**LAPORAN TUGAS UAS**

**PRACTICE ECLIPSE**

**PEMROGRAMAN BERORIRNTASI OBJEK**

****

Dosen:

PUTU BAGUS ADIDYANA ANUGRAH PUTRA, S.T ,M.Kom

**Disusun Oleh :**

|  |  |
| --- | --- |
| Luniko Jama | (223010503029) |
| Febrian Eka Putra | (223010503030) |
| Budi Aulyansyah Ahmad Trisna | (223020503051) |
| Bima Agung Saputra | (223020503060) |
| Thomas Zugildo Magnus | (223020503063) |

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS PALANGKA RAYA**

**KALIMANTAN TENGAH**

**2023**

## **BAB I**

## **PENDAHULUAN**

1. **Java**

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di *Sun Microsystems*, yang saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam p-code (*bytecode*) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM). Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik *(general purpose),* dan secara khusus didesain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun". Saat ini java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi.

OOP (Object Oriented Programming - Pemrogram Berorientasi Objek), Java merupakan salah satu bahasa pemrograman dengan konsep OOP. Di mana program yang dibangun berorientasikan kepada Object. Aplikasi yang dibangun dengan konsep OOP terdiri atas object-object yang saling berhubungan.

1. **Tipe Data**

Berikut ini macam-macam tipe data pada Java:

* + 1. *Char* : Tipe data karakter
    2. *Int* : angka atau bilangan bulat
    3. *Float*: bilangan decima
    4. *Double* : bilangan desimal juga, tapi lebih besar kapasistanya
    5. *String* : kumpulan dari karakter yang membentuk teks
    6. *Boolean* : tipe data yang hanya bernilai true dan false

1. **Pengertian Eclipse**

Eclipse adalah platform yang telah dirancang untuk membangun berbagai aplikasi yang dapat terintegrasi seperti website, *mobile* dan lain sebagainya. Eclipse bersifat *open source* yang biasanya digunakan untuk mengambangkan aplikasi berbasis java yang memungkinkan pengembang software untuk menciptakan lingkungan pengembangan yang disesuaikan (IDE). anda dapat mengembangkan dan mempelajari platform ini jika mengikuti kursus android dimanapun anda berada.

Eclipse bermula pada tahun 2001 oleh IBM yang pada saat itu perusahaan tersebut menyumbangkan tiga juta baris kode dari tool Java nya. Tujuan dari Eclipse adalah untuk menciptakan dan membina komunitas IDE open source yang akan melengkapi komunitas Apache.

Eclipse menyediakan antarmuka pengguna (UI) Model umum untuk bekerja dengan *tools* yang lainnya. Hal ini dirancang untuk berjalan pada beberapa sistem operasi sambil memberikan integrasi yang kuat dengan masing-masing OS yang mendasarinya. Plug-in digunakan untuk program API portabel Eclipse dan berubah pada salah satu sistem operasi yang didukung.

Inti dari Eclipse adalah sebuah arsitektur untuk penemuan dinamis, pemuatan, dan menjalankan *plug-in*. Platform ini menangani logistik untuk menemukan dan menjalankan kode yang tepat. Platform UI menyediakan model navigasi pengguna standar. Setiap plug-in kemudian dapat fokus pada melakukan sejumlah kecil tugas dengan baik.

1. **Sifat Eclipse**

Berikut ini adalah sifat dari Eclipse:

* + 1. *Multi*-*platform* : Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.
    2. *Mulit*-*language* : Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya.
    3. *Multi-role* : Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.

1. **Classes**

*Class* adalah struktur dasar dari OOP (*Object Oriented Programming*). Terdiri dari dua tipe yaitu : *field* (*attribute*/*property*) dan *method* (*behavior*). *Class* digunakan untuk mendeklarasikan sebuah variabel yang berupa objek atau dinamakan “referensi objek (*object reference*)”.

1. **Objects**

Setiap *Object*(objek) dibangun dari sekumpulan data(atribut) yang disebut "variabel" (untuk menjabarkan karakteristik khusus dari objek) dan juga terdiri dari sekumpulan *method*(menjabarkan tingkah laku dari objek) atau Objek adalah sebuah perangkat lunak yang berisi sekumpulan variabel dan *method* yang berhubungan. Objek merupakan sebuah *instance*(keturunan) dari *class*. Variabel dan *method* diketahui sebagai variabel *instance* dan *method instance*.

1. **Methods**

*Method* pada bahasa pemrograman Java adalah kumpulan baris kode yang dikelompokkan untuk menjalankan tugas tertentu. Sebuah *method* dapat mengembalikan nilai tertentu (memiliki *return value*), bisa juga tidak (*void*). *Method* dapat menerima argumen sebagai nilai masukan yang akan diproses di dalam method bersangkutan.

Method dapat digunakan(dipanggil) berulang-ulang dari mana saja dalam sebuah program. Dengan membuat *method*, tidak perlu lagi menulis kode program yang melakukan hal sama berkali-kali. Tidak kalah penting, penggunaan *method* juga membuat organisasi dan struktur program menjadi lebih baik.

1. **Parameters**

*Parameter* adalah data yang dapat ditambahkan dari luar m*ethod*, contoh : ketika membuat sebuah *method* untuk mengubah nama pada kelas Hewan, Oleh karena itu membutuhkan sebuah *parameter* baru untuk menggantikan nama tersebut. Dalam sebuah *method* dapat memiliki satu atau lebih *parameter*, dan untuk menambahkan *parameter*, pisahkan p*arameter* menggunakan koma.

1. **Scanner**

Inputasi melalui keyboard menggunakan *Scanner* membutuhkan pernyataan import berikut **import java.util.Scanner**. Untuk membaca masukan yang telah dimasukkan pengguna, gunakan objek Java *scanner*. Cara untuk menginisialisasi Scanner, tulis :

Scanner in = new Scanner(System.in);

Berikut ini adalah *method*-*method* dari *class Scanner* yang sering digunakan:

|  |  |
| --- | --- |
| **Method** | **Penjelasan** |
| next() | Mengembalikan token berikutnya  dari scanner |
| nextLine() | Memindahkan scanner ke baris  berikutnya dan mengembalikan nilai dalam string |
| nextByte() | Mengscan token berikutnya sebagai  suatu byte |
| nextShort() | Mengscan token berikutnya sebagai |
|  | suatu nilai short |
| nextInt() | Mengscan token berikutnya sebagai  suatu nilai int |
| nextLong() | Mengscan token berikutnya sebagai  suatu nilai long |
| nextFloat() | Mengscan token berikutnya sebagai  suatu nilai float |
| nextDouble() | Mengscan token berikutnya sebagai  suatu nilai double |
| hasNext() | Mengembalikan nilai true jika scanner memiliki token lainnya  dalam input |
| Close() | Untuk menutup scanner |
| Scanner useDelimiter (String pattern) | Method ini menetapkan pola pemisah  scanner ke pola yang dibangun dari String yang ditentukan. |

1. **Conditional dan Control Statement**

Statement kontrol dan kondisional digunakan untuk mengatur jalannya alur program sesuai degan yang kita inginkan. Statemen- statement ini dikategorikan kedalam tiga jenis yaitu: pemilihan, pengulangan, dan statement peloncatan.

1. Pemilihan

Pemilihan adalah suatu keadaan dimana pernyataan dapat dieksekusi apabila suatu kondisi memenuhi syarat untuk mengerjakan pernyataan tersebut.

* 1. Statement *If*

Untuk membandingkan kondisi pertama, yang berisi sebuah kondisi *boolean*, jadi bernilai true atau false. Statement *if* akan dijalankan jika kondisi bernilai *true*.

If (kondisi) {

//statement yang akan dieksekusi

}

* 1. Statement *if ... else*

Jenis pemilihan ini relatif lebih kompleks apabila dibandingan dengan *jenis* pertama diatas. Pada bentuk ini, program menyediakan kondisi tambahan untuk menangani kejadian yang kondisi pertamanya tidak terpenuhi.

If(kondisi){

//statement akan dieksekusi

}

else if(kondisi2){

//statement dieksekusi jika kondisi2 bernilai True.

}

* 1. *Switch*

Statement *Switch* adalah cara lain untuk mengubah aliran program tergantung pada nilai input. Pernyataan *switch* menggunakan 3 kata kunci: *switch*, *case*, dan *default*.

* + 1. *Switch*: menentukan variabel mana yang akan diuji nilainya.
    2. *Case*: membandingkan nilai variabel l sakelar.
    3. *Default*: ketika input tidak cocok dengan kasus manapun, kompilator memilih aksi default.

Switch(var){ case : //perintah

default : //perintah

}

1. Perulangan

Perulangan digunakan untuk melakukan eksekusi terhadap statemen secara berulang sesuai dengan kondisi yang ditentukan.

* 1. *For*

Perintah akan dijalankan secara terus-menerus selama kondisi dan inisialisasi bernilai *true*.

For(inisialisasi;kondisi;increment){

//statement yang akan diulang

}

Inisialisasi adalah tipe data dan nilainya. Kondisi adalah kondisi yang akan dibandingkan dengan inisialisasi *Increments* adalah kondisi yang dijalankan (bisa increment atau descrement).

* 1. *While*

Perulangan yang mendifinisikan kondisi diawal blok. Apabila kondisi tidak terpenuhi(bernilai *false*) maka perulangan pun tidak akan pernah dilakukan.

While(kondisi){

//statemen yang diulang ... Increment/perintah tertentu

}

* 1. *Do*…*While*

Pada struktur *while*, kondisi ditempatka diawal blok perulangan sedangkan pada struktur *do...while* kondisi berada diakhir blok.

Do{

//statemen yang diulang... Increment/perintah tertentu

}while(kondisi);

1. **Array**

*Array* adalah sebuah variabel yang bisa menyimpan banyak data dalam satu variabel. *Array* menggunakan indeks untuk memudahkan akses terhadap data yang disimpannya. Untuk mendeklarasikan sebuah *array* dapat dituliskan seperti berikut ini :

//cara pertama String[ ] nama;

// cara kedua String nama[ ];

//cara ketiga dengan kata kunci new String[ ] nama = new String[5];

Kurung siku [ ] digunakan untuk membuat *array*, kurung siku bisa diletakkan setelah tipe data atau nama *array* dan angka 5 dalam kurung artinya batas atau ukuran array-nya.

* + 1. Multidimensi *Array*

*Array* multidimensi adalah array yang berisi satu atau lebih *array*. Untuk membuat *array* dua dimensi, tambahkan setiap *array* dalam set kurungnya sendiri:

Tipedata [ ][ ] namaarray = new tipedata [n][m];

Dimana n adalah jumlah elemen untuk baris dan m adalah jumlah elemen kolom.

* + 1. Array List

*Array* list *merupakan* sebuah *class* yang memungkinkan kita membuat sebuah objek untuk menampung apapun. Contoh Program dengan Array List :

import java.util.ArrayList; public class Doraemon {

public static void main(String[] args) {

// membuat objek array list

ArrayList kantongAjaib = new ArrayList();

// Mengisi kantong ajaib dengan 5 benda

kantongAjaib.add("Senter Pembesar"); kantongAjaib.add(532); kantongAjaib.add("tikus"); kantongAjaib.add(1231234.132); kantongAjaib.add(true);

// menghapus tikus dari kantong ajaib kantongAjaib.remove("tikus");

// Menampilkan isi kantong ajaib System.out.println(kantongAjaib);

// menampilkan banyak isi kantong ajaib System.out.println("Kantong ajaib berisi "+ kantongAjaib.size() +" item");

}

}

1. **Overloading dan Overriding**

Overloadingadalah diperbolehkannya dalam sebuah *class* memiliki lebih dari satu nama *function*/*method* yang sama tetapi memiliki *parameter*/*argument* yang berbeda. Perbedaan *parameter* meliputi jumlah *parameter*, tipe data *parameter*, urutan tipe data *parameter*.

Sedangkan *Overriding* adalah kemampuan dari *subclass* untuk memodifikasi *method* dari *superclass*-nya, yaitu dengan cara menumpuk (mendefinisikan kembali) *method superclass*-nya. Hak akses pada *method overriding* tidak boleh lebih ketat dari *method* pada *superclass*.

1. **Inheritance (Pewarisan)**

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

*Subclass* tidak dapat mewarisi anggota *private* dari *superclass*-nya. Dengan *inheritance*, *class* yang baru(*subclass*) akan mirip dengan *class* yang lama(*superclass*) namun memiliki karakteristik yang baru. Dalam Java, *subclass* hanya bisa memiliki satu *superclass* (*single inheritance*) sedangkan *superclass* bisa memiliki satu *subclass* atau lebih. Untuk menerapkan *inheritance*, menggunakan statement “*extends*”. Keyword “*super*” digunakan oleh subclass untuk memanggil *constructor*, atribut dan *method* yang ada pada *superclass*-nya.

