

Budi Hartono. M.Kom

Pemrograman Java UNTUK Pemula



YAYASAN PRIMA AGUS TEKNIK

Pemrograman Java untuk Pemula

Penulis :

Budi Hartono, M.Kom

ISBN :

Editor :

Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

Penyunting :

Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :

Irdha Yunianto

Penebit :

Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :

Jl. Majapahit no 605 Semarang

Telp. (024) 6723456

Fax. 024-6710144

Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :

Universitas STEKOM

Jl. Majapahit no 605 Semarang

Telp. (024) 6723456

Fax. 024-6710144

Email : info@stekom.ac.id

Hak cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara apapun tanpa ijin tertulis dari penerbit

KATA PENGANTAR

Puji syukur pada Tuhan Yang Maha Esa bahwa buku yang berjudul ***“Pemrograman Java untuk Pemula”***. Pemrograman Java merupakan salah satu bahasa pemrograman terpopuler di dunia. Bahasa Java dapat dikembangkan ke aplikasi untuk web, desktop, mobile, hingga embedded dan Internet of The Things. Jika ingin menjadi pengembang dalam pemrograman Java harus memiliki pengetahuan dasar pemrograman Java seperti Java Dasar, Control Flow, Collection Type, dan Pemrograman berorientasi Objek. Selain itu Pemrograman Java merupakan open source program, sehingga siapa saja dapat dipelajari dan digunakan.

Buku ini ditujukan untuk pemula yang ingin mempelajari dasar-dasar pemrograman Java dengan mengacu pada standar industri. Buku ini juga ditulis khusus untuk pemula sehingga tidak ada syarat khusus dalam pemahaman pemrograman sebelumnya. Namun, jauh lebih baik jika memiliki pengetahuan dasar tentang pemrograman.

Buku ini dibagi menjadi, 7 bab yang akan menjelaskan serta menjabarkan secara rinci tentang pemrograman java. Bab 1 sebagai pembuka buku ini akan membahas tentang pengenalan tentang apa itu Java. Bab 2 akan memulai dengan menginstal Java dan memperkenalkan tentang menu-menu Java. Selanjutnya Bab 3 tentang variable-variabel yang akan ditemukan dalam program ini, serta cara penyelesaiannya. Bab 4 ini akan dibahas perintah pemrograman Java untuk struktur percabangan kode program atau dikenal juga dengan struktur kondisi / struktur logika. Di dalam bahasa Java terdapat kondisi IF, IF ELSE, IF ELSE IF, Nested IF ELSE (if bersarang), dan struktur SWITCH / CASE. Perulangan (atau dikenal dengan istilah loop) adalah metode pemrograman yang berfungsi untuk mengulang baris perintah. Akhir dari bab ini akan dibahas cara membuat perulangan menggunakan perintah FOR, WHILE dan DO WHILE.

Bab selanjutnya akan menjelaskan tentang Array, bagaimana cara membuat loop array, penyortiran array, penyortiran hingga daftar java array. Juga akan diberikan contoh agar pembaca dapat mudah memahaminya. Bab 6 akan membahas Method adalah kode program yang dirancang untuk menyelesaikan sebuah tugas tertentu, dan merupakan bagian dari program utama. Bab akhir dari buku ini akan meneruskan Bab sebelumnya, tetapi bab ini akan menjelaskan tentang pengaplikasian method. Akhir kata semoga buku ini berguna bagi para pembaca.

Semarang, Juli 2022

Penulis

Budi Hartono, S.Kom, M.Kom

DAFTAR ISI

Halaman Judul	i
Kata Pengantar	iii
Daftar Isi	iv
BAB 1 PENGENALAN PEMROGRAMAN JAVA	1
1.1 Pemrograman Java – Sejarah Singkat	1
1.2 Fitur Utama java	2
BAB 2 MEMULAI	3
2.1 Mesin Virtual Java	3
2.2 Kit Pengembangan Perangkat Lunak Java	3
2.3 Sistem Java	4
2.4 Perangkat Lunak Netbeans	4
2.5 Komentar Java	7
2.6 Menjalankan Program Java	11
2.7 Berbagi Program Java	13
BAB 3 VARIABEL JAVA	15
3.1 Variabel Ganda	18
3.2 Variabel Short dan Float	19
3.3 Aritmatika Dasar	19
3.4 Keutamaan Operator	20
3.5 Variabel String	22
3.6 Menerima Masukan	26
3.7 Panel Opsi	29
BAB 4 ALIRAN KONTROL	34
4.1 Pernyataan If	34
4.2 If... Else	38
4.3 If ... Else If	38
4.4 Pernyataan Bersatu	41
4.5 Nilai Boolean dalam Pemrograman Java	42
4.6 Pernyataan Switch Java	43
4.7 Java Loop	45
4.8 While Loops	49
4.9 Do ... While	49
BAB 5 JAVA ARRAY	51
5.1 Apa Itu Java Array?	51
5.2 Loop dan Array	53
5.3 Pengurutan Array	54
5.4 String dan Array	57
5.5 Java Multi-Dimensional Array	59

5.6	Daftar Java Array	61
BAB 6 METODE STRING JAVA		64
6.1	Bagaimana Java Menangani String	64
6.2	Metode IndexOf	67
6.3	Metode Substring	70
6.4	Metode Equals	73
6.5	Metode CharAt	74
6.6	Metode Replace	75
6.7	Metode Trim	75
BAB 7 METODE JAVA		76
7.1	Format Metode Java	76
7.2	Cara Memanggil Metode Java Anda	77
7.3	Cara Lulus Nilai ke Metode Java	79
7.4	Kesimpulan	81
Daftar Pustaka		84

BAB 1

PENGENALAN PEMROGRAMAN JAVA

Dengan Pemrograman Berorientasi Objek seperti Java, hari ini dimungkinkan untuk mengatur program yang kompleks dan besar melalui enkapsulasi, polimorfisme, pewarisan, objek, dan kelas. Selama bertahun-tahun, C++ menggunakan bahasa OOP. Dengan munculnya World Wide Web, pemrograman Java menjadi lebih populer, terutama dalam perkembangan elektronik konsumen seperti televisi, microwave, dan banyak lagi. Pakar komputer mencurahkan banyak waktu mereka untuk mencoba menemukan perangkat lunak yang aman, andal, ringkas, dan tidak bergantung pada prosesor. Pemrograman Java secara bertahap berkembang menjadi bahasa pemrograman penuh, mengubah fokusnya dari elektronik konsumen ke berbagai platform untuk mengembangkan aplikasi yang lebih kuat.

1.1 PEMROGRAMAN JAVA – SEJARAH SINGKAT

Bahasa Pemrograman Java dikembangkan pada tahun 1991 oleh lima ahli komputer – Ed Frank, Mike Sheridan, Chris Warth, Patrick Naughton, dan James Gosling yang semuanya bekerja untuk Sun Microsystems Inc. Mereka membutuhkan waktu 18 bulan untuk mengembangkan program, yang awalnya bernama “Oak.” Itu berganti nama menjadi Java pada tahun 1995 karena masalah hak cipta. Konsepnya adalah untuk membuat bahasa pemrograman yang dapat digunakan di seluruh platform dan yang dapat membangun perangkat lunak tertanam untuk elektronik konsumen. Bahasa pemrograman populer pada saat itu, C dan C++ tidak efisien untuk tujuan ini, karena mereka bergantung pada platform karena program yang ditulis di dalamnya harus dikompilasi terlebih dahulu untuk perangkat keras tertentu sebelum diluncurkan. Selain itu, kode yang dikompilasi tidak efisien untuk prosesor lain dan harus dikompilasi ulang. Oleh karena itu, tim lima, yang juga dikenal sebagai Tim Hijau, mulai bekerja membangun bahasa pemrograman yang lebih mudah. Mereka bermain-main selama satu setengah tahun dalam menciptakan bahasa pemrograman platform-independen yang kompak, yang memungkinkan seorang programmer untuk membangun kode yang dapat berjalan pada prosesor yang berbeda di bawah berbagai lingkungan.

Hal ini menyebabkan perkembangan Java. Bersamaan dengan itu, World Wide Web dan Internet menjadi populer. Program web masih bergantung pada platform, dan membutuhkan program yang dapat beroperasi pada OS apa pun terlepas dari konfigurasi perangkat lunak dan perangkat keras. Ini diperlukan untuk program yang ringkas dan kecil, yang dapat dengan mudah dibawa melalui jaringan. Java adalah bahasa yang memenuhi persyaratan tersebut. Pengembang web segera menyadari bahwa bahasa yang netral arsitektural seperti Java dapat menjadi ideal untuk menulis program untuk web. Oleh karena itu, Java menjadi lebih populer sebagai bahasa pemrograman untuk World Wide Web, dari awal yang sederhana sebagai bahasa untuk elektronik konsumen. Saat ini, Java jauh dari bahasa pemrograman dasar. Ini adalah teknologi yang dikembangkan dengan baik yang sederhana, aman, portabel, platform independen, multi-utas, berorientasi objek, terdistribusi, dan kuat.

1.2 FITUR UTAMA JAVA

Sederhana

Java dianggap sebagai bahasa yang sederhana, karena tidak memiliki fitur yang rumit seperti alokasi memori eksplisit, pointer, pewarisan berganda, dan overloading Operator.

Aman

Java dilengkapi dengan firewall virtual antara komputer dan aplikasi. Kode Java dibatasi di dalam Java Runtime Environment (JRE), yang tidak menyetujui akses tidak sah untuk sumber daya sistem.

Portabel

Sebuah kode yang ditulis dalam Java pada satu platform dapat berjalan pada platform lain pada mesin yang berbeda. Kode byte Java dapat diangkut ke platform apa pun untuk operasi, yang membuat kode java sangat portabel.

Platform Independen

Platform mengacu pada pengaturan yang telah ditetapkan sebelumnya untuk menjalankan program, mematuhi batasannya, dan menggunakan fitur-fiturnya. Selama fase kompilasi, program java diubah menjadi kode byte, yang dapat digunakan untuk platform apa pun seperti Mac/OS, Linux, atau Windows. Oleh karena itu, program yang telah dikompilasi di Linux masih dapat digunakan di Windows dan sebaliknya. Itulah mengapa Java adalah bahasa pemrograman platform independen.

Multi-utas

Java mendukung multi-threading karena memungkinkan program untuk melakukan beberapa tugas sekaligus.

Berorientasi pada objek

Java adalah bahasa pemrograman berorientasi objek, karena dapat mengatur program sebagai sekelompok objek yang masing-masing mewakili turunan dari kelas. Empat konsep utama OOP adalah: polimorfisme, pewarisan, enkapsulasi, dan abstraksi. Anda akan memahami setiap konsep saat Anda mempelajari pemrograman java.

Didistribusikan

Melalui pemrograman java, Anda dapat mengembangkan aplikasi terdistribusi. Enterprise Java Beans (EJB) dan Remote Method Invocation (RMI) digunakan untuk mengembangkan aplikasi terdistribusi menggunakan Java. Sederhananya, Anda dapat mendistribusikan program java pada beberapa sistem yang terhubung satu sama lain melalui Internet. Objek dalam Java Virtual Machine (JVM) dapat menjalankan protokol menggunakan JVM jarak jauh.

Kokoh

Kesalahan runtime yang salah penanganan dan kesalahan manajemen memori adalah dua masalah utama yang menyebabkan kegagalan program. Java dapat menangani masalah ini dengan efisiensi tinggi. Kesalahan runtime yang salah dapat diselesaikan melalui protokol Penanganan Pengecualian, sementara kesalahan manajemen memori dapat diselesaikan dengan pengumpulan sampah, yang merupakan de-alokasi otomatis objek yang sudah tidak diperlukan.

BAB 2

MEMULAI

Sebelum Anda dapat memulai dengan Java, Anda perlu menginstal semua yang Anda butuhkan. Perjuangannya nyata: Anda bisa mengalami sakit kepala bahkan sebelum Anda menulis satu kode pun. Bab ini dimaksudkan untuk membuat pengalaman Anda sedikit lebih mudah. Saya sarankan menulis kode Anda menggunakan NetBeans, yang merupakan perangkat lunak gratis dan salah satu Lingkungan Pengembangan Antarmuka (IDE) yang paling populer. Tetapi sebelum Anda bisa melakukannya, Anda perlu menginstal file dan komponen Java yang penting. Hal pertama yang perlu Anda instal adalah Java Virtual Machine.

2.1 MESIN VIRTUAL JAVA

Seperti yang telah kita bahas, Java adalah platform independen, sehingga dapat digunakan pada sistem operasi apa pun. Oleh karena itu, terlepas dari apakah Anda menggunakan Mac OS, Linux, atau Windows, semuanya akan sama dengan Java. Java Virtual Machine (JVM) adalah alasan utama mengapa program dapat berjalan di OS apa pun. Mesin Virtual adalah sebuah program, yang secara akurat dapat memproses semua kode Anda. Oleh karena itu, Anda perlu menginstal program ini sebelum Anda dapat mengoperasikan segala bentuk kode Java. Karena Java adalah milik Oracle, Anda harus terlebih dahulu mengunjungi situs web perusahaan untuk mengunduh JVM atau yang juga dikenal dengan JRE (Java Runtime Environment. Namun sebagai jalan pintas, Anda dapat mengklik tautan di bawah ini: <http://java.com/en/download/index.jsp>

Untuk memeriksa apakah JRE sudah terinstal di komputer Anda, cukup klik tautan "Apakah saya memiliki Java?" yang dapat Anda temukan di bawah tombol Unduh besar di bagian atas halaman. Setelah Anda mengklik tautan, PC Anda akan dipindai untuk mengetahui apakah Anda memiliki JRE. Pemberitahuan akan dikirimkan kepada Anda jika Anda memiliki JRE atau tidak. Jika Anda tidak memiliki JRE, Anda akan diminta untuk mengunduh dan menginstalnya. Namun sebagai jalan pintas, Anda dapat mengklik tautan di bawah ini: <http://java.com/en/download/manual/jsp>

Tautan di atas akan mengarahkan Anda ke halaman di mana Anda dapat mengunduh JRE secara manual. Halaman tersebut akan memberi Anda petunjuk dan tautan unduhan untuk OS yang berbeda. Setelah Anda mengunduh dan menginstal JRE, Anda mungkin perlu memulai ulang PC Anda, sebelum Anda dapat menggunakan program tersebut.

2.2 KIT PENGEMBANGAN PERANGKAT LUNAK JAVA

Bahkan jika Anda memiliki JRE, Anda tetap tidak dapat menulis program apa pun. JRE hanya akan mengizinkan program Java untuk berjalan di PC Anda. Untuk menulis kode dan mengujinya, Anda perlu mengunduh dan menginstal Java Software Development Kit. Anda dapat mengunduhnya dengan mengklik tautan di bawah ini:

<http://www.Oracle.com/technetwork/java/index.html>

Disarankan agar Anda menggunakan Java Standard Edition atau Java SE. Klik Unduhan Teratas yang terletak di sebelah kanan halaman, dan Anda akan diarahkan ke halaman terpisah yang berisi daftar opsi untuk mengunduh. Karena kami akan menggunakan NetBeans, Anda perlu menemukan JDK 8 dengan NetBeans. Klik tautan Unduh, yang akan mengarahkan Anda ke halaman lain. Temukan tautan untuk OS Anda. Tetapi perhatikan bahwa unduhan ini membutuhkan sekitar 290 MB untuk Windows (64 bit). Setelah mengunduh JDK dan NetBeans, Anda dapat menginstalnya di PC Anda. Dalam buku ini, kita akan menggunakan NetBeans untuk menulis kode. Tetapi sebelum menggunakan perangkat lunak, Anda harus terlebih dahulu memahami cara kerja di dunia Java.

2.3 SISTEM JAVA

Anda dapat menggunakan editor teks untuk menulis kode sebenarnya untuk program Anda. NetBeans menyediakan ruang khusus untuk menulis kode. Kode ini dikenal sebagai kode sumber, dan disimpan dalam ekstensi file - .java. Program khusus yang dikenal sebagai Javac akan digunakan untuk mengubah kode sumber menjadi kode byte. Proses ini dikenal sebagai kompilasi. Setelah Javac selesai mengkompilasi kode byte, itu akan membuat file baru dengan ekstensi .class. Ketika file kelas telah dibuat, itu dapat digunakan pada JVM.

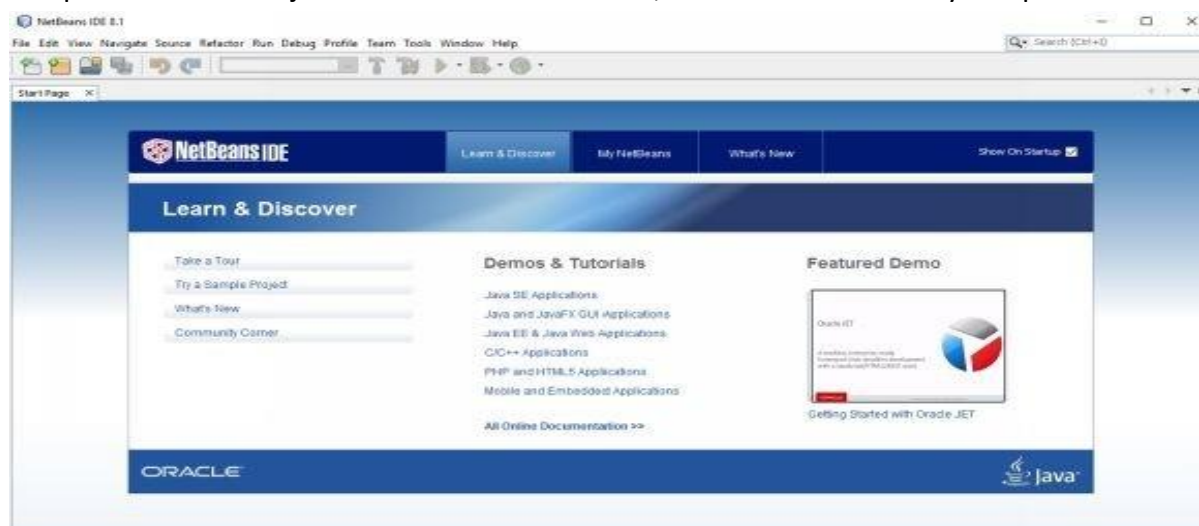
Sebagai ringkasan:

- Kode sumber dibuat menggunakan ekstensi .java.
- Kompilasi diproses melalui Javac dan akan membuat file dengan ekstensi .class.
- Memulai kelas yang dikompilasi.

Menggunakan NetBeans, Anda dapat dengan mudah mengkompilasi dan membuat file. Di latar belakang, itu akan mengambil kode sumber dan membuat file java. Ini akan menjalankan Javac dan mengkompilasi file kelas. NetBeans kemudian akan menjalankan program di dalam perangkat lunak, yang akan menyelamatkan Anda dari kesulitan memulai jendela terminal serta mengkodekan serangkaian perintah yang panjang. Setelah memahami cara kerja Java, kini Anda dapat menjalankan perangkat lunak NetBeans.

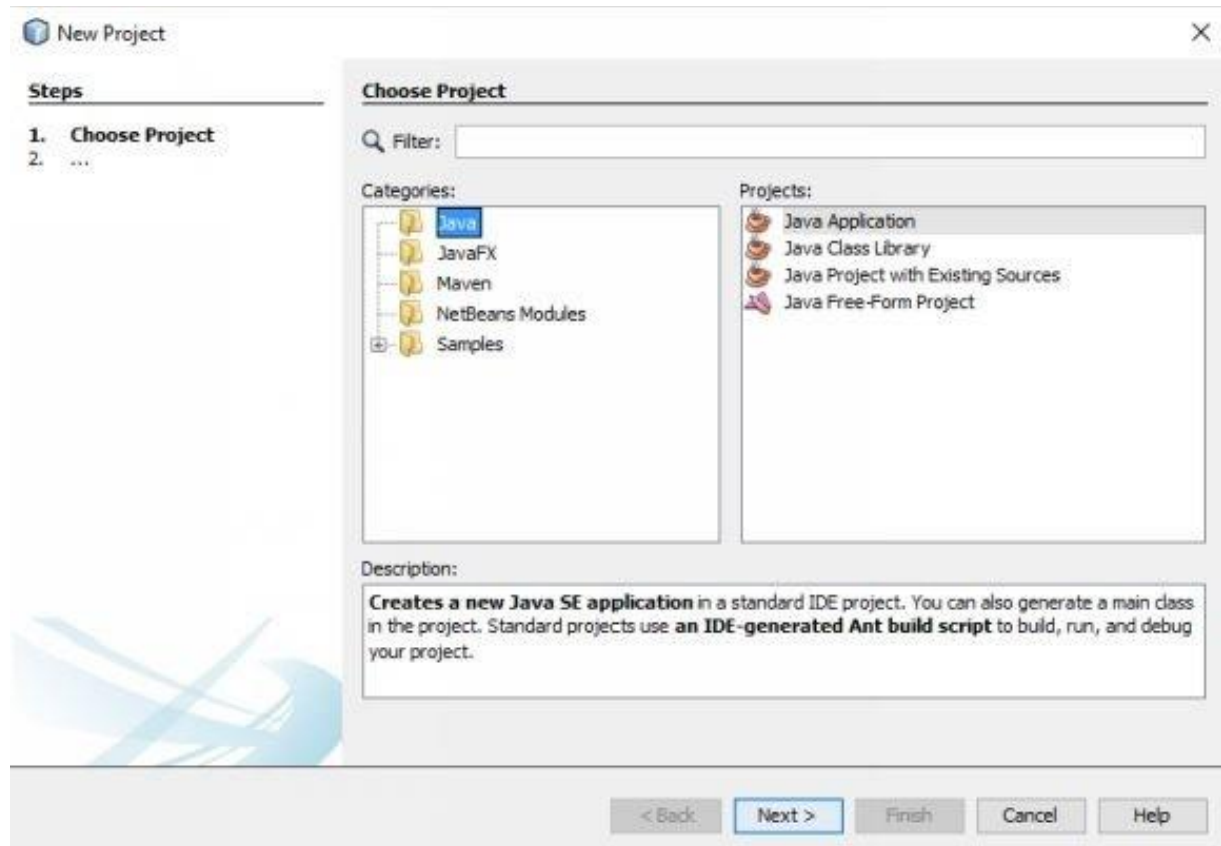
2.4 PERANGKAT LUNAK NETBEANS

Saat pertama kali menjalankan Software NetBeans, Anda akan melihat layar seperti ini:



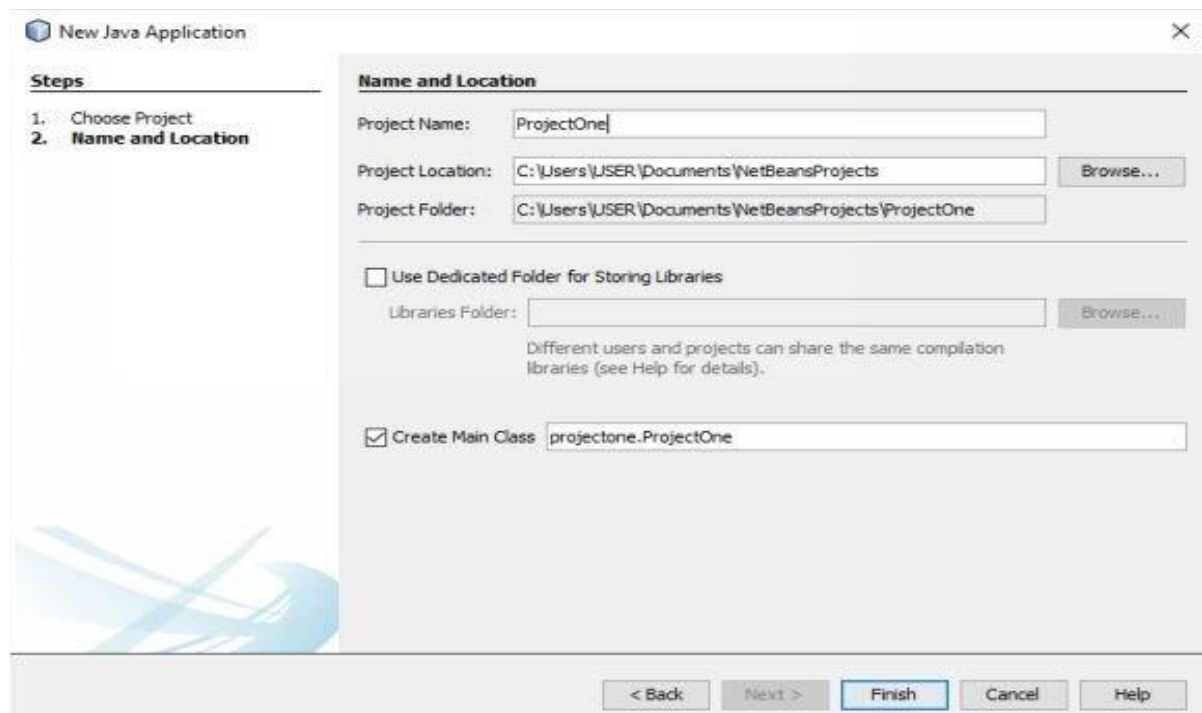
Gambar 2.1 Tampilan pertama ketika perangkat lunak netbeans dijalankan

Silakan dan nikmati secangkir kopi karena ini tidak dikenal karena kecepatannya. Di bagian atas menu NetBeans, Anda dapat mengklik File> Proyek Baru untuk memulai proyek baru. Akan muncul kotak dialog di bawah ini:



Gambar 2.2 Tampilan untuk memulai proyek baru

Anda perlu membuat Aplikasi Java, jadi pilih Java di bawah menu Kategori, lalu Aplikasi Java di bawah kategori Proyek. Untuk melanjutkan ke langkah kedua, klik tombol Berikutnya.



Gambar 2.3 Tampilan untuk memberi nama proyek dan lokasi menyimpannya

Area pertama menunjukkan Nama Proyek. Klik pada ruang kosong dan pilih judul untuk Proyek Anda. Anda akan melihat bahwa teks di bagian bawah juga berubah agar sesuai dengan nama proyek.

Kelas kami akan dikenal sebagai ProjectOne dengan huruf besar P dan O. Paket ini juga dikenal sebagai projectone, tetapi dengan huruf kecil p dan o. Perhatikan bahwa lokasi default untuk menyimpan proyek Anda juga muncul di kotak teks untuk Lokasi Proyek. Anda memiliki kebebasan untuk mengubah ini, jika Anda mau. NetBeans juga dapat membuat folder dengan nama proyek pilihan Anda dan akan disimpan di lokasi yang sama. Cukup klik tombol Finish dan NetBeans akan berjalan untuk membuat semua file yang dibutuhkan.

Setelah NetBeans mengarahkan Anda ke IDE, amati area Proyek yang terletak di kiri atas layar. Jika Anda tidak dapat melihat ini, cukup klik Windows > Projects dari bilah menu di bagian atas perangkat lunak.



