0; i < daftarKu.length; i++)
    {
      System.out.println(daftarKu[i]+" ");
    }

    // Menjumlahkan semua elemen
    double total =0;
    for(int i =0; i < daftarKu.length; i++)
    {
      total += daftarKu[i];
    }

    System.out.println("Jumlah total adalah "+ total);

    // Menemukan elemen dengan nilai paling besar
    double max = daftarKu[0];
    for(int i =1; i < daftarKu.length; i++)
    {
      if(daftarKu[i]> max)
       max = daftarKu[i];
    }

    System.out.println("Nilai paling besar adalah "+ max);
  }
}
```

Kode tersebut akan menghasilkan output sebagai berikut:

```
1.9
2.9
3.4
3.5
Jumlah total adalah 11.7
Nilai paling besar adalah 3.5
```

14.4 Perulangan foreach

Perulangan foreach memungkinkan Anda untuk mengakses elemen-elemen dalam array secara berurutan tanpa harus menggunakan variabel indeks.

Berikut ini contoh penggunaannya:

```
public class TesArray
{
  public static void main(String[] args)
  {
    double[] daftarKu ={1.9,2.9,3.4,3.5};

    // Menampilkan semua elemen dalam array
    for(double elemen : daftarKu)
    {
      System.out.println(elemen);
    }
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
1.9
2.9
3.4
3.5
```

14.5 Kelas Array

Kelas Array dalam Java memiliki berbagai method static untuk mengurutkan dan mencari array, membandingkan array, dan memasukkan elemen dalam array.

- **public static int binarySearch(Objek[] a, Objek kunci)** Mencari array dari Object (Byte, Int , double, dsb.) dari nilai tertentu dengan menggunakan algoritma pencarian biner.
- **public static boolean equals(long[] a, long[] a2)** Mengembalikan nilai true jika kedua array sama satu sama lain.
- **public static void fill(int[] a, int nilai)** Menambahkan nilai int yang ditentukan ke dalam setiap elemen dalam array a.
- **public static void sort(Objek[] a)** Mengurutkan array dari atas ke bawah berdasarkan tipe elemennya.

15 METHOD DALAM JAVA

Method dalam Java merupakan sekumpulan statemen yang dikumpulkan bersama untuk melaksanakan tugas tertentu.

Pada bagian ini Anda akan belajar cara membuat method Anda sendiri dengan atau tanpa nilai balikan, dengan atau tanpa parameter, melakukan overload method dengan nama yang sama, dan menggunakan abstraksi method.

15.1 Membuat Method

Berikut ini contoh struktur sebuah method:

```
public static int namaFungs(int a, int b)
{
  // tubuh
}
```

- public static: modifier
- int: jenis nilai balikan
- namaFungs: nama fungsi/method
- a, b: parameter formal
- int a, int b: daftar parameter

Method juga disebut prosedur atau fungsi:

- prosedur: tidak mengembalikan nilai
- fungsi: mengembalikan nilai

Berikut ini contoh method yang mengambil dua parameter n1 dan n2 dan mengembalikan nilai terbesar di antara keduanya:

```
public static int fungsiMin(int n1, int n2)
{
  int min;

  if (n1 > n2)
    min = n2;
  else
    min = n1;

  return min;
}
```

15.2 Memanggil Method

Untuk menggunakan method, Anda harus memanggilnya. Ada dua cara pemanggilan method yaitu method yang mengembalikan nilai dan yang tidak.

Berikut ini contoh pemanggilan method yang tidak mengembalikan nilai (void):

```
System.out.println("Ini adalah thinkjubilee.com!");
```

Berikut ini contoh pemanggilan method yang mengembalikan nilai:

```
int hasil = sum(6, 9);
```

Berikut ini contoh definisi dan pemanggilan method:

```
public class ContohMin
{
  public static void main(String[] args)
  {
    int a = 11;
    int b = 6;
    int c = fungsiMin(a, b);
    System.out.println("Nilai minimum = " + c);
  }

  /** mengembalikan nilai paling kecil */
  public static int fungsiMin(int n1, int n2)
```

```
  {
    int min;

    if (n1 > n2)
      min = n2;
    else
      min = n1;

    return min;
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Nilai minimum = 6
```

15.3 Kata Kunci void

Kata kunci void memungkinkan Anda untuk membuat method yang tidak mengembalikan nilai.

Berikut ini contoh penggunaannya:

```
public class ContohVoid
{
  public static void main(String[] args)
  {
    methodRankPoints(255.7);
  }

  public static void methodRankPoints(double points)
  {
    if (points >= 202.5)
    {
      System.out.println("Rank:A1");
    }
    else if (points >= 122.4)
    {
      System.out.println("Rank:A2");
    }
    else
    {
      System.out.println("Rank:A3");
    }
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Rank:A1
```

15.4 Overload Method

Ketika sebuah kelas memiliki dua method atau lebih dengan nama yang sama dan jumlah parameter yang berbeda, hal ini disebut overload method.

Berikut ini contoh penggunaannya:

```
public class ContohOverload
{
  public static void main(String[] args)
  {
    int a = 11;
    int b = 6;
    double c = 7.3;
    double d = 9.4;

    int hasil1 = fungsiMin(a, b);
    // nama fungsi sama dengan parameter yang berbeda
    double hasil2 = fungsiMin(c, d);

    System.out.println("Nilai minimum = " + hasil1);
    System.out.println("Nilai minimum = " + hasil2);
  }

  // untuk integer
  public static int fungsiMin(int n1, int n2)
  {
    int min;
    if (n1 > n2)
      min = n2;
    else
      min = n1; return min;
  }

  // untuk double
  public static double fungsiMin(double n1, double n2)
  {
    double min;
    if (n1 > n2)
      min = n2;
    else
      min = n1; return min;
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Nilai minimum = 6
Nilai minimum = 7.3
```

15.5 Konstruktor

Konstruktor menginisialisasi objek saat objek itu dibuat. Konstruktor memiliki nama yang sama dengan nama kelasnya dan sintaksnya sama dengan method, tetapi konstruktor tidak memiliki tipe nilai balikan.

Jika Anda tidak membuat konstruktor dalam kelas, maka Java akan membuat konstruktor default secara otomatis untuk Anda.

Berikut ini contoh sederhana penggunaan konstruktor:

```
// Contoh kelas.
class KelasKu
{
  int x;