Contoh untuk memanggil *constructor* milik *superclass*-nya :

super() super(paramet er)

Contoh untuk memanggil atribut dan *method* milik *superclass*-nya :

super.namaAtribut super.namaMethod(parame ter)

1. **Polymorphism**

Polimorfisme terbagi menjadi dua suku kata yaitu, Poly yang *berarti* banyak dan Morfisme yang berarti bentuk. Dalam ilmu sains, Polimorfisme (*polymorphism*) adalah sebuah prinsip dalam biologi di mana organisme atau spesies memiliki banyak bentuk serta tahapan (*stages*). Prinsip tersebut diterapkan juga pada bahasa Java.

Polimorfisme mempunyai makna sesuatu yang memiliki banyak bentuk, yaitu memiliki nama sama, tetapi memiliki kelakuan (*behaviour*) yang berbeda. Teknik polimorfisme digunakan untuk menyatakan satu nama yang merujuk(*reference*) ke beberapa fungsi yang berbeda. Java menginterprestasikan polimorfisme dengan 3 cara yaitu *method overloading*, *pewarisan*, *method overriding* dengan *interface* java.

1. Exception Handling

Exception *Handling* merupakan mekanisme yang paling diperlukan dalam menangani *error* yang terjadi pada saat *runtime* (program berjalan) atau yang lebih dikenal dengan sebutan *runtime error*. Secara umum, adanya kesalahan/*error* yang terjadi pada program pada saat *runtime* dapat menyebabkan program berhenti atau hang. Untuk itulah diperlukan mekanisme untuk memastikan bahwa program tetap dapat berjalan meskipun terdapat kesalahan yang terjadi. Ada 5 keyword penting dalam java dalam hal *exception handling* :

1. *try*, biasanya digunakan dalam suatu blok program. Keyword ini digunakan untuk mencoba menjalankan blok program kemudian mengenai sasaran dimana munculnya kesalahan yang ingin diproses. Keyword ini juga harus dipasangkan dengan keyword *catch* atau keyword *finally*.
2. *catch*, keyword catch harus dipasangkan dengan try. Kegunaan keyword *catch* adalah untuk menangkap kesalahan atau *bug* yang terjadi dalam blok *try*. Setelah 2 menangkap kesalahan yang terjadi maka programmer dapat melakukan hal apapun pada blok *catch* sesuai keinginan programmer.
3. *finally*, merupakan keyword yang menunjukan bahwa blok program tersebut akan selalu dieksekusi meskipun adanya kesalahan yang muncul atau pun tidak ada.
4. *throw*, keyword ini digunakan untuk melemparkan suatu *bug* yang dibuat secara manual
5. *throws*, digunakan dalam suatu method atau kelas yang mungkin menghasilkan suatu kesalahan sehingga perlu ditangkap *error*-nya. Cara mendefinisikannya dalam *method* adalah sebagai berikut :

<method modifier> type method-name throws exception-list1, exceptio-list2, … {}.

Beberapa exception yang sering terjadi pada Bahasa pemrograman Java di antaranya adalah:

1. *Arithmetic Exception*

Untuk perhitungan aritmatika, misalkan sebuah angka dibagi 0.

1. *Null Pointer Exception*

Bila *object* yang ingin dia akses masih NULL, maka tidak bisa akses *method* dari *object* tersebut.

1. *Number Format Exception*

Bila ingin melakukan casting tipe data *string* ke *int* maka akan terjadi kesalahan ini.

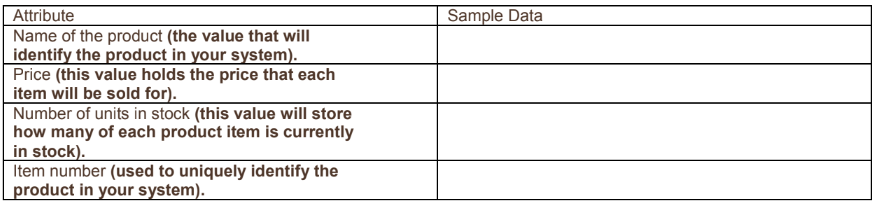
1. IO *Exception*

Bila yang diminta adalah angka, tapi diinputkan berupa *String*

## **BAB II**

## **PEMBAHASAN**

1. ***Practice JF\_4\_Project.***
2. Untuk bagian pertama proyek Anda diharuskan memikirkan apa yang akan disimpan oleh sistem inventaris Anda.
3. Pikirkan produk tertentu yang dapat disimpan dalam inventaris (misalnya, produk di rumah Anda, sekolah, atau tempat kerja: mereka dapat berasal dari kategori berikut; perlengkapan kantor, CD musik, film DVD, atau perangkat lunak). Tulis daftar minimal 6 produk yang ingin Anda simpan di sistem Anda, proyek ini dapat digunakan untuk menyimpan berbagai macam produk.
4. Untuk setiap produk yang Anda identifikasi, lengkapi tabel berikut:

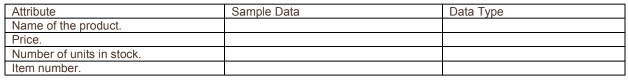


Tabel ini memberi Anda pemahaman tentang jenis data yang ingin Anda simpan untuk atribut setiap produk. Dia berguna untuk melakukan ini sehingga Anda memiliki pemahaman yang jelas tentang data yang akan Anda kerjakan !

Pembahasan :

|  |  |
| --- | --- |
| **Atribut** | **Contoh Data** |
| Nama produk (nilai yang akan  mengidentifikasi produk di sistem Anda). | Meja, Kursi, Pulpen,Tas, Jam,Kipas Angin,etc |
| Harga (nilai ini memegang harga masing-masing barang yang akan dijual). | 5000, 10000, 6000, 200000, 150000, etc. |
| Jumlah unit dalam stok (nilai ini akan disimpan berapa banyak dari setiap item produk saat ini persediaan). | 1, 2, 3, 4, 5, 6, etc. |
| Nomor item (digunakan untuk mengidentifikasi produk secara unik di sistem Anda). | 001, 002, 003, 004, etc. |

1. Langkah selanjutnya adalah memikirkan tipe data yang benar yang akan Anda gunakan untuk menyimpan nilai di sistem Anda. Untuk melakukan ini, tambahkan yang lain kolom ke tabel Anda yang akan mengidentifikasi tipe data yang benar untuk setiap nilai yang telah Anda identifikasi**.**



Pembahasan :

|  |  |  |
| --- | --- | --- |
| **Atribut** | **Contoh Data** | **Tipe Data** |
| Nama Produk | Meja | String |
| Harga | 150000 | Double |
| Jumlah Unit Dalam Stok | 4 | int |
| Nomor Item | 001 | int |

1. Buka Eclipse dan buat proyek bernama inventaris
2. Buat kelas objek bernama Produk



**Gambar 2.1** Class Produk (ProdukJF6)

Pada gambar menampilkan class yang di minta dibuat yaitu class ProdukJF6. Deklarasi kelas dimulai dengan kata kunci public yang menunjukkan bahwa kelas ini dapat diakses dari mana saja. Kata kunci class diikuti dengan nama kelas, yaitu ProdukJF6.

1. Tambahkan bidang instans privat berikut (variabel) dengan menggunakan tipe data yang anda identifikasi di tugas 2 :
2. Nomor Barang
3. Nama Produk
4. Jumlah Unit Yang Tersedia
5. Harga Setiap Unit

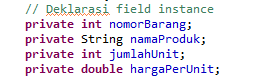


**Gambar 2.2** Kode Program pada Tugas

Pada gambar menampilkan kode private pada java. Deklarasi atribut dimulai dengan kata kunci private yang menunjukkan bahwa atribut ini hanya dapat diakses dari dalam kelas. Kode Java tersebut mendeklarasikan empat atribut untuk kelas ProdukJF6. Atribut-atribut tersebut adalah: nomorBarang adalah atribut tipe data int yang menyimpan nomor barang ,namaProduk adalah atribut tipe data String yang menyimpan nama produk.jumlahUnit adalah atribut tipe data int yang menyimpan jumlah unit produk hargaPerUnit adalah atribut tipe data double yang menyimpan harga per unit produk.

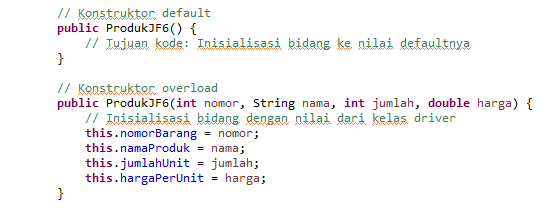
1. Tambahkan komentar di atas deklarasi kolom instance yang menyatakan:

//Deklarasi field instance

****

**Gambar 2.3** Menambahkan Komen

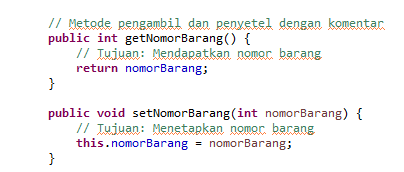
1. Buat dua konstruktor :
2. Konstruktor default tanpa parameter yang memungkinkan kompiler menginisialisasi bidang ke nilai defaultnya. Tambah sebuah beri komentar di atas konstruktor Anda yang menjelaskan tujuan kode.
3. Membebani konstruktor default dengan membuat konstruktor dengan parameter untuk keempat bidang instance kelas sehingga mereka dapat diinisialisasi dengan nilai dari kelas driver. Parameternya harus diberi nama; nomor, nama, jumlah, harga. Anda harus menggunakan notasi this.instance\_field\_name untuk mengukur bidang instance objek: *this.name = name;*



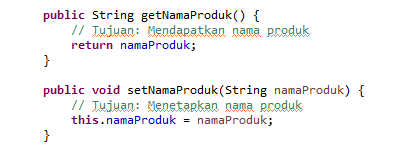
**Gambar 2.4** Kode Program Java

Konstruktor default adalah konstruktor yang tidak memiliki parameter. Konstruktor ini akan dipanggil secara otomatis ketika sebuah objek dibuat dari kelas ProdukJF6. Dalam kasus ini, konstruktor default akan menginisialisasi semua bidang ke nilai defaultnya. Konstruktor overload adalah konstruktor yang memiliki lebih dari satu parameter. Konstruktor ini akan dipanggil ketika sebuah objek dibuat dari kelas ProdukJF6 dengan memberikan nilai untuk semua parameternya. Dalam kasus ini, konstruktor overload akan menginisialisasi semua bidang dengan nilai yang diberikan oleh kelas driver. Kode Java tersebut mendeklarasikan dua konstruktor untuk kelas ProdukJF6. Konstruktor default akan dipanggil secara otomatis ketika sebuah objek dibuat dari kelas ProdukJF6. Konstruktor overload akan dipanggil ketika sebuah objek dibuat dari kelas ProdukJF6 dengan memberikan nilai untuk semua parameternya.

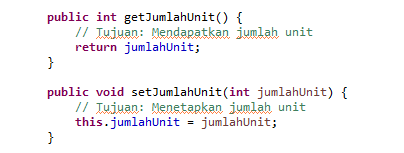
1. Tulis metode pengambil/pengakses dan penyetel/mutator untuk masing-masing dari empat variabel instan. Tulis pengambil/aksesor dan penyetel/mutator metode untuk masing-masing dari empat variabel instan. Tambahkan komentar di atasnya untuk menjelaskan tujuannya.