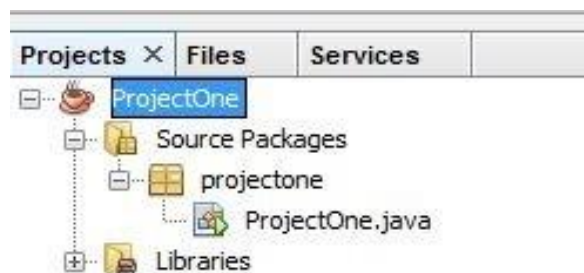
Gambar 2.4 Tampilan file proyek yang berhasil dibuat

Ketika Anda mengklik simbol plus, Anda dapat memperluas proyek, dan Anda akan melihat layar di bawah ini:



Gambar 2.5 File proyek ketika di klik simbol plus

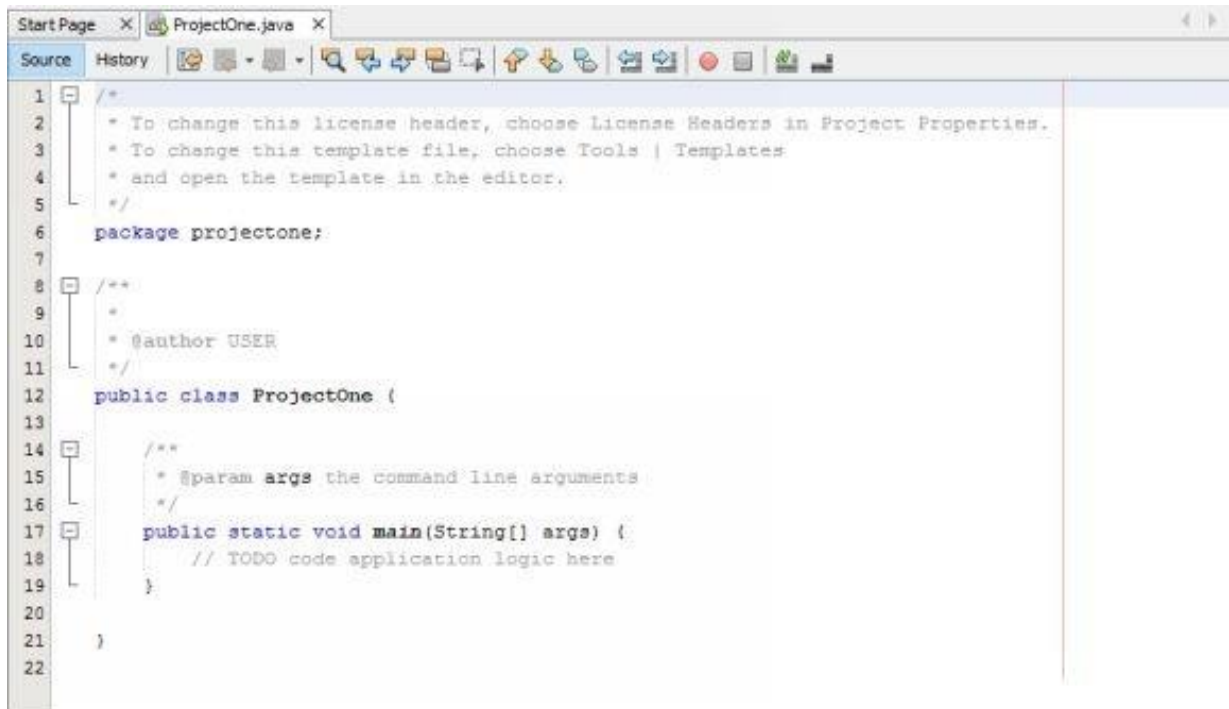
Selanjutnya, Anda dapat memperluas Paket Sumber untuk melihat nama proyek. Anda dapat memperluas ini dan Anda akan melihat file Java, yang sebenarnya adalah kode sumber Anda.



Gambar 2.6 Memperluas isi dari Source package

Kode sumber yang sama akan ditampilkan di sebelah kanan, di ruang teks yang lebih besar. Ini akan disebut sebagai ProjectOne.java. Jika tidak ada jendela kode, Anda hanya perlu mengklik dua kali ProjectOne.java di layar Proyek Anda seperti yang ditunjukkan di atas. Kode akan ditampilkan sehingga Anda dapat mulai mengerjakannya.

Layar pengkodean ditunjukkan di bawah ini:



Gambar 2.7 ProjectOne.java

Perhatikan bahwa kelas di sini dikenal sebagai ProjectOne:

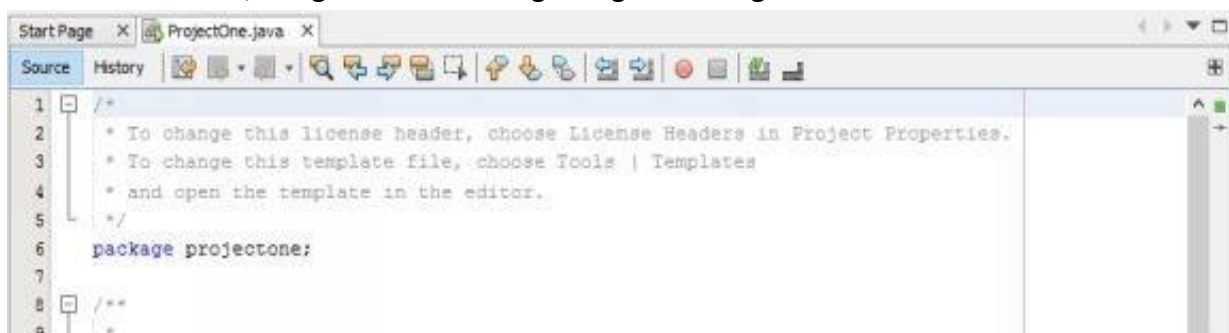
public class ProjectOne {

Ini mirip dengan nama file sumber java di jendela proyek: ProjectOne.java. Setelah Anda menjalankan program, kompiler akan membutuhkan file sumber serta nama kelas. Oleh karena itu, jika file .java dikenal sebagai projectOne tetapi kelasnya dikenal sebagai ProjectOne, maka Anda akan mengalami kesalahan saat kompilasi. Ini semua karena huruf kecil "p" dan yang kedua dikapitalisasi.

Perhatikan bahwa meskipun kami juga memanggil paket ProjectOne, ini tidak perlu. Anda dapat menggunakan nama paket yang berbeda, karena nama paket tidak harus sama dengan file sumber java atau kelas file sumber. Hanya nama kelas dan nama file sumber Java yang harus sama.

2.5 KOMENTAR JAVA

Setelah Anda membuat Proyek Baru di NetBeans, Anda akan melihat bahwa ada teks berwarna abu-abu, dengan tanda bintang dan garis miring.



Gambar 2.8 Teks komentar berwarna abu-abu

Teks berwarna abu-abu adalah komentar. Setelah Anda menjalankan program, ini akan diabaikan. Oleh karena itu, Anda memiliki kebebasan untuk mengetik apa pun yang Anda suka sebagai komentar. Biasanya komentar untuk menggambarkan apa yang Anda coba lakukan. Anda dapat memasukkan komentar satu baris dengan mengetikkan dua garis miring kemudian komentar Anda.

//Ini adalah contoh komentar satu baris di Java.

Jika Anda ingin memasukkan komentar dalam beberapa baris, Anda dapat melakukan ini:

```
/*
ini adalah sebuah contoh dari sebuah komentar
itu membutuhkan dua baris atau lebih.
```

```
*/ Atau ini:
//Ini adalah contoh lain dari sebuah komentar
// yang membutuhkan dua baris atau lebih.
```

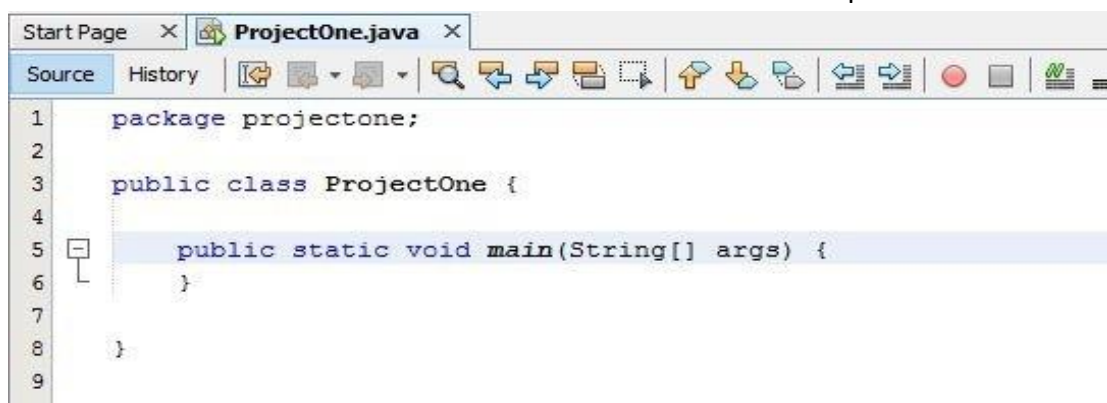
Pada opsi pertama, perhatikan bagaimana komentar dimulai dengan `/*` dan diakhiri dengan `*/`.

Lalu ada komentar Javadoc, yang dimulai dengan satu garis miring ke depan diikuti oleh dua tanda bintang (`/**`). Itu juga diakhiri dengan tanda bintang dengan satu garis miring (`*/`). Perhatikan bahwa setiap baris komentar dimulai dengan tanda bintang:

```
/**
*Ini adalah contoh komentar tipe Javadoc
*/
```

Komentar Javadoc digunakan untuk merekam kode. Kode yang direkam dapat diubah menjadi halaman HTML, yang akan membantu pemrogram lain untuk memahami kode tersebut. Klik Jalankan dari menu NetBeans untuk melihat seperti apa tampilannya. Dari menu Run, pilih Generate Javadoc. Jangan berharap untuk melihat banyak karena kami belum mulai menulis kode apa pun.

Pada titik ini, tidak apa-apa untuk menghapus komentar terlebih dahulu, yang telah dibuat oleh NetBeans untuk Anda. Di bawah ini adalah area kode tanpa komentar:



Gambar 2.9 Area kode tanpa komentar

Pada tangkapan layar di atas, Anda dapat melihat nama paket terlebih dahulu. Perhatikan bahwa garis berakhir menggunakan titik koma. Ingat, tanpa titik koma, program tidak akan memulai proses kompilasi.

```
package projectone;
```

Berikutnya adalah nama kelas:

```
Public class ProjectOne {  
}
```

Kelas dianggap sebagai segmen kode. Namun, Anda perlu menentukan di mana segmen dimulai dan diakhiri dengan menambahkan tanda kurung kurawal. Awal segmen kode ditunjukkan dengan tanda kurung kurawal kiri { dan diakhiri dengan tanda kurung kurawal kanan }. Setiap kode yang dibatasi dalam tanda kurung ini termasuk dalam segmen kode tersebut.

Sementara itu, apa pun yang ada di dalam tanda kurung kurawal kanan dan kiri untuk kelas juga dianggap sebagai segmen kode. Lihatlah ini:

Apa yang ada di dalam tanda kurung kurawal kiri dan kanan untuk kelas adalah segmen kode lainnya. Yang ini:

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

Perhatikan bahwa teks "utama" adalah kata yang paling penting di sini. Setelah Anda memulai program, ia akan mencari metode bernama main. Metode hanyalah segmen kode, dan Anda akan mempelajari lebih lanjut tentang segmen lainnya nanti. Ini kemudian akan meluncurkan kode apa pun di dalam tanda kurung melengkung untuk yang utama. Akan ada kesalahan tanpa metode utama dalam program. Tapi seperti namanya, ini mengacu pada titik masuk utama untuk program. Untuk saat ini, jangan khawatir tentang teks biru sebelum teks "utama". Tetapi jika Anda ingin melihat sekilas, public mengacu pada metode, yang dapat dilihat di luar kelas, sementara static menandakan bahwa tidak perlu membangun objek baru, dan void berarti tidak akan menghasilkan nilai. Bagian antara tanda kurung bulat utama dikenal sebagai argumen baris perintah. Bingung? Nah, Anda dapat mempelajari lebih lanjut tentang mereka nanti, jadi jangan khawatir.

Pointer penting yang perlu diperhatikan adalah Anda sekarang memiliki kelas yang dikenal sebagai ProjectOne. Kelas ini melibatkan metode yang dikenal sebagai main. Segmen ini memiliki tanda kurung melengkung mereka sendiri. Namun, bagian utama dari kode mengacu pada kelas ProjectOne.

Sekarang, mari kita pelajari bagaimana kita dapat mencetak ke layar atau jendela keluaran. Pada titik ini, Anda dapat menjalankan kode dan mengubahnya menjadi program nyata. Itu tidak akan melakukan apa-apa, tetapi Anda dapat memulai proses kompilasi. Sekarang, tambahkan baris kode lain untuk melihat cara kerjanya. Anda dapat menambahkan beberapa kata ke jendela konsol. Anda dapat memasukkan baris berikut ke metode utama:

```
public static void main( String[ ] args ) {  
System.out.println("Proyek Saya Satu");
```



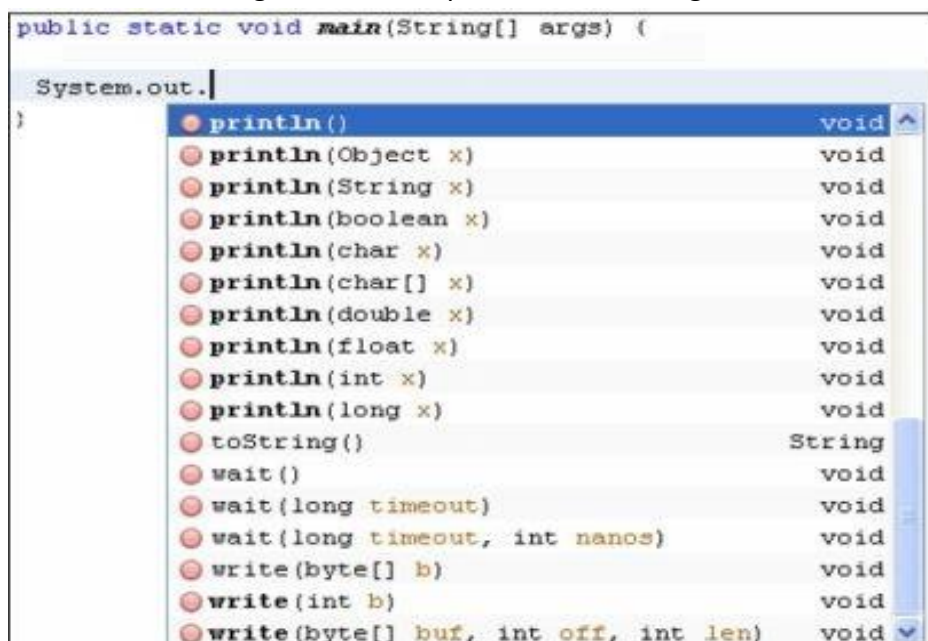
```
}
```

Setelah Anda mengakhiri kata Sistem dengan titik, program akan membantu Anda dengan menampilkan daftar opsi:



Gambar 2.10 Daftar opsi untuk membantu pengerjaan proyek

Klik opsi keluar dua kali sehingga Anda dapat menambahkannya ke kode Anda. Kemudian, tambahkan titik lagi, dan daftar opsi akan muncul lagi:



Gambar 2.11 klik opsi keluar dua kali untuk memilih kode dari daftar opsi

Pilih opsi **println()**, yang memungkinkan Anda mencetak satu baris teks ke jendela keluaran. Namun, Anda harus menempatkan teks di antara tanda kurung bulat untuk **println**. Teks harus diapit oleh tanda kutip ganda.

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

Gambar 2.12 Memilih opsi println()

Saat Anda memasukkan tanda kutip ganda, Anda dapat menambahkan teks pilihan Anda:

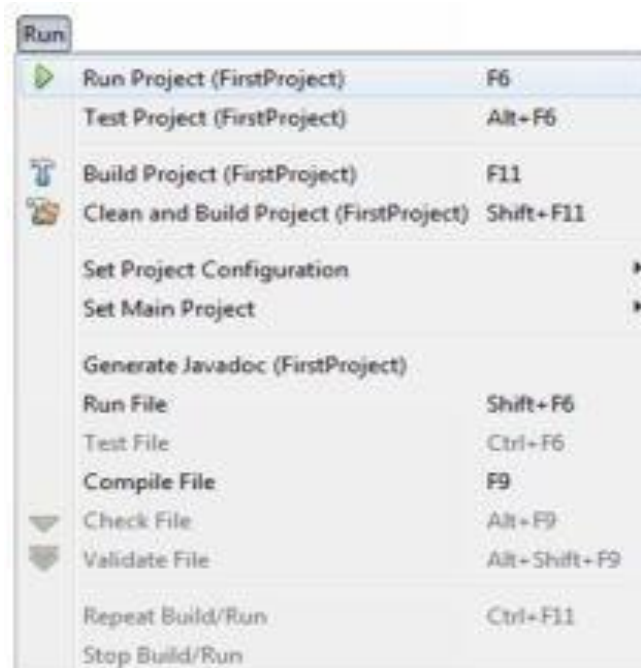
```
public static void main(String[] args) {
    System.out.println( "My Project One" );
}
```

Gambar 2.13 Menambah kan kalimat di dalam tanda kutip ganda

Perhatikan bahwa garis berakhir menggunakan titik koma. Setiap baris kode lengkap di Java harus diakhiri dengan titik koma. Tanpa titik koma, program tidak akan memulai proses kompilasi. Sekarang, kita bisa menguji program pertama kita. Tetapi sebelum Anda dapat melakukannya, pastikan untuk menyimpan kode Anda. Anda dapat melakukannya dengan mengklik ikon Simpan di toolbar NetBeans atau mengikuti baris perintah ini: **File > Save or File > Save All**.

2.6 MENJALANKAN PROGRAM JAVA

Setelah Anda menjalankan program di NetBeans, perangkat lunak akan menjalankannya di layar Output di bawah layar di bagian bawah kode. Ini lebih mudah karena tidak perlu memulai jendela konsol atau terminal. Layar Output ADALAH konsol Anda. Ada beberapa metode menjalankan program di NetBeans. Cara paling sederhana adalah dengan menekan F6 di Keyboard. Anda juga dapat menjalankan program menggunakan menu pada toolbar NetBeans: **Run > Run Project (Nama proyek Anda)**.



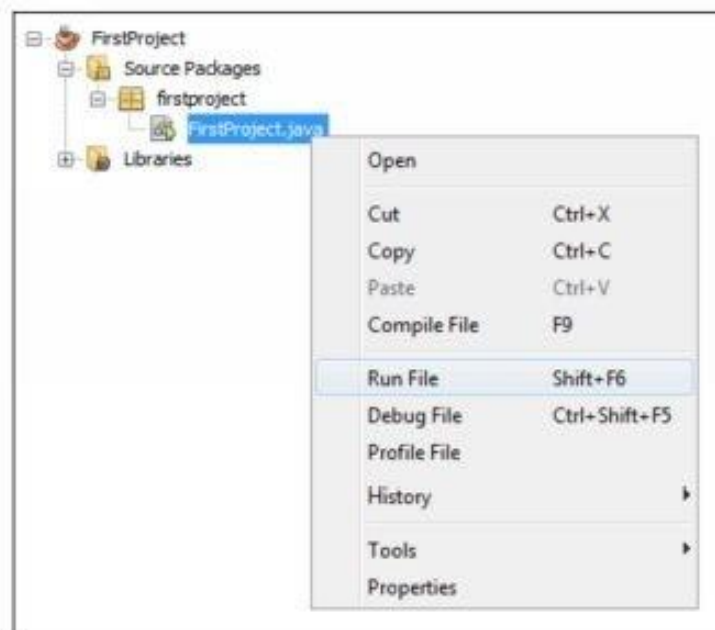
Gambar 2.14 Cara menjalankan proyek

Pilihan lainnya adalah dengan menekan ikon panah hijau pada toolbar.



Gambar 2.15 tombol run berupa ikon panah hijau

Metode lain dalam menjalankan program adalah melalui layar Proyek. Ini akan memastikan bahwa Anda menjalankan kode sumber yang benar. Cukup klik kanan file sumber Java di layar proyek dan menu akan muncul. Pilih Jalankan Berkas.



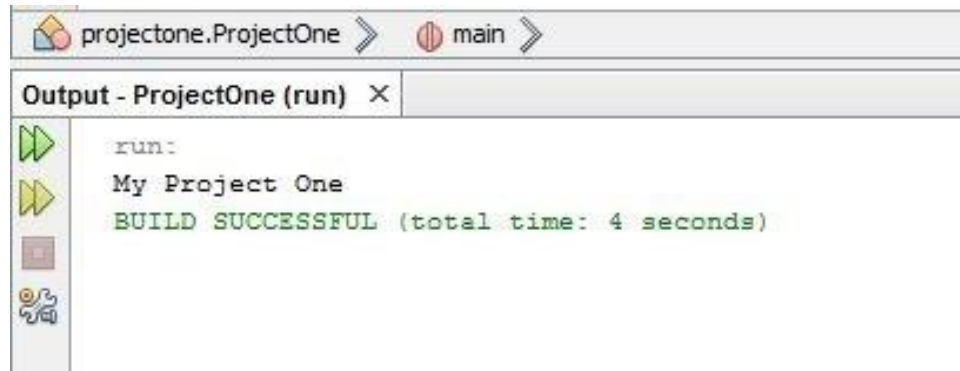
Gambar 2.16 Menjalankan proyek melalui layar proyek

Anda juga dapat menjalankan program dengan mengklik kanan di dalam layar kode. Seperti yang ditunjukkan pada tangkapan layar di bawah, kami telah mengklik kanan sebelum braket melengkung terakhir.



Gambar 2.17 Menjalankan proyek di layar kode

Pilih metode pilihan Anda dan jalankan program. Di layar Output, Anda akan melihat sesuatu seperti ini:

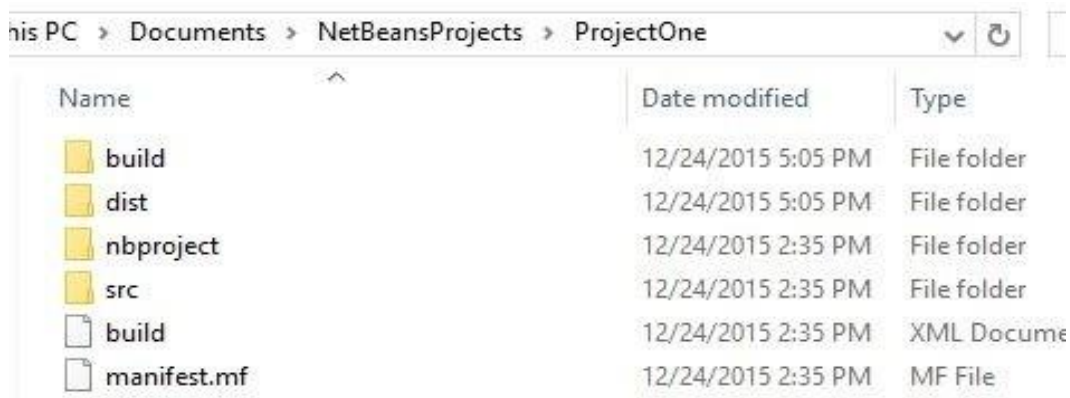


Gambar 2.18 Output dari kode PrintIn ketika berhasil dijalankan

Baris kedua di layar Output pada gambar di atas adalah kode kita: My Project One. Sangat mudah untuk memulai menjalankan ulang dengan mengklik dua panah hijau di bilah alat sisi kiri.

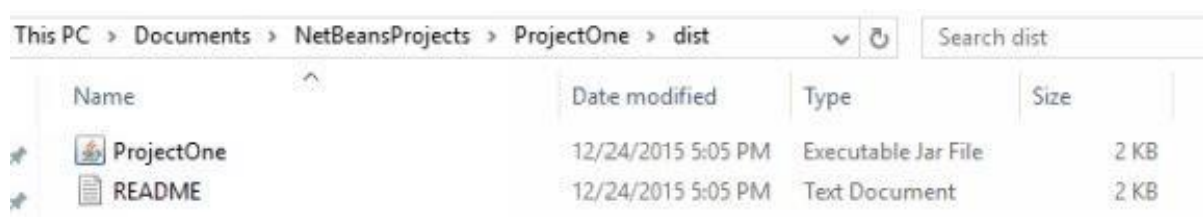
2.7 BERBAGI PROGRAM JAVA

Java memungkinkan Anda untuk membagikan program Anda kepada orang lain, sehingga mereka juga dapat mencoba menjalankannya. Untuk melakukan ini, Anda harus terlebih dahulu membuat file Java Archive atau file JAR. NetBeans akan membantu Anda melakukan ini. Pada menu Run, pilih Clean and Build Main project. NetBeans kemudian akan menyimpan kode Anda dan akan membuat semua file yang dibutuhkan. Itu juga akan membuat folder baru bernama dist di mana ia akan menyimpan semua file. Lihatlah lokasi di mana proyek berada dan Anda juga akan melihat folder bernama dist.



Gambar 2.19 Folder “dist” tempat menyimpan semua file proyek

Buka folder dist dan lihat ke dalam:



Gambar 2.20 Isi folder dist

Harus ada file JAR dan file txt README, yang menyimpan instruksi tentang bagaimana Anda dapat menjalankan program dari layar konsol atau terminal. Pada titik ini, Anda sudah tahu bagaimana Anda dapat menjalankan file sumber java. Bab berikutnya akan membantu Anda mempelajari lebih lanjut tentang pemrograman yang sebenarnya.

BAB 3

VARIABEL JAVA

Dalam pemrograman, selalu ingat bahwa bahasa bekerja melalui penggunaan data yang disimpan dalam memori. Data ini bisa berupa objek, teks, atau angka penunjuk ke area data lainnya. Data ini disediakan dalam sebuah nama. Oleh karena itu, itu akan diambil setelah Anda membutuhkannya. Variabel mengacu pada nama serta nilainya. Kita akan mulai belajar tentang nilai bilangan. Di Jawa, ada opsi berbeda dalam menyimpan angka. Bilangan bulat seperti 2, 5, 10 dan seterusnya dapat ditahan menggunakan variabel `int`, yang berarti bilangan bulat. Angka titik seperti 2.4, 5.8, 10.2, dan seterusnya disimpan melalui penggunaan variabel `double`. Kita dapat menyimpannya menggunakan simbol sama dengan (=). Di bagian ini, kami akan mengerjakan contoh, menggunakan kode ProjectOne Anda.

Untuk membuat bilangan bulat, cukup masukkan metode kode baris utama ProjectOne.

```
public static void main(String[ ] args) {
    int first_number;
    System.out.println("My Project One");
}
```

Untuk memerintahkan program Java untuk menyimpan bilangan bulat, Anda harus terlebih dahulu mulai mengetik `int` dan spasi. Penting untuk memikirkan judul untuk variabel bilangan bulat. Secara umum, Anda memiliki kebebasan untuk memilih nama, selama Anda mengikuti aturan di bawah ini:

1. Anda tidak dapat memulai nama variabel menggunakan angka. Oleh karena itu, **first_number** baik-baik saja, tetapi `1st_number` tidak. Dapat diterima untuk memasukkan angka di mana saja pada variabel nama kecuali di awal.
2. Anda tidak dapat menggunakan kata kunci Java sebagai variabel nama. Kata kunci ini akan langsung berubah menjadi biru saat Anda mengetiknya seperti `int`.
3. Spasi tidak diperbolehkan dalam penamaan variabel. Variabel ekspresi **int first number** akan menghasilkan kesalahan. Sebagai gantinya, Anda dapat menggunakan garis bawah sebagai pengganti spasi. Ini adalah praktik industri bahwa kata utama dimulai dengan teks kecil dan kata berikutnya dikapitalisasi seperti `myFirstnumber` atau `firstNumber`.
4. Sensitivitas huruf besar-kecil sangat penting untuk variabel nama. Oleh karena itu, `FirstNumber` dan `firstNumber` tidak sama.

Untuk menyimpan apa pun dalam variabel yang dikenal sebagai **first_number**, cukup sertakan simbol sama dengan diikuti oleh nilai yang ingin Anda pertahankan.

```
public static void main(String[ ] args) {
    int first_number;
    first_number = 5;
    System.out.println("My Project One");
}
```

Java akan menginterpretasikan bahwa Anda ingin menyimpan nilai 5 dalam variabel `int`, yang mengacu pada `first_number`.

Anda juga dapat menulis ini semua dalam satu kode baris:

```
public static void main(String[] args) {
    int first_number = 5;
    System.out.println("My Project One");
}
```

Untuk melihat cara kerja kode ini, buat sedikit perubahan pada metode `println`:

```
System.out.println( "First number = " + first_number );
```

Di dalam kurung bulat `println`, Anda sekarang memiliki teks langsung yang dibatasi dalam tanda kutip ganda:

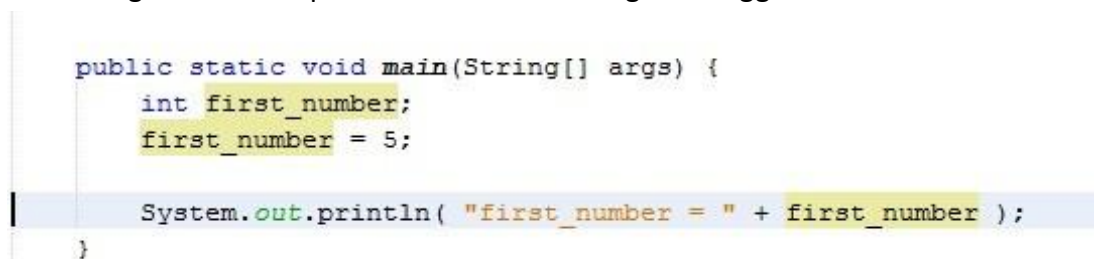
```
("First number = "
```

Diikuti dengan simbol plus dan nama variabel:

```
+ first_number );
```

Menambahkan simbol plus akan ditafsirkan oleh Java bahwa Anda ingin menggabungkan nama variabel dengan teks langsung. Proses ini disebut penggabungan.