  // Berikut ini adalah konstruktornya
  KelasKu()
  {
    x = 10;
  }
}
```

Anda perlu memanggil konstruktor untuk menginisialisasi objek seperti ini:

```
public class DemoKons
{
  public static void main(String args[])
  {
    KelasKu t1 = new KelasKu();
    KelasKu t2 = new KelasKu();

    System.out.println(t1.x +" "+ t2.x);
  }
}
```

Anda juga dapat menggunakan parameter dalam konstruktor. Parameter tersebut ditambahkan seperti pada method.

Berikut ini contohnya:

```
// Contoh kelas.
class KelasKu
{
  int x;

  // Berikut ini adalah konstruktornya
  KelasKu(int i)
  {
    x = i;
  }
}
```

Anda perlu memanggil konstruktor untuk menginisialisasi objek seperti ini:

```
public class DemoKons
{
  public static void main(String args[])
  {
    KelasKu t1 = new KelasKu(10);
    KelasKu t2 = new KelasKu(20);

    System.out.println(t1.x +" "+ t2.x);
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
10 20
```

16 PEWARISAN DALAM JAVA

Pewarisan dapat didefinisikan sebagai proses di mana satu objek menggunakan properti dari objek lainnya.

Dalam pewarisan, kata kunci yang paling umum digunakan adalah extends dan implements. Kata kunci tersebut menentukan apakah sebuah objek Adalah tipe dari objek lainnya.

16.1 Hubungan 'Adalah'

Berikut ini contoh penggunaan kata kunci extends untuk menandakan hubungan 'Adalah'.

```
public class Binatang
{ }

public class Mamalia extends Binatang
{ }

public class Reptil extends Binatang
{ }

public class Anjing extends Mamalia
{ }
```

Berdasarkan contoh tersebut, berikut adalah penjelasannya:

- Binatang adalah superkelas dari kelas Mamalia
- Binatang adalah superkelas dari kelas Reptil

- Mamalia dan Reptil adalah subkelas dari kelas Binatang
- Anjing adalah subkelas dari kelas Mamalia dan Binatang

Berikut ini adalah hubungan 'Adalah' dari contoh tersebut:

- Mamalia 'adalah' Binatang
- Reptil 'adalah' Binatang
- Anjing 'adalah' Mamalia
- Anjing 'adalah' Binatang

Dengan menggunakan kata kunci extends, subkelas mewarisi dan dapat menggunakan semua properti dari superkelas kecuali properti private dalam superkelas.

Kata kunci implements digunakan oleh kelas dengan mewarisi sebuah interface. Interface tidak dapat di-extends oleh subkelas.

Berikut adalah contohnya:

```
public interface Binatang
{ }

public class Mamalia implements Binatang
{ }

public class Anjing extends Mamalia
{ }
```

16.2 Kata Kunci instanceof

Kata kunci instanceof digunakan untuk memeriksa apakah sebuah objek 'Adalah' tipe objek lain.

Berikut ini contoh penggunaan operator instanceof untuk memeriksa apakah Mamalia adalah Binatang, dan apakah Anjing adalah Binatang.

```
interface Binatang
{ }

class Mamalia implements Binatang
{ }

public class Anjing extends Mamalia
{
  public static void main(String args[])
  {
```

```
        Mamalia m = new Mamalia();
        Anjing a = new Anjing();

        System.out.println(m instanceof Binatang);
        System.out.println(a instanceof Mamalia);
        System.out.println(a instanceof Binatang);
    }
}
```

Berikut ini hasil dari kode tersebut:

```
true
true
true
```

16.3 Hubungan 'Memiliki'

Hubungan ini menentukan apakah sebuah kelas Memiliki sesuatu.

Berikut contohnya:

```
public class Kendaraan
{ }

public class Kecepatan
{ }

public class Van extends Kendaraan
{
  private Kecepatan sp;
}
```

Contoh tersebut menunjukkan bahwa kelas Van Memiliki Kecepatan. Dengan membuat kelas Kecepatan yang terpisah, Anda tidak perlu memasukkan semua kode kecepatan dalam kelas Van, dan dapat menggunakan kelas Kecepatan secara langsung.

17 EKSEPSI DALAM JAVA

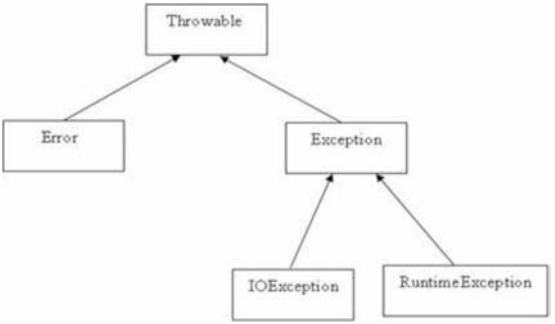
Eksepsi adalah masalah yang muncul saat mengeksekusi program. Sebuah eksepsi dapat terjadi karena berbagai hal seperti:

- Pengguna memasukkan data yang tidak valid.
- File yang perlu dibuka tidak ditemukan.
- Koneksi jaringan terputus.
- Kesalahan-kesalahan lainnya.

Semua kelas eksepsi merupakan subtype dari kelas java.lang.Exception. Kelas Exception merupakan subkelas dari kelas Throwable. Selain kelas Exception ada juga subkelas Error.

Kelas Exception memiliki dua subkelas utama yaitu IOException dan RuntimeException.

Berikut ini gambar diagramnya:



Gambar 17.1 Diagram kelas Throwable

17.1 Eksepsi built-in dalam Java

Terdapat dua jenis eksepsi built-in dalam Java.

- Checked exceptions: eksepsi yang biasanya merupakan kesalahan pengguna atau masalah yang tidak dapat dilihat oleh programer. Sebagai contoh, jika file yang ingin dibuka tidak bisa ditemukan.
- Runtime exceptions: eksepsi yang merupakan kesalahan logika atau kesalahan penulisan kode yang dibuat oleh programer.
- Error: bukan merupakan eksepsi melainkan masalah yang muncul di luar masalah dari pengguna atau programer.

Berikut ini daftar Unchecked RuntimeException dalam Java:

Eksepsi	**Deskripsi**
ArithmeticException	Kesalahan aritmatika, seperti pembagian denga nol.
ArrayIndexOutOfBoundsExceptio n	Indeks array di luar batas.
ArrayStoreException	Penambahan elemen array dengan tipe yang tidak kompatibel.
ClassCastException	Penggunaan cast yang tidak valid.
IllegalArgumentException	Menggunakan argumen ilegal untuk

	memanggil method.
IllegalMonitorStateException	Operasi monitor ilegal.
IllegalStateException	Aplikasi sedang berada dalam keadaan yang tidak tepat.
IllegalThreadStateException	Operasi yang diminta tidak kompatibel dengan keadaan aplikasi.
IndexOutOfBoundsException	Beberapa tipe indeks di luar batas.
NegativeArraySizeException	Array dibuat dengan ukuran negatif.
NullPointerException	Penggunaan referensi null yang tidak valid.
NumberFormatException	Konversi string menjadi numerik yang tidak valid.
SecurityException	Percobaan mengubah keamanan aplikasi.
StringIndexOutOfBounds	Percobaan untuk menggunakan indeks di luar batas string.
UnsupportedOperationException	Ditemukan operasi yang tidak didukung.

Berikut ini daftar Checked Exceptions dalam java.lang.

Exception	**Description**
ClassNotFoundException	Kelas tidak ditemukan.
CloneNotSupportedException	Percobaan untuk melakukan cloning objek yang tidak mengimplimentasikan interface Cloneable.
IllegalAccessException	Akses terhadap kelas tidak diijinkan.
InstantiationException	Percoban untuk membuat objek dari kelas abstrak atau interface.
InterruptedException	Sebuah thread diinterupsi oleh thread yang lain.
NoSuchFieldException	File yang diminta tidak tersedia.
NoSuchMethodException	Method yang diminta tidak tersedia.

17.2 Method-Method dalam Exceptions

Berikut ini daftar method penting dalam kelas Throwable:

- public String getMessage()

  Mengembalikan pesan rinci mengenai eksepsi yang didapatkan.

- public Throwable getCause()

  Mengembalikan penyebab eksepsi.

- public String toString()

  Mengembalikan nama kelas disambungkan dengan hasil dari getMessage().

- public void printStackTrace()

  Menampilkan hasil dari toString() bersama dengan stack trace.

- public StackTraceElement [] getStackTrace()

  Mengembalikan sebuah array yang berisi setiap elemen pada stack trace.

- public Throwable fillInStackTrace()

  Mengisi stack trace dari objek Throwable dengan stack trace yang ada.

17.3 Menangkap Eksepsi

Sebuah method menangkap eksepsi dengan menggunakan kombinasi kata kunci try dan catch.

Berikut sintaksnya:

```
try
{
  // Kode yang diproteksi
}
catch(NamaEksepsi e1)
{
  // Blok catch
}
```

Berikut ini contoh penggunaannya:

```
// Nama File : TesEksep.java
import java.io.*;

public class TesEksep
{
  public static void main(String args[])
  {
    try
    {
      int a[]= new int[2];
      System.out.println("Mengakses elemen ketiga :"+ a[3]);
    }
    catch(ArrayIndexOutOfBoundsException e)
    {
      System.out.println("Eksepsi yang muncul : " + e);
    }

    System.out.println("Di luar batas");
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Eksepsi yang muncul :
        java.lang.ArrayIndexOutOfBoundsException:3
Di luar batas
```

17.4 Menggunakan Beberapa Blok Catch

Sebuah blok try dapat diikuti dengan beberapa blok catch.

Berikut sintaksnya:

```
try
{
  // Kode yang diproteksi
}
catch(TipeEksepsi1 e1)
{
  // Blok catch
}
catch(TipeEksepsi2 e2)
{
  // Blok catch
}
catch(TipeEksepsi3 e3)
{
  // Blok catch
}
```

Berikut ini contoh penggunaannya:

```
try
{
  file = newFileInputStream(namaFile);
  x =(byte) file.read();
}
catch(IOException i)
{
  i.printStackTrace();
  return-1;
}
catch(FileNotFoundException f) // Tidak valid!
{
  f.printStackTrace();
  return-1;
}
```

17.5 Kata Kunci Throws/Throw

Jika sebuah method tidak mengatasi sebuah eksepsi checked, method tersebut harus mendeklarasinya dengan menggunakan kata kunci throws.

Anda dapat melempar sebuah eksepsi, baik eksepsi baru maupun eksepsi yang baru saja ditangkap, dengan menggunakan kata kunci throw. Hal ini yang membedakan kedua kata kunci throws dan throw.

Berikut ini contoh penggunaannya:

```
import java.io.*;

public class namaKelas
{
  public void deposit(double jumlah) throws RemoteException
  {
    // Implimentasi method
    throw new RemoteException();
  }

  // Definisi kelas selanjutnya
}
```

17.6 Kata Kunci Finally

Kata kunci finally digunakan untuk membuat blok kode yang mengikuti blok try. Blok kode finally selalu dieksekusi, baik terjadi eksepsi ataupun tidak.

Berikut sintaksnya:

```
try
{
  // Kode yang diproteksi
}
catch(TipeEksepsi1 e1)
{
  // Blok catch
}
catch(TipeEksepsi2 e2)
{
  // Blok catch
}
catch(TipeEksepsi3 e3)
{
  // Blok catch
}
finally
{
  // Blok finally selalu dieksekusi.
}
```

Berikut ini contoh penggunaannya:

```
public class TesEksep
{
  public static void main(String args[])
  {
    int a[]= new int[2];

    try
    {
      System.out.println("Mengakses elemen ketiga :"+ a[3]);
    }
    catch(ArrayIndexOutOfBoundsException e)
    {
      System.out.println("Eksepsi yang muncul :"+ e);
    }
    finally
    {
      a[0]=6;

      System.out.println("Nilai elemen pertama: " + a[0]);
      System.out.println("Statemen finally dieksekusi");
    }
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Eksepsi yang muncul:
        java.lang.ArrayIndexOutOfBoundsException:3
Nilai elemen pertama:6
Statemen finally dieksekusi
```

17.7 Deklarasi Eksepsi Buatan Sendiri

Anda dapat membuat eksepsi Anda sendiri dalam Java.

Berikut ini sintaks untuk membuat kelas eksepsi Anda sendiri:

```
class EksepsiKu extends Exception
{ }
```

Berikut ini contoh penggunaannya:

```
// Nama file DanaTidakValid.java
import java.io.*;

public class DanaTidakValid extends Exception
{
  private double jumlah;

  public DanaTidakValid(double jumlah)
  {
    this.jumlah = jumlah;
  }

  public double ambilJumlah()
  {
    return jumlah;
  }
}
```

18 OVERRIDE DALAM JAVA

Jika sebuah kelas mewarisi method dari superkelasnya, maka kelas tersebut dapat melakukan override method yang tidak dideklarasikan sebagai final.

Berikut contohnya:

```
class Binatang
{
  public void bergerak()
  {
    System.out.println("Binatang bisa bergerak");
  }
}

class Anjing extends Binatang
{
  public void bergerak()
  {
    System.out.println("Anjing bisa berjalan dan lari");
  }
}

public class TesAnjing
{
  public static void main(String args[])
  {
    // Referensi dan objek Binatang
    Binatang a = new Binatang();

    // Objek Anjing yang juga merupakan referensi Binatang
    b = new Anjing();

    a.bergerak(); // menjalankan method pada kelas Binatang
    b.bergerak(); // menjalankan method pada kelas Anjing
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Binatang bisa bergerak
Anjing bisa berjalan dan lari
```

18.1 Aturan dalam Melakukan Override Method

- Daftar argumen harus persis sama seperti pada method yang di-override.
- Tipe balikan harus sama atau merupakan subtipe dari tipe balikan yang dideklarasikan dalam method asli pada superkelas.
- Level akses tidak boleh lebih terbatas dari method yang di-override.
- Method instance dapat di-override hanya jika method tersebut diwarisi oleh subkelas.
- Method yang dideklarasikan final tidak bisa di-override.
- Konstruktor tidak bisa di-override.

18.2 Menggunakan Kata Kunci Super

Untuk menggunakan method versi superkelas dari method yang di-override, Anda dapat menggunakan kata kunci super.

```
class Binatang
{
  public void bergerak()
  {
    System.out.println("Binatang bisa bergerak");
  }
}

class Anjing extends Binatang
{
  public void bergerak()
  {
    super.bergerak(); // menggunakan method dari superkelas
    System.out.println("Anjing bisa berjalan dan lari");
  }
}

public class TesAnjing
{
  public static void main(String args[])
  {
    // Referensi Binatang, objek Anjing
```

```
        Binatang b = new Anjing();

        b.bergerak(); // menjalankan method pada kelas Anjing
    }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Binatang bisa bergerak
Anjing bisa berjalan dan lari
```

19 POLIMORFISME DALAM JAVA

Polimorfisme adalah kemampuan suatu objek untuk memiliki banyak bentuk. Perhatikan contoh berikut:

```
public interface Vegetarian{}
public class Binatang{}
public class Rusa extends Binatang implements Vegetarian{}
```

Kelas Rusa memiliki sifat polimorfis karena memiliki lebih dari satu pewarisan. Berikut ini penjelasan dari contoh di atas:

- Rusa adalah Binatang
- Rusa adalah Vegetarian
- Rusa adalah Rusa
- Rusa adalah Objek

Saat Anda menggunakan variabel referensi pada objek referensi Rusa, deklarasi berikut dapat digunakan:

```
Rusa r = new Rusa();
Binatang b = r;
Vegetarian v = r;
Object o = r;
```

19.1 Method Virtual

Sebuah method yang di-override pada dasarnya tersembunyi dalam super kelas, dan tidak digunakan kecuali subkelas menggunakan kata kunci super.

Perhatikan contoh berikut:

```
/* File : Karyawan.java */
public class Karyawan
{
  private String nama;
  private String alamat;
  private int nomor;

  public Karyawan(String nama,String alamat,int nomor)
  {
    System.out.println("Membuat satu Karyawan");
    this.nama = nama;
    this.alamat = alamat;
    this.nomor = nomor;
  }

  public void mailCheck()
  {
    System.out.println("Mengirim sebuah check untuk " +
                this.nama + " " + this.alamat);
  }

  public String toString()
  {
    return nama +" "+ alamat +" "+ nomor;
  }

  publicString getName()
  {
    return nama;
  }

  public String getAddress()
  {
    return alamat;
  }

  public void setAddress(String alamatBaru)
  {
    alamat = alamatBaru;
  }


  public int getNumber()
  {
    return nomor;
  }
}
```

Misalkan Anda meng-extend kelas Karyawan seperti ini:

```
/* File : Gaji.java */
public class Gaji extends Karyawan
{
  private double gaji; // Gaji tahunan

  public Gaji(String nama, String alamat, int nomor, double
              gaji)
  {
    super(nama, alamat, nomor);
    setSalary(gaji);
  }

  public void mailCheck()
  {
    System.out.println("Dalam mailCheck dari kelas Gaji ");
    System.out.println("Mengirim sebuah check untuk " +
                       getName() +" dengan gaji "+ gaji);
  }

  public double getSalary()
  {
    return gaji;
  }

  public void setSalary(double gajiBaru)
  {
    if(gajiBaru >=0.0)
    {
      gaji = gajiBaru;
    }
  }

  public double computePay()
  {
    System.out.println("Menghitung pembayaran gaji untuk " +
                        getName());
    return gaji/52;
  }
}
```

Perhatikan kode berikut:

```
/* File : VirtualDemo.java */
public class VirtualDemo
{
  public static void main(String[] args)
  {
    Gaji g = new Gaji("Mohd Mohtashim","Ambehta, UP",
                      3,3600.00);

    Karyawan k = new Gaji("John Adams","Boston, MA",
                          2,2400.00);

    System.out.println("Memanggil mailCheck dengan menggunakan
                        referensi Gaji --");
    g.mailCheck();
```

```
        System.out.println("\n Memanggil mailCheck dengan
                           menggunakan referensi Karyawan --");
        k.mailCheck();
    }
}
```

Berikut ini hasil output dari kode tersebut:

```
Membuat satu Karyawan
Membuat satu Karyawan
Memanggil mailCheck dengan menggunakan referensi Gaji --
Dalam mailCheck dari kelas Gaji
Mengirim sebuah check untuk MohdMohtashim dengan gaji 3600.0

Memanggil mailCheck dengan menggunakan referensi Karyawan --
Dalam mailCheck dari kelas Gaji
Mengirim sebuah check untuk JohnAdams dengan gaji 2400.0
```

Pada contoh tersebut, dibuat dua objek Gaji, satu dengan referensi Gaji g, dan yang lain dengan referensi Karyawan k.

Pada saat memanggil mailCheck() pada g, yang dijalankan adalah method yang ada pada kelas Gaji. Sedangkan saat memanggil mailCheck() pada k, yang dijalankan adalah method yang ada pada kelas Karyawan.

Pada saat compile, mailCheck() pada Karyawan digunakan untuk memvalidasi statemen ini. Tetapi pada saat program dijalankan, mailCheck() pada Gaji yang dijalankan. Perilaku ini merupakan pemanggilan method virtual.

20 Abstraksi dalam Java

Kelas abstrak adalah kelas yang tidak dapat dibuat instance nya. Semua fungsionalisasi dari suatu kelas tetap ada, dan field, method, dan konstruktornya tetap dapat diakses. Tetapi Anda tidak dapat membuat instance dari kelas abstrak tersebut.

20.1 Kelas Abstrak

Gunakan kata kunci abstract untuk mendeklarasikan suatu kelas sebagai kelas abstrak. Kata kunci ini muncul pada deklarasi kelas sebelum kata kunci class.