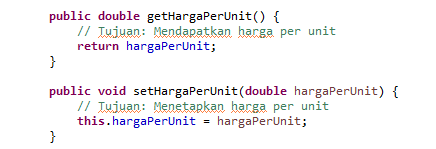
**Gambar 2.5 Kode** Java pada NomorBarang

****

**Gambar 2.6** Kode Java Pada NamaProduk

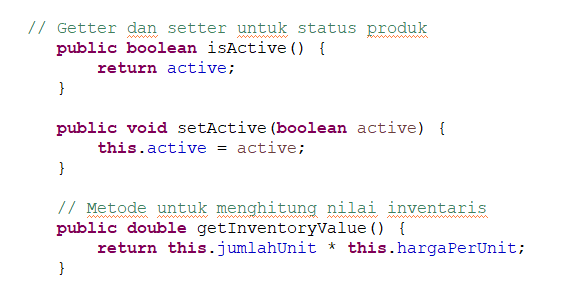
****

**Gambar 2.7** Kode Java Pada JumlahUnit

****

**Gambar 2.8** Kode Java Pada HargaPerUnit

Kode Java tersebut mendeklarasikan delapan metode pengambil dan penyetel untuk kelas ProdukJF6. Metode pengambil digunakan untuk mendapatkan nilai dari atribut kelas, sedangkan metode penyetel digunakan untuk menetapkan nilai ke atribut kelas. Metode pengambil dimulai dengan kata kunci public yang menunjukkan bahwa metode ini dapat diakses dari mana saja. Kata kunci void menunjukkan bahwa metode ini tidak mengembalikan nilai. Nama metode pengambil biasanya diawali dengan kata get diikuti dengan nama atribut yang terkait. Kode dalam metode pengambil hanya mengembalikan nilai dari atribut yang terkait. Metode penyetel dimulai dengan kata kunci public yang menunjukkan bahwa metode ini dapat diakses dari mana saja. Kata kunci void menunjukkan bahwa metode ini tidak mengembalikan nilai. Nama metode penyetel biasanya diawali dengan kata set diikuti dengan nama atribut yang terkait. Kode dalam metode penyetel menetapkan nilai ke atribut yang terkait.

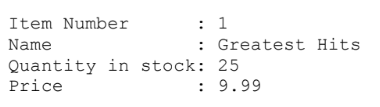


**Gambar 2.9** Kode Program Getter dan setter

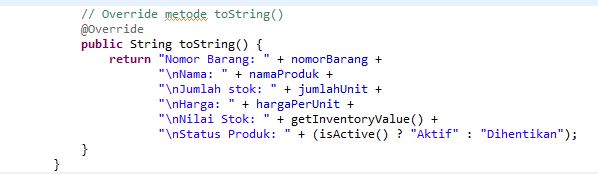
Metode isActive() merupakan metode getter yang mengembalikan nilai atribut active. Metode ini mengembalikan nilai true jika atribut active bernilai true, dan mengembalikan nilai false jika atribut active bernilai false. Metode setActive() merupakan metode setter yang mengubah nilai atribut active. Metode ini menerima parameter boolean active yang menentukan nilai baru untuk atribut active. Metode getInventoryValue() merupakan metode yang menghitung nilai inventaris. Metode ini mengembalikan nilai hasil perkalian antara atribut jumlahUnit dan hargaPerUnit.

Secara keseluruhan, kode Java tersebut mendefinisikan kelas dengan atribut active, jumlahUnit, dan hargaPerUnit, serta metode isActive(), setActive(), dan getInventoryValue(). Metode isActive() dan setActive() digunakan untuk mengakses dan mengubah nilai atribut active, sedangkan metode getInventoryValue() digunakan untuk menghitung nilai inventaris.

1. Ganti metode toString() dari kelas objek untuk menampilkan deskripsi setiap objek Produk yang menyertakan instance tersebut nilai bidang dalam format berikut :

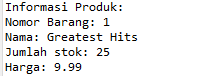


Pembahasan :



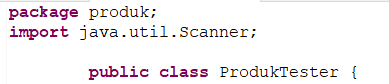
**Gambar 2.10** Kode Program Java Override Metode toString

Kode Java tersebut merupakan contoh override metode toString() pada kelas Produk. Metode toString() merupakan metode bawaan dari kelas Object yang digunakan untuk mengembalikan representasi string dari objek. Metode ini biasanya digunakan untuk menampilkan informasi tentang objek ke layar atau ke output lainnya. Pada kode tersebut, metode toString() dioverride untuk mengembalikan representasi string dari objek Produk. Representasi string tersebut terdiri dari informasi berikut yaitu , Nomor barang, Nama produk, Jumlah stok, Harga per unit ,Nilai stok, Status produk. Informasi-informasi tersebut diperoleh dari atribut-atribut kelas Produk. Atribut-atribut tersebut adalah nomorBarang, namaProduk, jumlahUnit, hargaPerUnit, dan isActive().



**Gambar 2.11** Hasil Run Program

1. Kelas utama Java bernama ProductTester.

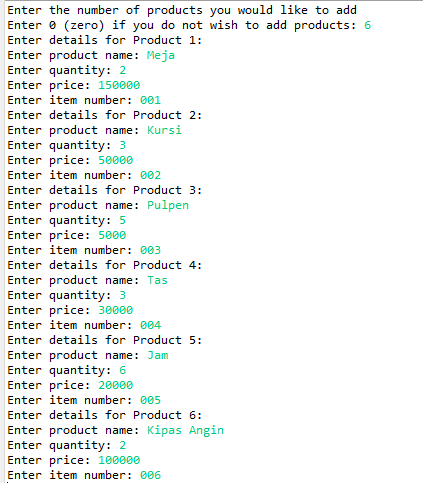


**Gambar 2.12** Kode Program Java

Pada gambar menampilkan class yang di minta dibuat yaitu class ProdukTester. Deklarasi kelas dimulai dengan kata kunci public yang menunjukkan bahwa kelas ini dapat diakses dari mana saja. Kata kunci class diikuti dengan nama kelas, yaitu ProdukTester.

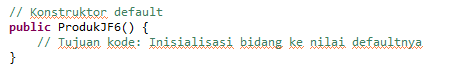
1. Buat dan inisialisasi enam objek Produk berdasarkan daftar yang Anda buat di tugas 1:
2. Dua Produk harus dibuat menggunakan konstruktor default
3. Empat lainnya harus dibuat dengan memberikan nilai argumen yang cocok dengan parameter konstruktor.

Pembahasan :

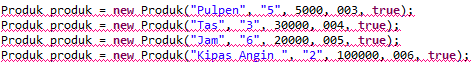


**Gambar 2.13** Hasil Run dari perintah yang diminta

1. Konstruktor default adalah konstruktor yang tidak memiliki parameter. Konstruktor ini secara otomatis dibuat oleh kompiler Java jika tidak ada konstruktor lain yang dideklarasikan dalam kelas. Konstruktor default digunakan untuk menginisialisasi objek dengan nilai default. Misalnya, jika kelas Produk memiliki atribut nomorBarang, namaProduk, jumlahUnit, hargaPerUnit, dan isActive(), maka konstruktor default akan menginisialisasi atribut-atribut tersebut dengan nilai default sebagai berikut:

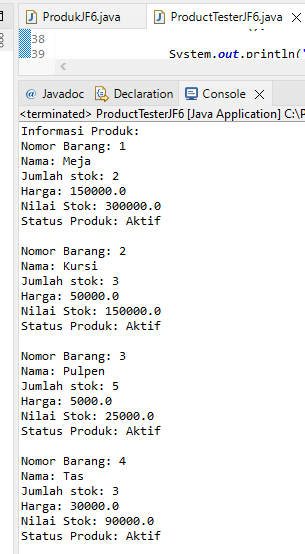
**Gambar 2.14** Kode Java Konstruktor default

1. Konstruktor yang harus dibuat dengan memberikan nilai argumen yang cocok dengan parameter konstruktor adalah konstruktor Produk() pada kelas Produk. Konstruktor ini memiliki lima parameter, yaitu nomorBarang, namaProduk, jumlahUnit, hargaPerUnit, dan isActive(). Oleh karena itu, untuk membuat objek Produk dengan konstruktor ini, kita harus memberikan nilai argumen yang cocok dengan lima parameter tersebut. Sebagai contoh, berikut adalah kode untuk membuat objek Produk dengan konstruktor Produk():



**Gambar 2.15** Kode Java Nilai Argumen yang Cocok dengan Parameter

1. Menggunakan kelas ProductTester, tampilkan detail setiap produk ke konsol.



**Gambar 2.16** Tampilan Pada Konsul

1. Simpan proyek Anda.
2. ***Practice JF\_5\_Project.***
3. Buka program inventaris yang telah dibuat di Bagian 4: Membuat Proyek inventaris
4. Ubah kelas ProductTester
5. Tambahkan Pemindai yang dipanggil ke awal metode utama Anda.
6. Buat variabel lokal yang akan menyimpan nilai untuk setiap atribut kelas Produk. Beri nama variabelnya tempNumber, tempName, tempQty dan tempPrice.
7. Minta pengguna untuk memasukkan nilai untuk setiap atribut kelas Produk. Tanyakan nama, jumlah, harga dan nomor item, simpan nilai dalam variabel lokal sementara yang baru saja Anda buat.
8. Gunakan nilai yang dimasukkan oleh pengguna untuk membuat objek p1. Ini berarti Anda akan menggunakan konstruktor yang membutuhkan 4 parameter, bukan konstruktor default.

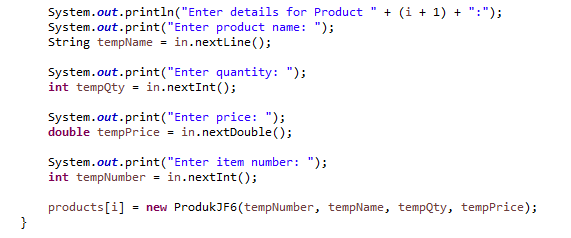
Pembahasan :

1. Menambahkan Scanner



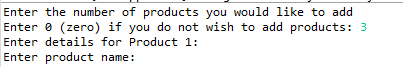
**Gambar 2.17** Kode Program Java Scanner

1. Variabelnya : tempNumber, tempName, tempQty dan tempPrice.



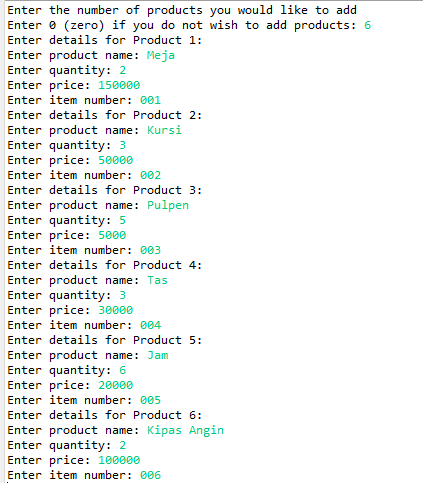
**Gambar 2.18** Kode Program Java Menambah Variable sesuai Perintah

1. Meminta Pengguna Memasukkan nilai pada Setiap Atribut ;



**Gambar 2.19** Hasil Run

1. Gunakan nilai yang dimasukkan oleh pengguna untuk membuat objek p1.



**Gambar 2.20** Memasukan nilai pada P1

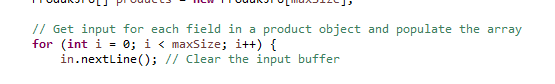
1. Anda akan meminta pengguna memberi Anda nilai untuk p2.
   * 1. Gunakan variabel lokal yang sama seperti sebelumnya untuk mendapatkan masukan dari pengguna untuk membuat objek p2. Salin dan tempel kode setelah baris yang membuat objek p1.
     2. Jalankan program dan identifikasi di mana kesalahan terjadi.
     3. Program tampaknya tidak menanyakan nilai nama kepada Anda. Hal ini karena nilai terakhir yang dimasukkan adalah a nilai numerik dan meninggalkan beberapa karakter khusus di buffer input. Untuk menghapus buffer input, tambahkanpernyataan berikut sebelum Anda menanyakan nilai apa pun untuk p2:

di.nextLine();

Ini mengambil nilai apa pun yang disimpan dalam buffer dan membuangnya sehingga meninggalkan buffer kosong.

* + 1. Jalankan program sekarang, program harus bebas dari kesalahan dan menampilkan semua nilai termasuk nilai yang dimasukkan pengguna di dalamnya menghibur.
    2. Tutup objek Pemindai setelah Anda selesai menggunakannya.

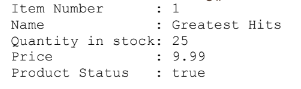
Pembahasan :



**Gambar 2.21** Kode Program Java Menambahkan in.nextLine ()

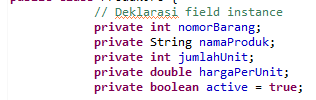
Kode diatas digunakan untuk membaca input dari pengguna dan mengisi array objek produk. Variabel maxSize digunakan untuk menentukan jumlah maksimum produk yang dapat disimpan array. Loop for mengiterasi array objek produk, dan untuk setiap objek, meminta pengguna untuk memasukkan nilai untuk bidang produk. Pernyataan in.nextLine() digunakan untuk mengosongkan buffer input setelah setiap bidang dimasukkan. Ini diperlukan karena metode nextLine() meninggalkan karakter baris baru di buffer input, yang dapat menyebabkan masalah jika input berikutnya diharapkan berupa angka atau string.