Layar pengkodean sekarang akan muncul seperti tangkapan layar di bawah ini. Perhatikan bagaimana setiap baris kode diakhiri dengan menggunakan titik koma.

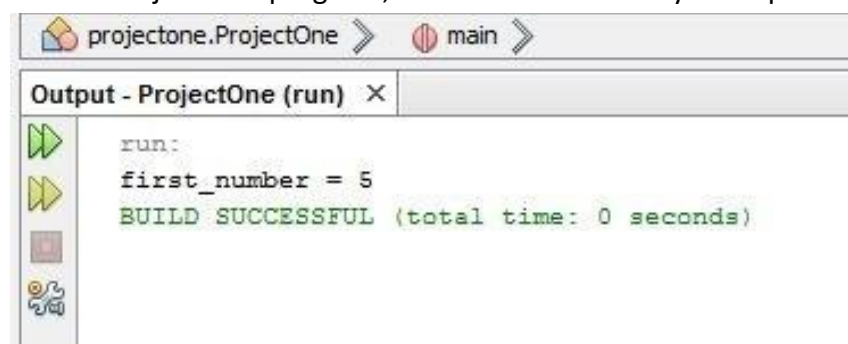


```
public static void main(String[] args) {
    int first_number;
    first_number = 5;

    System.out.println( "first_number = " + first_number );
}
```

Gambar 3.1 Menggabungkan nama variabel dengan teks number

Saat Anda mencoba menjalankan program, Anda akan melihat layar Output di bawah:



```
projectone.ProjectOne > main >
Output - ProjectOne (run) X
run:
first_number = 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar 3.2 Output dari penggabungan

Oleh karena itu, angka yang Anda simpan dalam variabel yang kita sebut sebagai `first_number` mengacu pada teks di sebelah kanan simbol sama dengan.

Sekarang, coba tambahkan dasar. Ketik dua variabel int tambahan ke dalam kode Anda. Yang lain untuk menentukan angka ke-2 sementara yang lain untuk mendefinisikan "jawaban":

```
int first_number, second_number, answer;
```

Perhatikan bahwa ada tiga nama variabel dalam satu baris. Dimungkinkan untuk melakukan ini menggunakan Java selama variabel bisa bertipe serupa. Dalam hal ini, mereka semua bilangan bulat. Setiap nama variabel kemudian dibagi menggunakan koma. Kemudian, Anda dapat menyimpan teks lain di variabel yang ditambahkan:

```
first_number = 5;
```

```
second_number = 10;
```

```
answer = first_number + second_number;
```

Untuk menentukan jawaban variabel, Anda perlu mendapatkan jumlah angka 1 dan angka 2. Anda dapat melakukan penjumlahan dengan menyertakan simbol penjumlahan (+). Ini akan memerintahkan program Java untuk menambahkan nilai pada `first_number` dan `second_number`. Setelah selesai, itu juga akan menyimpan variabel total yang terletak di bagian kiri simbol sama dengan. Oleh karena itu, alih-alih mendefinisikan 5 atau 10 untuk nama variabel, itu akan dijumlahkan dan akan melakukan penetapan. Tetapi Java telah menginterpretasikan apa nilai dari variabel ganda, jadi boleh saja menggunakan nama mereka.

Sekarang, ubah metode `println` ke baris di bawah ini:

```
System.out.println("Addition Total = " + answer );
```

Ingat, kami menambahkan teks langsung dalam tanda kutip ganda dengan nama variabel. Layar pengkodean akan muncul seperti tangkapan layar di bawah ini:

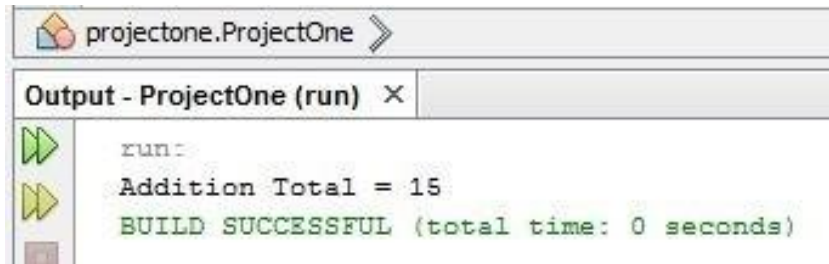
```
public class ProjectOne {
    |
    public static void main(String[] args) {
        int first_number, second_number, answer;

        first_number = 5;
        second_number = 10;
        answer = first_number + second_number;

        System.out.println( "Addition Total = " + answer );
    }
}
```

Gambar 3.3 Membuat kode penjumlahan

Jalankan programnya, dan hasilnya akan terlihat seperti pada gambar di bawah ini:



Gambar 3.4 Output kode penjumlahan

Sejauh ini, kami telah melakukan hal-hal ini di program pertama kami:

- Menyimpan bilangan bulat (angka pertama)
- Disimpan bilangan bulat lain (angka kedua)
- Gabungkan bilangan bulat ini
- Memegang jumlah bilangan bulat dalam variabel ketiga
- Cetak outputnya

Atau, juga dimungkinkan untuk menggunakan nomor langsung. Putar saja jawaban baris ini:

answer = first_number + second_number + 10;

Klik program run, dan lihat hasilnya.

Dimungkinkan untuk menyimpan bilangan bulat besar menggunakan tipe `int`, tetapi nilai maksimalnya adalah 2147483647. Untuk menyimpan bilangan bulat negatif, nilai minimum yang dapat kita simpan adalah -2147483648. Untuk menyimpan angka yang lebih tinggi atau lebih rendah, disarankan untuk menggunakan tipe variabel ganda.

3.1 VARIABEL GANDA

Variabel ganda dapat menyimpan angka yang sangat kecil atau sangat besar. Nilai maks adalah 17 diikuti oleh 307 nol, dan nilai minimum adalah -17 diikuti oleh 307 nol. Nilai floating point seperti 7,8, 11,6, atau 14,5 juga dapat disimpan dalam variabel ganda. Saat Anda mencoba menyimpan nilai floating point menggunakan variabel `int`, NetBeans akan menafsirkannya sebagai kesalahan. Mari kita coba berlatih dengan variabel ganda, menggunakan kode ProjectOne Anda. Pertama, ubah variabel `int` menjadi variabel ganda. Oleh karena itu, kode ini:

int first_number, second_number, answer;

harus diubah menjadi ini:

double first_number, second_number, answer;

Kemudian ubah nilai yang sesuai dari `first_number` dan `second_number`:

first_number = 5.2;

second_number = 10.4;

Area pengkodean akan terlihat seperti tangkapan layar di bawah ini:

```

public static void main(String[] args) {
    double first_number, second_number, answer;

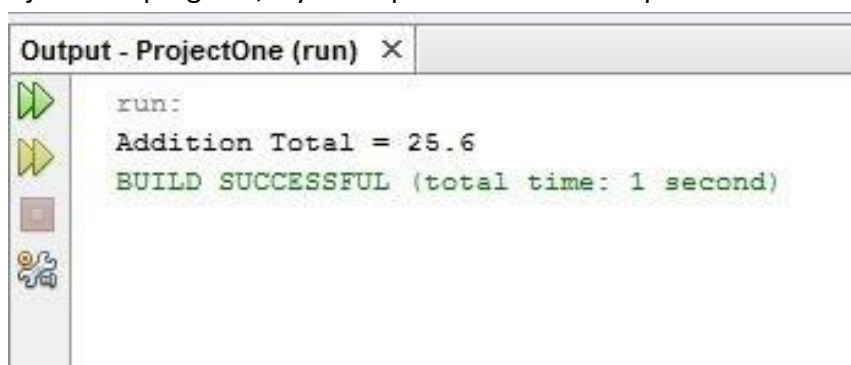
    first_number = 5.2;
    second_number = 10.4;
    answer = first_number + second_number + 10;

    System.out.println( "Addition Total = " + answer );
}

```

Gambar 3.5 Penggunaan kode variabel ganda

Saat Anda menjalankan program, layar Output akan terlihat seperti ini:



Gambar 3.6 Output penggunaan kode variabel ganda

Ubah nilai yang disimpan di `first_number` dan `second_number`. Pilih nilai yang Anda suka. Jalankan programnya dan lihat hasilnya.

3.2 VARIABEL SHORT DAN FLOAT

Short dan **float** adalah jenis variabel lain yang dapat Anda gunakan. Variabel short dapat digunakan untuk menyimpan angka yang lebih kecil, mulai dari -32768 hingga 32767. Namun daripada menggunakan int dalam kode seperti pada kode sebelumnya, Anda dapat menggunakan short. Perhatikan bahwa Anda hanya boleh menggunakan variabel short jika Anda yakin bahwa nilai yang ingin Anda simpan tidak akan melebihi 32767 atau lebih rendah dari -32768. Variabel ganda dapat menampung angka yang sangat besar untuk floating range point. Tapi daripada menggunakan double, Anda bisa menggunakan float. Dalam memegang nilai dalam float variabel, Anda harus menambahkan f di akhir angka, seperti:

```

float first_number, second_number, answer;
first_number = 5.2f;
second_number = 10.4f;

```

Perhatikan bahwa f harus ditambahkan setelah nilai aktual tetapi tidak setelah titik koma.

3.3 ARITMATIKA DASAR

Dengan menggunakan variabel yang telah kita pelajari sejauh ini, kita dapat menggunakan simbol-simbol ini untuk melakukan perhitungan:

- Simbol plus (+) adalah untuk penambahan.

- Simbol minus (-) untuk pengurangan.
- Tanda bintang (*) untuk perkalian.
- Tanda hubung ke depan (/) adalah untuk pembagian.

Mari kita lakukan beberapa latihan.

Singkirkan tanda plus, yang telah kita gunakan untuk menggabungkan `first_number` dan `second_number`. Sebagai gantinya, masukkan simbol minus, tanda bintang, dan tanda hubung ke depan. Hasil pembagian harus berupa bilangan yang besar (15.600000000000001), karena tipe variabel yang digunakan adalah `double`. Tetapi Anda perlu mengubah variabel ganda menjadi mengambang dan memasukkan huruf `f` di sebelah nilai. Oleh karena itu, kode harus muncul mirip dengan tangkapan layar ini:

```
public static void main(String[] args) {
    float first_number, second_number, answer;

    first_number = 5.2f;
    second_number = 10.4f;
    answer = first_number / second_number;

    System.out.println( "Addition Total = " + answer );
}
```

Gambar 3.7 Menghapus tanda plus dan menambahkan huruf `f` di sebelah nilai

Jika Anda menjalankan kode ini, Anda akan mendapatkan 0,5. Program ini telah mengumpulkan ini. Oleh karena itu, variabel tipe ganda dapat menyimpan lebih banyak angka dibandingkan dengan `float`. (`Float` memiliki kapasitas 32 bit sedangkan `double` memiliki 64 bit.

3.4 KEUTAMAAN OPERATOR

Tentu saja, dimungkinkan untuk menghitung lebih banyak angka. Namun Anda harus mendefinisikan secara spesifik apa yang harus dihitung. Mari tambahkan nomor lain dalam kode kita.

```
first_number = 50;
second_number=25;
third_number=15;
answer = first_number - second_number + third_number;
```

Dalam melakukan perhitungan mulai dari kiri ke kanan, hasilnya adalah 50-25, dan jawaban akhirnya adalah 25. Kemudian sertakan angka ketiga (15). Totalnya akan menjadi 40. Tapi bagaimana jika ini bukan niat Anda? Katakanlah Anda ingin mendapatkan jumlah `second_number` dan `third_number`, lalu kurangi hasilnya dari `first_number`. Jadi, itu akan menjadi 25 + 15 = 40. Kemudian, kurangi ini dari angka_pertama, yaitu 50. Jawabannya adalah 10.

Untuk memastikan bahwa program melakukan apa yang Anda inginkan, Anda harus menggunakan tanda kurung bulat. Oleh karena itu, perhitungan pertama akan terlihat seperti ini:

```
answer = (first_number - second_number) + third_number;
```

Area pengkodean Anda akan terlihat seperti ini:

```
public static void main(String[] args) {
    int first_number, second_number, third_number, answer;

    first_number = 50;
    second_number = 25;
    third_number = 15;
    answer = (first_number - second_number) + third_number;

    System.out.println( " Total = " + answer );
}
```

Gambar 3.8 kode untuk pengurangan dengan tanda kurung akan dihitung terlebih dahulu

Ini adalah perhitungan kedua:

answer = first_number - (second_number + third_number);

Sementara itu, area kodenya adalah ini:

```
public static void main(String[] args) {
    int first_number, second_number, third_number, answer;

    first_number = 50;
    second_number = 25;
    third_number = 15;
    answer = first_number - (second_number + third_number);

    System.out.println( " Total = " + answer );
}
```

Gambar 3.9 kode untuk penjumlahan dengan tanda kurung akan dihitung terlebih dahulu

Selanjutnya, kita akan melakukan beberapa latihan untuk penjumlahan dan perkalian.

Ubah operator menjadi simbol tambahan dan tanda asterisk:

answer = first_number + second_number * third_number;

Singkirkan semua kurung bulat sebelum menjalankan program. Tanpa tanda kurung, biasanya orang mengira Java akan melakukan perhitungan mulai dari kiri ke kanan. Oleh karena itu, Anda akan berpikir bahwa itu akan menambahkan angka_pertama ke angka_kedua untuk mendapatkan 75. Kemudian jawaban akan dikalikan dengan angka_ketiga, yaitu 15. Oleh karena itu, jawabannya adalah 1125. Jalankan program, dan jawaban yang akan Anda dapatkan hanya 425. Jawabannya berbeda dari yang kita harapkan.

Prioritas Operator adalah alasan mengapa Java menghasilkan hasil yang berbeda. Java memprioritaskan beberapa operator daripada operator lain. Ini memprioritaskan perkalian daripada penambahan, maka itu melakukan perkalian terlebih dahulu. Kemudian akan melakukan penambahan. Oleh karena itu, Java melakukan kode di bawah ini:

answer = first_number + (second_number * third_number);

Di dalam kurung bulat yang ditambahkan, `second_number` dikalikan dengan `third_number`. Jumlah ini kemudian akan ditambahkan di atas `first_number`. Jadi, 25 dikalikan 15 adalah 375. Tambahkan 50, dan Anda akan mendapatkan 425.

Jika Anda lebih suka cara alternatif, pastikan untuk menginstruksikan Java dengan menambahkan tanda kurung bulat:

```
answer = (first_number + second_number) * third_number;
```

Selain perkalian, pembagian dianggap sebagai operator yang lebih penting untuk Java. Program akan melakukan pembagian terlebih dahulu sebelum melakukan pengurangan atau penambahan. Coba ubah baris jawaban:

```
answer = first_number + second_number / third_number;
```

Hasil yang akan kita dapatkan adalah 51. Kemudian, tambahkan tanda kurung bulat:

```
answer = (first_number + second_number) / third_number;
```

Kali ini jawabannya adalah 5. Jadi, jika Anda menghilangkan tanda kurung, Java akan melakukan pembagian terlebih dahulu dan menambahkan 50 ke totalnya. Java tidak akan melakukan perhitungan dari kiri ke kanan.

Berikut adalah Prioritas Operator:

- Perkalian dan Pembagian dianggap sama, tetapi dianggap lebih penting dibandingkan Pengurangan dan Penjumlahan.
- Penjumlahan dan Pengurangan dianggap sama, tetapi kurang diprioritaskan dibandingkan perkalian dan pembagian.

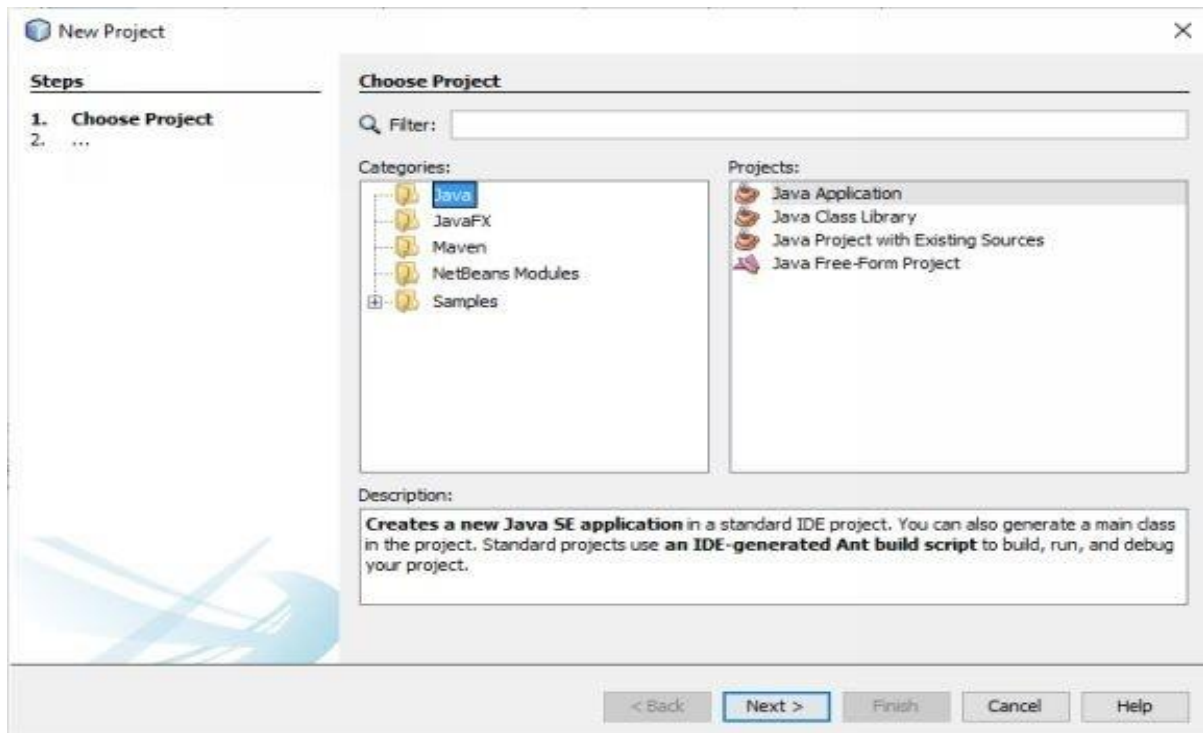
Oleh karena itu, jika Java memberikan hasil yang salah, perhatikan bahwa Precedence sangat penting, dan sertakan tanda kurung bulat.

3.5 VARIABEL STRING

Selain menyimpan nilai angka, variabel juga dapat menyimpan teks. Anda dapat menyimpan satu karakter atau banyak karakter. Anda dapat menggunakan variabel `char` jika Anda ingin menyimpan satu karakter. Tapi umumnya, Anda perlu memegang beberapa karakter.

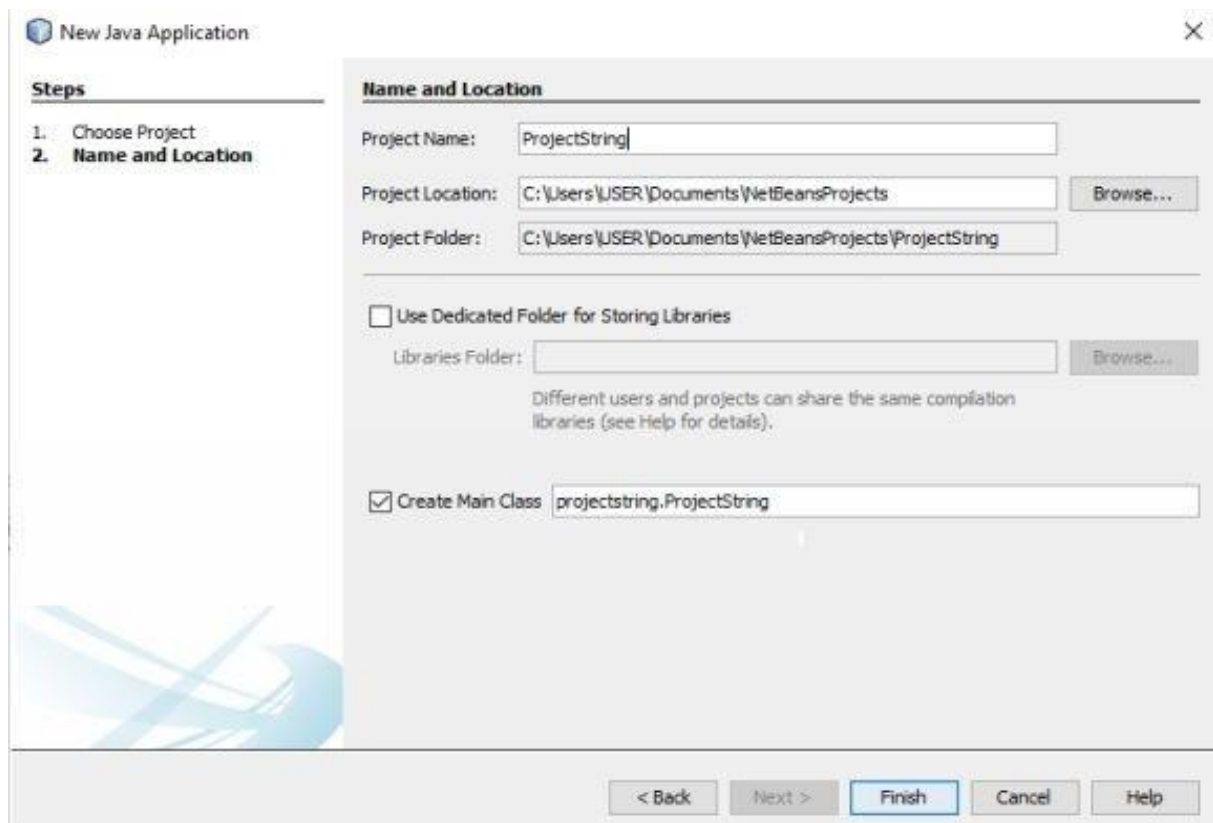
Untuk menyimpan hanya satu karakter, variabel `char` digunakan. Namun, biasanya Anda ingin menyimpan lebih dari satu karakter. Untuk melakukannya, Anda memerlukan tipe variabel `string`.

Anda perlu memulai proyek baru untuk ini di NetBeans. **Click File > New Project** di menu. Setelah kotak dialog Proyek Baru akan muncul, pastikan Anda memilih Aplikasi Java dan Java.



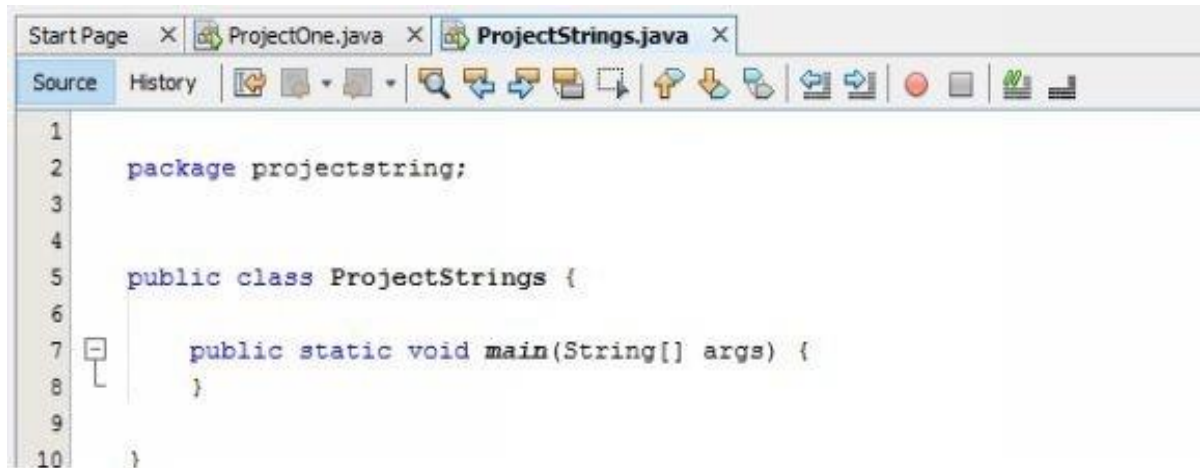
Gambar 3.10 Tampilan langkah pertama ketika akan membuat proyek baru

Klik Next dan masukkan ProjectString sebagai nama proyek. Pastikan bahwa kotak Buat Kelas Utama telah dipilih. Kemudian, hapus Main setelah projectstring, dan ketik ProjectStrings sebagai gantinya seperti yang Anda lihat di bawah:



Gambar 3.11 Tampilan langkah kedua membuat proyek baru

Oleh karena itu nama proyeknya adalah ProjectString, dan nama kelasnya adalah ProjectStrings. Tekan tombol Selesai dan area pengkodean akan terlihat seperti ini, setelah menghapus semua komentar. Perhatikan bahwa nama paket dalam huruf kecil (projectstring), tetapi nama proyeknya adalah ProjectString.



Gambar 3.12 Tampilan awal kode setelah membuat proyek baru

Untuk membuat variabel string, Anda perlu menggunakan istilah String dan kemudian nama variabel. Ingat, itu harus dimulai dengan huruf besar S. Tentu saja, itu harus diakhiri dengan titik koma.

String first_name;

Pilih nilai yang akan disimpan dalam variabel string baru dengan menambahkan tanda sama dengan. Mengikuti simbol sama dengan, teks yang ingin Anda pegang akan berada di antara dua set tanda kutip ganda.

first_name = "Harry";

Jika mau, Anda dapat mengetik semuanya dalam satu baris:

String first_name = "Harry";

Tambahkan variabel string lain untuk menyimpan nama keluarga:

String family_name = "Potter";

Untuk mencetak nama-nama ini, ikuti dengan metode `println()`

System.out.println(first_name + " " + family_name);

Di dalam kurung bulat `println`, tambahkan ini:

first_name + " " + family_name

Pada dasarnya, kami memberi tahu Java untuk mencetak konten variabel yang dikenal sebagai `first_name`. Kemudian, kami memiliki tanda plus dan spasi. Ruang dibatasi dalam tanda kutip ganda. Ini penting untuk memastikan bahwa Java akan mengerti bahwa kita ingin menambahkan spasi. Setelah spasi ini, pastikan untuk menambahkan tanda plus lain

kemudian variabel `family_name`. Meskipun ini mungkin sedikit rumit, kami hanya meminta Java untuk mencetak nama depan, spasi, lalu nama keluarga. Kode harus terlihat seperti yang ditunjukkan di bawah ini:

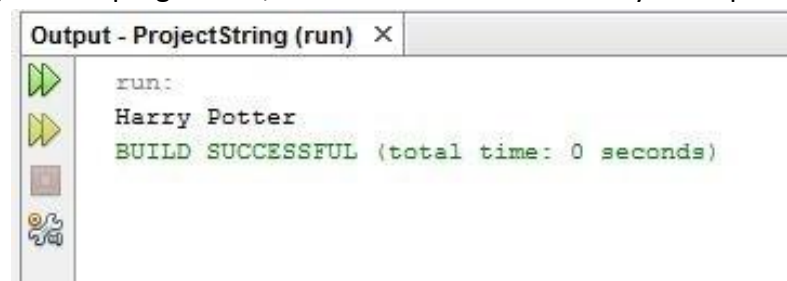
```
public class ProjectStrings {

    public static void main(String[] args) {
        String first_name = "Harry";
        String family_name = "Potter";

        System.out.println( first_name + " " + family_name);
    }
}
```

Gambar 3.13 Penggunaan kode variabel string

Saat Anda menjalankan program ini, Anda harus melihat ini di layar Output:



Gambar 3.14 Hasi penggunaan kode variabel string

Jika Anda hanya perlu menahan satu karakter, maka variabel yang harus Anda gunakan adalah `char`. Perhatikan bahwa itu dalam huruf kecil `c`. Untuk menahan satu karakter, Anda harus menggunakan tanda kutip tunggal daripada tanda kutip ganda. Perhatikan program di bawah ini dengan menggunakan variabel `char`.

