LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK



Laporan Untuk Melengkapi Tugas Terstruktur Mata Kuliah Pemrograman Berorientasi Object

Dosen Pengajar

Imron Rosadi, MM

Disusun oleh :

TEKNIK INFORMATIKA-3A

Elmaida Khoirotuzzuhria (201869040003)

**UNIVERSITAS YUDHARTA** **PASURUAN**

Jl.Yudharta No.7 (Pondok Pesantren Ngalah)

Sengonagung Purwosari Pasuruan Telp.(0343)611186

**2019**

**KATA PENGANTAR**

Segala puji dan syukur kami panjatkan kehadirat Allah.S.W.T atas hidayahnya, sehingga kami dapat menyelesaikan laporan praktikum pada mata kuliah pemrgraman berorientasi objek dengan baik, penulisan laporan ini dibuat agar dapat memenuhi salah satu syarat untuk mendapatkan nilai tambahan serta dalam mengikuti ujian semester. Dengan sepenuh hati penulis menyadari bahwa tersusunnya tugas ini berkat bantuan dan bimbingan dari semua pihak baik berupa material, spiritual, maupun informasi. Oleh karena itu, dalam kesempatan ini penulis tidak lupa mengucapkan terima kasih sebanyak-banyaknya kepada:

1. Kepada Tuhan Yang Maha Esa, karena dengan anugerahnya penulis dapat menyelesaikan laporan praktikum ini dengan baik.
2. Orang tua saya yang telah memberikan semangat dan dukungan sehingga laporan praktikum ini dapat terselesaikan dengan baik serta mendapatkan nilai yang diinginkan.
3. Bapak Imron Rosadi, MM selaku dosen pemrograman berorientasi objek
4. Semua pihak yang telah membantu dalam penyusunan laporan ini.

Akhirnya penulis mohon maaf apabila ada kekurangan atau kesalahan dalam penyusunan laporan praktikum ini. Semoga laporan praktikum ini dapat bermanfaat bagi penulis, maupun pembacanya.