1. Anda ingin dapat menandai produk Anda sebagai produk aktif atau dihentikan. Jika suatu produk dihentikan, itu berarti sisa stok akan menjadi yang terakhir, dan tidak ada lagi pesanan yang dibuat.
   * 1. Tambahkan bidang instance Boolean ke kelas Produk Anda yang disebut aktif yang memiliki nilai default true.
     2. Buat metode pengambil/penyetel untuk bidang baru ini.
     3. Tambahkan nilai kolom baru ini ke metode toString() sehingga outputnya cocok dengan yang berikut:

****

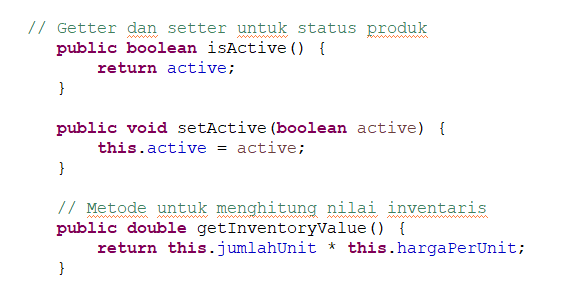
Pembahasan :

* + 1. Tambahkan bidang Boolean Ke kelas Produk :

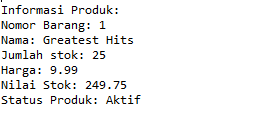


**Gambar 2.22** Kode Program Java Tambahan Boolean

* + 1. Buat Metode Getter/Setter :

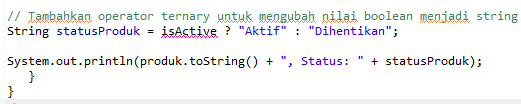


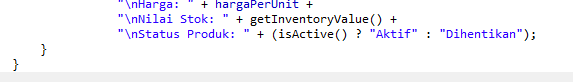
**Gambar 2.23** Kode Java Getter/Setter

* + 1. Output yang cocok seperti perintah :

**Gambar 2.24** Hasil Run Program

1. Saat Anda menjalankan kode, Anda mendapatkan kembali nilai yang dicetak untuk aktif sebagai benar atau salah. Ini tidak ramah pengguna dan kan lebih baik jika keluarannya menyatakan Aktif (benar) atau Dihentikan (salah). Tambahkan operator ternary di file toString() metode untuk mencapai ini.

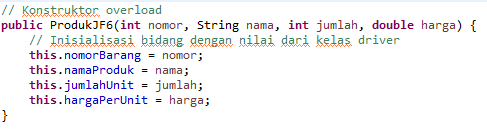




**Gambar 2.25** Kode Program java

Dalam kasus ini, ekspresi yang digunakan adalah isActive. Jika isActive bernilai true, maka nilai yang dikembalikan adalah "Aktif". Jika isActive bernilai false, maka nilai yang dikembalikan adalah "Dihentikan".

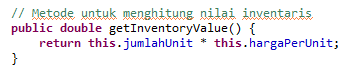
1. Panggil setter dari kelas driver dan atur nilai aktif ke false untuk objek p6 sebelum Anda menampilkan nilai ke layar. Jalankan dan uji kode Anda.



**Gambar 2.26** Kode Program Java

Konstruktor ini memiliki empat parameter, yaitu: nomor: tipe data int, nama: tipe data String, jumlah: tipe data int, harga: tipe data double. Konstruktor ini digunakan untuk menginisialisasi atribut nomorBarang, namaProduk, jumlahUnit, dan hargaPerUnit dengan nilai yang diberikan dari kelas driver.

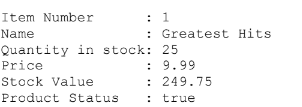
1. Buatmetode di kelas Produk yang akan mengembalikan nilai inventaris setiap item. Gunakan harga produk dikalikan dengan jumlah stok untuk menghitung nilai persediaan. Jangan gunakan variabel lokal apa pun dalam metode ini mengembalikan nilai dalam satu baris kode .



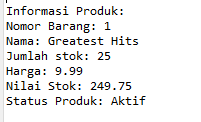
**Gambar 2.27** Kode program Java Menghitung Stok Persedian

Kode Java tersebut adalah metode getInventoryValue() untuk kelas ProdukJF6. Metode ini digunakan untuk menghitung nilai inventaris produk. Metode ini memiliki tipe data double sebagai nilai kembaliannya. Metode ini menggunakan operator perkalian untuk menghitung nilai inventaris, yaitu dengan mengalikan jumlah unit dengan harga per unit.

1. Perbarui metode toString() di kelas Produk untuk menyertakan pemanggilan metode ke metode getInventoryValue() yang Anda baru saja dibuat sehingga outputnya seperti berikut:



Pembahasan :



**Gambar 2.28** Hasil Run Program

1. Simpan proyek Anda.
2. ***Practice JF\_6\_Project.***
3. Buka program inventaris yang telah diperbarui di Bagian 5: Membuat Proyek inventaris
4. Minta pengguna memasukkan jumlah produk yang ingin ditambahkan. Terima bilangan bulat positif untuk jumlah produk dan menangani nilai nol.
   * 1. Buat variabel bernama maxSize yang dapat menyimpan bilangan bulat.
     2. Buat prompt di awal metode utama Anda yang akan menginstruksikan pengguna untuk memasukkan nilai yang diperlukan untuk jumlah produk yang ingin mereka simpan:



* + 1. Gunakan perulangan do while sehingga program tidak akandilanjutkan sampai nilai positif yang valid dimasukkan. Jika nilainya kurang dari nol dimasukkan, pesan kesalahan yang menyatakan “Nilai yang dimasukkan salah” harus ditampilkan sebelum pengguna Kembali diminta untuk memasukkan nilai baru. Anda tidak boleh meninggalkan perulangan sampai nilai nol atau lebih besar dimasukkan.

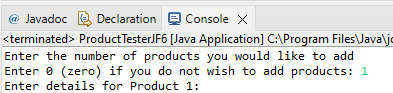
Pembahasan :

* + 1. Buat Variable Maxsize :



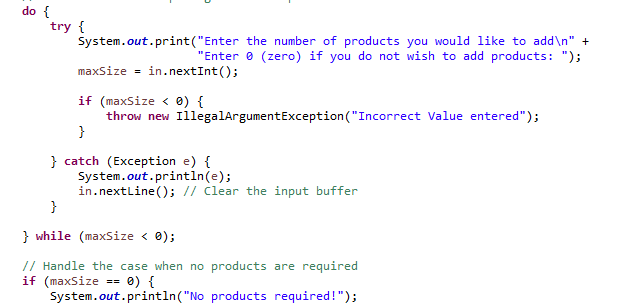
**Gambar 2.29** Kode Variable Maxsize

* + 1. Buat prompt di awal metode utama Anda yang akan menginstruksikan pengguna untuk memasukkan nilai yang diperlukan untuk jumlah produk :



**Gambar 2.30** Hasil Run Program

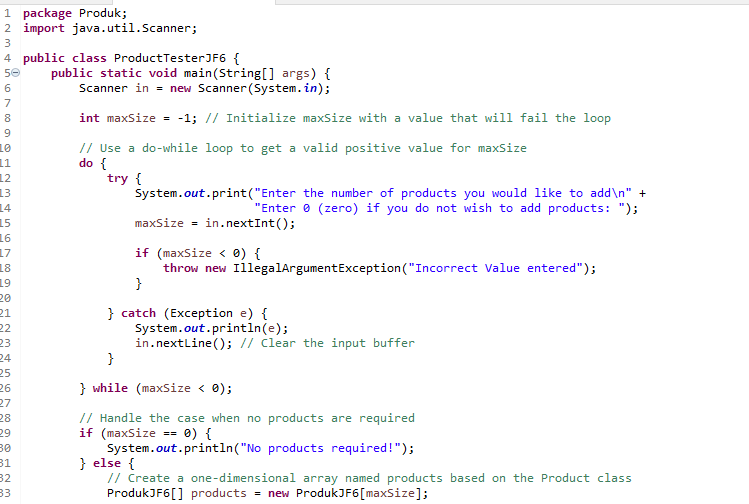
* + 1. Gunakan perulangan do while :

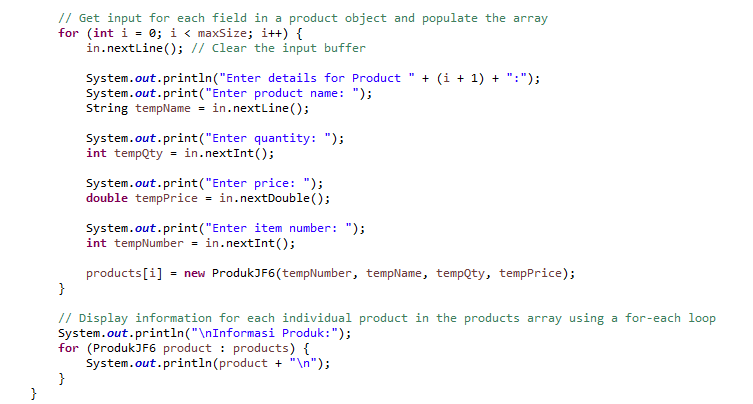


**Gambar 2.31** Kode Program Java Do While

1. Anda sekarang akan menambahkan beberapa penanganan kesalahan untuk menangani kesalahan run-time dalam kode Anda. Saat ini program Anda berhubungan dengan nomor yang dimasukkan di luar rentang yang diberikan tetapi tidak dapat menangani entri tipe data yang salah.
   * 1. Tambahkan blok try yang mengelilingi semua kode di dalam loop do while.
     2. Tambahkan pernyataan catch di atas while yang akan mengambil parameter Exception e. Program ini harus menggunakan konsol pernyataan keluaran untuk menampilkan nilai e ke layar.
     3. Saat Anda sekarang menetapkan nilai maxSize di dalam pernyataan try, ada kemungkinan maxSize tidak akan memiliki ben diberi nilai ketika Anda sampai ke klausa while. Untuk memastikan hal ini tidak terjadi, tetapkan nilai awal -1 ke maxSize saat dideklarasikan. PETUNJUK: Selalu tetapkan nilai yang akan menggagalkan perulangan sehingga kode Anda terpaksa menetapkan a nilai yang benar sebelum melanjutkan.
     4. Jalankan dan uji kode Anda dengan memasukkan karakter, bukan angka.
     5. Tambahkan baris kode dalam pernyataan catch Anda yang akan menghapus buffer input sehingga prompt akan ditampilkan, dan sistem akan menunggu masukan pengguna.
     6. Catat jenis Pengecualian tertentu yang dihasilkan saat Anda memasukkan karakter dan buat pernyataan catch saja untuk pengecualian itu. Kesalahan ini seharusnya menampilkan tipe data yang dimasukkan salah! pesan ke konsol dan juga harus menghapus buffer input.
     7. Jalankan dan uji kode Anda dengan memasukkan berbagai nilai masukan yang berbeda.

Pembahasan :



****

**Gambar 2.32** Kode Program Java Setelah Ditambahkan Sesuai Perintah

1. Ubah kelas ProductTester untuk menangani beberapa produk menggunakan array dimensi tunggal jika nilai yang dimasukkan lebih besar dari nol.
2. Buat pernyataan if yang akan menampilkan pesan “Tidak diperlukan produk!” ke konsol jika nilai maxSize adalah nol.
3. Tambahkan pernyataan Else untuk menangani nilai apa pun selain nol. Buat array satu dimensi bernama produk berdasarkan kelas Produk yang akan memiliki jumlah elemen yang ditentukan oleh pengguna dalam variabel maxSize.

Pembahasan :

* + 1. Buatlah Pernyataan pesan “No products required!” :



**Gambar 2.33** Kode Program Java Membuat Pesan “No products required!”

* + 1. Tambahkan pernyataan Else :

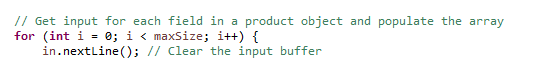


**Gambar 2.34** Kode Program Java else

1. Anda sekarang akan mengisi array, mendapatkan nilai dari pengguna untuk setiap bidang dalam objek produk.
   * 1. Di dalam pernyataan else di bawah tempat Anda membuat array, tulislah perulangan for yang akan melakukan iterasi melalui array dari nol menjadi 1 kurang dari maxSize.
     2. Karena masukan terakhir yang Anda terima dari pengguna adalah numerik, Anda perlu menambahkan pernyataan yang menghapus buffer masukan sebagai baris pertama dalam loop for Anda.
     3. Salin kode yang Anda gunakan untuk mendapatkan masukan dari pengguna untuk semua bidang produk ke dalam perulangan for. Ini termasuk nama, jumlah, harga dan nomor barang.
     4. Tambahkan objek produk baru ke dalam array menggunakan nilai indeks untuk posisi dan konstruktor yang mengambil 4 parameter.