```
/* File : Karyawan.java */
public abstract class Karyawan
{
  private String nama;
  private String alamat;
  private int nomor;

  public Karyawan(String nama,String alamat,int nomor)
  {
    System.out.println("Membuat satu Karyawan");
    this.nama = nama;
    this.alamat = alamat;
    this.nomor = nomor;
  }

  public double computePay()
  {
    System.out.println("Dalam computePay Karyawan");
    return0.0;
  }
```

```
  public void mailCheck()
  {
    System.out.println("Mengirim sebuah check untuk " +
                this.nama + " " + this.alamat);
  }

  public String toString()
  {
    return nama +" "+ alamat +" "+ nomor;
  }

  publicString getName()
  {
    return nama;
  }

  public String getAddress()
  {
    return alamat;
  }

  public void setAddress(String alamatBaru)
  {
    alamat = alamatBaru;
  }

  public int getNumber()
  {
    return nomor;
  }
}
```

Kelas tersebut tidak berbeda dengan kelas Karyawan sebelumnya. Kelas ini sekarang merupakan kelas abstrak, tetapi masih memiliki tiga field, tujuh method, dan satu konstruktor.

Berikut contoh kode untuk mencoba kelas abstrak ini:

```
/* File name : AbstractDemo.java */
public class AbstractDemo
{
  public static void main(String[] args)
  {

    /* Kode berikut akan menyebabkan error */
    Karyawan k = new Karyawan("George W.","Houston, TX",43);

    System.out.println("\n Memanggil mailCheck dengan
                        menggunakan referensi Karyawan --");
    k.mailCheck();
  }
}
```

Kode tersebut akan menghasilkan error berikut:

```
Karyawan.java:46:Karyawan is abstract; cannot be instantiated
Karyawan k = new Karyawan("George W.","Houston, TX",43);
^
1 error
```

20.2 Meng-Extend Kelas Abstrak

Anda dapat meng-extend kelas Karyawan sebagai berikut:

```
/* File : Gaji.java */
public class Gaji extends Karyawan
{
  private double gaji; // Gaji tahunan

  public Gaji(String nama, String alamat, int nomor, double
              gaji)
  {
    super(nama, alamat, nomor);
    setSalary(gaji);
  }

  public void mailCheck()
  {
    System.out.println("Dalam mailCheck dari kelas Gaji ");
    System.out.println("Mengirim sebuah check untuk " +
                       getName() +" dengan gaji "+ gaji);
  }

  public double getSalary()
  {
    return gaji;
  }

  public void setSalary(double gajiBaru)
  {
    if(gajiBaru >=0.0)
    {
      gaji = gajiBaru;
    }
  }

  public double computePay()
  {
    System.out.println("Menghitung pembayaran gaji untuk " +
                        getName());
    return gaji/52;
  }
}
```

Pada bagian ini Anda tidak dapat membuat instance Karyawan baru, tetapi jika Anda membuat instance dari Gaji, objek Gaji tersebut akan mewarisi tiga field dan tujuh method dari Karyawan.

```
/* File : AbstractDemo.java */
```

```
public class AbstractDemo
{
  public static void main(String[] args)
  {
    Gaji g = new Gaji("Mohd Mohtashim","Ambehta, UP",
                      3,3600.00);

    Karyawan k = new Gaji("John Adams","Boston, MA",
                          2,2400.00);

    System.out.println("Memanggil mailCheck dengan menggunakan
                        referensi Gaji --");
    g.mailCheck();

    System.out.println("\n Memanggil mailCheck dengan
                        menggunakan referensi Karyawan --");
    k.mailCheck();
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Membuat satu Karyawan
Membuat satu Karyawan
Memanggil mailCheck dengan menggunakan referensi Gaji --
Dalam mailCheck dari kelas Gaji
Mengirim sebuah check untuk MohdMohtashim dengan gaji 3600.0

Memanggil mailCheck dengan menggunakan referensi Karyawan --
Dalam mailCheck dari kelas Gaji
Mengirim sebuah check untuk JohnAdams dengan gaji 2400.0
```

20.3 Method Abstrak

Jika Anda ingin suatu kelas memiliki method tetapi implementasi methodnya dilakukan pada subkelas, Anda dapat mendeklarasi method tersebut sebagai abstrak.

Method abstrak tidak memiliki definisi, dan method tersebut diakhiri dengan tanda titik koma (;).

```
public abstract class Karyawan
{
  private String nama;
  private String alamat;
  private int nomor;

  public abstract double computePay();

  // Definisi kelas selanjutnya....

}
```

Mendeklarasikan sebuah method sebagai abstrak menyebabkan:

- Kelas tersebut juga harus dideklarasikan sebagai abstrak.
- Semua subkelas harus meng-override method tersebut atau mendeklarasikan diri sebagai abstrak.

Jika Gaji mewarisi Karyawan, maka Gaji perlu mengimplimentasikan method computePay():

```
/* File : Gaji.java */
public class Gaji extends Karyawan
{
  private double gaji; // Gaji tahunan

  public double computePay()
  {
    System.out.println("Menghitung pembayaran gaji untuk " +
                       getName());
    return gaji/52;
  }

  // Definisi kelas selanjutnya....
}
```

21 Enkapsulasi dalam Java

Enkapsulasi adalah teknik untuk membuat field dalam kelas menjadi private dan menyediakan akses pada field melalui method public. Jika field dideklarasikan sebagai private, field tersebut tidak dapat diakses dari luar kelas.

Kegunaan enkapsulasi adalah untuk memungkinkan modifikasi kode implementasi tanpa mengubah kode yang sudah ada.

Berikut ini contoh penggunaannya:

```
/* File : EncapTest.java */
public class EncapTest
{
  private String nama;
  private String id;
  private int umur;

  public int getUmur()
  {
    return umur;
  }

  publicString getNama()
  {
    return nama;
  }


  publicString getId()
  {
    return id;
  }

  publicvoid setUmur(int umurBaru)
  {
    umur = umurBaru;
```

```
  }

  publicvoid setNama(String namaBaru)
  {
    nama = namaBaru;
  }

  public void setId(String idBaru)
  {
    id = idBaru;
  }
}
```

Method-method public adalah titik akses untuk menggunakan field pada kelas ini dari luar, method-method ini disebut getters dan setters. Kelas lain yang ingin mengakses variabel-variabel yang ada dapat menggunakan get dan set ini.

Berikut ini contoh mengakses variabel yang ada:

```
/* File : RunEncap.java */
public class RunEncap
{
  public static void main(String args[])
  {
    EncapTest encap = new EncapTest();

    encap.setNama("James");
    encap.setUmur(20);
    encap.setId("12343");

    System.out.print("Nama : " + encap.getNama() + " Umur : "
                    + encap.getAge());
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Nama : James Umur : 20
```

22 INTERFACE DALAM JAVA

Interface merupakan kumpulan method-method abstrak. Sebuah kelas yang mengimplementasikan interface, mewarisi method-method abstrak dari interface tersebut.

Berikut ini persamaan interface dengan kelas:

- Interface dapat memiliki banyak method.
- Interface ditulis dalam file dengan ekstensi .java.

Berikut ini perbedaan interface dengan kelas:

- Anda tidak bisa membuat instance dari sebuah interface.
- Interface tidak memiliki konstruktor.
- Semua method dalam interface adalah abstrak.
- Interface tidak bisa memiliki field instance.
- Interface tidak di-extend, tetapi di-implements oleh kelas.
- Sebuah interface dapat meng-extend beberapa interface lainnya.

22.1 Mendeklarasikan Interface

Kata kunci interface digunakan untuk mendeklarasikan sebuah interface.

```
/* File: NamaInterface.java */
import java.lang.*;
// Satu atau beberapa statemen import lainnya

public interface NamaInterface
{
  // Satu atau beberapa field static final

  // Satu atau beberapa deklarasi method abstrak
}
```

Interface memiliki properti berikut:

- Interface secara implisit merupakan abstrak. Anda tidak perlu menggunakan kata kunci abstract ketika mendeklarasikan sebuah interface.
- Setiap method dalam interface juga secara implisit merupakan abstrak, sehingga Anda tidak perlu menggunakan kata kunci abstract.
- Method dalam interface secara implisit merupakan public.

Berikut ini contohnya:

```
/* File : Binatang.java */
interface Binatang
{
  public void makan();
  public void berjalan();
}
```

22.2 Mengimplementasikan Interface

Ketika sebuah kelas meng-implements interface, kelas tersebut harus mengimplimentasikan method yang ada dalam interface.

Berikut ini contohnya:

```
/* File : MammalInt.java */
public class MammalInt implements Binatang
{
  public void makan()
  {
    System.out.println("Mamalia makan");
  }

  public void berjalan()
  {
    System.out.println("Mamalia berjalan");
  }
```

```
  public int jumlahKaki()
  {
    return 0;
  }

  public static void main(String args[])
  {
    MammalInt m = new MammalInt();

    m.makan();
    m.berjalan();
  }
}
```

Kode tersebut akan menghasilkan output berikut:

```
Mamalia makan
Mamalia berjalan
```

22.3 Meng-Extends Interface

Sebuah interface dapat meng-extends interface lainnya.

Berikut ini contohnya:

```
//File: Olahraga.java
public interface Olahraga
{
  public void setTim1(String nama);
  public void setTim2(String nama);
}

//File: SepakBola.java
public interface SepakBola extends Olahraga
{
  public void skorTim1(int poin);
  public void skorTim2(int poin);
  public void akhirBabak(int babak);
}

//File: Hoki.java
public interface Hoki extends Olahraga
{
  public void skorTim1();
  public void skorTim2();
  public void akhirPeriod(int period);
  public void hasilAkhir(int hasil);
}
```

23 Distribusi Aplikasi Java

Untuk mendistribusikan aplikasi Java yang telah Anda buat, Anda dapat menggunakan IDE Netbeans ***(https://netbeans.org/downloads/)*** untuk memudahkan baik pembuatan maupun pendistribusian aplikasi.

Untuk menggunakan aplikasi yang sudah jadi (file JAR), Anda dapat menggunakan tiga cara:

- Klik dua kali file aplikasi Java Archive (JAR).
- Memanggil aplikasi dari command line.
- Memanggil aplikasi dari file script.

23.1 Membuat dan Mendistribusikan File JAR

Berikut langkah-langkah untuk membuat dan mendistribusikan file JAR.

Membuat Proyek Aplikasi

- Pada IDE NetBeans, pilih File - New Project.
- Pada bagian General Category, pilih proyek Java sesuai dengan aplikasi yang akan Anda buat.
- Klik Finish.

Proyek tersebut akan ditampilkan pada IDE dan Anda dapat mulai memasukkan kode aplikasi tersebut.

Build Proyek dan Membuat file JAR

- Pilih Build - Build Main Project.
- Folder build dan dist akan ditambahkan secara otomatis ke dalam folder proyek Anda.
- Semua kode sumber akan dicompile ke dalam file .class, dalam folder build.
- Sebuah file JAR yang berisi proyek Anda akan dibuat secara otomatis di dalam folder dist.
- Jika Anda menggunakan library dalam proyek, library tersebut akan ditempatkan dalam folder lib yang akan dibuat secara otomatis dalam folder dist.

Menjalankan Aplikasi di Luar IDE

Untuk menjalankan aplikasi di luar IDE, Anda dapat mengklik dua kali file .jar pada folder dist.

Mendistribusikan Aplikasi untuk Pengguna Lain

Untuk mendistribusikan aplikasi untuk pengguna lain (komputer lain), lakukan langkah-langkah berikut:

- Buat file zip yang berisi file JAR (.jar) dari aplikasi dan folder lib yang mengandung library-library.
- Kirim file tersebut untuk pengguna / komputer lain. Pastikan file zip kemudian diekstrak sehingga file .jar dan folder lib berada dalam folder yang sama.
- Pengguna kemudian dapat mengklik dua kali file .jar untuk menjalankan aplikasi.

23.2 Menjalankan Aplikasi melalui Command Line

Untuk menjalankan aplikasi melalui command line, lakukan langkah-langkah berikut:

- Buka jendela command prompt.

- Ubah direktori menjadi folder LOKASI_PROYEK/dist (gunakan perintah cd).
- Masukkan perintah berikut untuk menjalankan aplikasi:

```
java -jar NamaAplikasi.jar
```

24 Membuat Aplikasi CRUD Sederhana

Aplikasi CRUD (Create, Read, Update, Delete) adalah aplikasi yang terintegrasi dengan database dan dapat melakukan proses Membuat, Menampilkan, Mengubah, dan Menghapus data.

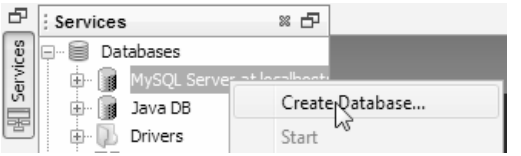
Pada bagian ini Anda akan belajar membuat aplikasi CRUD sederhana dengan menggunakan NetBeans IDE dan database MySQL.

24.1 Menyiapkan Database

Pertama-tama Anda perlu menyiapkan aplikasi dengan membuka NetBeans dan menjalankan MySQL.

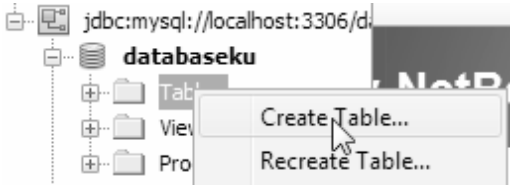
Anda perlu menyiapkan database MySQL terlebih dahulu dengan langkah-langkah berikut:

1. Pilih **Services** di sebelah kiri atas IDE dan kemudian klik kanan **MySQL Server at…** dan pilih **Create Database**.



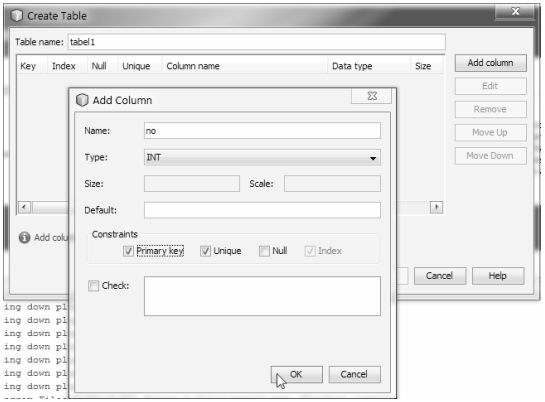
Gambar 24.1 Menyiapkan database MySQL

2. Setelah itu masukkan nama database dan klik **OK**.
3. Perluas database, klik kanan folder **Tables** dan pilih **Create Table**.



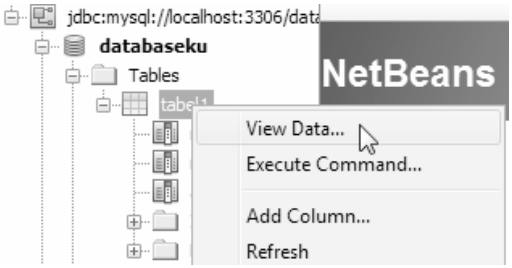
Gambar 24.2 Membuat tabel

4. Beri nama **tabel1**, pilih **Add column** dan beri nama kolom no, tipe INT, dan pilih Primary key, kemudian klik **OK**.



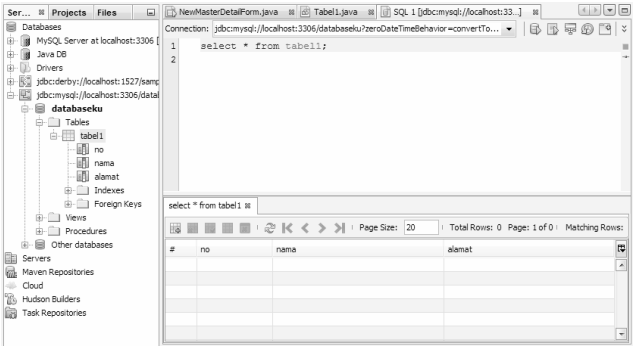
Gambar 24.3 Menambahkan tabel beserta kolom

5. Dengan cara yang sama, tambahkan kolom nama, tipe VARCHAR, ukuran 100, dan kolom alamat, tipe TEXT. Kemudian klik **OK**.
6. Perluas tabel1, klik kanan dan pilih **View Data…** Anda kemudian dapat melihat tabel1 ditampilkan dalam IDE.



Gambar 24.4 Menampilkan tabel1

7. Tabel database Anda akan muncul pada **IDE**.

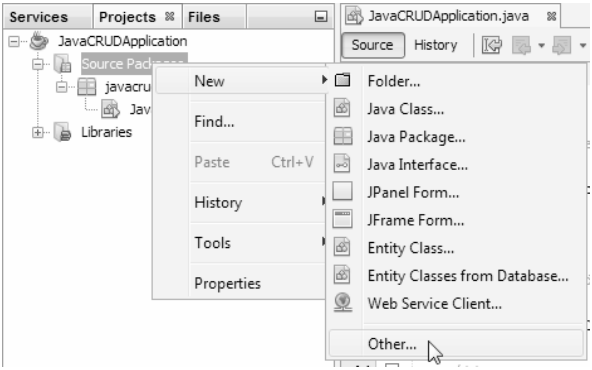


Gambar 24.5 Tampilan tabel database

24.2 Menyiapkan Aplikasi

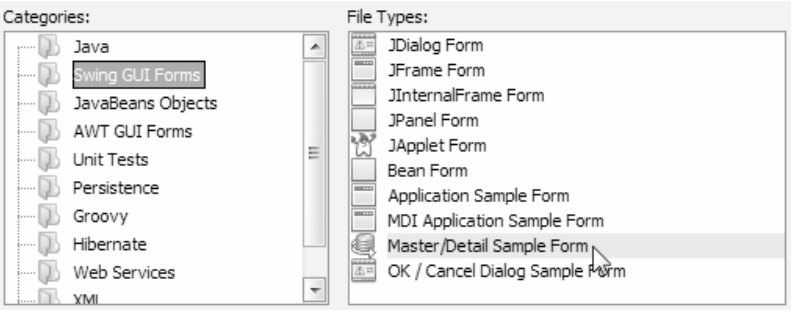
Setelah membuat tabel database, Anda perlu membuat proyek aplikasi Java dengan langkah-langkah berikut:

1. Pilih **File - New Project**.
2. Pilih **Java - Java Application**, kemudian klik **Next**.