```
public class ProjectStrings {

    public static void main(String[] args) {
        char first_name = 'H';
        char family_name = 'S';

        System.out.println( first_name + " " + family_name);
    }
}
```

Gambar 3.15 Jika hanya satu huruf, string diganti menjadi `char`

Cobalah untuk membatasi variabel `char` dalam tanda kutip ganda, dan NetBeans akan menandainya dengan garis bawah merah, dan akan menghasilkan kesalahan yang tidak kompatibel. Namun, Anda masih dapat memiliki variabel string hanya dengan menggunakan satu karakter. Tetapi Anda harus menggunakan tanda kutip ganda. Karenanya, baris ini baik-baik saja:

String first_name = "H";

Tapi ini tidak apa-apa:

```
String first_name = 'S';
```

Contoh pertama memiliki tanda kutip ganda, sedangkan yang kedua memiliki tanda kutip tunggal.

Masih banyak lagi yang harus dipelajari tentang variabel string, dan kita akan membahasnya di bab-bab selanjutnya. Pada titik ini, kita perlu melanjutkan dan mendapatkan beberapa masukan.

3.6 MENERIMA MASUKAN

Dengan Java, Anda dapat memanfaatkan berbagai pustaka kode yang dapat Anda gunakan. Kode-kode ini telah ditetapkan untuk melakukan tugas-tugas tertentu. Anda hanya perlu menentukan kode spesifik yang ingin Anda gunakan, lalu menentukan metode. Kelas yang berguna, yang dapat menangani input pengguna dikenal sebagai kelas Pemindai. Ini dapat dirujuk di perpustakaan: **java.util**. Anda harus merujuknya sebelum Anda dapat menggunakan Pemindai kelas. Ini dapat dilakukan dengan menggunakan kata kunci impor.

```
import java.util.Scanner;
```

Impor baris harus ditempatkan di atas pernyataan Kelas:

```
import java.util.Scanner;  
public class ProjectStrings {  
}
```

Ini akan menginstruksikan program Java untuk menggunakan kelas tertentu dalam pustaka tertentu (kelas Pemindai) yang dapat ditemukan di pustaka: **java.util**.

Selanjutnya, Anda perlu membuat objek dari class Scanner. Perhatikan bahwa kelas hanyalah sekelompok kode. Itu tidak dapat melakukan sesuatu kecuali Anda membuat objek lain.

Untuk menyiapkan Pemindai objek baru, Anda perlu mengetikkan kode ini:

```
Scanner input_user = new Scanner( System.in );
```

Oleh karena itu, Anda perlu menyiapkan variabel Pemindai alih-alih variabel String atau variabel int. Nama variabel kita adalah **input_user**. Di sebelah simbol sama dengan, kita telah memasukkan kata kunci baru, yang dapat digunakan untuk membuat objek baru di kelas. Perhatikan bahwa objek yang kita siapkan ini bersumber dari class Scanner. Di dalam kurung bulat, kita perlu menginstruksikan Java bahwa kita menginginkan ini untuk **System.in** atau **System Input**.

Untuk menerima masukan dari pengguna, kami perlu menentukan tindakan dari berbagai metode yang dapat Anda gunakan untuk objek Pemindai Anda. Metode dikenal sebagai berikut, yang dapat memperoleh string teks berikut, yang dapat diketik pengguna.

```
String first_name;  
first_name = input_user.next( );
```

Setelah `input_user`, kami mengikutinya dengan titik. Sebuah daftar popup akan muncul berisi daftar metode yang dapat Anda gunakan. Pilih metode selanjutnya dan ikuti menggunakan titik koma di akhir baris. Juga, dimungkinkan untuk mencetak teks sebagai panduan:

```
String first_name;  
System.out.print("Type your first name here: ");  
first_name = input_user.next( );
```

Perhatikan bahwa kami telah menggunakan `print` alih-alih `println` yang telah kami gunakan dalam kode kami sebelumnya. Perhatikan bahwa hasil cetak akan tetap pada satu baris, sedangkan `println` akan menggeser kursor ke baris lain di sebelah output. Selanjutnya, tambahkan prompt lain untuk nama keluarga atau nama keluarga:

```
String family_name;  
System.out.print("Type your family name here: ");  
family_name = input_user.next( );
```

Perhatikan bahwa ini adalah kode yang serupa, tetapi program sekarang akan menyimpan informasi apa pun yang diberikan pengguna ke dalam variabel `family_name` daripada variabel `first_name`.

Untuk menampilkan input, cukup tambahkan kode di bawah ini:

```
String full_name;  
full_name = first_name + " " + family_name;  
System.out.println("Hello ");
```

Sekarang, kami telah menetapkan `full_name` sebagai String variabel lain. Kami menyimpan informasi apa pun dalam variabel `family_name` dan `first_name`. Perhatikan bahwa ada spasi di antara variabel-variabel ini, dan baris terakhir akan mencetak semuanya dari layar Output.

Kode Anda akan terlihat seperti tangkapan layar di bawah ini:


```

package projectstring;

import java.util.Scanner;

public class ProjectStrings {

    public static void main(String[] args) {

        Scanner input_user = new Scanner(System.in);

        String first_name;
        System.out.print("Type your first name here:");
        first_name = input_user.next();

        String family_name;
        System.out.print("Type your family name here:");
        family_name = input_user.next();

        String full_name;
        full_name = first_name + " " + family_name;

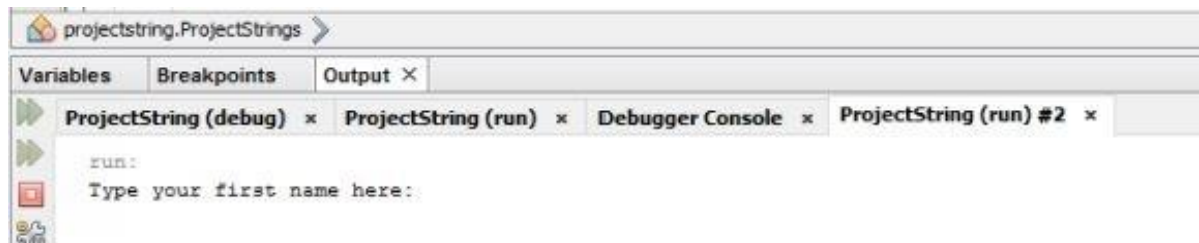
        System.out.print( "Hello, ");

        System.out.println( first_name + " " + family_name);
    }
}

```

Gambar 3.16 Pengembangan kode variabel string

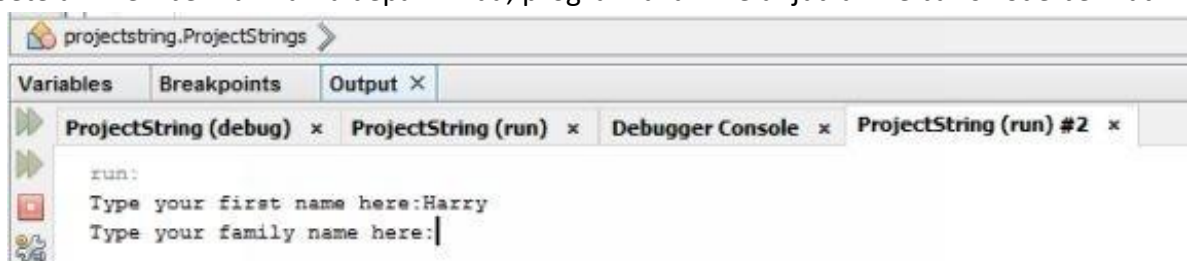
Saat Anda menjalankan program, layar Output akan terlihat seperti ini:



Gambar 3.17 Mengetikkan nama depan

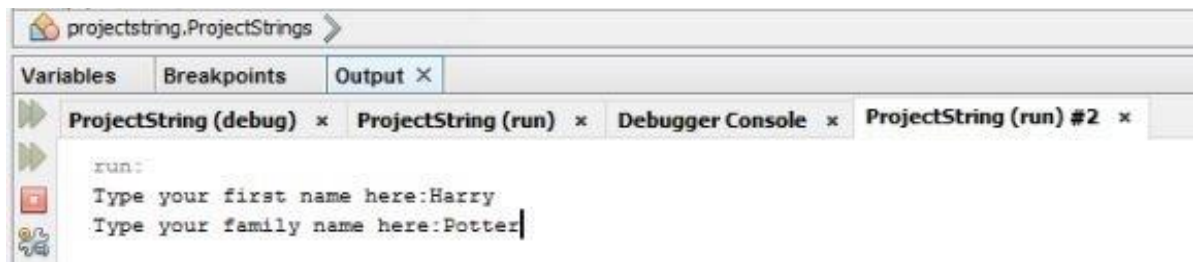
Pada titik ini, Java sedang menunggu Anda untuk mengetik sesuatu. Ini tidak akan dilanjutkan sampai Anda menekan tombol Enter di keyboard Anda. Cukup klik tombol kiri mouse di sebelah "Ketik nama depan Anda di sini:" dan kursor akan berkedip. Masukkan nama depan Anda, dan tekan tombol Enter. Setelah menekan tombol Enter, program akan menyimpan informasi yang Anda berikan dan menyimpannya di dalam variabel nama setelah simbol sama dengan. Dalam kode kami, ini mengacu pada variabel `first_name`.

Setelah memberikan nama depan Anda, program akan melanjutkan ke baris kode berikut:



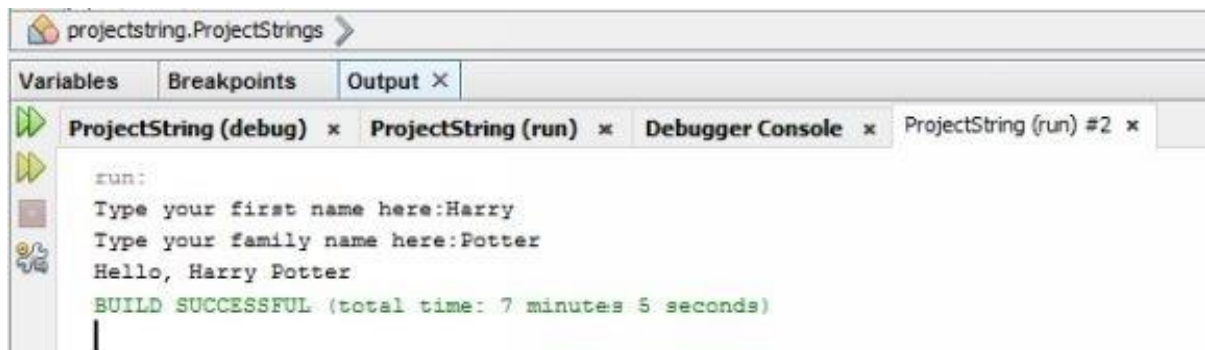
Gambar 3.18 Mengetik nama keluarga

Berikan nama keluarga, dan tekan lagi tombol Enter:



Gambar 3.19 Tampilan ketika sudah memberikan nama keluarga

Kami sekarang telah menyelesaikan input dari pengguna, dan program akan mulai memproses kode, yang merupakan kombinasi dari dua nama. Hasil akhir harus terlihat seperti ini:



Gambar 3.20 Output setelah menekan tombol enter di keyboard

Sekarang, Anda telah mempelajari cara menggunakan Pemindai kelas untuk memperoleh informasi dari pengguna program. Setiap informasi yang diberikan oleh pengguna akan disimpan dalam variabel, dan hasilnya akan dicetak di layar Output.

3.7 PANEL OPSI

Kelas JOptionPane adalah kelas lain yang berharga untuk mendapatkan masukan dari pengguna dan menampilkan hasil. Anda dapat menemukan ini di perpustakaan: javax.swing. Kelas ini akan memungkinkan Anda menampilkan kotak input yang menarik seperti ini:



Gambar 3.21 Contoh opsi berupa input

Atau kotak pesan seperti ini:



Gambar 3.22 Panel opsi berupa pesan

Kami dapat menggunakan kode `ProjectStrings` kami dan menambahkan panel tipe opsi. Penting terlebih dahulu untuk merujuk perpustakaan yang kita butuhkan. Masukkan ini ke dalam kode:

`import javax.swing.JOptionPane;`

Ini akan menginstruksikan program bahwa Anda memerlukan kelas `JOptionPane`, terletak di perpustakaan `javax.swing`.

Jika Anda suka, kita bisa memulai proyek lain. Saya percaya, sekarang Anda sudah tahu bagaimana memulai sebuah proyek. Pastikan untuk mengubah nama kelas menjadi nama pilihan Anda sendiri. Kami akan menggunakan nama kelas `DisplayPanels` untuk yang satu ini, dan nama paketnya adalah `inputuser`.

Ketik baris impor baru di proyek baru. Layar pengkodean harus muncul seperti ini:

```
package inputuser;
import javax.swing.JOptionPane;

public class DisplayPanels {

    public static void main(String[] args) {

    }
}
```

Gambar 3.23 Memulai kelas `JOptionPane`

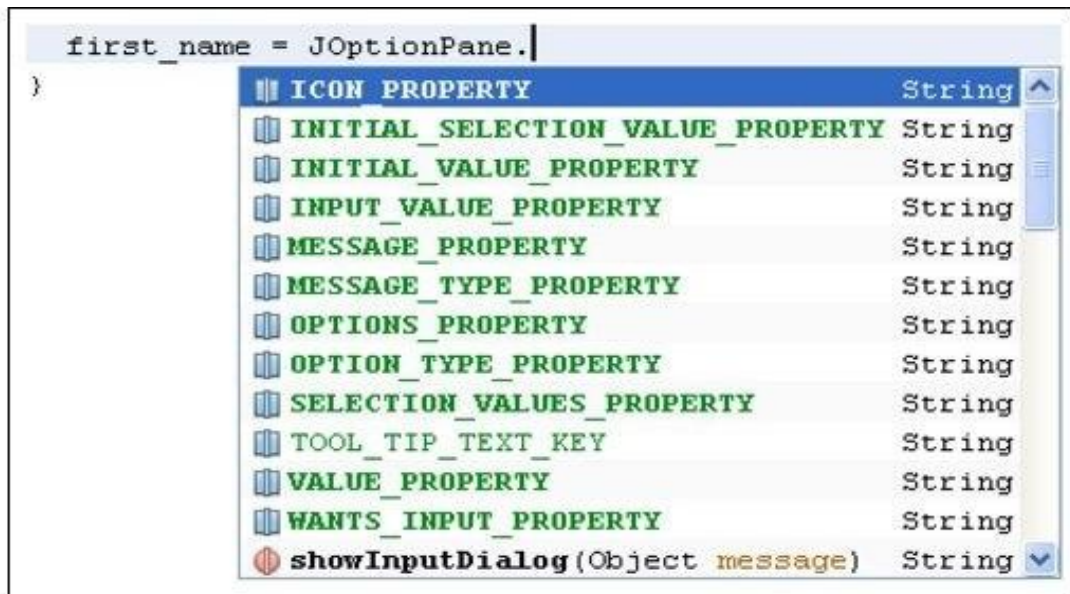
Perhatikan bahwa ada garis kuning bergelombang pada kode impor karena kita belum menggunakan kelas apa pun. Setelah kami melakukannya, itu akan hilang begitu saja, jadi jangan khawatir tentang itu.

Untuk membuat kotak input, yang dapat diberikan oleh pengguna, Anda dapat menggunakan metode `showInputDialog` dari `JOptionPane`. Sama seperti di bagian sebelumnya, kita akan menangani input langsung ke nama depan variabel. Tambahkan baris berikut di bawah metode utama sehingga Anda dapat melihat cara kerjanya:

`String first_name;`

`first_name = JOptionPane.showInputDialog("First Name");`

Setelah Anda memasukkan periode setelah `JOptionPane`, daftar popup akan muncul seperti ini:



Gambar 3.24 Daftar opsi kelas JOptionPane

Pilih `showInputDialog`. Di dalam tanda kurung bulat dari tipe `showInputDialog`, masukkan pesan, yang ingin Anda tampilkan di atas kotak teks untuk input. Kami telah memilih "Nama Depan". Mirip dengan string, ini harus dibatasi di dalam tanda kutip ganda.

Ketik kode di bawah ini, sehingga kami bisa mendapatkan nama keluarga pengguna:

```
String family_name;
```

```
family_name = JOptionPane.showInputDialog("Family Name");
```

Gabungkan ini bersama-sama, dan sertakan pesan:

```
String full_name;
```

```
full_name = "Hello" + first_name + " " + family_name;
```

Untuk menampilkan pesan dalam kotak tampilan, sertakan baris ini:

```
JOptionPane.showMessageDialog( null, full_name );
```

Pada titik ini, Anda harus memilih `showMessageDialog`. Di dalam tanda kurung bulat, Anda harus memasukkan kata `null`. Kata kunci ini menandakan bahwa kotak untuk pesan tidak boleh dihubungkan ke aspek program apa pun. Koma harus diikuti oleh teks yang Anda suka untuk ditampilkan dari prompt pesan. Kode Anda harus muncul seperti:

```

public class DisplayPanes {

    public static void main(String[] args) {

        String first_name;
        first_name=JOptionPane.showInputDialog("Enter Your First Name");

        String family_name;
        family_name = JOptionPane.showInputDialog("Enter Your Family Name");

        String full_name;
        full_name = "Hello, " + first_name + " " + family_name;

        JOptionPane.showMessageDialog(null, full_name);
        System.exit(0);
    }
}

```

Gambar 3.25 Menambahkan kode showInputDialog

Perhatikan kode akhir di bawah ini:

System.exit(0);

Codeline ini akan memastikan bahwa kotak akan keluar. Itu juga akan merapikan barang-barang karena bisa menyingkirkan benda-benda dari memori.

Setelah Anda menjalankan kode, Anda akan melihat kotak input untuk Nama Depan. Masukkan teks dan klik OK.



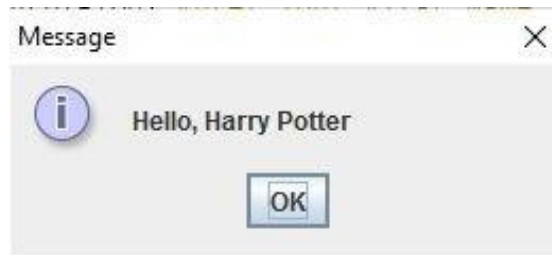
Gambar 3.26 Memasukkan nama depan dan klik ok

Setelah kotak input untuk Nama Keluarga muncul, masukkan nama dan tekan OK:



Gambar 3.27 Memasukkan nama keluarga dan tekan ok

Kotak ini akan muncul setelah mengklik OK.



Gambar 3.28 Tampilan akhir dari kotak dialog berupa pesan

Sekarang Anda telah belajar cara membuat kotak display. Tapi ini hanyalah puncak gunung es karena Anda dapat melakukan hal-hal yang jauh lebih besar dengan Java. Bab selanjutnya akan membahas semua tentang Aliran Kontrol.

BAB 4

ALIRAN KONTROL

Pemrograman yang telah Anda pelajari pada saat ini dikenal sebagai pemrograman sekuensial, yang menandakan bahwa kode dimulai dari atas ke bawah. Ini cukup linier, karena setiap baris kode akan ditafsirkan, mulai dari baris pertama yang Anda masukkan hingga akhir baris terakhir. Tapi itu adalah pemrograman yang sangat primitif. Anda memerlukan kode untuk dimulai setelah kondisi tertentu terpenuhi. Misalnya, Anda mungkin memerlukan satu pesan untuk ditampilkan jika pengguna adalah pria dan pesan yang berbeda jika pengguna adalah wanita. Anda perlu mengontrol aliran program. Ini bisa dilakukan melalui logika kondisional. Logika kondisional terutama tentang kata "IF." IF pengguna adalah laki-laki, maka tampilkan pesan ini. IF pengguna adalah perempuan, maka tampilkan pesan ini. Untungnya, cukup mudah menggunakan logika kondisional dalam pemrograman Java. Logika kondisional pertama yang akan kita pelajari adalah pernyataan IF.

4.1 PERNYATAAN IF

Dalam pemrograman, sangat umum untuk menjalankan kode jika kondisi tertentu terpenuhi. Itulah mengapa Pernyataan IF dikembangkan. Di Jawa, struktur pernyataan IF adalah ini:

```
if ( Statement ) {  
}
```

Anda mulai dengan kata huruf kecil jika diikuti oleh sepasang tanda kurung bulat. Kurung melengkung kemudian digunakan untuk mengelompokkan sekumpulan kode. Kode ini adalah satu-satunya kode yang ingin Anda jalankan setelah kondisi IF terpenuhi. Kondisi itu sendiri harus dibatasi di antara tanda kurung bulat:

```
if ( user < 21 ) {  
}
```

Kondisi ini menyatakan bahwa IF user kurang dari 21. Namun daripada menggunakan less than, kita bisa menggunakan tanda less than (<). Jika pengguna berusia kurang dari 21 tahun, maka Anda perlu melakukan tindakan tertentu seperti menampilkan pesan:

```
if ( user < 21 ) {  
    //DISPLAY MESSAGE  
}
```

Ketika pengguna tidak kurang dari 21, maka kode dalam kurung kurawal harus dilanjutkan, dan program akan terus berjalan hingga baris kode terakhir. Apa pun yang Anda sertakan dalam kurung kurawal hanya akan berlaku JIKA kondisinya telah terpenuhi, dan kondisi ini dibatasi dalam kurung bulat. Simbol lain yang bisa kita gunakan adalah simbol lebih besar dari (>). Pernyataan IF di atas dapat diubah sedikit untuk memeriksa apakah pengguna lebih besar dari 21.


```
if ( user > 21) {
    //DISPLAY MESSAGE
}
```

Satu-satunya perubahan yang telah kita lakukan sejauh ini dalam kode adalah mengubah simbol kurang dari (<) menjadi lebih besar dari simbol (>). Pernyataan IF sekarang akan memeriksa apakah pengguna lebih besar dari 21.

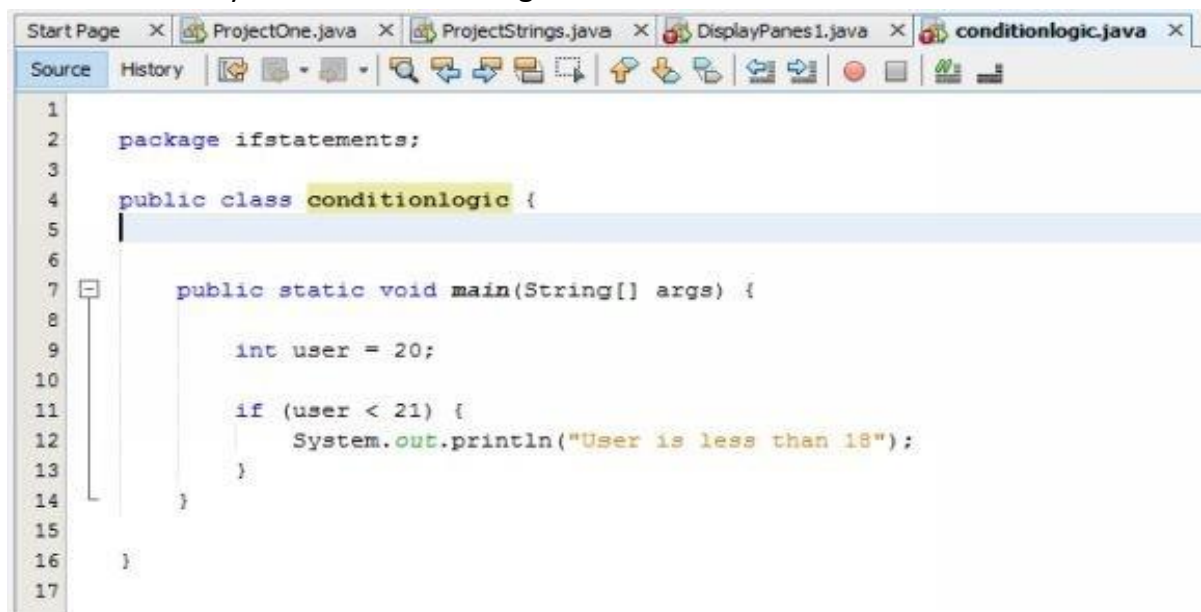
Namun, kode ini tidak akan memeriksa pengguna yang secara spesifik berusia 21 dan bukan mereka yang lebih besar dari 21. Jika Anda perlu memeriksa pengguna yang berusia 21 tahun atau lebih, Anda dapat menyertakan simbol sama dengan. Dengan demikian:

```
if ( user >= 21 ) {
    //DISPLAY MESSAGE
}
```

Demikian pula, Anda dapat memeriksa tanda kurang dari atau sama dengan dengan cara yang sama:

```
if ( user <= 21 ) {
    //DISPLAY MESSAGE
}
```

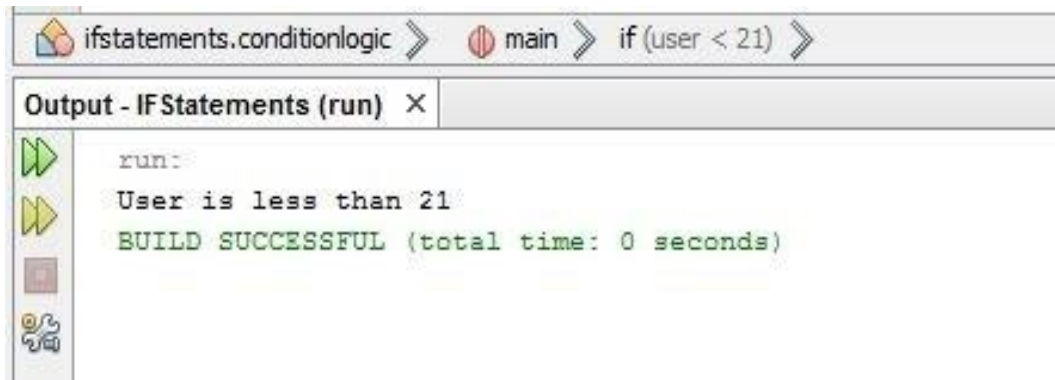
Kode di atas berisi tanda kurang dari (<) dan simbol sama dengan (=). Sekarang, kita dapat mencoba kode ini dalam program dasar. Memulai proyek baru. Anda dapat memberi nama paket dan nama kelas apa pun yang Anda sukai. Kami telah menamai paket **IFStatements** sementara kelasnya adalah **conditionlogic**.



Gambar 4.1 Membuat proyek baru dengan kelas conditionlogic

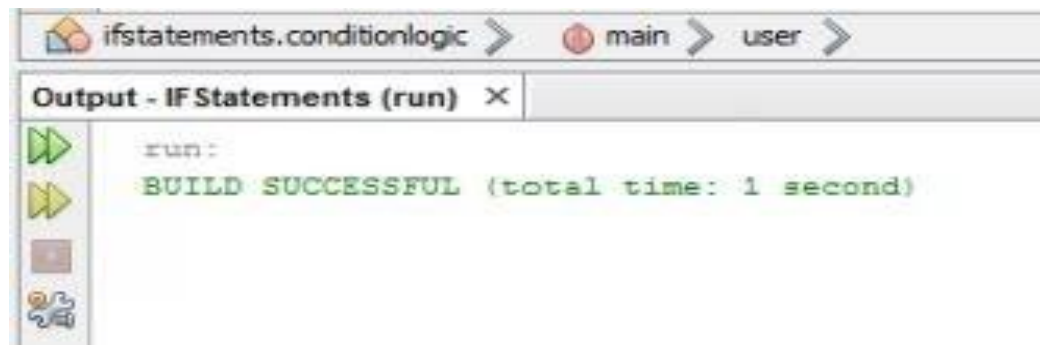
Kami telah menetapkan variabel integer, dan mendefinisikan nilai 20. Pernyataan IF akan memeriksa "kurang dari 21". Oleh karena itu, pesan dalam kurung melengkung akan ditampilkan. Anda sekarang dapat menjalankan program. Jika Anda perhatikan, NetBeans

menjalankan program dalam teks kotak di layar Proyek dan bukan kode yang Anda tunjukkan. Untuk menjalankan kode di layar pengkodean, cukup klik kanan di area kode. Daftar popup akan muncul. Pilih Jalankan Berkas. Anda dapat melihat ini di layar Output:



Gambar 4.2 Output ketika nilai user kurang dari 21

Selanjutnya, ubah nilai variabel pengguna dari 20 menjadi 21. Jalankan, dan Anda harus melihat ini:



Gambar 4.3 Output ketika nilai user diubah menjadi 21

Program berjalan dengan baik, dan tidak ada pesan kesalahan. Namun, tidak ada print out, karena pesan di dalam tanda kurung kurawal Pernyataan IF, yang memeriksa nilai kurang dari 21. Kondisi tidak terpenuhi sehingga Java mengabaikan tanda kurung kurawal dan melanjutkan dengan sisa kode. Sekarang, coba ganti `<` dengan `< =` simbol. Ubah juga pesan yang sesuai seperti “pengguna kurang dari atau sama dengan 21”. Jalankan programnya. Pesan apa yang Anda lihat? Selanjutnya, Anda dapat mengubah nilai pengguna menjadi 22, dan menjalankan program.

Apakah pesannya masih ada? Anda juga dapat memiliki beberapa Pernyataan IF dalam kode. Coba yang ini:

```

public class conditionlogic {

    public static void main(String[] args) {

        int user = 21;

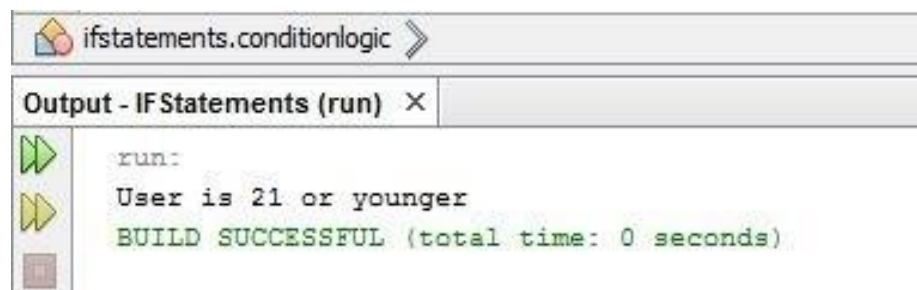
        if (user <= 21) {
            System.out.println("User is 21 or younger");
        }

        if (user > 21) {
            System.out.println("User is greater than 21");
        }
    }
}

```

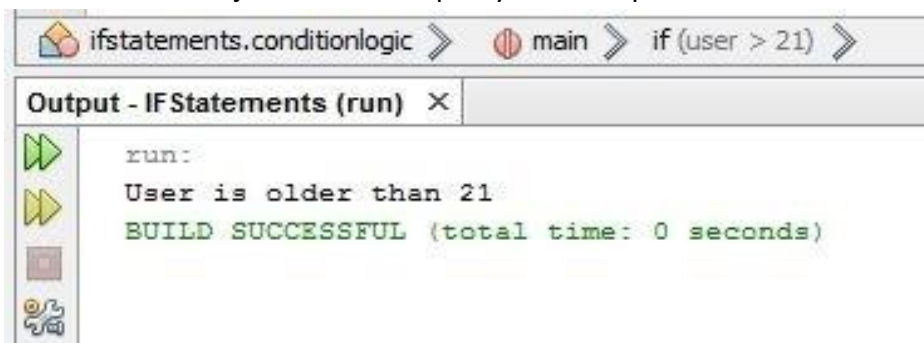
Gambar 4.4 Mengembangkan kode pernyataan IF

Pada titik ini, kami memiliki dua Pernyataan IF. Yang pertama memeriksa nilai yang kurang dari atau sama dengan 21. Pernyataan IF kedua memeriksa nilai yang lebih besar dari 21. Setelah Anda menjalankan kode dengan nilai 21 atau kurang untuk pengguna variabel, Outputnya adalah:



Gambar 4.5 Output ketika nilai variabel user kurang atau sama dengan 21

Ubah nilai variabel user menjadi 22 dan Outputnya akan seperti ini:



Gambar 4.6 Output ketika nilai variabel user lebih dari 21

Oleh karena itu, hanya satu dari pernyataan IF yang akan mencetak pesan di layar. Dan pesan akan tergantung pada nilai yang diberikan dalam variabel pengguna.

4.2 IF... ELSE

Daripada menggunakan dua Pernyataan IF, lebih mudah menggunakan Pernyataan IF...Else. Di bawah ini adalah format pernyataan IF...Else:

```
if ( condition ) {  
}  
else {  
}
```

Ini dimulai dengan kode "if" dan kemudian tanda kurung bulat yang berisi kondisi yang ingin Anda periksa. Ingatlah untuk menggunakan tanda kurung kurawal untuk memotong beberapa opsi. Opsi kedua berjalan setelah "else" dan dibatasi dengan tanda kurung melengkungnya sendiri. Sekali lagi, ini kodenya, yang menguji usia pengguna:

```
public static void main(String[] args) {  
  
    int user = 22;  
  
    if (user <= 21) {  
        System.out.println("User is 21 or younger");  
    }  
    else {  
        System.out.println("User is older than 21");  
    }  
}
```

Gambar 4.7 Penggunaan kode if dan else

Sekarang kami hanya memiliki dua opsi: apakah pengguna 21 atau kurang dari atau pengguna lebih besar dari itu. Ubah kode Anda sehingga akan menyerupai screenshot di atas, dan jalankan. Anda akan menemukan bahwa pesan pertama akan ditampilkan. Anda dapat mengubah nilai variabel pengguna menjadi 22 dan menjalankannya kembali. Teks di antara tanda kurung kurawal setelah ELSE akan ditampilkan di Layar Output.

4.3 IF... ELSE IF

Dimungkinkan untuk memeriksa beberapa opsi. Misalnya, katakanlah kami ingin memeriksa rentang usia yang lebih banyak seperti 20 hingga 40 tahun, dan 41 tahun ke atas? Untuk lebih dari dua opsi, kita dapat menggunakan pernyataan If...Else If. Format pernyataan If...Else If adalah sebagai berikut:

```
if ( condition_1 ) {  
}  
else if ( condition_2 ) {  
}  
else {  
}
```

Ini adalah bagian baru:

```
else if ( condition_2 ) {  
    }  
}
```

Oleh karena itu, IF pertama memeriksa kondisi_1 (21 atau lebih rendah, misalnya). Kemudian yang lain jika diikuti oleh tanda kurung bulat. Kondisi_2 dibatasi di dalam tanda kurung bulat yang baru. Setiap informasi yang tidak dibatasi oleh 2 kondisi pertama akan dibatasi dalam if yang lain. Ingat, kode Anda harus dipotong melalui tanda kurung kurawal, dengan setiap if, else if, atau else, dengan kumpulan tanda kurung kurawalnya sendiri. Hapus satu tanda kurung, dan pesan kesalahan akan ditampilkan.

Sebelum menjalankan kode ini, penting terlebih dahulu untuk mempelajari operator lain untuk logika kondisional. Sejauh ini, kami telah menggunakan yang berikut:

Kurang Dari (<)

Lebih Besar dari (>)

Kurang Dari atau Sama Dengan (<=)

Lebih Besar dari atau Sama Dengan (>=)

Berikut adalah empat operator lagi yang dapat kita gunakan:

Dan (&&)

Bukan (!)

Atau (||)

Memiliki nilai (==)

Operator And terdiri dari dua ampersand (&&), dan dapat digunakan untuk memeriksa lebih dari satu kondisi secara bersamaan. Kita bisa menggunakan ini untuk memeriksa dua rentang usia:

```
else if ( user > 21 && user < 60)
```

Pada titik ini, kami bermaksud menguji apakah pengguna lebih besar dari 21 tetapi kurang dari 60. Perhatikan bahwa kami mencoba menguji apa yang terkandung dalam variabel pengguna. Kondisi pertama adalah "lebih besar dari 21" (pengguna > 21), dan kondisi kedua adalah "kurang dari 60" (pengguna < 60). Di antara keduanya, kami memiliki operator &&. Oleh karena itu, seluruh baris kode berarti "lain jika pengguna lebih besar dari 21 DAN pengguna kurang dari 60."

Kami akan mencoba operator kondisional lainnya di bagian selanjutnya. Pertama, mari kita coba kode baru:

```

public static void main(String[] args) {

    int user = 21;

    if (user <= 21) {
        System.out.println("User is 21 or younger");
    }
    else if (user > 21 && user < 60) {
        System.out.println("User is between 22 and 59");
    }
    else {
        System.out.println("User is older than 60");
    }
}

```

Gambar 4.8 Contoh penggunaan kode if,else, dan else if dengan nilai user 21

Jalankan program dan uji. Sekarang, Anda seharusnya telah mengembangkan keterampilan untuk menebak dengan tepat apa yang akan dicetak sebelum Anda menjalankan kode. Karena kami memiliki nilai 21 untuk pengguna variabel, pesan yang dibatasi di antara tanda kurung kurawal dari else akan menampilkan layar Output. Ubah nilai variabel pengguna dari 21 menjadi 50. Tampilan untuk bagian lain dari kode sekarang harus ditampilkan. Anda memiliki kebebasan untuk menambahkan sebanyak mungkin bagian if. Katakanlah kita ingin menguji apakah pengguna berusia 50 atau 55 tahun. Sekarang kita dapat menggunakan operator lain. Kita dapat menguji apakah variabel untuk pengguna memiliki nilai 50 atau memiliki nilai 55.

else if (user == 50 || user == 55)

Gunakan dua tanda sama dengan untuk memeriksa apakah variabel pengguna memiliki nilai sesuatu. Perhatikan bahwa tidak boleh ada spasi di antara tanda sama dengan. Java akan memeriksa nilai ini saja. Karena kami ingin memeriksa apakah pengguna juga berusia 55, kami dapat menyertakan kondisi lain dalam tanda kurung bulat yang sama: user==55. Ini akan menginstruksikan Java untuk memeriksa apakah variabel pengguna memiliki nilai 55. Perhatikan bahwa di antara kedua kondisi ini, kami telah menambahkan operator OR. Ingat, tidak boleh ada spasi di antara dua karakter. Seluruh kode baris berarti "Lain jika pengguna memiliki nilai 55 ATAU pengguna memiliki nilai 55". Di bawah ini adalah kode dengan bagian lain yang baru ditambahkan.

```

public static void main(String[] args) {

    int user = 50;

    if (user <= 21) {
        System.out.println("User is 21 or younger");
    }
    else if (user > 21 && user < 60) {
        System.out.println("User is between 22 and 59");
    }
    else if (user == 50 || user == 55) {
        System.out.println("User is either 50 or 55");
    }
    else {
        System.out.println("User is older than 60");
    }
}

```

Gambar 4.9 Contoh penggunaan kode if,else, dan else if dengan nilai user 50

Sekarang, ubah nilai variabel pengguna menjadi 55 dan jalankan kodenya. Selanjutnya, ubah menjadi 60 dan jalankan kodenya. Dalam kedua kasus, pesan baru harus ditampilkan. Cobalah sendiri. Ubah nilai variabel pengguna menjadi 45 dan jalankan kode Anda. Kemudian ubah menjadi 50 dan jalankan kembali kodenya. Dalam kedua kasus, pesan baru akan ditampilkan. Mungkin sulit untuk menggunakan operator kondisional yang berbeda. Namun, kami hanya memeriksa variabel untuk kondisi tertentu. Ini pada dasarnya hanya tentang memilih operator kondisional yang sesuai untuk fungsi yang Anda sukai.

4.4 PERNYATAAN BERSATU

Dimungkinkan untuk membuat sarang pernyataan logika bersyarat. Bersarang berarti membatasi satu pernyataan di antara yang lain. Misalnya, jika kita ingin memeriksa apakah pengguna berusia kurang dari 21 tahun, tetapi lebih dari 18 tahun, kita dapat menampilkan pesan yang berbeda untuk pengguna di atas 18 tahun. Kita dapat mulai dengan pernyataan IF pertama:

```

if ( user < 21 ) {
    System.out.println( "21 or younger");
}

```

Untuk menguji lebih dari 18, Anda dapat menambahkan pernyataan kedua di dalam pernyataan pertama. Strukturnya masih sama:

```

if ( user < 22 ) {
    if ( user > 18 && user < 22 ) {
        System.out.println( "You are 18 or 22");
    }
}

```

Oleh karena itu, pernyataan pertama mencakup variabel pengguna jika lebih rendah dari 22. Pernyataan kedua akan membatasi variabel pengguna ke bawah untuk nilai lebih dari 18 dan lebih rendah dari 22. Untuk mencetak pesan terpisah, Anda dapat menggunakan pernyataan If...else daripada pernyataan If seperti yang ditunjukkan di atas.

```
if ( user < 22 ) {
if ( user > 18 && user < 22 ) {
System.out.println( "You are 18 or 22");
}
else {
System.out.println( "18 or younger");
}
}
```

Perhatikan lokasi kurung kurawal: salah tempatkan satu kurung dan program Anda tidak akan berjalan. Pernyataan IF bersarang bisa jadi rumit. Namun, kami mencoba mempersempit opsi.

4.5 NILAI BOOLEAN DALAM PEMROGRAMAN JAVA

Nilai Boolean adalah salah satu dari dua opsi: 1 atau 0, ya atau tidak, benar atau salah. Dalam pemrograman Java, ada tipe variabel untuk Boolean:

```
boolean user = true;
```

Oleh karena itu, daripada memasukkan string atau double atau int, Anda dapat menggunakan boolean (dengan b kecil). Setelah nama variabel, Anda dapat menentukan nilai benar atau salah. Perhatikan bahwa operator penugasan mengacu pada satu tanda sama dengan (=). Jika Anda perlu menguji apakah suatu variabel "memiliki nilai" sesuatu, Anda dapat menggunakan tanda sama dengan ganda (==).

Anda dapat mencoba kode dasar ini:

```
boolean user = true;
if ( user == true ) {
System.out.println("it's true");
}
else {
System.out.println("it's false");
}
```

Oleh karena itu, pernyataan pertama menguji apakah nilai variabel pengguna benar. Bagian lain menguji apakah nilainya salah. Tidak perlu menyatakan "else if (user==false)", karena jika tidak benar, maka salah. Karena itu, kita bisa menggunakan yang lain. Hanya ada dua opsi dalam hal nilai boolean. Operator lain pada opsi kami adalah operator NOT. Kita bisa menggunakan ini dengan nilai boolean. Perhatikan kode ini:

```
boolean user = true;
if ( !user ) {
```

```

System.out.println("it's flase");
}
else {
System.out.println("it's true");
}

```

Ini hampir mirip dengan nilai boolean lainnya, tetapi untuk yang ini:

```
if ( !user ) {
```

Tetapi pada titik ini, kita dapat menggunakan operator NOT sebelum variabel pengguna. Operator NOT adalah satu tanda seru (!) dan ditempatkan sebelum variabel yang Anda coba periksa. Ini memeriksa negasi yang berarti memeriksa kebalikan dari nilai sebenarnya. Karena variabel user telah ditetapkan sebagai true, maka operator ! akan memeriksa nilai-nilai palsu. Jika pengguna telah ditetapkan ke false, maka operator ! akan memeriksa nilai sebenarnya. Sederhananya, jika suatu nilai TIDAK benar, lalu apa itu? Atau jika suatu nilai TIDAK salah, lalu apakah itu?

4.6 PERNYATAAN SWITCH JAVA

Pernyataan switch juga digunakan untuk mengontrol aliran program. Pernyataan-pernyataan ini memberi Anda pilihan untuk memeriksa rentang nilai untuk variabel. Anda dapat menggunakannya daripada pernyataan if...else if yang panjang dan canggung. Ini adalah format pernyataan switch:

```

switch ( variable_2_check ) {
case value:
code_here;
break;
case value:
code_here;
break;
default:
values_not_caught_above;
}

```

Mulailah dengan kata switch dan ikuti dengan sepasang tanda kurung bulat. Variabel yang ingin Anda uji harus dibatasi di antara tanda kurung bulat sakelar. Kemudian, tambahkan sepasang kurung melengkung. Bagian lain dari pernyataan switch semuanya harus berada dalam dua kurung melengkung. Anda harus menggunakan kata case untuk setiap nilai yang ingin Anda uji. Anda kemudian dapat menguji nilai ini:

```
case value:
```

Case value harus diikuti oleh titik dua. Kemudian, Anda dapat menempatkan apa yang Anda butuhkan untuk melihat apakah mereka dapat mencocokkan nilai. Ini adalah kode yang ingin Anda jalankan. Kata kunci break harus digunakan untuk memecahkan setiap kasus

pernyataan. Adalah opsional untuk memasukkan nilai standar. Anda dapat menempatkannya jika ada nilai lain, yang dapat disimpan dalam variabel, tetapi Anda belum mengujinya di mana pun dalam pernyataan tersebut.

Jika Anda bingung dengan ini, Anda dapat mencoba kode ini. Anda dapat memulai proyek baru untuk ini, atau Anda dapat menambahkan komentar baru. Cara cepat untuk mengomentari kode dalam NetBeans adalah dengan mengklik ikon komentar yang ada di bilah alat. Langkah pertama adalah menyorot kode yang ingin Anda komentari. Langkah selanjutnya adalah mengklik ikon untuk berkomentar.



Gambar 4.10 Ikon untuk menambahkan komentar baru

Sekarang di sini adalah kode:

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

    int user = 21;

    switch ( user ) {
        case 21:
            System.out.println("You are 21");
            break;
        case 22:
            System.out.println("You are 22");
            break;
        case 23:
            System.out.println("You are 23");
            break;
        default:
            System.out.println("You are not 21, 22 or 23");
    }
}
```

Gambar 4.11 Tampilan area kode setelah klik ikon komentar

Kode ini akan menentukan nilai yang perlu Anda periksa. Kami sekarang telah menetapkan variabel integer dan menyebutnya sebagai pengguna. Kami telah mendefinisikan nilainya menjadi 18. Pernyataan tersebut akan menguji variabel pengguna dan memeriksa nilainya. Ini kemudian akan melewati setiap pernyataan. Setelah menemukan satu yang setuju dengan, kode akan berhenti dan berjalan untuk kasus itu dan kemudian akan mematahkan pernyataan tersebut. Anda dapat mencoba menjalankan program ini. Ketik nilai yang berbeda untuk pengguna variabel dan lihat apa yang akan terjadi. Sayangnya, tidak ada cara untuk

memeriksa rentang nilai setelah kasus. Oleh karena itu, Anda dapat memeriksa satu nilai saja. Oleh karena itu, tidak mungkin melakukan kode ini:

case (user <= 21):

Sementara itu, dimungkinkan untuk melakukan ini:

case 1: case 2: case 3: case 4:

Tes di atas akan memeriksa rentang nilai dari satu hingga empat. Namun, Anda perlu menentukan setiap nilai. Pastikan untuk mencatat penempatan semua kasus dan titik dua.

4.7 JAVA LOOP

Selama ini pemrograman yang kita lakukan sekarang adalah sekuensial. Ini berjalan dari atas ke bawah dengan setiap baris kode dijalankan, kecuali jika Anda menyertakan bagian yang memberi tahu Java untuk tidak membaca kode itu. Di bagian sebelumnya, menggunakan pernyataan IF untuk memotong area kode adalah metode untuk menginstruksikan Java agar tidak membaca setiap baris. Metode lain untuk mengganggu aliran dari atas ke bawah adalah melalui loop. Dalam pemrograman, loop adalah salah satu yang memaksa program untuk menjalankan baris kode lagi dan lagi. Misalnya, kita ingin menambahkan angka 11 hingga 20. Anda dapat dengan mudah menghitungnya di Java seperti ini:

int addition = 11 + 12 + 13 + 14 + 15 + 16 + 17 + 18 + 19 + 20;

Namun, menggunakan metode ini untuk menjumlahkan rangkaian angka yang panjang seperti 1 hingga 10.000 memakan waktu. Metode yang lebih mudah adalah menggunakan loop untuk mengembalikan kode baris berulang kali hingga Anda mencapai 10.000. Anda kemudian dapat keluar dari loop dan melanjutkan. Jenis perulangan yang umum disebut For Loops. Di bawah ini adalah format LOOP ini:

```
for ( start_value; end_value; increment_number ) {  
    //INSERT_CODE_HERE  
}
```

Kata kunci for diikuti oleh tanda kurung bulat yang berisi tiga elemen: nilai awal, nilai akhir, dan metode untuk memperoleh dari satu angka ke angka lainnya. Ini dikenal sebagai angka kenaikan, dan sering kali nilainya satu. Anda memiliki kebebasan untuk memilih kenaikan. Di sebelah tanda kurung bulat adalah tanda kurung kurawal, yang digunakan untuk memotong kode yang perlu Anda jalankan berulang kali. Bingung? Berikut adalah contoh kode. Mulai proyek baru, dan beri nama proyek dan kelas apa pun yang Anda suka. Untuk ini, saya telah menamai proyek ForLoops dan kelas JavaLoops. Selanjutnya, tambahkan kode di bawah ini:

```

package forloops;

public class javaloops {

    public static void main(String[] args) {

        int loopVal;
        int end_value = 21;

        for (loopVal = 0; loopVal <end_value; loopVal++) {

            System.out.println("Loop Value = " + loopVal);

        }

    }
}

```

Gambar 4.12 Penggunaan kode java loops

Kita bisa mulai dengan mendefinisikan variabel integer yang kita beri nama loopVal. Baris kedua mendefinisikan variabel int lain, yang dapat kita gunakan untuk nilai akhir loop, yang disetel ke 21. Apa yang ingin kita lakukan adalah mengulang untuk mencetak angka dari nol hingga 21. Kami memiliki kode berikut dalam tanda kurung bulat dari for loop:

loopVal =0; loopVal < end_value; loopVal++

Potongan pertama menginstruksikan Java untuk memulai pada nilai ini dalam perulangan. Dalam hal ini, kami mendefinisikan nilai nol untuk variabel loopVal. Ini akan digunakan sebagai nomor pertama dalam loop. Potongan berikutnya menggunakan logika kondisional.

loopVal < end_value

Baris di atas berarti bahwa "loopVal kurang dari nilai_end". Perulangan for kemudian akan berjalan dan berjalan jika nilai dari variabel loopVal masih lebih rendah dibandingkan dengan nilai yang didefinisikan pada variabel end_value. Java akan terus menjalankan kode yang dibatasi di dalam kurung kurawal selama nilai loopVal masih kurang dari nilai variabel end_value. Di bawah ini adalah potongan terakhir di dalam tanda kurung bulat dari for loop

loopVal++

Pada titik ini, kami menginstruksikan Java metode yang kami suka untuk diikuti untuk beralih dari nilai awal di loopVal ke nomor berikutnya dalam seri. Kami bermaksud menghitung dari nol hingga 20. Angka berikutnya adalah 1. Kode loopVal++ berarti "menambahkan kenaikan 1 ke nilai variabel. Daripada mengekspresikan loopkVal++, kita bisa menulis ini:

loopVal = loopVal + 1

Setelah loopVal+1, Java akan menambahkan kenaikan 1 untuk apa pun yang saat ini disimpan dalam variabel loopVal. Setelah menambahkan satu ke nilai ini, itu akan menyimpan

hasil dalam variabel sebelum simbol sama dengan. Sekali lagi, ini adalah variabel loopVal. Hasilnya adalah 1 akan terus ditambahkan ke loopVal. Ini disebut sebagai penambahan variabel. Ini sangat umum dalam pemrograman sehingga notasi variabel yang ditandai dengan dua simbol plus (++) dikembangkan untuk ini:

```
int set_number = 0;  
set_number++;
```

Nilai set_number adalah 1 setelah Anda menjalankan kode. Ini adalah tangan pendek untuk mengungkapkan ini:

```
int set_number = 0;  
set_number = set_number + 1;
```

Singkatnya, for loop mengacu pada ini:

Nilai mulai lingkaran: 0

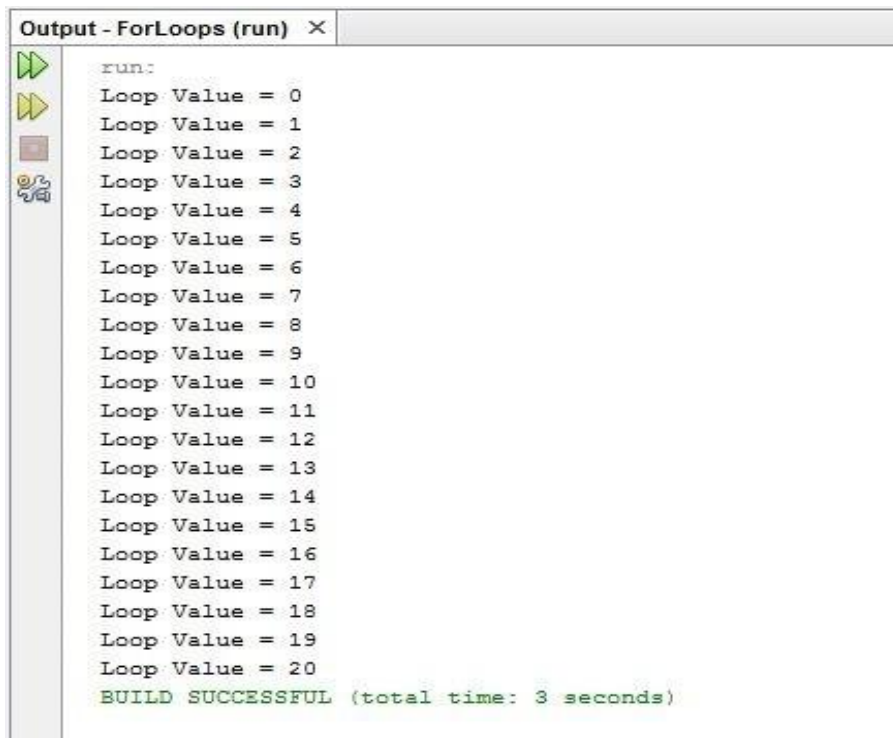
Terus Perulangan sampai: Nilai awal kurang dari 21

Metode untuk meningkatkan nilai akhir: Terus tambahkan 1 ke nilai awal

Kami memiliki tanda kurung kurawal dari for loop:

```
System.out.println("Loop Value = " + loopVal);
```

Apa pun yang saat ini dibatasi variabel loopVal dapat dicetak dengan pesan apa pun. Sekarang, jalankan program dan pesan ini akan muncul di layar Output:



```
Output - ForLoops (run) X
run:
Loop Value = 0
Loop Value = 1
Loop Value = 2
Loop Value = 3
Loop Value = 4
Loop Value = 5
Loop Value = 6
Loop Value = 7
Loop Value = 8
Loop Value = 9
Loop Value = 10
Loop Value = 11
Loop Value = 12
Loop Value = 13
Loop Value = 14
Loop Value = 15
Loop Value = 16
Loop Value = 17
Loop Value = 18
Loop Value = 19
Loop Value = 20
BUILD SUCCESSFUL (total time: 3 seconds)
```

Gambar 4.13 Output ketika kode loop yang sudah dibuat sebelumnya dijalankan

Oleh karena itu, kami telah membatasi program dalam satu lingkaran, dan menginstruksikannya untuk berjalan berputar-putar. Setiap loop, kenaikan 1 akan

ditambahkan ke variabel loopVal. Loop ini akan terus berjalan lagi dan lagi saat nilai loopVal lebih rendah dari nilai yang ditentukan di end_value. Apa pun di dalam kurung melengkung dari loop mengacu pada kode yang akan berjalan lagi dan lagi. Ini adalah keseluruhan ide dari loop: untuk menjalankan kode kurung kurawal berulang-ulang. Berikut adalah kode yang memungkinkan Anda untuk menambahkan angka 0 hingga 20. Coba:

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

    int loopVal;
    int end_value = 21;
    int addition = 0;

    for (loopVal = 0; loopVal < end_value; loopVal++) {

        addition = addition + loopVal;

    }

    System.out.println("Total = " + addition);