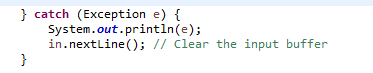
Pembahasan :

* + 1. Perulangan for :



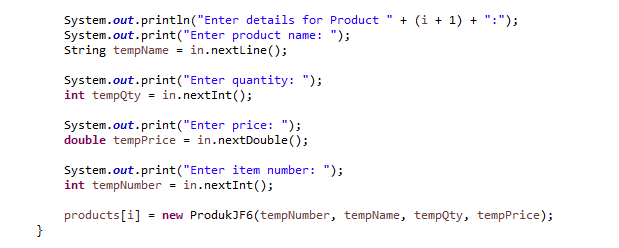
**Gambar 2.35** Kode Program Java Perulangan For

* + 1. Mengahapus buffer :



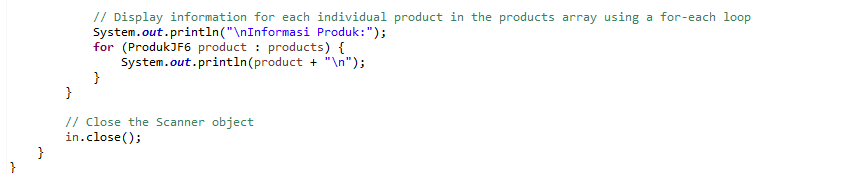
**Gambar 2.36** Kode Program Java Clear Buffer

* + 1. Salin kode pengguna produk ke dalam perulangan for :



**Gambar 2.37** Kode Program Penggunaan Produk

1. Gunakan perulangan for every untuk menampilkan informasi setiap produk dalam larik produk.

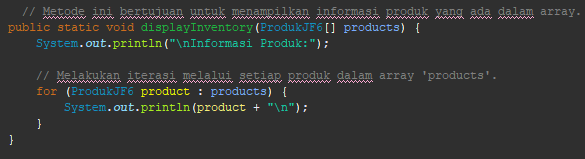


**Gambar 2.38** Kode Program Java Perulangan for display the information

Kode Java tersebut digunakan untuk menampilkan informasi dari setiap produk yang ada di array products. Berikut penjelasan baris per baris: System.out.println("\nInformasi Produk:"); Baris ini digunakan untuk mencetak string "Informasi Produk:" ke layar, diikuti dengan newline character. for (ProdukJF6 product : products) Baris ini merupakan awal dari loop for-each. Loop ini akan iterasi melalui setiap elemen dalam array products. Setiap elemen dalam array adalah objek dari kelas ProdukJF6. Variabel product digunakan untuk menunjuk ke objek produk saat ini yang sedang diiterasi. System.out.println(product + "\n"); Baris ini digunakan untuk mencetak informasi dari objek produk saat ini. Operator + digunakan untuk menggabungkan objek product dengan string newline character ("\n"). Objek product akan memanggil metode toString() secara otomatis untuk mengubah dirinya menjadi string yang berisi informasi produk. String yang dihasilkan kemudian dicetak ke layar, diikuti dengan newline character. in.close(); Baris ini digunakan untuk menutup objek Scanner. Menutup objek Scanner penting untuk memastikan sumber daya sistem dibebaskan dengan benar. Secara keseluruhan, kode ini akan menampilkan informasi dari setiap produk yang ada di array products ke layar. Informasi yang ditampilkan tergantung pada implementasi metode toString() pada kelas ProdukJF6.

1. Hapus kode yang tidak diperlukan dan tidak digunakan dalam latihan ini.
2. Simpan proyek Anda.
3. ***Practice JF\_7\_1 Project.***
   1. Buka program inventaris yang telah diperbarui di Bagian 6: Membuat Proyek inventaris.
   2. Anda sekarang akan memodifikasi kode Anda sehingga kelas utama tidak akan melakukan pemrosesan apa pun tetapi cukup memanggil metode statis kapan diperlukan.
4. Buat metode statis di kelas ProductTester setelah akhir metode utama yang disebut displayInventory. Ini metode tidak akan mengembalikan nilai apa pun dan akan menerima array produk sebagai parameter. Ingat ketika Anda lewat array sebagai parameter Anda menggunakan nama kelas sebagai tipe data, sekumpulan tanda kurung siku kosong dan kemudian nama array (NamaKelas[] Namaarray)
5. Salin kode yang menampilkan array dari metode utama ke metode displayInventory yang baru.

**Pembahasan** :

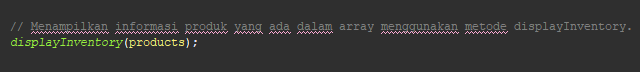


**Gambar 2.39** Kode Program Java Menampilkan Informasi Produk

Kode Java tersebut adalah metode displayInventory() untuk kelas ProdukJF6. Metode ini digunakan untuk menampilkan informasi produk yang ada dalam array. Didalam metode ini, berisi kode sebelumnya yang menampilkan array dari metode utama yang akan disalin dan dipindahkan ke metode displayInventory().

1. Jika Anda menghapus kode tampilan dari main, gantilah dengan pemanggilan metode ke metode displayInventory. Ingatlah untuk menyertakan daftar argumen yang benar agar sesuai dengan daftar parameter dalam metode yang Anda panggil.

**Pembahasan :**

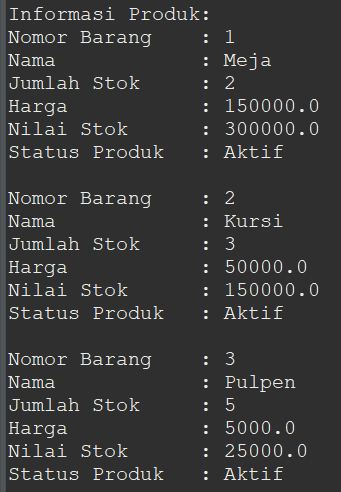


**Gambar 2.40** Pemanggilan metode displayInventory()

Pada **Gambar 2.40** tersebut merupakan pemanggilan method displayInventory() dengan parameter products dibagian main method yang digunakan untuk menampilkan informasi produk yang ada dalam array.

1. Jalankan dan uji kode Anda

**Pembahasan :**

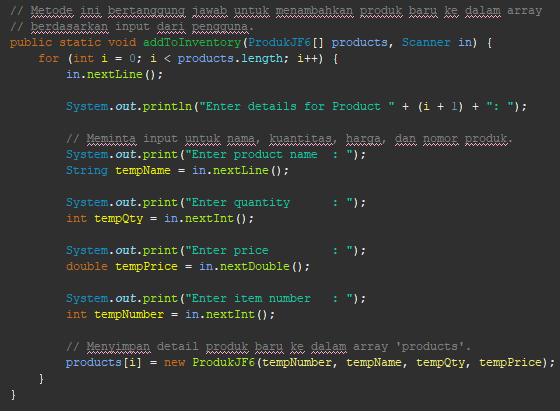


**Gambar 2.41** Hasil Run Program

Berikut adalah hasil run program yang mana akan menampilkan informasi produk seperti Nomor Barang, Jumlah Stok, Harga, Nilai Stok, dan Status Produk.

1. Buat metode statis di kelas ProductTester setelah akhir metode utama yang disebut addToInventory. Ini metode tidak akan mengembalikan nilai apa pun dan akan menerima array produk dan pemindai sebagai parameter.
2. Salin kode yang menambahkan nilai ke array dari metode utama ke metode addToInventory yang baru.
3. Untuk mengatasi kesalahan yang Anda alami dalam kode, pindahkan variabel lokal yang diperlukan (tempNumber, tempName, tempQty, tempPrice) dari metode utama ke bagian atas metode addInventory.

**Pembahasan :**

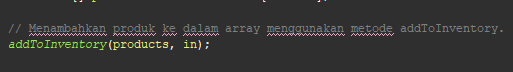


**Gambar 2.42** Kode Program Java Menambahkan Produk Baru

Kode Java tersebut adalah metode addToInventory() untuk kelas ProdukJF6. Metode ini digunakan untuk menambahkan produk baru ke dalam array. Didalam metode ini, berisi kode sebelumnya yang menampilkan array dari metode utama yang akan disalin dan dipindahkan ke metode addToInventory().

1. Tambahkan pemanggilan metode di main ke metode addToInventory() tempat Anda menghapus perulangan for.

**Pembahasan :**

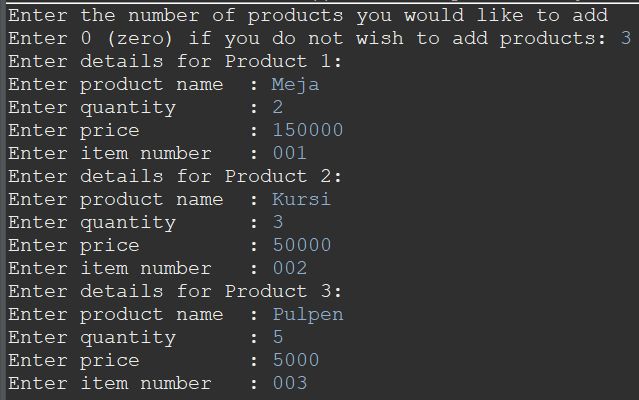


**Gambar 2.43** Kode Program Java Pemanggilan Metode addToInventory()

Pada **Gambar 2.43** tersebut merupakan pemanggilan method addToInventory() dengan parameter products dan objek in sebagai scanner dibagian main method yang digunakan untuk menambahkan produk baru ke dalam array.

1. Jalankan dan uji kode Anda

**Pembahasan :**



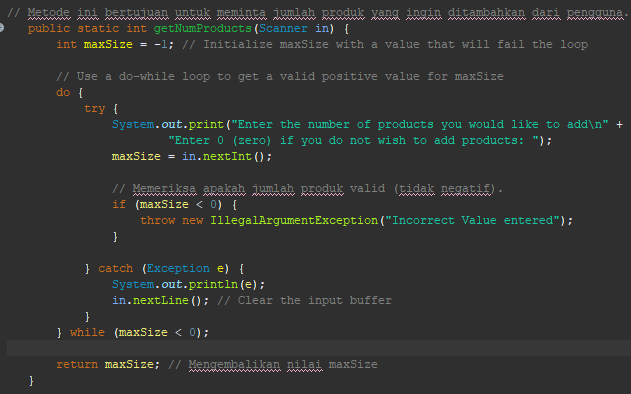
**Gambar 2.44** Hasil Run Program

Berikut adalah hasil run program yang mana akan menambahkan data produk seperti Nomor Barang, Jumlah Stok, Harga, dan Nilai Stok.

1. Buat metode di ProductTester yang akan mengembalikan nilai integer bernama getNumProducts() yang menerima pemindai sebagai parameter. Pindahkan semua kode yang mendapatkan jumlah produk maksimum dari pengguna ke dalamnya metode, masukkan pemanggilan metode main ke metode baru Anda. Anda akan menyimpan nilai yang dikembalikan dalam variabel maxSize Anda jadi Anda perlu mendeklarasikan 2 di antaranya, satu di metode utama Anda dan satu lagi di metode getNumProducts(). Anda dapat menghapus nilai awal -1 dari deklarasi di main.

**Pembahasan :**

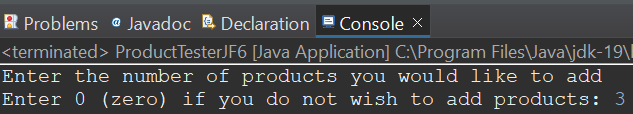
Di class ProductTester, akan dibuatkan method getNumProducts() yang dapat mengembalikan nilai dan menerima scanner sebagai parameter. Kemudian kode yang berada dimain method yang menginput jumlah produk maksimum dari pengguna akan disalin dan dipindahkan ke method getNumProducts(). Tambahkan “return maxSize” pada akhir method untuk mengembalikan nilai maxSize. Berikut adalah kode program java untuk meminta jumlah produk.