3. Ubah nama proyek sesuai dengan keinginan Anda kemudian klik Finish.
4. Perluas aplikasi, klik kanan **Source Packages**, kemudian pilih **New - Other…**



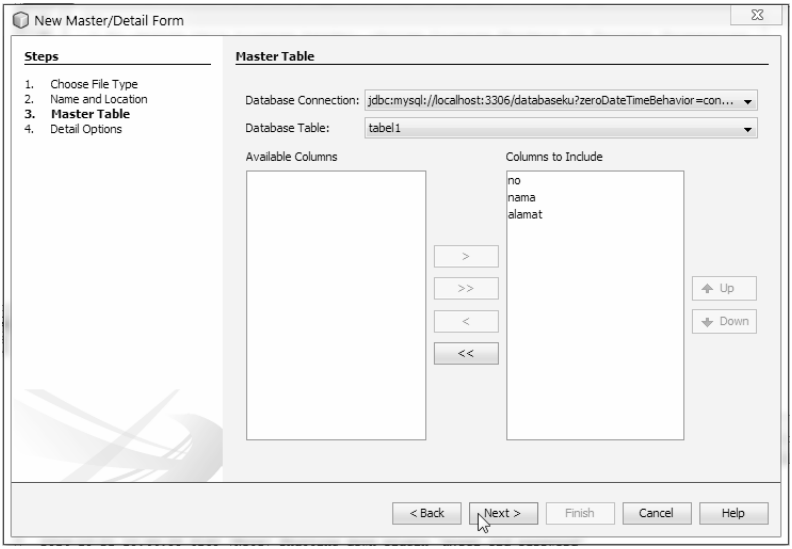
Gambar 24.6 Menambahkan Form

5. Pilih kategori **Swing GUI Forms** dan tipe file **Master/Detail Sample Form** kemudian klik **Next**.



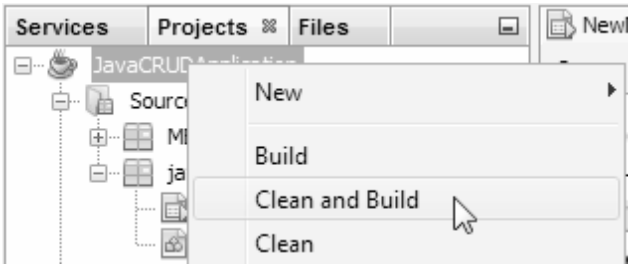
Gambar 24.7 Memilih Form

6. Anda kemudian dapat mengubah nama kelas sesuai dengan keinginan Anda dan kemudian mengklik **Next**.
7. Pada **Master Table** pilih koneksi database yang baru saja Anda buat, pilih tabel1, klik **Next** dan kemudian klik **Finish**.



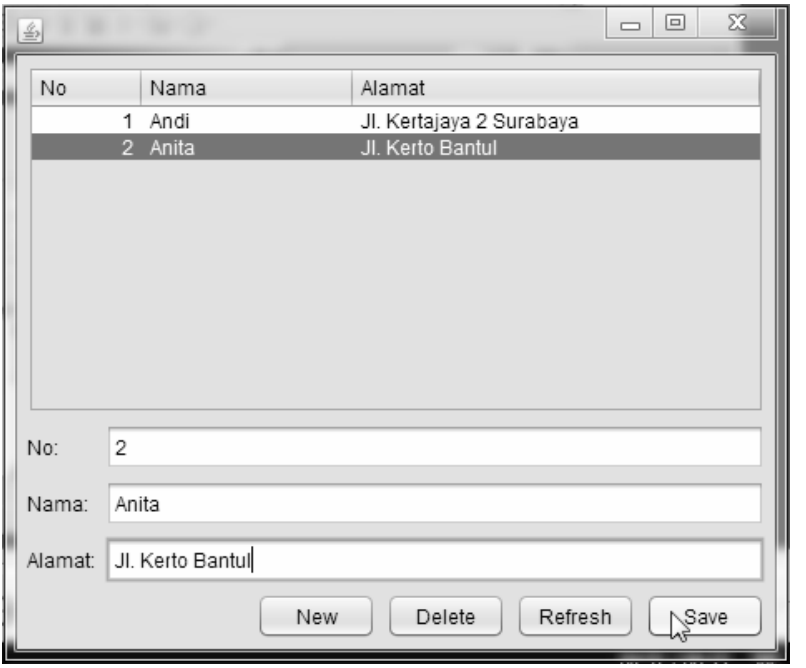
Gambar 24.8 Memilih tabel database

8. Hapus NamaProyek.java di sebelah kiri untuk menghindari error saat menjalankan aplikasi.
9. Klik kanan proyek kemudian pilih **Clean and Build**.



Gambar 24.9 Membersihkan dan mem-build aplikasi

10. Anda kemudian dapat menjalankan aplikasi dengan memilih **Run**.



Gambar 24.10 Aplikasi CRUD sederhana

Jika Anda ingin belajar pemrograman Java dari nol, maka buku ini cocok untuk Anda baca. Buku ini mengupas Java benar-benar dari awal sehingga programmer pemula yang baru ingin mencicipi Java tidak akan kesulitan.

Buku ini mengupas tentang:

- Sintaks Dasar Java
- Variabel, Objek, dan Kelas
- Tipe Data
- Tipe-Tipe Modifier
- Operator-Operator Dasar
- Kontrol Perulangan
- Override dalam Java
- Abstraksi dalam Java
- Enkapsulasi dalam Java
- Dan banyak lagi

Buku ini penting dipelajari kalau Anda ingin masuk ke dunia pemrograman Java.

PT ELEX MEDIA KOMPUTINDO
Kompas Gramedia Building
Jl. Palmerah Barat 29-37, Jakarta 10270
Telp. (021) 53650110-53650111, Ext 3214
Webpage: http://www.elexmedia.co.id

gramediana

Kelompok
Pemrograman
Keterampilan
- [x] Tingkat Pemula
- [x] Tingkat Menengah
- [ ] Tingkat Mahir

Jenis Buku
- [x] Referensi
- [x] Tutorial
- [ ] Latihan

ISBN 978-602-02-5545-3
9 786020 255453
121142672