}
```

Gambar 4.14 Kode yang memungkinkan anda untuk menambahkan angka 0 - 20

Hasil yang harus ditampilkan di layar Output adalah 210. Kurang lebih, kodenya sendiri mirip dengan perulangan for sebelumnya. Anda memiliki set variabel yang sama yang ditulis di atas – loopVal dan end_value. Kemudian, kita telah menambahkan variabel int lain yang kita sebut sebagai penambahan. Ini akan menyimpan nilai jumlah 0 hingga 20. Di dalam tanda kurung bulat dari loop, itu juga serupa untuk kode sebelumnya. Kami telah mengulang saat nilai variabel loopVal lebih rendah dari nilai yang ditentukan di end_value. Sementara itu, kami menambahkan kenaikan 1 ke variabel loopVal setiap kali loop berjalan (loopVal++). Di dalam kurung melengkung, kita sekarang memiliki satu baris kode:

addition = addition + loopVal;

Baris kode yang satu ini menambahkan angka dari 0 hingga 20. Jika Anda masih belum mengerti cara kerjanya, Anda dapat memulai di sebelah simbol sama dengan:

addition + loopVal;

Putaran pertama dari loop mendefinisikan penambahan variabel, yang menyimpan 0 sebagai nilainya. Sedangkan variabel loopVal, masih menyimpan nilai 1 (nilai awalnya). Program akan menambahkan 0 ke 1, dan akan menyimpan jawaban variabel sebelum simbol sama dengan. Ingat, ini adalah penambahan variabel. Nilai apa pun yang disimpan dalam kode sebelumnya akan dihilangkan dan akan diganti dengan nilai baru. Putaran kedua loop, nilai variabel akan menjadi baru lagi.

addition (1) + loopVal (2);;

Tentu saja, jawabannya adalah 3. Oleh karena itu, nilai baru ini akan disimpan ke variabel sebelum simbol sama dengan. Ini sekarang adalah nilai loop untuk putaran ketiga.

addition (3) + loopVal (3);

Program akan menambahkan nilai dan akan menyimpan hasilnya ke variabel sebelum simbol sama dengan. Ini akan terus berjalan lagi dan lagi sampai loop berakhir. Jawaban akhirnya adalah 210. Perhatikan bahwa garis untuk mencetak terletak di luar for loop, mengikuti tanda kurung kurawal akhir untuk loop.

4.8 WHILE LOOPS

Jenis perulangan lain, yang dapat Anda gunakan di Java dikenal sebagai perulangan while. Anda akan menemukan bahwa perulangan while lebih mudah dipelajari dibandingkan dengan perulangan for. Berikut adalah format dari while loop:

```
while ( condition ) {  
}
```

Kami akan mulai dengan kata kunci dalam huruf kecil. Kondisi yang ingin Anda periksa akan dibatasi di dalam tanda kurung bulat. Ini akan diikuti oleh sepasang kurung kurawal serta kode yang ingin Anda jalankan. Berikut adalah while loop, yang mencetak pesan. Coba yang ini:

```
int loopVal = 0;  
while ( loopVal < 10) {  
System.out.println("Display Some Message");  
loopVal++;  
}
```

Kondisi yang harus diperiksa dibatasi di antara tanda kurung bulat. Kami bermaksud untuk terus mengulang sampai nilai yang disimpan dalam variabel loopVal lebih rendah dari 10. Di dalam kurung kurawal, program kami akan menampilkan sebaris pesan. Kami kemudian dapat menambah nilai variabel loopVal. Tanpa mendefinisikan ini, kita akan memiliki infinite loop, karena tidak ada cara bagi variabel loopVal untuk mendapatkan nilai di luar nilai awal nol. Meskipun kami telah menggunakan penghitung untuk melanjutkan ke kondisi akhir, loop sementara ideal untuk digunakan jika Anda benar-benar tidak membutuhkan nilai penghitungan, tetapi hanya nilai untuk diperiksa. Misalnya, tidak apa-apa untuk mempertahankan loop saat pengguna masih tidak menekan tombol apa pun. Ini berguna dalam pemrograman game. Tombol tertentu dapat ditekan untuk keluar dari loop while. Ini lebih dikenal sebagai game loop, dan karena itu bisa menjadi game itu sendiri. Contoh bagus lainnya adalah mengulang file teks saat program belum mencapai akhir file.

4.9 DO...WHILE

Do...While terkait dengan perulangan while. Formatnya adalah ini:

```
int loopVal = 0;  
do {
```

```
System.out.println("Display Some Message");  
loopVal++;  
}  
while ( loopVal < 10 );
```

Perhatikan bahwa program akan berulang lagi dan lagi sampai Anda memenuhi kondisi akhir. Pada titik ini, bagian `while` terletak di bagian bawah. Namun, kondisinya serupa: terus perulangan sementara nilai variabel `loopVal` kurang dari 10. Perbedaan utama antara keduanya adalah bahwa kode di dalam tanda kurung kurawal untuk `do... while` dapat dijalankan setidaknya sekali. Menggunakan `while` loop, kondisi dapat dengan mudah dipenuhi. Program hanya akan keluar dari loop, dan bahkan tidak menjalankan kode di dalam kurung kurawal. Untuk memeriksa ini, Anda dapat mencoba loop `while` terlebih dahulu. Anda dapat mengubah nilai variabel `loopVal` menjadi 10, lalu jalankan. Anda akan menemukan bahwa pesan tidak tercetak. Sekarang, Anda dapat mencoba loop `do` dengan nilai 5 yang ditentukan untuk `loopVal`. Pesan akan ditampilkan sekali, dan kemudian Java akan keluar dari loop. Pada bab berikutnya, kita akan mempelajari lebih lanjut tentang Java Array.

BAB 5

JAVA ARRAY

Konsep pemrograman yang perlu Anda pelajari untuk membuat kode secara efektif adalah array. Ini adalah konsep yang sangat penting, sehingga layak mendapatkan babnya sendiri.

5.1 APA ITU JAVA ARRAY?

Pada titik ini, kami telah mengkodekan dengan variabel yang menyimpan nilai tunggal. Variabel string yang telah kita siapkan hanya menampung string teks yang panjang, sedangkan variabel int hanya menyimpan satu angka. Untuk menyimpan lebih dari satu nilai pada saat yang sama, kita perlu menggunakan array. Ini mirip dengan daftar item. Array mirip dengan kolom di perangkat lunak spreadsheet Anda.

	Array_Values
0	10
1	14
2	36
3	27
4	43
5	18

Gambar 5.1 Tampilan array pada spreadsheet

Mirip dengan spreadsheet, array menyertakan nomor posisi untuk setiap baris. Posisi dalam array dimulai dari nol dan secara berurutan akan meningkat. Setiap posisi dalam array kemudian dapat menyimpan nilai. Pada gambar di atas larik posisi 0 menyimpan nilai 10, larik posisi 1 menampung 14, dan larik posisi 2 menyimpan nilai 36, dan seterusnya dan seterusnya. Untuk membuat larik angka yang mirip dengan tangkapan layar ini, kita perlu menginstruksikan Java jenis data yang akan ada dalam larik (nilai boolean, string, bilangan bulat, dan lainnya). Kemudian, Anda perlu mengatakan berapa banyak posisi array. Formatnya seperti ini:

```
int[ ] numArray;
```

Perbedaan utama antara mendefinisikan variabel integer normal dan array adalah sepasang tanda kurung siku tepat setelah tipe data. Tanda kurung siku sudah cukup untuk memerintahkan Java yang ingin Anda buat sebuah array. Nama array kita di atas adalah numArray. Mirip dengan variabel biasa, Anda memiliki kebebasan untuk memberi nama variabel sesuka Anda. Namun, ini akan menginstruksikan Java bahwa Anda bermaksud membuat integer array. Itu tidak akan mengatakan berapa banyak posisi yang harus disimpan oleh array. Untuk melakukan ini, kita perlu mendefinisikan array objek baru:

```
numArray = new int[5];
```


Kita akan mulai dengan nama array dan kemudian tanda sama dengan. Di sebelah simbol sama dengan, kami menambahkan kata kunci baru, dan kemudian jenis datanya. Di sebelah tipe data adalah sepasang tanda kurung siku. Dalam kurung ini, kita perlu menunjukkan ukuran array. Ukurannya adalah berapa banyak posisi yang harus disimpan oleh array. Kita dapat menulis semuanya dalam satu baris:

```
int[ ] numArray = new int[6];
```

Kami memerintahkan Java untuk membuat array dengan lima posisi di dalamnya. Setelah baris ini dijalankan, Java dapat menentukan nilai array default. Karena kita telah menentukan integer array, nilai default untuk lima posisi adalah 0. Untuk mendefinisikan nilai ke posisi yang berbeda dalam array, kita dapat melakukannya dengan metode biasa:

```
numArray[0] = 5;
```

Pada baris ini, kita menetapkan nilai 5 ke posisi 0 dalam array bernama numArray. Ingat, kami menggunakan tanda kurung siku untuk merujuk ke setiap posisi. Jika Anda bermaksud untuk menentukan nilai 25 ke posisi 1, kodenya harus seperti ini:

```
numArray[1] = 25;
```

Sementara itu, kita dapat menetapkan nilai 50 ke posisi array 2 melalui ini:

```
numArray[2] = 50;
```

Perhatikan, karena array dimulai dari 0, posisi ketiga dalam array menyertakan indeks nomor 2. Jika Anda mengetahui nilai yang ingin Anda sertakan dalam array, Anda dapat menuliskannya seperti ini:

```
int[ ] numArray = { 1, 2, 3, 4 };
```

Cara membuat array ini menggunakan tanda kurung kurawal di sebelah simbol yang sama. Di dalam kurung melengkung, kita dapat menentukan nilai yang dapat disimpan oleh setiap array. Nilai pertama bisa posisi 0 sedangkan nilai kedua posisi 1, dst. Ingat, Anda harus menambahkan tanda kurung siku di sebelah int, tetapi tidak pada kata kunci new atau siklus dari jenis data dan tanda kurung siku. Namun, ini hanya untuk tipe data nilai char, string, dan int. Jika tidak, Anda harus menggunakan kata kunci baru. Karenanya, Anda dapat menulis ini:

```
String[ ] stringsArray = { "Harry", "Ron", "Hermione", "Snape" };
```

Tapi tidak ini:

```
boolean[ ] boolsArray = {true, false, true, false};
```

Untuk membuat array boolean, kita masih perlu menggunakan kata kunci new:

```
boolean[ ] boolsArray= new boolean[ ] {true, false, true, false};
```

Untuk memindahkan nilai yang disimpan dalam array, kita dapat menulis nama array dan memasukkan array posisi dalam tanda kurung siku seperti ini:

```
System.out.println( numArray[3] );
```

Kode ini akan menampilkan nilai apa pun yang disimpan pada posisi array 3 dalam array bernama numArray. Mari kita lakukan beberapa latihan. Mulailah proyek baru dan beri nama apa pun yang Anda suka. Pastikan untuk memilih nama baru untuk Kelas. Masukkan kode di bawah ini ke dalam metode Utama yang baru:

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

    int[] numArray;

    numArray = new int[5];

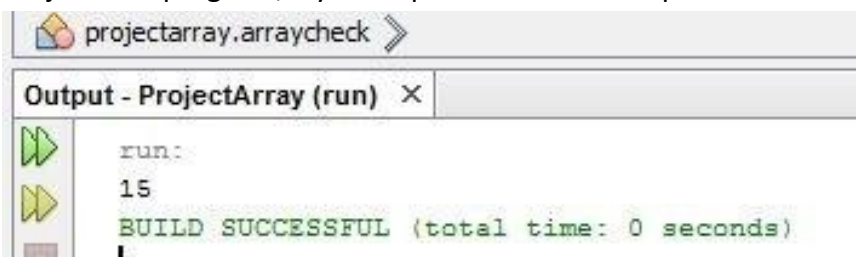
    numArray[0] = 5;
    numArray[1] = 10;
    numArray[2] = 15;
    numArray[3] = 20;
    numArray[4] = 25;

    System.out.println( numArray[2] );

}
```

Gambar 5.2 Penggunaan kode numArray

Setelah kita menjalankan program, layar Output akan terlihat seperti ini:



Gambar 5.3 Output penggunaan kode numArray

Coba ubah posisi larik dalam angka di dalam cetakan garis dari 2 menjadi 4, dan 25 akan ditampilkan sebagai gantinya.

5.2 LOOP DAN ARRAY

Array secara alami memiliki loop mereka sendiri. Di bagian sebelumnya, kode di bawah ini digunakan untuk menetapkan nilai ke posisi array:

numArray[0] = 5;

Namun, ini tidak disarankan jika Anda memiliki daftar panjang angka yang perlu Anda tetapkan ke array. Misalnya, katakanlah kita perlu mengembangkan program lotere dan kita perlu menetapkan angka 1 hingga 60 ke posisi dalam array. Daripada mengkodekan daftar posisi lengkap dengan nilainya, kita dapat menggunakan loop. Di bawah ini adalah kode untuk menyelesaikannya:

```

package projectarray;

public class arraycheck {

    public static void main(String[] args) {

        int[] lotto_nums = new int[60];
        int i;

        for (i=0; i < lotto_nums.length; i++) {
            lotto_nums[i] = i + 1;
            System.out.println( lotto_nums[i] );
        }

    }
}

```

Gambar 5.4 Kode untuk loop dan array

Dalam kode ini, kita telah mendefinisikan sebuah array untuk menyimpan 60 nilai int. Kemudian, kami menambahkan kode loop. Perhatikan kondisi akhir loop:

`i < lottery_numbers.length`

Panjang mengacu pada properti objek larik, yang dapat Anda gunakan sehingga Anda dapat memperoleh ukuran larik atau jumlah posisi. Oleh karena itu, loop ini akan terus berjalan sementara nilai variabel di `i` lebih rendah dibandingkan dengan ukuran array. Kode di bawah ini digunakan untuk mendefinisikan nilai untuk setiap posisi array:

`lottery_numbers[i] = i + 1;`

Daripada nilai hard-code di dalam tanda kurung siku array nama, kami menggunakan variabel `i`, yang meningkat satu nilai setiap kali loop berjalan. Kami kemudian dapat mengakses setiap posisi array dengan memanfaatkan loop nilai. Nilai yang kita tetapkan untuk setiap posisi adalah `i` ditambah 1. Oleh karena itu, ini adalah contoh nilai inkremental dalam satu lingkaran tetapi dengan 1 ditambahkan. Karena nilai loop dimulai dari 0, ini akan memberi Anda angka 1 hingga 60. Baris yang ditambahkan dalam loop hanya akan menampilkan nilai di setiap posisi array. Jika Anda suka, Anda bahkan dapat mengetikkan kode untuk mencampur nomor larik. Ketika Anda memiliki nilai yang tercampur, Anda kemudian bisa mendapatkan enam yang pertama sehingga Anda dapat menetakannya sebagai nomor lotre. Kemudian ketik blok kode lain, yang membandingkan jumlah pengguna dengan nomor pemenang yang ditetapkan dan Anda sekarang akan memiliki kode lotre!

5.3 PENGURUTAN ARRAY

Ada metode Java yang dapat Anda gunakan untuk pengurutan array. Untuk menggunakan metode pengurutan ini, pertama-tama penting untuk merujuk pustaka yang

dikenal sebagai Array. Anda dapat melakukan ini menggunakan pernyataan impor. Anda dapat menggunakan program numArry dan menambahkan pernyataan impor di bawah ini:

import java.util.Arrays;

Kode harus terlihat seperti tangkapan layar ini:

```
package projectarray;

import java.util.Arrays;

public class arraycheck {

    public static void main(String[] args) {

        int[] numArray;
        numArray = new int[5];

        numArray[0] = 5;
        numArray[1] = 10;
        numArray[2] = 15;
        numArray[3] = 20;
        numArray[4] = 25;

    }

}
```

Gambar 5.5 Memulai pengurutan array

Setelah mengimpor Array perpustakaan, sekarang kita dapat menggunakan metode pengurutan. Anda harus menemukan ini mudah.

Arrays.sort(numArry);

Langkah pertama adalah mengkodekan kata "Array diikuti oleh titik. Setelah Anda menambahkan titik, program akan menampilkan daftar opsi yang dapat Anda lakukan dengan array. Cukup masukkan kata: urutkan. Di dalam kurung bulat, tempatkan nama array yang ingin Anda urutkan. Perhatikan bahwa tidak perlu menggunakan tanda kurung siku setelah nama array. Dan voila! Anda sekarang dapat mengurutkan array. Coba kode di bawah ini untuk melihat pekerjaan ini:

```

package projectarray;

import java.util.Arrays;

public class arraycheck {

    public static void main(String[] args) {

        int[] numArray;
        numArray = new int[5];

        numArray[0] = 5;
        numArray[1] = 10;
        numArray[2] = 15;
        numArray[3] = 20;
        numArray[4] = 25;

        Arrays.sort(numArray);

        int i;

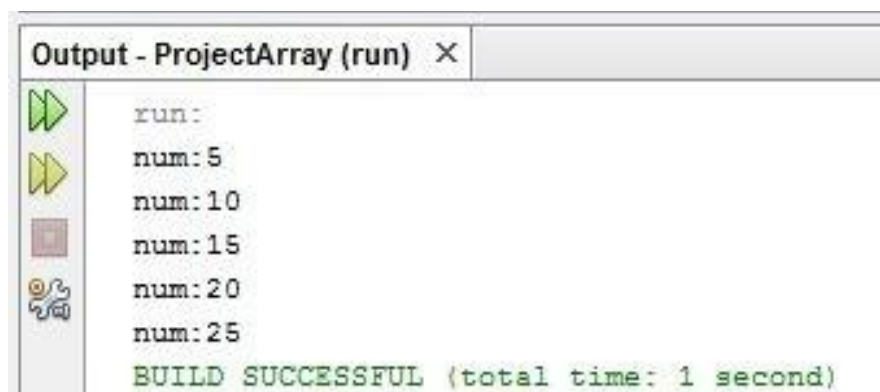
        for (i=0; i < numArray.length; i++) {
            System.out.println("num: " + numArray[i]);
        }

    }
}

```

Gambar 5.6 Menambahkan kode metode pengurutan

Perulangan for yang ditambahkan di akhir baris akan berjalan berulang kali untuk menampilkan nilai dalam setiap posisi larik. Setelah kode dijalankan, layar Output akan terlihat seperti ini:



```

Output - ProjectArray (run) X
run:
num: 5
num: 10
num: 15
num: 20
num: 25
BUILD SUCCESSFUL (total time: 1 second)

```

Gambar 5.7 Ouput dari proyek array

Seperti yang akan Anda lihat, program mengurutkan array dalam urutan menaik. Jika Anda ingin mengurutkan dalam urutan menurun, kita perlu menuliskan kode pengurutan, atau mengubah array menjadi objek int lalu mengimpor dari Koleksi perpustakaan. Di bawah ini adalah kode jika Anda ingin mengurutkan dalam urutan menurun:

```

package projectarray;

import java.util.Arrays;
import java.util.Collections;

public class DescendingSort {

    public static void main(String[] args) {

        int[] numArray;
        numArray = new int[5];

        numArray[0] = 5; numArray[1] = 10; numArray[2] = 15; numArray[3] = 20;
        numArray[4] = 25;

        Integer[] integerArray = new Integer[numArray.length];

        for (int i = 0; i < numArray.length; i++) {
            integerArray[i] = new Integer(numArray[i]);
        }

        Arrays.sort(integerArray, Collections.reverseOrder());

        for (int i = 0; i < numArray.length; i++) {
            System.out.println("num: " + integerArray[i]);
        }
    }
}

```

Gambar 5.8 Penggunaan kode dengan urutan menurun

Kode di atas bisa sedikit berantakan, tetapi bisa melakukan pekerjaan itu.

5.4 STRING DAN ARRAY

Konsep selanjutnya yang harus dipelajari adalah bagaimana menempatkan string teks di dalam array. Anda juga dapat melakukan ini menggunakan metode pemrosesan bilangan bulat yang serupa:

```

String[] stringArray = new String[5];
stringArray[0] = "What";
stringArray[1] = "does";
stringArray[2] = "the";
stringArray[3] = "fox";
stringArray[4] = "say";

```

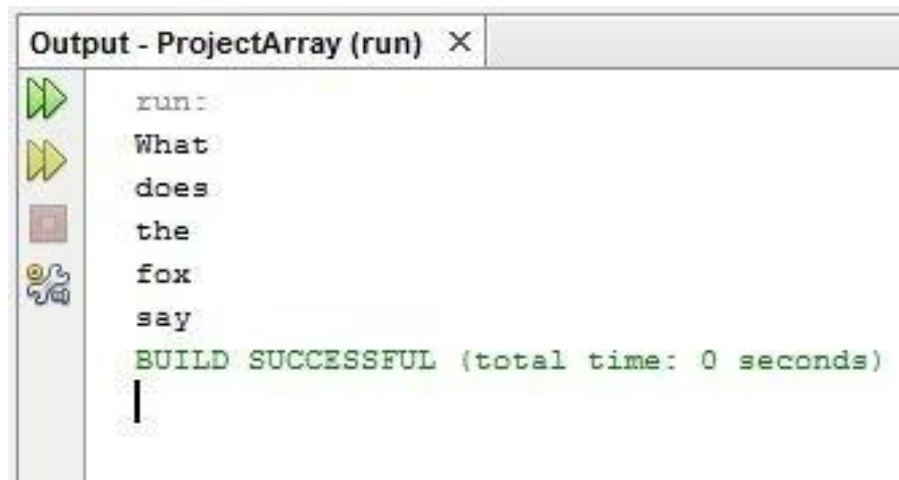
Kode ini mendefinisikan string array dengan lima posisi. Sebuah teks tertentu kemudian didefinisikan untuk setiap posisi array. Di bawah ini adalah loop, yang berjalan di sekitar semua posisi array, yang dapat menampilkan pesan untuk setiap posisi:

```

int i;
for ( i=0; i < stringArray.length; i++ ) {
    System.out.println( stringArray[i] );
}

```

Loop ini akan terus berjalan selama nilai dalam variabel yang dikenal sebagai `i` lebih rendah dari panjang array yang dikenal sebagai `stringArray`. Setelah Anda menjalankan program ini, layar Output akan terlihat seperti ini:



Gambar 5.9 Otuput proyek string dan array

Dimungkinkan juga untuk mengurutkan string array, mirip dengan apa yang telah kita lakukan dengan bilangan bulat. Namun, perhatikan bahwa pengurutan akan berdasarkan abjad. Oleh karena itu, `a` akan datang sebelum `b`. Java menggunakan teks Unicode untuk melakukan perbandingan huruf dengan string. Oleh karena itu, huruf besar akan muncul sebelum huruf kecil. Perhatikan kode di bawah ini:

```
package arraystrings;

import java.util.Arrays;

public class stringsarry {

    public static void main(String[] args) {

        String[] stringArray = new String [5] ;

        stringArray[0] = "What";
        stringArray[1] = "does";
        stringArray[2] = "the";
        stringArray[3] = "fox";
        stringArray[4] = "say";

        Arrays.sort(stringArray);

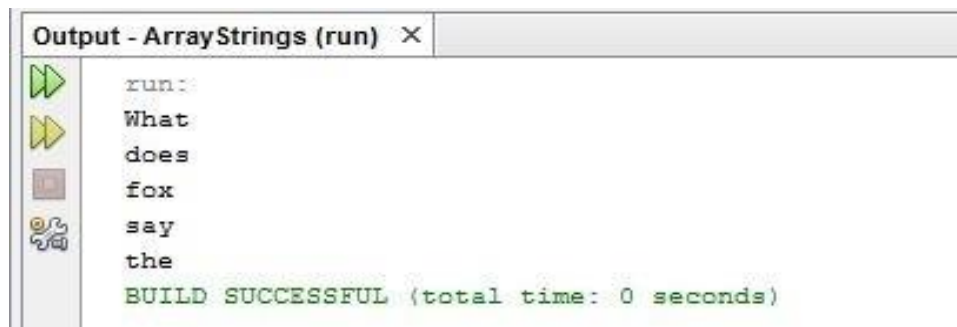
        int i;
        for (i=0; i < stringArray.length; i++) {
            System.out.println( stringArray[i] );
        }

    }

}
```

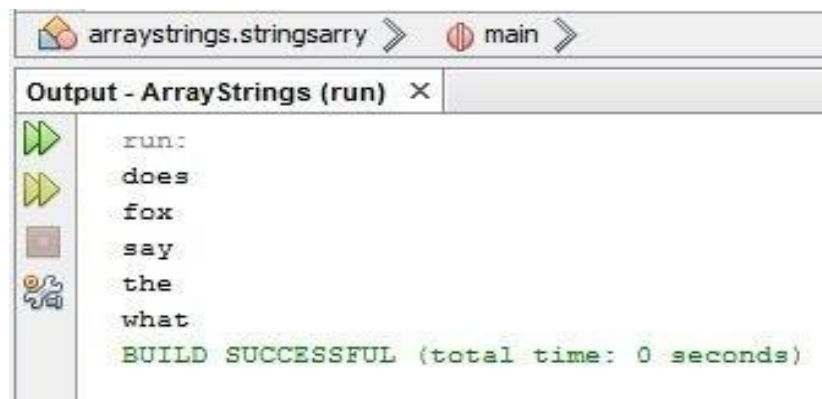
Gambar 5.10 Penggunaan kode pengurutan dengan nilai berupa abjad

Setelah Anda menjalankan kode ini, layar Output akan terlihat seperti ini:



Gambar 5.11 Output proyek pengurutan dengan nilai abjad

Meskipun kita sudah mengurutkan array, kata "What" tetap ditempatkan di urutan pertama. Jika ini adalah pengurutan abjad, kami mengharapkan kata "does" akan dicetak terlebih dahulu. Ini akan benar jika semua huruf dalam huruf kecil. Dalam kode, coba ubah huruf kapital "W" dari "What" menjadi "w" kecil. Jalankan kode lagi. Layar keluaran sekarang harus seperti ini:



Gambar 5.12 Output ketika huruf "W" diganti menjadi "w"

Pada titik ini, kata "apa" sekarang ditempatkan terakhir. Kita akan belajar lebih banyak tentang string di bagian selanjutnya.

5.5 JAVA MULTI-DIMENSIONAL ARRAY

Array yang kita gunakan selama ini hanya menyimpan satu kolom data. Namun, kita masih dapat mendefinisikan sebuah array untuk menyimpan beberapa kolom. Ini dikenal sebagai array multi-dimensi. Misalnya, pertimbangkan file excel dengan kolom dan baris. Jika Anda memiliki lima baris dan enam kolom, maka file excel dapat menyimpan 30 nilai seperti ini:

	A	B	C	D	E	F
1	50	18	64	63	15	75
2	34	88	43	46	19	14
3	23	12	65	35	22	77
4	56	43	45	44	34	85
5	78	11	32	36	32	20

Gambar 5.13 Contoh array multi-dimensi di dalam excel

Array multi-dimensi dapat menyimpan semua nilai di atas. Untuk mengatur array ini, gunakan kode di bawah ini:

```
int[ ][ ] numArrays = new int[5][6];
```

Ini didefinisikan sama seperti array biasa, kecuali Anda memiliki dua pasang kurung kurawal. Kurung kotak pertama untuk baris dan kurung kurawal kedua untuk kolom. Pada baris kode di atas, kita menginstruksikan Java untuk membuat array dengan lima kolom dan enam baris. Untuk menetapkan nilai dalam array multi-dimensi, Anda perlu melacak kolom dan baris. Berikut adalah kode untuk menetapkan baris pertama dari tangkapan layar excel.

```
numArrays[0][0] = 50;  
numArrays[0][1] = 18;  
numArrays[0][2] = 64;  
numArrays[0][3] = 63;  
numArrays[0][4] = 15;  
numArrays[0][5] = 75;
```

Oleh karena itu, baris pertama adalah baris 0. Kolom kemudian akan berubah dari nol menjadi 5, yaitu 6 item. Untuk mengisi baris berikutnya, kita perlu menulis baris berikut:

```
numArrays[1][0] = 34;  
numArrays[1][1] = 88;  
numArrays[1][2] = 43;  
numArrays[1][3] = 46;  
numArrays[1][4] = 19;  
numArrays[1][5] = 14;
```

Perhatikan bahwa jumlah kolom masih sama, tetapi jumlah baris sekarang ditetapkan menjadi 1. Untuk mengakses item dalam array multi-dimensi, strateginya adalah menggunakan satu loop di dalam yang lain. Berikut adalah kode untuk mengakses nomor dari atas. Ini akan menggunakan dua for loop:

```
public static void main(String[] args) {  
  
    int [ ][ ] numArrays = new int [5][6];  
  
    numArrays[0][0] = 50;      numArrays[1][0] = 34;  
    numArrays[0][1] = 18;      numArrays[1][1] = 88;  
    numArrays[0][2] = 64;      numArrays[1][2] = 43;  
    numArrays[0][3] = 63;      numArrays[1][3] = 46;  
    numArrays[0][4] = 15;      numArrays[1][4] = 19;  
    numArrays[0][5] = 75;      numArrays[1][5] = 14;  
  
    numArrays[2][0] = 23;      numArrays[3][0] = 56;  
    numArrays[2][1] = 12;      numArrays[3][1] = 43;  
    numArrays[2][2] = 65;      numArrays[3][2] = 45;  
    numArrays[2][3] = 35;      numArrays[3][3] = 44;  
    numArrays[2][4] = 22;      numArrays[3][4] = 34;  
    numArrays[2][5] = 77;      numArrays[3][5] = 85;
```

```

numArrays[4][0] = 78;
numArrays[4][1] = 11;
numArrays[4][2] = 32;
numArrays[4][3] = 36;
numArrays[4][4] = 32;
numArrays[4][5] = 20;

int rows = 5;
int columns = 6;
int i, j = 0;

for ( i = 0; i < rows; i++) {
    System.out.print(numArrays[i][j] + " ");
}
System.out.println( "" );
}

```

Gambar 5.14 Kode untuk mengakses nomor dari atas

Loop pertama akan digunakan untuk baris, sedangkan loop kedua akan digunakan untuk kolom. Pada putaran pertama loop pertama, nilai variabel *i* adalah 0. Kode di dalam loop *for* juga merupakan loop. Seluruh loop kedua akan berjalan selama nilai variabel *i* adalah 0. Untuk loop kedua menggunakan variabel *j*. Variabel *j* dan *i* kemudian dapat digunakan untuk mengakses array.

numArray[i][j]

Oleh karena itu, sistem loop ganda digunakan untuk mengakses semua nilai dalam array multidimensi, baris demi baris.