**Gambar 2.45** Kode Program Java Meminta Jumlah Produk

1. Jalankan dan uji kode Anda

**Pembahasan :**



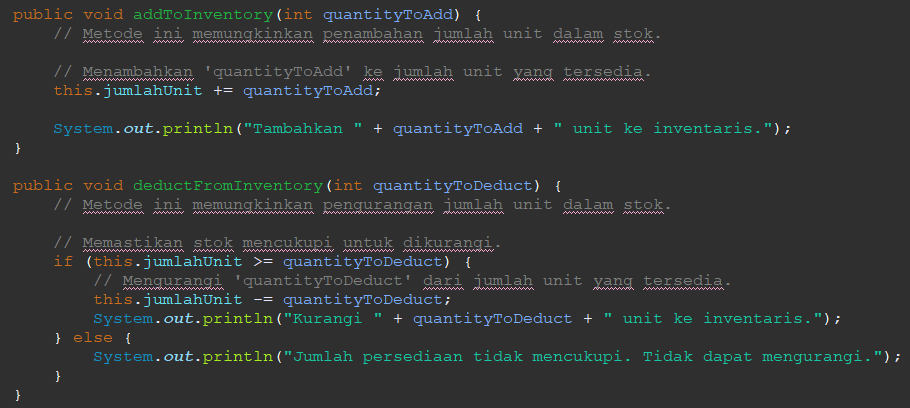
**Gambar 2.46** Hasil Run Program

Berikut adalah hasil run program yang mana akan meminta jumlah produk yang akan ditambahkan.

* 1. Buat dua metode baru di kelas Produk, yang memungkinkan pengguna menambah jumlah unit dalam stok (addToInventory), dan yang memungkinkan pengguna mengurangi jumlah unit dalam stok (deductFromInventory). Keduanya metode harus menerima parameter (kuantitas) yang menampung jumlah item yang akan ditambahkan/dikurangi. Tempatkan ini di bawah konstruktor anda.

**Pembahasan :**

Di class Product, akan dibuatkan method addToInventory() dan deductFromInventory(). Method addToInventory() digunakan untuk menambah jumlah unit dalam stok dan method deductFromInventory() digunakan untuk mengurangi jumlah unit dalam stok. Berikut adalah Kode Program Java untuk Penambahan dan Pengurangan Jumlah Unit Dalam Stok.



**Gambar 2.47** Kode Program Java Penambahan dan Pengurangan Jumlah Unit Dalam Stok

* 1. Ubah kelas ProductTester sehingga pengguna dapat melihat, memodifikasi, atau menghentikan produk melalui antarmuka pengguna yang berdasarkan sistem menu.

1. Menampilkan sistem menu yang akan menampilkan pilihan dan mengembalikan pilihan menu yang dimasukkan oleh pengguna.
   1. Metode ini harus dipanggil getMenuOption, mengembalikan nilai integer dan mengambil objek Scanner sebagai sebuah parameter. Tulis kode di bawah main.
   2. Menunya akan terlihat seperti berikut:

1. View Inventory

2. Add Stock

3. Deduct Stock

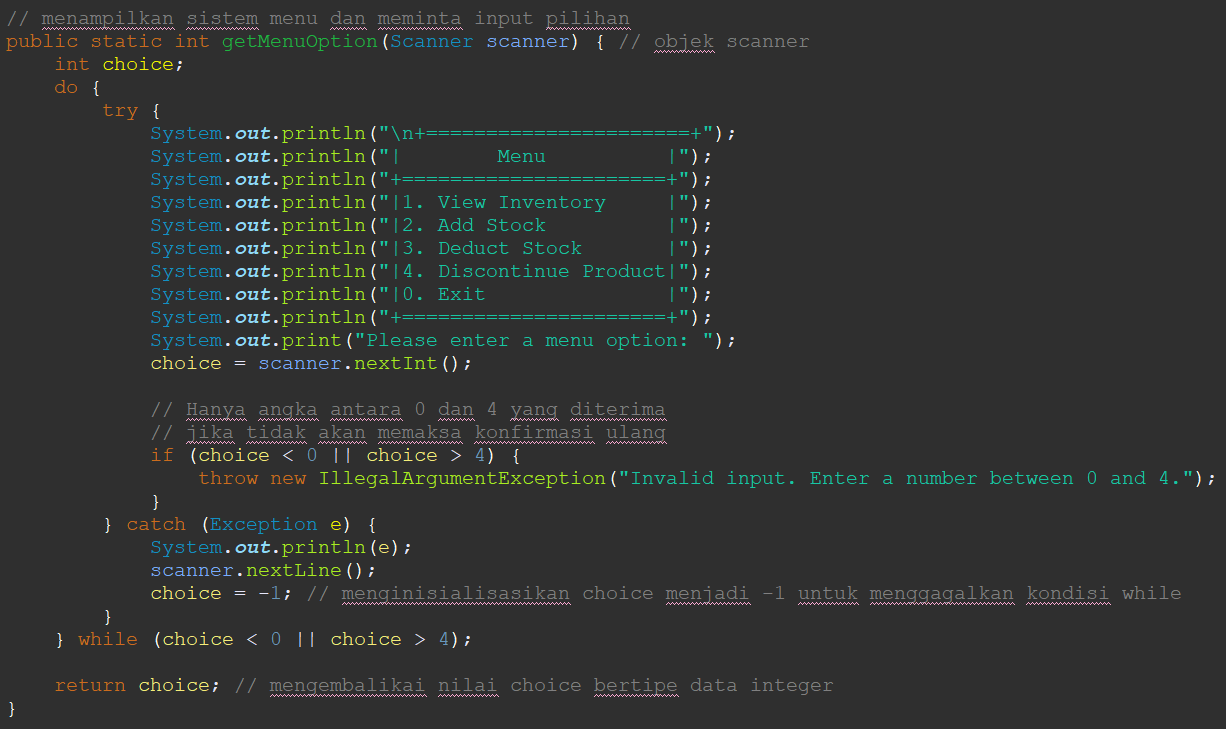
4. Discontinue Product

0. Exit

Please enter a menu option:

* 1. Hanya angka antara 0 dan 4 yang boleh diterima, masukan lainnya akan memaksa konfirmasi ulang ke pengguna. Ingat ketika menambahkan pernyataan try catch Anda harus menginisialisasi variabel menjadi sesuatu yang akan menggagalkan kondisi while!

**Pembahasan :**

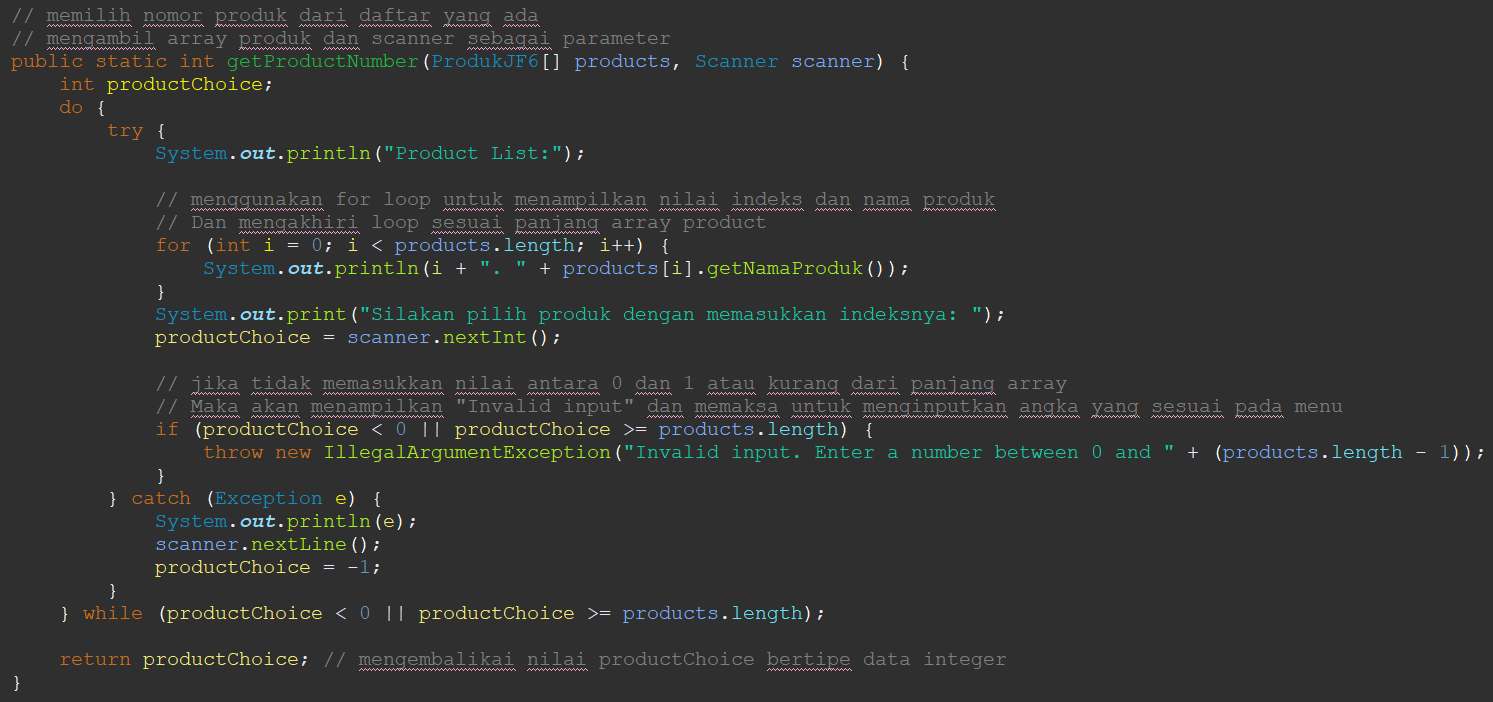


**Gambar 2.48** Kode Program Java Menampilkan Sistem Menu dan Meminta Input Pilihan

Kode Java tersebut adalah method getMenuOption() untuk class ProductTesterJF6. Pada blok program method tersebut, akan terlihat menu seperti View Inventory, Add Stock, Deduct Stock, Discontinue, dan Exit. Selanjutnya program akan meminta pengguna menginputkan menu tersebut. Hanya inputan angka antara 0 dan 4 yang bisa diterima. Jika menginputkan selain dari itu, maka akan terjadi pesan invalid input dan program akan memaksa pengguna melakukan konfirmasi ulang hingga menu yang input benar. Tambahkan “return choice” pada akhir method untuk mengembalikan nilai choice.

1. Buat metode yang akan menampilkan nilai indeks array dan nama setiap produk yang memungkinkan pengguna untuk melakukannya pilih produk yang ingin diperbarui (tambah/kurangi).
   1. Metode ini harus dipanggil getProductNumber, mengembalikan nilai integer dan mengambil array produk dan objek Pemindai sebagai parameter. Itu harus memiliki satu variabel lokal bernama productChoice of ketik integer yang diinisialisasi ke -1. Tulis kode di bawah main.
   2. Perulangan FOR tradisional harus digunakan untuk menampilkan nilai indeks dan nama produk. Menggunakan panjang array untuk mengakhiri loop. Nama setiap produk dapat diakses melalui nya metode pengambil yang sesuai.
   3. Pengguna hanya diperbolehkan memasukkan nilai antara 0 dan 1 kurang dari panjang array. Semua masukan harus memiliki penanganan kesalahan yang sesuai.

**Pembahasan :**

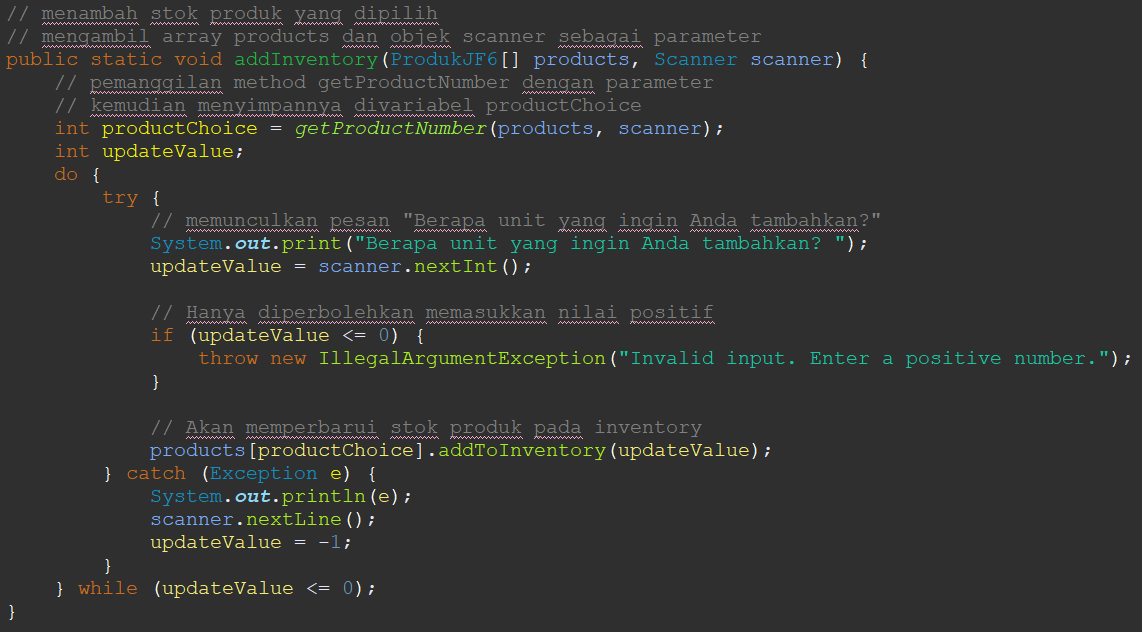


**Gambar 2.49** Kode Program Java Memilih Nomor Produk

Kode Java tersebut adalah method getProductNumber() dengan array product dan objek scanner sebagai parameter. Kemudian menggunakan for loop untuk menampilkan nilai indeks dan menggunakan getter untuk menampilkan nama produk. Setelah itu user akan diminta untuk memilih produk dengan memasukkan indeksnya. Jika nilai yang dipilih tidak diantara 0 dan panjang dari array maka input akan invalid dan program akan memaksa untuk menginputkan angka yang sesuai pada menu.

1. Buat metode yang akan menambahkan nilai stok pada setiap produk yang diidentifikasi.
   * 1. Metode ini harus disebut addInventory, tidak memiliki nilai kembalian dan mengambil array produk dan objek pemindai sebagai parameter. Seharusnya ada dua variabel lokal bernama productChoice yang berfungsi tidak perlu diinisialisasi dan nama lain updateValue yang harus diinisialisasi ke -1. Kedua variabel lokal harus dapat menyimpan nilai integer. Tulis kode di bawah main.
     2. Tambahkan pemanggilan metode ke metode getProductNumber dengan meneruskan parameter yang benar dan menyimpan menghasilkan variabel productChoice.
     3. Pengguna akan diarahkan dengan pesan “Berapa banyak produk yang ingin Anda tambahkan?” dan hanya diperbolehkan memasukkan nilai positif 0 ke atas. Semua masukan harus memiliki kesalahan yang sesuai penanganan dan pesan kesalahan.
     4. Setelah nilai pembaruan yang valid ditambahkan, maka tingkat stok produk yang dipilih harus diperbarui melalui metode addToInventory yang Anda buat sebelumnya. Variabel productChoice digunakan untuk mengidentifikasi nilai indeks produk dalam array dan updateValue adalah jumlah stok yang akan ada ditambahkan.

**Pembahasan :**

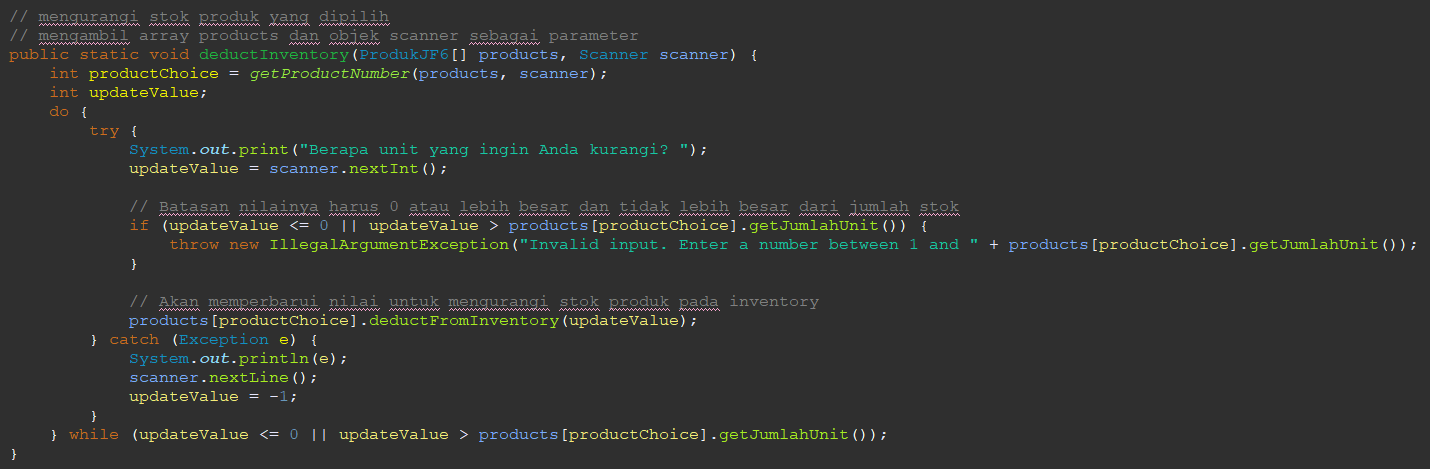


**Gambar 2.50** Kode Program Java Menambah Stok Produk

Kode Java tersebut adalah method addInventory() dengan array products dan objek scanner sebagai parameter.Method ini digunakan untuk menambah stok produk yang dipilih. Pada blok program, pengguna diminta untuk menginputkan berapa unit atau stok yang ingin ditambahkan. Input yang dapat diterima hanya nilai positif. Jika nilai negatif (kurang dari 0), maka input akan invalid dan program akan memaksa pengguna untuk melakukan konfirmasi ulang hingga menu yang diinpu sesuai. Kemudian pada product akan diperbarui stok barangnya dengan memanggil method addToInventory() pada class Product.

* 1. Buat metode yang akan mengurangi nilai stok untuk setiap produk yang diidentifikasi.
     1. Ikuti prosedur yang sama seperti yang Anda lakukan untuk menambah stok tetapi beri nama metode deductInventory. Itu batasan inputnya adalah nilainya harus 0 atau lebih besar dan tidak boleh lebih besar dari arus jumlah stok untuk produk tersebut. Semua masukan harus memiliki penanganan kesalahan dan kesalahan yang sesuai pesan. Gunakan metode deductFromInventory untuk melakukan perubahan pada objek produk di Himpunan.

**Pembahasan :**

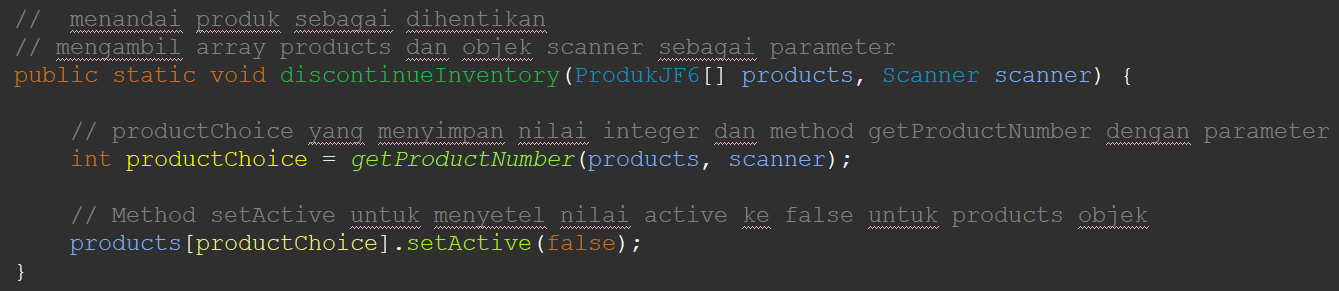


**Gambar 2.51** Kode Program Java Menambah Stok Produk

Kode Java tersebut adalah method deductInventory() dengan array products dan objek scanner sebagai parameter. Method ini digunakan untuk mengurangi stok produk yang dipilih. Pada blok program, pengguna diminta untuk menginputkan berapa unit atau stok yang ingin dikurangi. Input yang dapat diterima hanya nilai positif atau tidak lebih dari nilai jumlah unit pada produk. Jika nilai negatif (kurang dari 0) atau lebih dari nilai jumlah unit pada produk, maka input akan invalid dan program akan memaksa pengguna untuk melakukan konfirmasi ulang hingga menu yang diinpu sesuai. Kemudian pada product akan diperbarui stok barangnya dengan memanggil method deductInventory() pada class Product.

* 1. Opsi menu terakhir untuk diterapkan adalah kemampuan untuk menandai stok sebagai dihentikan.
     1. Metode ini harus disebut discontinueInventory, tidak memiliki nilai kembalian dan mengambil array produk dan objek Pemindai sebagai parameter. Itu harus memiliki satu variabel lokal bernama productChoice yang menyimpan nilai integer dan tidak perlu diinisialisasi. Tulis kode di bawah main.
     2. Tambahkan pemanggilan metode ke metode getProductNumber dengan meneruskan parameter yang benar dan menyimpan menghasilkan variabel productChoice.
     3. Sekarang gunakan metode setActive untuk menyetel nilai active ke false untuk objek produk yang dipilih.

**Pembahasan :**



**Gambar 2.52** Kode Program Java Menandai Produk untuk Dihentikan

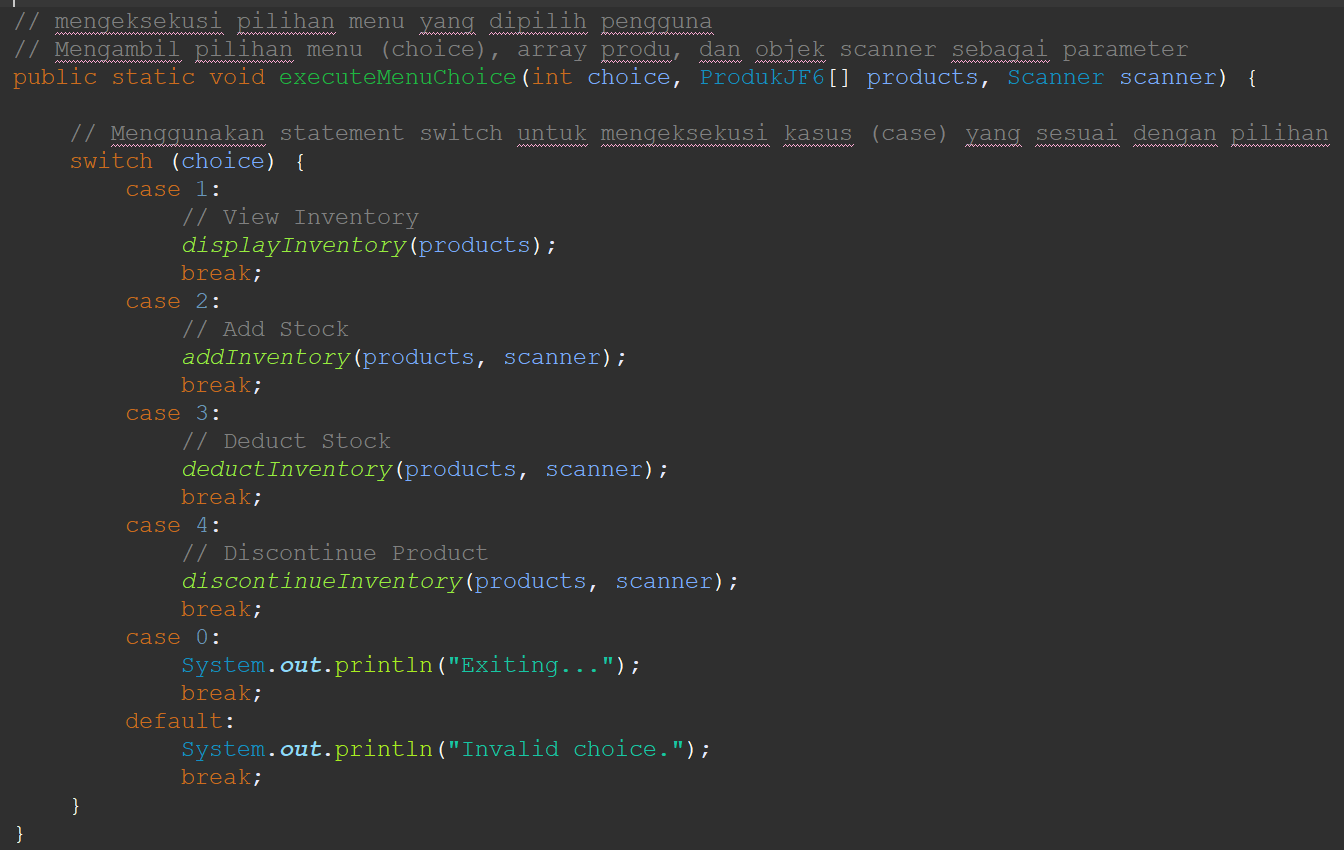
Kode Java tersebut adalah method discontinueInventory() dengan array products dan objek scanner sebagai parameter. Variabel productChoice bertipe data integer akan menyimpan method getProductNumber() dengan parameternya. Kemudian pada product akan dilakukan perbaruan pada nilai active ke false untuk objek products dengan memanggil method setActive() dengan diisikan parameter nilai boolean FALSE.