5.6 DAFTAR JAVA ARRAY

Jika Anda tidak yakin tentang angka yang perlu Anda simpan dalam array, sangat ideal untuk menggunakan *ArrayList*. Struktur data ini bersifat dinamis, artinya item dapat dimasukkan dan dihilangkan dari daftar. Array Java biasa adalah struktur statis karena Anda harus berurusan dengan ukuran array tetap. Untuk mendefinisikan *ArrayList*, pertama-tama penting untuk mendefinisikan paket impor yang bersumber dari perpustakaan *java.util*:

import java.util.ArrayList;

Selanjutnya, kita perlu membuat objek baru *ArrayList* menggunakan kode ini:

ArrayList listTest = new ArrayList();

Perhatikan, kita tidak memerlukan kurung kurawal pada saat ini. Saat kita membuat *ArrayList*, kita dapat mulai menambahkan elemen menggunakan metode *add*:

```

listTest.add( "item one" );
listTest.add( "item two" );
listTest.add( "item three" );
listTest.add( 8 );

```

Di dalam tanda kurung bulat dari metode `add`, kita perlu menempatkan teks atau angka yang ingin kita sertakan dalam `ArrayList`. Namun, kami hanya dapat menyertakan objek. Perhatikan bahwa 3 objek pertama yang kami sertakan dalam daftar adalah `Strings`, sedangkan objek terakhir adalah angka tertentu. Namun, perhatikan bahwa ini akan menjadi nomor objek tipe `integer` dan bukan tipe data primitif `int`. Daftar ini dapat dirujuk menggunakan nomor indeks serta melalui penggunaan metode `get`:

```
listTest.get( 3 )
```

Kode ini dapat memperoleh objek pada indeks posisi 3 dalam daftar ini. Ingat, nomor indeks dimulai dari nol, maka ini bisa menjadi item empat. Dimungkinkan juga untuk menyingkirkan item dari daftar array. Anda dapat menggunakan nilai daftar ini:

```
listTest.remove("item one");
```

Atau yang ini:

```
listTest.remove(1);
```

Setelah Anda menghapus item, `ArrayList` akan diubah ukurannya. Oleh karena itu, Anda harus berhati-hati dalam mendapatkan objek dari daftar jika Anda menggunakan indeks angka. Dalam contoh ini, jika Anda menghapus item 1, maka daftar hanya akan berisi tiga item. Mendapatkan objek menggunakan indeks nomor 2 kemudian akan menghasilkan kembali kesalahan. Untuk mengakses setiap item dalam `ArrayList`, kita dapat menambahkan `Iterator`, yang juga terletak di perpustakaan: `java.util`:

```
import java.util.Iterator;
```

Langkah selanjutnya adalah melampirkan `ArrayList` ke objek baru `Iterator`:

```
Iterator it = listTest.iterator( );
```

Ini akan menambahkan objek baru `Iterator`, yang dapat digunakan untuk mengakses objek dalam daftar bernama `listTest`. Sangat ideal untuk menggunakan objek `Iterator` karena ia dilengkapi dengan metode yang dikenal sebagai `hasNext` dan selanjutnya, yang dapat Anda gunakan dalam satu lingkaran.

```
while ( it.hasNext( ) ) {  
System.out.println( it.next( ) );  
}
```

Metode `hasNext` akan menghasilkan nilai `Boolean`. Nilai ini harus salah setelah tidak ada objek yang tersisa di dalam `ArrayList`. Kita dapat menggunakan metode selanjutnya untuk mengakses semua objek dalam daftar. Anda dapat memeriksa teori ini dengan menggunakan kode di bawah ini:

```

public static void main(String[] args) {

    ArrayList listTest = new ArrayList();

    listTest.add("item one");
    listTest.add("item two");
    listTest.add("item three");
    listTest.add (8);

    Iterator it = listTest.iterator();

    while (it.hasNext()) {
        System.out.println(it.next());
    }

    //REMOVE AN ITEM FROM THE LIST
    listTest.remove("item two");

    //PRINT OUT THE NEW LIST
    System.out.println("Whole list = " + listTest);

    //GET THE ITEM AT INDEX POSITION 1
    System.out.println("Position 1 = " + listTest.get(1));
}

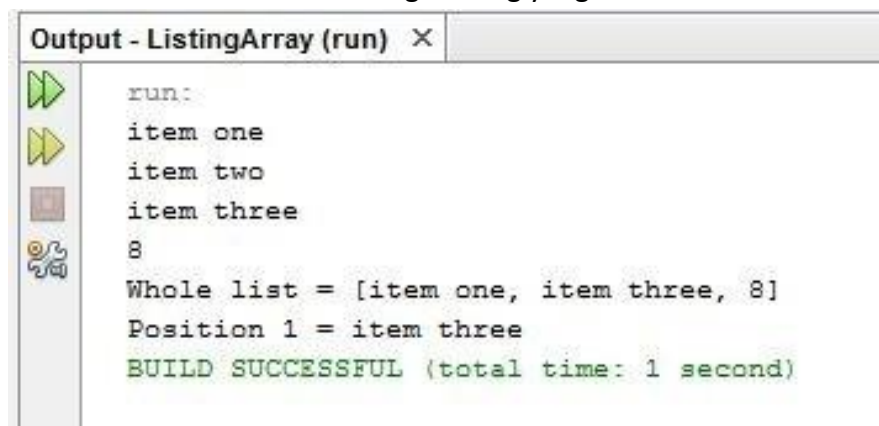
```

Gambar 5.15 Penggunaan kode hasNext

Perhatikan baris, yang menampilkan seluruh daftar:

System.out.println("Whole list=" + listTest);

Melalui ini, Anda bisa melihat sekilas barang-barang yang termasuk dalam daftar.



```

Output - ListingArray (run) X
run:
item one
item two
item three
8
Whole list = [item one, item three, 8]
Position 1 = item three
BUILD SUCCESSFUL (total time: 1 second)

```

Gambar 5.16 Output proyek listingarray

Ingat, ArrayList berguna jika Anda tidak yakin dengan jumlah elemen yang harus disimpan dalam daftar item.

BAB 6

METODE STRING JAVA

Tidak seperti variabel ganda dan variabel int, string adalah objek, sehingga Anda dapat melakukan hal-hal tertentu dengan string teks yang tidak dapat Anda lakukan dengan variabel ganda atau int. Prinsip yang sama berlaku untuk tipe primitif seperti short, long, float, char, single, byte, dan boolean. Sebelum kita membahas manipulasi string teks, mari kita bahas dulu string dasar.

6.1 BAGAIMANA JAVA MENANGANI STRING

String mengacu pada urutan karakter Unicode yang disimpan dalam variabel nama. Perhatikan string di bawah ini:

String sampleText = "Ron";

Ini menginstruksikan Java untuk membuat objek string dengan tiga karakter: R, o, dan n. Dalam kumpulan karakter Unicode, nilai-nilai ini adalah U+0052, U+006F, dan U+006E. Nilai Unicode ditangani sebagai heksadesimal. Di bagian sebelumnya, kami mengerjakan larik yang menampung string teks yang telah kami urutkan:

```
package arraystrings;

import java.util.Arrays;

public class stringsarry {

    public static void main(String[] args) {

        String[] stringArray = new String [5] ;

        stringArray[0] = "What";
        stringArray[1] = "does";
        stringArray[2] = "the";
        stringArray[3] = "fox";
        stringArray[4] = "say";

        Arrays.sort(stringArray);

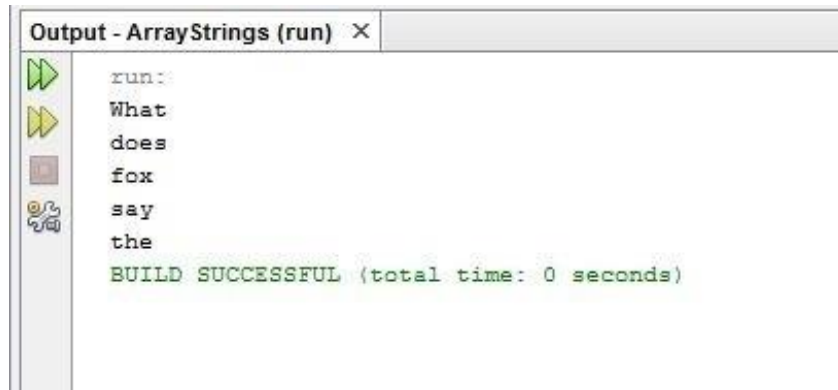
        int i;
        for (i=0; i < stringArray.length; i++) {
            System.out.println( stringArray[i] );
        }

    }

}
```

Gambar 6.1 Kode string array karakter unicode

Setelah Anda menjalankan program, layar output akan menjadi ini:



Gambar 6.2 Output kode stringarray karakter unicode

Kami telah memperhatikan bahwa kata "What" terdaftar terlebih dahulu. Jika array disusun menurut abjad, kata "does" harus didahulukan. Namun, "d" memiliki nilai heksadesimal `u\0064`, yang merupakan angka desimal 100. Huruf besar "W" memiliki nilai heksadesimal `u\0057`, yang merupakan angka desimal 87. 87 kurang dari 100, jadi "W" akan dicantumkan terlebih dahulu. Sekarang, saatnya untuk memanipulasi beberapa teks.

Huruf besar dan kecil

Mengubah teks string Java menjadi huruf besar atau kecil cukup mudah. Anda dapat menggunakan metode yang sudah ada sebelumnya `toUpperCase` dan `toLowerCase`. Memulai proyek baru dan ketik kode berikut:

```
package stringmanipulation;

public class projectstrings {

    public static void main(String[] args) {

        String caseChange = "object to change";
        System.out.println( caseChange );

        String result;
        result = caseChange.toUpperCase();

        System.out.println( result );

    }

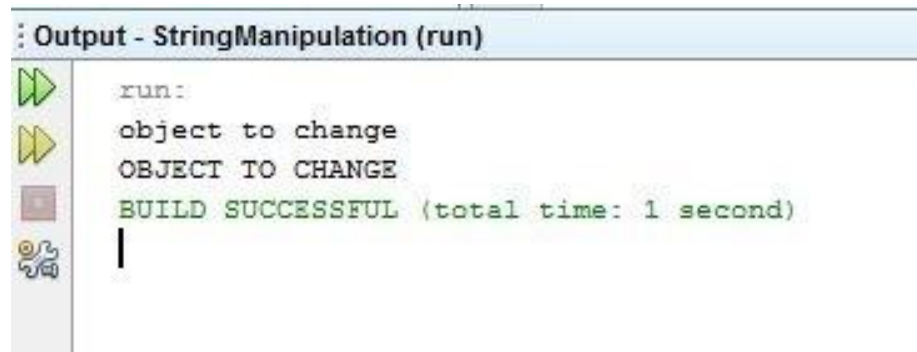
}
```

Gambar 6.2 Kode mengubah huruf kecil ke huruf besar

Fungsi dari dua baris pertama kode ini adalah untuk mengatur String variabel untuk menyimpan teks "object to change", dan kemudian mencetaknya. Baris berikutnya menetapkan String variabel lain yang dikenal sebagai hasil. Kemudian, kami memiliki baris lain untuk membantu kami dalam konversi:

result = caseChange.toUpperCase();

Untuk menggunakan metode string, pertama-tama kita perlu mengkodekan string yang ingin kita kerjakan. Dalam hal ini, kami menggunakan string variabel yang kami beri nama `caseChange`. Sertakan titik setelah variabel nama dan daftar metode yang tersedia akan muncul, yang dapat Anda gunakan pada string. Pilih `toUpperCase`. Ingat, metode ini masih membutuhkan tanda kurung kurawal kosong. Setelah Java mengonversi kata menjadi teks huruf besar, kami akan menyimpan string baru dalam string variabel. Setelah Anda menjalankan program, layar Output akan seperti ini:



Gambar 6.3 Output Konversi kata menjadi huruf besar

Namun, kita tidak perlu menahan teks yang diubah ke variabel lain. Kode di bawah ini hanya akan berfungsi dengan baik:

```
System.out.println( caseChange.toUpperCase() );
```

Pada titik ini, Java akan melanjutkan dengan konversi string tanpa perlu menetapkan hasilnya ke variabel lain. Jika Anda perlu mengubah ke huruf kecil, Anda dapat menggunakan metode **`toLowerCase`**. Anda dapat menggunakan ini dengan cara yang persis sama seperti **`toUpperCase`**.

Perbandingan String

Dimungkinkan untuk melakukan perbandingan string. Sebagai perbandingan, Java akan menggunakan nilai heksadesimal sebagai ganti huruf. Misalnya, jika Anda ingin membandingkan istilah Map dan Man untuk mengetahui mana yang akan dicantumkan terlebih dahulu, Anda dapat menggunakan metode string yang dikenal sebagai `compareTo`. Periksa cara kerjanya. Tidak perlu memulai proyek. Anda tinggal menghapus atau mengomentari kode yang telah kami tulis. Kemudian, ketikkan kode di bawah ini:

```
package stringmanipulation;

public class projectstrings {

    public static void main(String[] args) {

        int result;
        String Word1 = "Map";
        String Word2 = "Man";

        result = Word1.compareTo(Word2);
    }
}
```



```

    if (result < 0) {
        System.out.println("Word1 is lower than Word2");
    }
    else if (result > 0) {
        System.out.println("Word1 is more than Word2");
    }
    else if (result == 0) {
        System.out.println("The same word ");
    }
}
}

```

Gambar 6.4 Kode perbandingan string

Kami telah menetapkan dua string variabel untuk menampung kata-kata "Peta" dan "Lap". Metode `compareTo` adalah kode di bawah ini:

result = Word1.compareTo(Word2);

Metode `compareTo` mengembalikan nilai. Nilai yang dikembalikan akan lebih tinggi dari 0, lebih rendah dari 0, atau nol sebagai nilainya. Ketika `Word1` terdaftar sebelum `Word2`, nilai yang akan dikembalikan lebih rendah dari nol. Ketika `Word1` terdaftar setelah `Word2`, maka nilai yang akan dikembalikan lebih tinggi dari nol. Jika kata-katanya sama, maka nilai pengembaliannya akan menjadi nol. Oleh karena itu, penting untuk menentukan nilai yang dikembalikan oleh metode `compareTo` ke variabel. Kita dapat menetapkan nilai untuk variabel `int` yang dikenal sebagai hasil. Dalam kode ini, Pernyataan IF pada dasarnya memeriksa hasil variabel.

Tetapi jika kita membandingkan string teks dengan yang lain, Java akan membandingkan nilai heksadesimalnya alih-alih huruf sebenarnya. Karena huruf besar memiliki nilai heksadesimal yang lebih rendah dibandingkan dengan huruf kecil, huruf besar "M" di "Peta" akan dicantumkan sebelum huruf kecil "m" di man. Anda dapat memeriksanya. Ubah "Peta" menjadi "peta" dalam kode Anda. Anda akan mendapatkan output yang menunjukkan `Word1` lebih tinggi dari `Word2`, yang berarti bahwa Java akan mencantumkan `peta` setelah kata `man` dalam urutan abjad. Untuk mengatasi masalah ini, kita dapat menggunakan metode serupa yang dikenal sebagai **`compareToIgnoreCase`**. Seperti yang Anda duga, program akan mengabaikan huruf besar dan huruf kecil. Dengan menggunakan ini, "manusia" akan didahulukan.

6.2 METODE INDEXOF

Untuk menemukan string atau karakter di dalam string lain, kita bisa menggunakan metode `indexOf`. Misalnya, kita dapat menggunakan ini untuk memeriksa apakah ada tanda (@) di alamat email yang diberikan. Kita dapat menggunakan contoh ini dalam sebuah kode. Perhatikan bahwa Anda dapat mengomentari atau menghapus kode yang telah kami tulis. Di bawah ini adalah kode yang dapat Anda coba:


```

package stringmanipulation;

public class projectstrings {

    public static void main(String[] args) {

        char ampersand = '@';
        String email_address = "harrypotter@hogwarts.com";

        int result;
        result = email_address.indexOf( ampersand );

        System.out.println( result );

    }

}

```

Gambar 6.5 Penggunaan kode indexOf untuk menemukan string atau karakter di dalam string lain

Kita perlu menguji apakah tanda (@) ada di email add. Oleh karena itu, pertama-tama kita perlu menetapkan karakter variabel dan mendefinisikan nilai @. Perhatikan bahwa kita telah menggunakan sepasang tanda kutip tunggal dalam variabel char. Setelah menentukan tambahan email, kami telah menyertakan hasil variabel, yang merupakan variabel int. Anda mungkin bertanya-tanya kode tersebut menghasilkan bilangan bulat. Metode indexOf akan memberikan nilai. Ini dapat mengembalikan posisi nomor karakter ampersand dalam string alamat_email. Di bawah ini adalah kode terkait:

result = email_address.indexOf(ampersand);

String teks yang kami coba temukan akan terdaftar terlebih dahulu. Di sebelah titik, kami memiliki metode indexOf. Di dalam kurung kurawal indexOf, kita dapat menambahkan beberapa opsi seperti mengetik simbol atau variabel char. Dalam contoh ini, kita menempatkan variabel ampersand di dalam tanda kurung bulat dari indexOf. Kemudian, Java akan memposisikan tanda @ di add email dan akan menyimpan nilai dalam hasil variabel. Setelah kode ini dijalankan, hasilnya akan menjadi 11. Anda mungkin berpikir bahwa simbol @ adalah karakter ke-12 dari string e-mail add.

Namun, indexOf dimulai dari nol. Saat karakter tidak berada di dalam string yang Anda cari, metode indexOf akan menampilkan hasil negatif (-1). Untuk memeriksa ini, hapus tanda @ dari string email_address. Kemudian jalankan lagi kodenya. Hasilnya harus negatif satu (-1). Kita dapat menggunakan nilai negatif 1 untuk keuntungan kita. Di bawah ini adalah kodenya, tetapi kali ini, kami memiliki pernyataan IF, yang menganalisis nilai dalam hasil variabel.

```

package stringmanipulation;

public class projectstrings {

    public static void main(String[] args) {

        char ampersand = '@';
        String email_address = "hogwarts.com";

        int result;
        result = email_address.indexOf( ampersand );

        if (result== -1) {
            System.out.println( "Email Add is Invalid");
        }
        else {
            System.out.println( "Email Add is Fine");
        }
    }
}

```

Gambar 6.6 Metode indexOf akan menampilkan hasil negatif (-1)

Oleh karena itu, hasil indexOf adalah negatif 1. Else akan mengizinkan pengguna untuk melanjutkan. Jika Anda ingin memeriksa lebih banyak karakter, kami juga dapat menggunakan indexOf. Di bawah ini adalah kode untuk memeriksa add e-mail jika diakhiri dengan .com.

```

package stringmanipulation;

public class projectstrings {

    public static void main(String[] args) {

        String dotCom = ".com";
        String email_address = "hogwarts.com";

        int result;
        result = email_address.indexOf( dotCom );

        if (result== -1) {
            System.out.println( "Email Add is Invalid");
        }
        else {
            System.out.println( "Email Add is Fine");
        }
    }
}

```

Gambar 6.7 Kode memeriksa add e-mail

Dalam kode ini, kami menggunakan variabel String untuk menyimpan teks yang ingin kami uji (.com) dan bukan variabel char. Ingat, Anda akan mendapatkan hasil -1 jika objek yang Anda cari tidak terletak di String, yang muncul sebelum periode indexOf. Atau, indexOf akan menghasilkan posisi di karakter pertama yang cocok. Dalam kode di atas, titik adalah karakter ke-8 dari "hogwarts.com" jika Anda mulai menghitung dari nol. Kami juga dapat menentukan posisi awal untuk pencarian. Pada contoh di atas, kita dapat mulai mencari .com di sebelah tanda at. Di bawah ini adalah kode:

```
result = email_address.indexOf( dotCom, atPos );
```

Perbedaannya adalah dimasukkannya variabel tambahan dalam kurung indexOf. Kami tidak mendapatkan string yang ingin kami cari, yang mengacu pada teks yang ada di dalam variabel dotcom. Namun, kami sekarang memiliki posisi pencarian awal. Ini mengacu pada nilai variabel bernama atPos. Kami memperoleh nilai atPos melalui indexOf untuk menemukan posisi tanda at di add email. Java dapat memulai pencarian yang berasal dari posisi, alih-alih mulai dari nol yang merupakan pengaturan default.

StartWith...EndWith

Dalam program di atas, kita dapat menggunakan metode endWith:

```
Boolean ending = email_address.endsWith( dotcom );
```

Kita dapat mendefinisikan var Boolean untuk endWith, karena metode ini akan mengembalikan jawaban benar-salah. String yang perlu kita periksa berada di dalam tanda kurung kurawal dari endWith. Oleh karena itu, objek yang kita cari akan mendahuluinya. Ketika karakter berada di dalam pencarian string, maka nilai sebenarnya akan dikembalikan, atau bisa juga salah. Jadi, Anda dapat menyertakan pernyataan if...else untuk pemeriksaan nilai.

```
if (ending == false ) {  
System.out.println( "Email Add is Invalid" );  
}  
else {  
System.out.println( "Email Add is Fine " );  
}
```

Metode startWith dapat digunakan dengan cara yang sama:

```
Boolean startVal = email_address.startsWith( dotcom );
```

Ingat, nilai yang dikembalikan adalah Boolean true-false.

6.3 METODE SUBSTRING

Substring adalah metode lain yang berguna dalam pemrograman Java. Metode ini akan memungkinkan Anda mendapatkan bagian teks dari string lain. Dalam program sebelumnya, kita dapat memperoleh lima huruf atau tanda terakhir dari alamat ini dan memeriksa apakah ini co.us. Untuk mencoba substring, kita dapat membuat program Name Swapper sederhana.

Untuk ini, kita dapat mengubah 2 karakter pertama dari nama keluarga dan mengubahnya dengan 2 huruf pertama dari nama depan, dan sebaliknya. Katakanlah kita memiliki nama: "Harry Potter"

Kami kemudian akan mengubah "Po" dari "Potter" dengan "Ha" dari "Harry" untuk mendapatkan "Hatter. "Ha" dari "Harry" bisa ditukar dengan "Po" dari "Potter" untuk mendapatkan "Porry". Hasil print akan menjadi "Porry Hatter".

Kita dapat menggunakan substring seperti di bawah ini:

```
String FullName = "Harry Potter";
String FirstNameChars = "";
FirstNameChars = FullName.substring( 0, 2 );
```

Kita dapat membuat string yang akan terlihat, dalam contoh ini string "Harry Potter". String yang kita coba cari akan ditulis setelah simbol sama dengan. Di sebelah titik, tulis nama metode substring. Kami memiliki dua metode dalam menggunakan substring, dan perbedaannya adalah angka yang menempati dalam tanda kurung bulat. Dalam kode ini, ada dua angka: nol dan 2. Kami memberi tahu Java untuk mendapatkan karakter yang menempati pos 1 dalam string dan menghentikan panen jika kami telah mengumpulkan dua. Kedua karakter tersebut kemudian akan dikembalikan dan diposisikan dalam variabel FirstNameChars. Jika Anda ingin pergi ke kanan ke ujung string, kami dapat menyertakan baris ini:

```
String test = FullName.substring( 2 );
```

Pada titik ini, kami hanya memiliki satu angka di dalam kurung kurawal substring. Jadi, Java dapat dimulai dari karakter 2 dalam string FirstName, dan kemudian mendapatkan karakter dari pos 2 di sebelah akhir string. Untuk memeriksa ini, mulai proyek baru dan pada akhirnya, sertakan garis cetak. Substring metode akan memungkinkan Anda untuk mendapatkan dua karakter pertama dari nama "Harry". Untuk mendapatkan karakter pertama, 0 dan 2 harus ditetapkan di dalam tanda kurung bulat dari substring. Anda mungkin berpikir bahwa untuk mendapatkan "Po" dari "Potter" kita bisa memasukkan baris ini:

```
= FullName.substring(5, 2);
```

Bagaimanapun, kami bermaksud untuk mendapatkan dua karakter. Tetapi pada titik ini, 5 akan menginstruksikan Java untuk memulai dari "P" dari "Potter". Perhatikan bahwa posisi 1 dalam string adalah nol dan bukan 1. Oleh karena itu, mulai dari pos 5 dalam string & dapatkan dua karakter. Tetapi menjalankan kode ini akan mengembalikan kesalahan. Ini karena angka ke-2 dalam kurung kurawal substring yang dibulatkan tidak akan menunjukkan jumlah karakter yang ingin kita peroleh. Ini menandakan posisi string, yang ingin kita hentikan. Dalam menetapkan 2, kami menginstruksikan Java untuk mengakhiri pada karakter yang terletak di posisi 2 dari string. Karena kita tidak bisa bergerak dari pos 6 secara terbalik menuju posisi 2, maka akan muncul pesan error. Perhatikan bahwa jika kita mulai menghitung dari nol dalam string "Harry" Anda mungkin berpikir bahwa pos 2 adalah huruf "r". Anda benar.

Namun, substring dimulai sebelum karakter pada posisi ini dan bukan di sebelahnya. Oleh karena itu, untuk mendapatkan "Po" dari "Potter", kita dapat melakukan kode ini:

```
FullName.substring( 5, FullName.length() - 3 );
```

Sekarang, angka ke-2 mengacu pada panjang string, yang untuk contoh ini adalah -3 karakter. Panjang string mengacu pada jumlah karakter yang dimiliki teks. "Harry Potter" memiliki 12 karakter, termasuk spasi. Hilangkan 3 dan kami memiliki 9. Oleh karena itu, kami menginstruksikan substring untuk memulai pada char 5 dan berakhir pada char 7. Kita juga harus memperhatikan posisi spasi di antara nama-nama tersebut. Dua karakter yang ingin kita peroleh dari nama ke-2 akan ditempatkan di sebelah ruang karakter. Kami membutuhkan beberapa kode, yang memperoleh dua karakter pertama ini di sebelah spasi.