* 1. Anda sekarang perlu membuat metode yang akan menyatukan semuanya.
     1. Metode ini harus disebut executeMenuChoice, tidak memiliki nilai kembalian dan mengambil pilihan menu, array produk dan objek Pemindai sebagai parameter. Itu tidak memerlukan variabel lokal apa pun. Menulis kode di bawah main.
     2. Gunakan pernyataan switch untuk mengeksekusi metode yang telah Anda buat dalam latihan ini. Untuk setiap pernyataan kasus menggunakan pernyataan keluaran yang akan menampilkan salah satu dari judul berikut sebelumnya menjalankan metode yang sesuai:

View Product List Add Stock

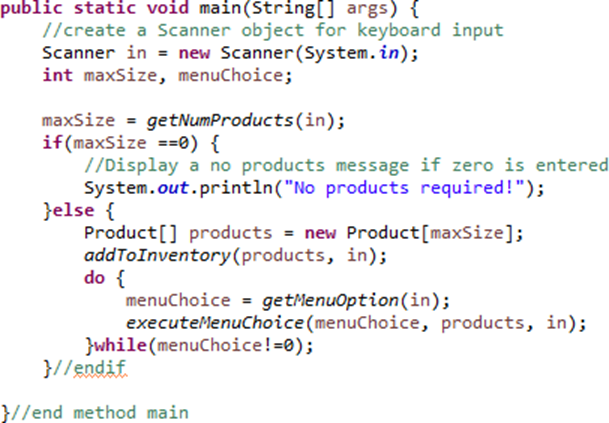
Deduct Stock Discontinue Stock

**Pembahasan :**

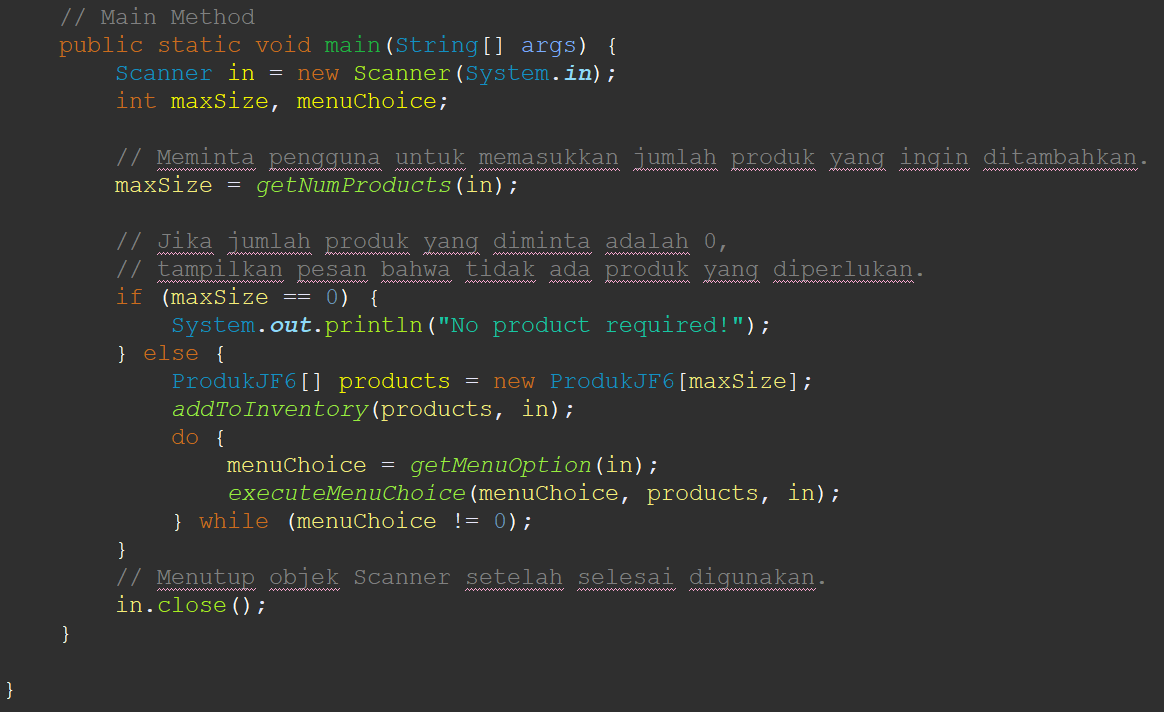
 **Gambar 2.53** Kode Program Java Menandai Produk untuk Dihentikan

Kode Java tersebut adalah method executeMenuChoice() dengan variabel choice, array products, dan objek scanner sebagai parameter. Method ini digunakan mengeksekusi pilihan menu yang dipilih pengguna. Pada blok program, terdapat statement switch yang digunakan untuk dapat mengeksekusi kasus (case) yang sesuai dengan pilihan menu.

1. Tahap terakhir dari latihan ini adalah memperbarui metode utama untuk memanfaatkan fungsionalitas baru. Perbarui Anda kode sehingga metode utama Anda cocok dengan kode berikut:



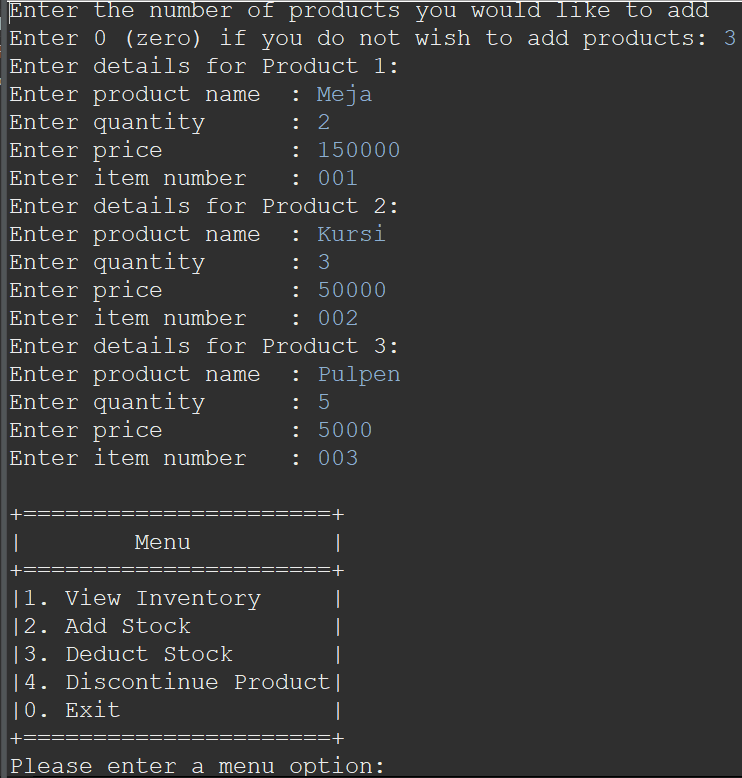
**Pembahasan :**



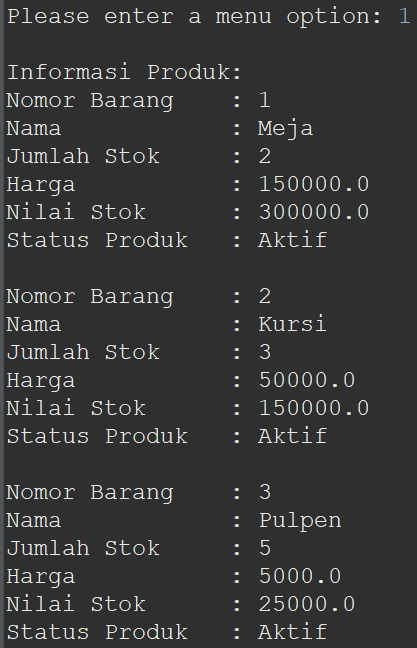
**Gambar 2.54** Kode Program Java Main Method

1. Jalankan dan uji kode Anda.

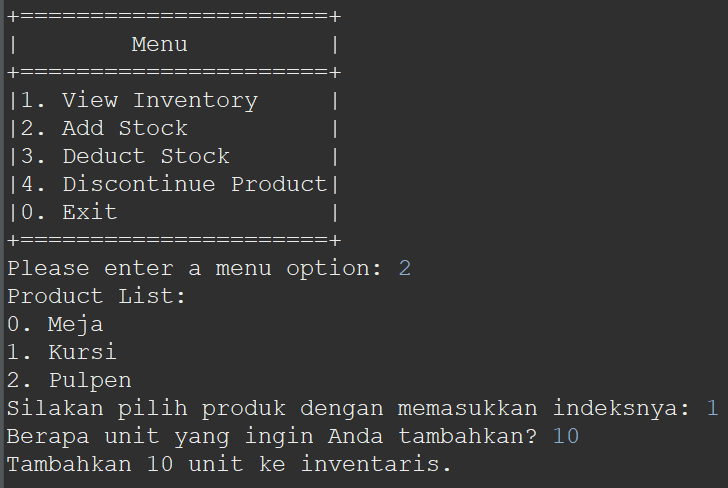
**Pembahasan :**



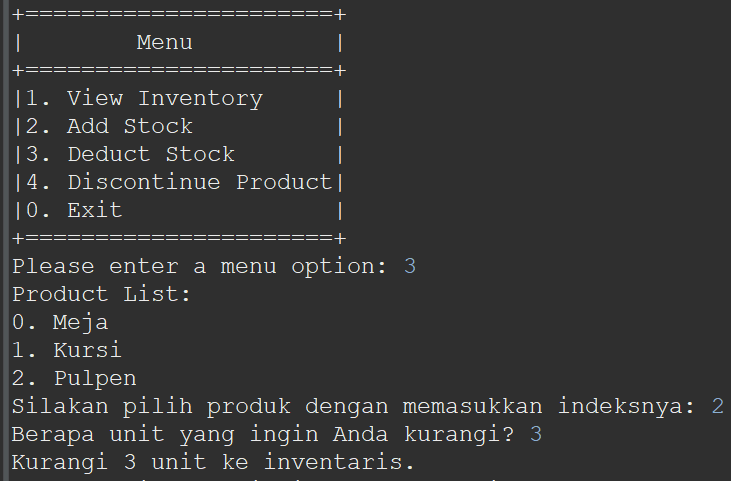
**Gambar 2.55** Output menambahkan product dan langsung ke menu



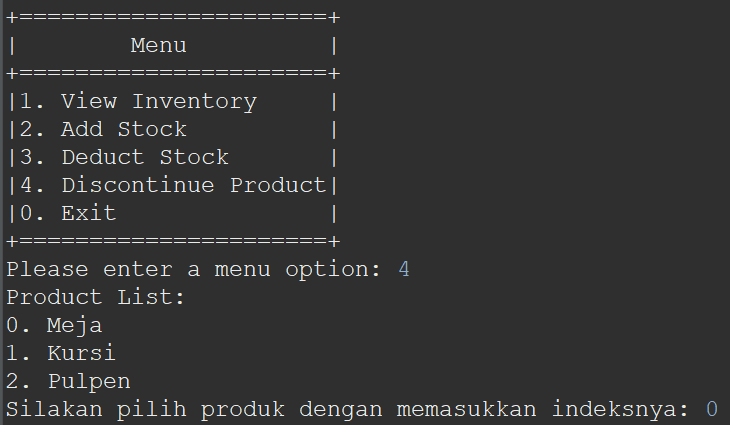
**Gambar 2.56** Output melihat informasi produk



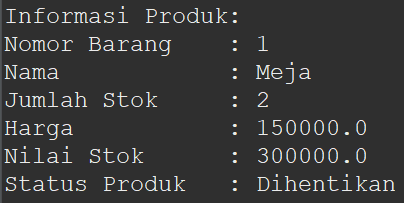
**Gambar 2.57** Output menambah stok barang



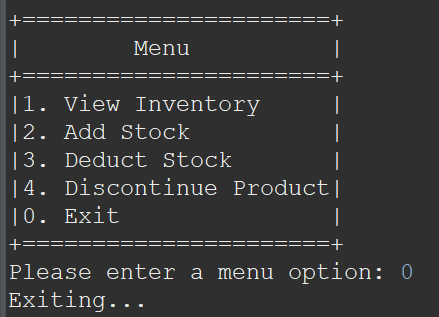
**Gambar 2.58** Output mengurangi stok barang



**Gambar 2.59** Output Menghentikan Produk



**Gambar 2.60** Output Menghentikan Produk



**Gambar 2.60** Output Keluar Menu

* 1. Simpan proyek Anda.