Anda dapat menggunakan indexOf untuk mencatat posisi spasi:

```
int spacePos = FullName.indexOf(" ");
```

Untuk menentukan spasi karakter, kita dapat menyertakan spasi di antara tanda kutip ganda. Ini kemudian akan masuk ke dalam kurung kurawal indexOf. Nilai yang akan ditampilkan bisa berupa bilangan bulat, dan ini adalah lokasi kemunculan pertama ruang karakter dalam string Nama Lengkap. Periksa dengan memasukkan kode di atas. Sertakan cetakan garis untuk menguji Output. Dalam program ini, spasi terletak pada posisi 5 pada string. Anda dapat menggunakan data ini untuk mendapatkan 2 karakter pertama "Potter". Anda perlu menginstruksikan Java untuk dijalankan dari karakter pertama di sebelah spasi dan berhenti pada 2 karakter berikutnya:

```
FullName.substring( spacePos + 1, (spacePos + 1) + 2)
```

Oleh karena itu, 2 angka yang ditetapkan dalam tanda kurung bulat sebenarnya adalah substring dari kode ini:

```
spacePos + 1, (spacePos + 1) + 2
```

Maksudnya adalah untuk memulai pada karakter pertama setelah spasi (spasi +1), dan mengakhiri 2 karakter setelah posisi – (spacePos+1)+2. Sertakan baris dalam kode. Substring metode baru akan tumpah ke dua baris berikutnya, tetapi jika Anda suka, Anda dapat menyimpan kode Anda sendiri. Sekarang kita memiliki "Ha" dari Harry dan "Po" dari Potter. Kita hanya perlu mendapatkan sisa teks dari 2 nama tersebut kemudian melakukan swapping. Ingat, Anda dapat menggunakan substring untuk mendapatkan sisa karakter dari nama pertama:

```
String OtherFirstChars = "";
```

```
OtherFirstChars = FullName.substring( 2, spacePos );
```

```
System.out.println( OtherFirstChars );
```

Serta sisa karakter yang berasal dari nama ke-2:

```
String OtherSurNameChars = "";
```

```
OtherSurNameChars = FullName.substring((spacePos + 1) + 2,
```

```
FullName.length() );
```

System.out.println(OtherSurNameChars);

Kami tidak mencari angka yang dibatasi dalam tanda kurung bulat dari substring. Untuk mendapatkan karakter nama pertama, Anda dapat menggunakan angka:

2, spacePos

Ini memberitahu Java untuk memulai di pos 2 & melanjutkan ke posisi spasi. Tetapi ketika sampai pada karakter yang tersisa dari nama ke-2, itu bisa menjadi kode yang rumit:

(spasiPos + 1) + 2, FullName.length()

Perhatikan bahwa (spasiPos+1)+2 menandakan posisi awal dari karakter ke-3 dari nama ke-2. Kita dapat mengakhiri panjang string yang memungkinkan kita untuk menjalankan karakter yang tersisa. Anda dapat menghilangkan cetakan garis dan mengizinkan pengguna mengetikkan nama_depan dan nama_keluarga. Dalam kode baru ini, kami telah menambahkan input dari keyboard yang telah Anda pelajari di bab sebelumnya. Tentu saja, kita perlu menyertakan beberapa baris untuk menjalankan beberapa pemeriksaan kesalahan. Namun, kami akan menganggap pengguna akan mengetikkan nama_depan serta nama keluarga dengan spasi di antara nama-nama tersebut. Tanpa spasi, hasilnya akan error.

6.4 METODE EQUALS

Metode yang dikenal sebagai Equals akan memungkinkan Anda menguji string jika serupa. Berikut adalah kode yang dapat Anda coba:

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

    String email_address1 = "meme@me.cob";
    String email_address2 = "meme@me.com";
    Boolean isMatch = false;

    isMatch = email_address1.equals(email_address2);

    if (isMatch == true) {
        System.out.println( "Email Address Match " );
    }
    else {
        System.out.println("Email addresses don't match");
    }
}
```

Gambar 6.8 Metode equals

Kode ini akan memungkinkan kita untuk menguji apakah sebuah email add mirip dengan email add lainnya. 2 baris kode pertama membentuk 2 string variabel untuk setiap penambahan email. Baris kode berikutnya membuat variabel Boolean. Metode equals akan menghasilkan nilai true | Salah. Baris 4 menandakan fungsi metode:

isMatch = email_address1.equals(email_address2);

Di dalam kurung kurawal dari metode equals, kita akan meletakkan string yang kita periksa. String kedua akan pergi sebelum metode sama. Java selanjutnya akan menampilkan true | false jika kedua string ini sama. Yang diperiksa adalah pernyataan IF. Namun, metode equals hanya dapat membandingkan objek. Itu dapat menguji string, karena mereka adalah objek. Namun, tidak mungkin menggunakan metode untuk membandingkan variabel int. Sebagai contoh, kode ini akan menghasilkan pesan kesalahan:

```
int num1 = 10;  
int num2 = 11  
Boolean isMatch = false;  
isMatch = num1.equals(num2);
```

Ingat, variabel int bukanlah objek karena merupakan tipe data primitif. Namun, dimungkinkan untuk mengonversi tipe int primitif data di dalam objek melalui kode ini:

```
int num1 = 10;  
Integer num_1 = new Integer(num1);
```

Dalam kode ini, variabel int yang dikenal sebagai num1 akan diubah menjadi objek Integer. Perhatikan bahwa kami telah menggunakan kata kunci. Di dalam kurung bulat untuk Integer, kita akan menempatkan tipe data primitif int yang ingin kita ubah menjadi objek.

6.5 METODE CHARAT

Anda dapat menguji untuk melihat karakter apa yang terkandung dalam string tertentu. Di Java, kami menggunakan metode charAt. Di bawah ini adalah kode yang dapat Anda kerjakan:

```
String email_address = "albus@hogwarts.com";  
char aChar = email_address.charAt( 5 );  
System.out.println( aChar );
```

Codeline ini akan memeriksa huruf seperti pada posisi 5 pada string e-mail add. Nilai yang akan dikembalikan adalah variabel tipe char.

```
char aChar = email_address.charAt( 5 );
```

Ketika Anda menjalankan kode di atas, output akan menjadi tanda (@). Angka di dalam kurung kurawal charAt adalah posisi string yang kami coba uji. Dalam codeline ini, kita perlu mencari char di pos 5 dari string: email_address. Ingat, hitungan dimulai dari 0 mirip dengan substring. Kegunaan hebat lainnya untuk charAt adalah untuk mendapatkan karakter dari string variabel, yang disediakan oleh pengguna, & akan mengubahnya menjadi satu karakter variabel. Misalnya, kita dapat meminta pengguna untuk memasukkan Y untuk melanjutkan atau N untuk menutup. Tidak mungkin untuk secara langsung menggunakan Pemindai kelas untuk mendapatkan satu karakter untuk menampung karakter variabel. Oleh karena itu, kita dapat menggunakan metode next() untuk mendapatkan string berikut, yang dapat dimasukkan oleh pengguna. Kemudian kita memiliki bilangan bulat berikutnya, lalu panjang, lalu ganda, dan bahkan Boolean. Namun, tidak ada karakter selanjutnya. Ini adalah kasus

bahkan jika pengguna mengetik char, itu bisa dibaca sebagai string, bukan sebagai char. Perhatikan bahwa variabel char menyimpan nomor Unicode dalam bentuk integer. Anda sekarang dapat menggunakan charAt untuk mendapatkan karakter yang berasal dari string, yang akan dimasukkan pengguna meskipun pengguna mengetikkan huruf:

```
char aChar = aString.charAt( 1 );
```

Dalam kode ini, kami memberi tahu Java untuk mendapatkan char di pos 0 dalam string yang dikenal sebagai aString, lalu menahannya menggunakan variabel aChar. Kami juga menyertakan pernyataan IF untuk memeriksa isi variabel aChar. Perhatikan penggunaan sepasang tanda kutip tunggal untuk membatasi Y.

6.6 METODE REPLACE

Di Java, metode replace digunakan untuk mengganti semua kemunculan karakter dalam string tertentu. Pertimbangkan kalimat ini:

"Di mana tongkatmu?"

Kita perlu mengganti "Anda" dengan "Anda".

Ada beberapa cara berbeda untuk menggunakan metode replace, dan perbedaannya terutama terletak pada apa yang Anda tempatkan di dalam kurung bulat metode tersebut. Kami mengganti satu seri karakter dengan yang lain. Pertimbangkan koma yang memisahkan keduanya sebagai istilah dengan. Kemudian, Anda akan memiliki "ganti Anda dengan Anda". Kami juga dapat mengganti satu karakter:

```
aString.replace('$', '%')
```

Kode di atas berarti "Ganti \$ dengan %". Kami juga dapat menggunakan ekspresi normal dalam metode ganti, tetapi itu di luar cakupan ebook ini.

6.7 METODE TRIM

Kita bisa memangkas ruang putih dari string. Ruang putih mengacu pada karakter spasi, karakter baris baru, dan tab. Sangat mudah untuk menggunakan metode trim:

```
String amend = " blank space ";
```

```
amend = amend.trim( );
```

Oleh karena itu, metode trim akan mengikuti string yang ingin kita ubah. Spasi kosong sebelum kata "kosong" dan setelah "spasi" pada kode di atas harus dihilangkan. Jika kita mendapatkan input pengguna, maka sangat ideal untuk menggunakan metode trim pada mereka.

BAB 7

METODE JAVA

Dalam bab-bab sebelumnya, kita telah menggunakan metode java, dan pada titik ini, Anda harus tahu bahwa metode yang sudah ada sangat berguna. Sekarang saatnya Anda belajar bagaimana menulis metode Java Anda sendiri.

7.1 FORMAT METODE JAVA

Metode hanyalah bagian dari kode, yang melakukan tugas tertentu. Namun, metode terstruktur dalam format tertentu. Muncul dengan header metode dan badan. Header akan menandakan Java jenis nilai seperti string, double, atau int. Ini juga menyertakan tipe pengembalian dengan nama metode yang juga disertakan dalam header. Anda dapat menentukan nilai di atas metode, dan nilai tersebut dapat ditulis di dalam tanda kurung bulat. Tubuh metode adalah tempat kode akan dijalankan.

```
int total( int aNumber) {  
  
    int a_Value = aNumber + 10;  
  
    return a_Value;  
  
}
```

Tipe pengembalian metode Java harus didahulukan, yang merupakan tipe int pada contoh di atas. Di sebelah metode ini adalah spasi diikuti dengan nama metode. Dalam contoh, kami menamakannya total. Di dalam kurung bulat, kami telah menginstruksikan Java bahwa kami menyimpan metode variabel yang dikenal sebagai aNumber, dan ini bisa berupa bilangan bulat. Untuk memisahkan metode ini dari kode lain, Anda harus menggunakan tanda kurung kurawal. Kode untuk metode harus dibatasi dalam kurung melengkung. Catat pengembalian kata kunci dalam metode. Ini jelas merupakan nilai, yang Anda sukai sebagai pengembalian metode setelah Anda menjalankan kode. Namun, itu harus tipe yang sama seperti tipe pengembalian dalam metode header. Oleh karena itu, nilai yang dikembalikan tidak dapat berupa string jika Anda memulai metode dengan total int. Ada beberapa contoh di mana Anda tidak suka Java mengembalikan nilai sama sekali. Sebuah metode yang tidak akan memberikan nilai apapun dapat dibuat dengan menggunakan kata void. Dalam hal ini, metode tidak memerlukan pengembalian kata kunci. Di bawah ini adalah contoh metode, yang tidak mengembalikan nilai.

```
void print_text(String someText) {  
  
    System.out.println( "Some Text Here" );  
  
}
```

Metode di atas dapat mencetak beberapa pesan. Itu masih bisa menjalankan fungsinya bahkan tanpa nilai jadi kami telah menetakannya sebagai metode batal. Ingat, metode tidak

perlu diteruskan dengan nilai. Kami hanya dapat menjalankan beberapa kode. Berikut adalah metode batal lain tanpa nilai.

```
void print_text() {
    System.out.println( "Some Text Here" );
}
```

Berikut adalah metode int lain tanpa nilai yang diteruskan:

```
int total() {
    int a_Value = 10 + 10;
    return a_Value;
}
```

Perhatikan bahwa tanda kurung bulat tidak memiliki nilai apa pun, namun tetap penting. Tanpa tanda kurung bulat, kode akan mengalami kesalahan.

7.2 CARA MEMANGGIL METODE JAVA ANDA

Perhatikan bahwa metode tidak dapat melakukan apa pun kecuali Anda memanggilnya ke dalam tindakan. Sebelum kita mengujinya, tambahkan kelas baru ke proyek, jadi kita bisa menempatkan semua metode daripada mengacaukan kelas utama. Memulai proyek aplikasi baru. Seperti biasa, berikan nama proyek dan ubah nama metode utama menjadi yang lain. Dalam kode di bawah ini, kami telah memanggil proyek **projectmethods** kami, dan kelas **MethodTests**.

```
package projectmethods;

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

Gambar 7.1 Cara memanggil metode java

Untuk menyisipkan kelas baru ke proyek, pilih File Baru dari menu File di NetBeans. Sebuah kotak dialog akan muncul. Di bagian Kategori, pilih Java, dan di bagian Jenis File, pilih Kelas Java. Di bagian bawah, klik tombol Berikutnya. Pada langkah kedua, ketikkan judul untuk kelas baru. Kami akan memanggil jenis SampleMethods kami. Oleh karena itu, kami membuat kelas kedua yang dikenal sebagai SampleMethods yang akan ada dalam metode proyek Project. Cukup klik tombol Finish dan program akan membuat file kelas baru. Tab lain akan muncul dengan komentar default tentang bagaimana Anda dapat mengubah template. Hapus saja komentarnya.

Anda akan melihat bahwa tidak ada metode Main dalam kode ini, tetapi kelas kosong dengan nama kelas yang telah Anda pilih serta sepasang tanda kurung kurawal untuk kode tersebut. Kita bisa menambahkan metode lain. Ini adalah metode `int` yang telah Anda kerjakan sebelumnya di nama `total`. Itu tidak memiliki apa pun di dalam tanda kurung bulat sehingga kami tidak menyimpan nilai apa pun. Metode ini hanya menambahkan nilai `5 + 5` dan menangani jawabannya dalam variabel yang dikenal sebagai `a_Value`. Metode akan mengembalikan nilai ini, yang harus cocok dengan jenis pengembalian dari metode header. Kode ini baik-baik saja karena nilai-nilai ini adalah tipe `int`.

Perhatikan bahwa variabel `a_Value` tidak boleh terlihat di luar metode `total`. Kami tidak dapat mengatur di dalam metode yang tidak dapat kami akses di luar metode. Ini disebut variabel lokal, yang lokal ke metode. Untuk memanggil metode `total`, pilih tab `TestMethods` di `NetBeans`, yang merupakan metode utama. Kami akan memanggil metode `total` dari metode Utama. Pertama-tama kita perlu membuat objek baru dari kelas `SampleMethods`. Untuk membuat objek baru dari kelas, kita akan mulai dengan nama kelas, `SampleMethods` dalam contoh ini. Ini menggantikan variabel `String`, `double`, atau `int`. Jenis variabel yang kita buat adalah variabel `SampleMethods`. Di sebelah spasi, kami menambahkan judul untuk variabel `SampleMethods` baru. Nama variabel kami adalah `test1`. Perhatikan bahwa itu digarisbawahi dalam warna abu-abu karena kita belum menambahkan perintah apa pun.

Ini akan diikuti oleh simbol sama dengan dan kemudian kata kunci: `baru`, yang mengacu pada objek baru. Di sebelah kata kunci ini, tambahkan spasi diikuti dengan nama kelas. Ingat, nama kelas membutuhkan tanda kurung bulat. Baris harus diakhiri seperti biasa dengan titik koma. Dengan menggunakan kode ini, kita telah membuat objek `SampleMethods` baru menggunakan nama `test1`. Metode `total` dalam kelas `SampleMethods` sekarang akan dapat diakses menggunakan metode Utama dari kelas `SampleMethods`. Kami mendefinisikan variabel `int` menggunakan nama `aVal`. Di sebelah simbol sama dengan muncul nama kelas `test1`. Untuk mengakses metode kelas, masukkan titik, dan `NetBeans` dapat menampilkan kotak popup menggunakan metode yang tersedia:

Variabel `total` didefinisikan pada daftar, sedangkan variabel lainnya dibangun dalam metode. Tanda kurung bulat tidak memiliki nilai yang ditetapkan, karena metode ini tidak akan menerima nilai apa pun. Namun, tipe pengembalian `int` ditampilkan di sebelah kanan. Pilih `total` dengan mengklik dua kali untuk menambahkan ini dalam kode. Selanjutnya, tambahkan titik koma di akhir baris. Kemudian, kita akan menambahkan garis cetak. Untuk memanggil metode, yang mengembalikan nilai, perhatikan nilai yang dikembalikan oleh metode. Kemudian tentukan nilai ini ke variabel lain, yaitu `aVal` dalam contoh ini. Namun, metode harus tersedia jika Anda memasukkan titik setelah nama objek. Tetapi jika metodenya adalah tipe kosong, tidak perlu menentukannya ke variabel lain seperti `aVal`. Anda dapat beralih kembali ke kelas `SampleMethods` dan menyertakan metode `void`, yang telah Anda pelajari. `print_text` adalah metode baru. Perhatikan bahwa ia juga memiliki sepasang tanda kurung bulat yang kosong karena kami tidak bermaksud untuk menyimpan nilai apa pun. Fungsinya untuk mencetak pesan. Setelah menambahkan metode `void`, sekarang kita dapat beralih kembali ke kelas `SampleMethods`. Anda sekarang dapat menyisipkan baris di bawah ini:

```
test1.print_text()
```

Setelah menambahkan periode, kita dapat melihat metode baru pada daftar. Metode ini sekarang termasuk dalam daftar – `print_text` dan `total`. Nilai yang akan mereka kembalikan akan ditampilkan di sebelah kanan, `batal` dan `int`. Karena `print_text` adalah metode `batal`, tidak perlu menetapkan nilai kembalian. Kita hanya perlu nama objek, titik, dan metode `void` yang ingin kita panggil. Java hanya akan melanjutkan dengan menjalankan kode di dalam metode.

7.3 CARA LULUS NILAI KE METODE JAVA

Untuk melakukan sesuatu dengan nilai, kita dapat meneruskan nilai ke metode. Nilai harus ditulis dalam pasangan tanda kurung bulat dari metode. Sekarang kami memiliki dua metode menggunakan `total` sebagai namanya. Satu-satunya perbedaan adalah bahwa metode baru berisi nilai dalam tanda kurung bulat. Java akan membiarkan Anda melakukan ini, melalui metode proses `overloading`. Karenanya, Anda dapat memasukkan sebanyak mungkin metode yang Anda suka menggunakan nama yang sama yang berisi nilai apa pun. Tetapi perhatikan bahwa tidak diperbolehkan memiliki jenis variabel yang sama di dalam tanda kurung bulat. Oleh karena itu, Anda tidak dapat memiliki dua metode `total`, yang mengembalikan nilai `int` yang memiliki nilai `int` yang sama yang dibatasi dalam tanda kurung bulat. Misalnya, baris di bawah ini tidak diperbolehkan:

```
int total( int aNumber ) {
int a_Value = aNumber + 10;
return a_Value;
}
int total( int aNumber ) {
int a_Value = aNumber + 20;
return a_Value;
}
```

Meskipun kedua metode melakukan hal yang berbeda, mereka masih memiliki header metode yang serupa. Masukkan beberapa komentar terlebih dahulu sebelum mencoba metode baru.

```
/**
 * Returns an integer value, which is
 * 20 plus the number passed as a paramater.
 * @param aNumber Any Integer value
 * @return 20 + the value of aNumber
 */
int total(int aNumber) {
    int a_Value = aNumber + 20;

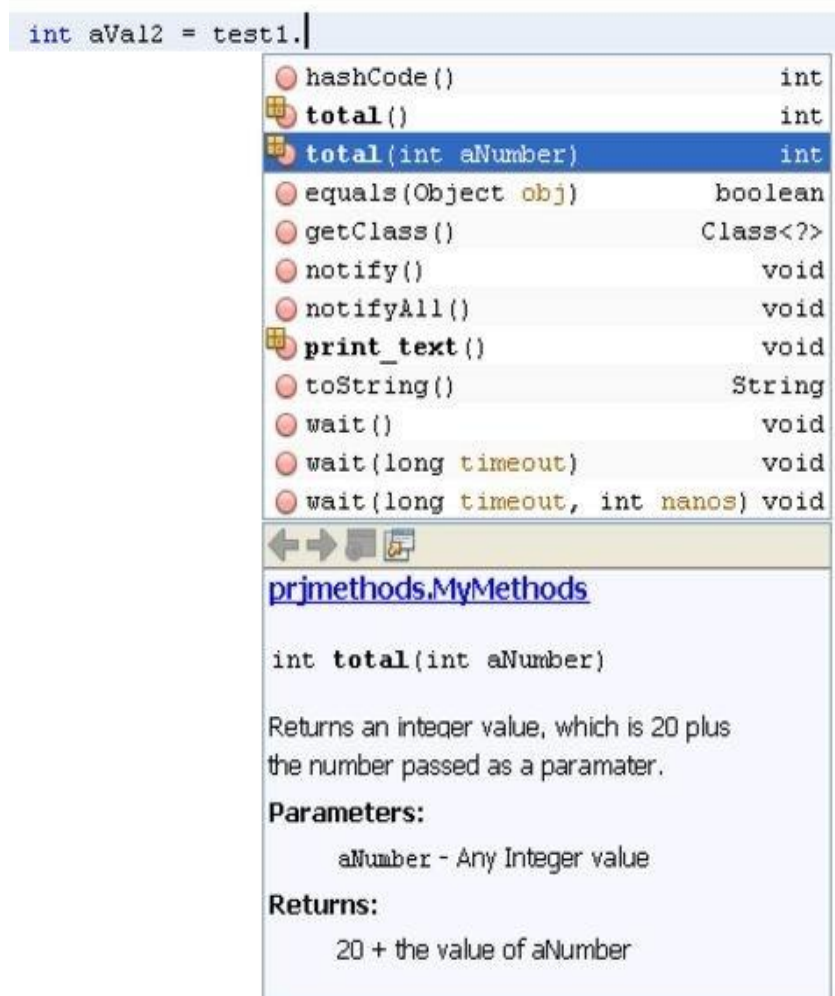
    return a_Value;
}
```

Gambar 7.2 memasukan beberapa komentar

Param dalam komentar di atas mengacu pada parameter, yang merupakan kata teknis untuk nilai dalam tanda kurung bulat dari header metode. Parameter ini dikenal sebagai aNumber dengan nilai integernya. Perhatikan bahwa kita menggunakan karakter tanda at sebelum param dan return. Dalam kode ini, kami melewati nilai integer dan menambahkan 20 ke nilai ini. Return_value akan menjadi jumlah dari keduanya. Langkah selanjutnya adalah kembali ke kode dan masukkan baris di bawah ini:

```
int aVal2 = test1.total(30);
```

Mengetik periode setelah objek test1 akan menampilkan daftar opsi Metode total akan disertakan. Pilih metode baru. Komentar yang ditambahkan sekarang dibatasi dalam kotak biru di bawah opsi metode. Setiap pengguna yang membaca metode ini dapat dengan mudah mengetahui tentang apa kode itu. Saat Anda menambahkan metode total2, masukkan angka 30 di dalam kurung bulat. Akhiri baris ini dengan titik koma.

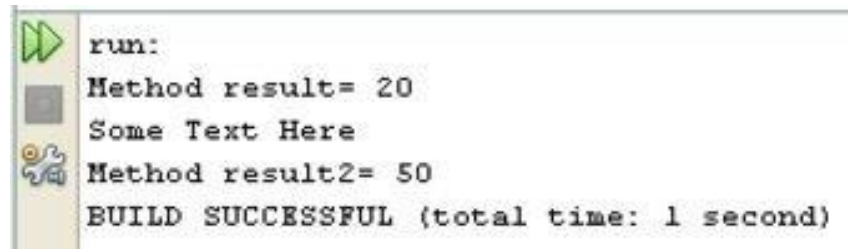


Gambar 7.3 daftar opsi metode total

Saat Anda menyerahkan nilainya, metode akan menjalankan fungsinya. Sertakan baris cetak ke kode:

```
System.out.println( "Method result2= " + aVal2 );
```

Jalankan programnya. Layar Output harus seperti ini:

A screenshot of a Java IDE's output window. On the left, there is a vertical toolbar with icons for running (a green play button), a gray square, and a blue bug icon. The output text is as follows:

```
run:
Method result= 20
Some Text Here
Method result2= 50
BUILD SUCCESSFUL (total time: 1 second)
```

Gambar 7.4 Output dari coding metode java

7.4 KESIMPULAN

Saya harap buku ini dapat membantu Anda mempelajari Pemrograman Java Dasar. Langkah selanjutnya adalah melatih keterampilan pemrograman Java Anda dan terus mempelajari keterampilan tingkat lanjut termasuk:

- Menulis Metode Java Anda Sendiri
- Menulis Kelas Java Anda Sendiri
- Menangani Kesalahan Java
- Mengelola File Teks Java
- Kontrol Formulir
- Basis Data Java

DAFTAR PUSTAKA

- Cosmina, Luliana. Java for Absolute Beginners. Springer Science + Business Media Finance Inc. Edinburgh, UK. ISBN-13 (pbk): 978-1-4842-3777-9
- Herbert Schildt, "Java: A Beginner's Guide", McGraw-Hill, 2005.
- Profile, S. E. E. (2014). New Learning Methodology for Student of Java Programming Language
Tejinder Singh.
- R.H. Sianipar, "Java: Teori, Algoritma Dan Aplikasi", Penerbit ANDI, 2014.
- Schidt, Herbert. 2019. Java A Beginner's Guide, McGraw-Hill Education.
- Sharan, Kishori. Beginning Java 9 Fundamentals. Springer Science + Business Media Finance Inc. ISBN-13 (pbk): 978-1-4842-2843-2.
- Singh, T. (2012). New learning methodology for student of Java programming language. Int. J. Eng. Res. Dev., 3(11).
- Tandika, Bima. 2021. "Apa itu Bahasa Pemrograman Java? Ini Penjelasannya". Melalui [https:// glints.com/id/lowongan/bahasa-pemrogramanjava/#.YXEPbxbW2w](https://glints.com/id/lowongan/bahasa-pemrogramanjava/#.YXEPbxbW2w) diakses tanggal 23 Juni 2022.
- Tirtobisono. Yan, 2009. Pembuatan Aplikasi Dalam Komputer Menggunakan Bahasa Pemograman. Andi Offset. Yogyakarta.
- Y. Daniel Liang, "Introduction to Java Programming", Pearson Higher Education, 2011.



Pemrograman **Java** UNTUK **Pemula**

Budi Hartono. M.Kom



Budi Hartono, S.Kom, M.Kom Lahir di Sleman pada Tanggal 19 September 1968, dan sekarang menetap di Semarang. Menyelesaikan Pendidikan Dasar hingga SLTA di Sleman, Yogyakarta. Menempuh pendidikan Sarjana dari Program Studi S1-Sistem Komputer Universitas Sains dan Teknologi Komputer (Universitas STEKOM) Lulus Tahun 2004, Program Magister diselesaikan pada Tahun 2006 pada Program Magister Teknik Informatika Universitas Dian Nuswantoro.

Saat ini penulis tercatat aktif sebagai Dosen Tetap di Universitas Sains dan Teknologi Komputer (Universitas STEKOM) pada Program Studi S1 Teknik Informatika. Untuk matakuliah yang di ampu antara lain adalah Logika dan Algoritma Pemrograman, Pemrograman Visual, Pemrograman Berorientasi Objek, Pengantar Bahasa Query, Sistem Pendukung Keputusan, Sistem Basis Data, Struktur Data, Analisa dan Perancangan Sistem Informasi.

Buku Pemrograman Berorientasi Objek ini adalah hasil karyanya pada tahun 2022. Adapun buku karya penulis yang sudah terbit antara lain Pemrograman WAP, Pemrograman Visual dengan Delphi, Sistem Pendukung Keputusan dan Cara Mudah dan Cepat Belajar Pengembangan Sistem Informasi.



YAYASAN PRIMA AGUS TEKNIK

PENERBIT :

YAYASAN PRIMA AGUS TEKNIK

Jl. Majapahit No. 605 Semarang

Telp. (024) 6723456. Fax. 024-6710144

Email : penerbit_ypat@stekom.ac.id

Pemrograman Java UNTUK Pemula

Budi Hartono. M.Kom



YAYASAN PRIMA AGUS TEKNIK

PENERBIT :

YAYASAN PRIMA AGUS TEKNIK

Jl. Majapahit No. 605 Semarang

Telp. (024) 6723456. Fax. 024-6710144

Email : penerbit_ypat@stekom.ac.id