Pruwosari, 05 desember 2019

Penulis

Elmaida Khoirotuzzuhria

**DAFTAR ISI**

**HALAMAN JUDUL i**

**KATA PENGANTAR ii**

**DAFTAR ISI iii**

**BAB I PENDAHULUAN 1**

1. Latar Belakang 1
2. Tujuan 1
3. Manfaat 2

**BAB II LANDASAN TEORI 3**

A. Java OOP 3

A.1 Class dan Objects 3

B. Java Classes/Objects 4

B.1 Class 4

B.2 Objects 5

B.2.1 Buat Object 5

B.2.2 Banyak Object 5

B.2.3 Menggunakan Multiple Object 6

C. Java Class Attributes 7

C.1 Java Class Attributes 7

C.2 Accessing Attributes 7

C.3 Modify Attributes 7

C.4 Multiple Objects 8

D. Java Class Methods 9

D.1 Static vs Non-Static 10

D.2 Access Methods with Object 11

E. Java Constructors 12

E.1 Constructors Parameters 13

F. Java Modifiers 13

F.1 Macam-macam Modifiers dalam Java 18

G. Java Encapsulation 19

G.1 Get and Set 19

H. Java Package/API 20

H.1 Paket Built-In 21

H.1.1 Import Class 21

H.1.1.1 Class Scanner 21

H.1.1.2 Buffer Reader 22

H.1.1.3 Class JOptionPane 23

H.1.1.4 Class Console 24

H.1.2 Import Package 25

H.2 Daftar Package pada Program Java 25

I. Java Inheritance (Pewarisan) 26

J. Java Polymorphism 27

K. Java Inner Class 29

K.1 Static Nested Class (Kelas Bersarang Statis) 29

K.2 Java Inner Class 30

K.2.1 Private Inner Class 31

K.2.2 Static Inner Class 32

K.2.3 Access Outer Class dari Inner Class 32

L. Java Abstraksi 33

L.1 Java Abstract Classes dan Methods 33

M. Java Interface 35

N. Java Enum 38

N.1 Enum dalam Pernyataan SWITCH-CASE 39

N.2 Loop Melalui Enum 40

O. Java User Input (Scanner) 41

O.1 Jenis Input 42

P. Java Date 44

P.1 Menampilkan Tanggal Saat Ini 44

P.2 Menampilkan Waktu Saat Ini 44

P.3 Menampilkan Tanggal dan Waktu Saat Ini 45

P.4 Memformat Tanggal dan Waktu 45

Q. Java ArrayList 48

Q.1 Membuat ArrayList dan Menambahkan Elemen Baru ke Dalamnya 48

Q.2 Mengakses Elemen dari ArrayList 49

Q.3 Mengubah Elemen pada ArrayList 50

Q.4 Menghapus Elemen pada ArrayList 50

Q.5 Mengetahui Ukuran ArrayList 52

Q.6 Loop Through ArrayList 52

Q.7 Tipe yang Lain 53

Q.8 Sortir Daftar Array 53

R. Java HashMap 54

R.1 Menambahkan Item pada HashMap 55

R.2 Mengakses Item pada HashMap 55

R.3 Menghapus Item pada HashMap 55

R.4 Mengetahui Ukuran HashMap 56

R.5 Loop Through HashMap 56

S. Java Wrapper Classes 56

S.1 Membuat Object Wrapper 57

T. Java Exceptions 58

T.1 Java Try and Catch 59

T.2 Finally 59

T.3 Kata Kunci Throw 60

**BAB III HASIL PERCOBAAN PRAKTIKUM 61**

1. Java Classes and Object 61

A.1 Membuat Object 61

A.2 Membuat Banyak Object 61

A.3 Membuat Banyak Class 62

B. Java Class Attributes 62

B.1 Accessing Attributes 62

B.2 Modify Attributes 62

B.3 Multiple Objects 63

B.4 Multiple Attributes 64

C. Java Class Methods 64

C.1 Static Vs Non-Static 64

C.2 Access Method with Objects 65

D. Java Constructor 66

D.1 Constructor with Parameters 66

E. Java Modifiers 67

E.1 Static 67

E.2 Abstract 67

F. Java Encapsulations 68

F.1 Get and Set 68

G. Java Package / API 68

G.1 Class Scanner 68

G.2 Buffer Reader 69

G.3 JOption Pane 72

H. Java Inheritance 73

I. Java Polymorphism 73

J. Java Inner Class 74

J.1 Inner Class Static 74

J.2 Inner Class Non- Static 75

K. Java Abstraksi 76

L. Java Interface 77

M. Java Enum 77

N. Java User Input 78

O. Java Date 79

O.1 Menampilkan Tanggal Saat Ini 79

O.2 MenampilkanWaktu Saat Ini 80

O.3 Menampilkan Waktu dan Tanggal Saat Ini 80

O.4 Memformat Tanggal dan Waktu 81

P. Java ArrayList 83

P.1 Menambahkan Item 83

P.2 Mengakses Item 83

P.3 Mensortir Daftar Array 84

Q. Java HashMap 84

Q.1 Menambahkan Item 84

R. Java Wrapper 85

R.1 Membuat Object 85

S. Java Exceptions 86

**BAB IV STUDI KASUS 87**

1. Java Class and Objects 87
2. Java Class Attributes 87
3. Java Class Methods 88
4. Java Constructors 89
5. Java Modifiers 90

E.1 Public 90

E.2 Protected 91

E.3 Private 92

1. Java Encapsulation 93
2. Java Package / API 94

G.1 Class Scanner 94

G.2 Buffer Reader 96

G.3 Class Console 98

G.4 JOptionPane 99

1. Java Inheritance 100
2. Java Polymorphism 104
3. Java Inner Class 105

J.1 Static 105

J.2 Non-Static 106

1. Java Abstraksi 108
2. Java Interface 108
3. Java Enum 109
4. Java User Input (Scanner) 111
5. Java Date 112
6. Java ArrayList 116
7. Java HashMap 117
8. Java Wrapper 119
9. Java Exception 119

**BAB V PENUTUP 122**

1. Kesimpulan 122
2. Saran 122

DAFTAR PUSTAKA 123

**BAB I**

**PENDAHULUAN**

1. **Latar Belakang**

Pemrograman Berorientasi Objek (*Object Oriented Programming* atau OOP) merupakan paradigma pemrograman yang berorientasikan kepada objek. Objek adalah struktur data yang terdiri dari bidang data dan metode bersama dengan interaksi mereka untu merancang aplikasi dan program computer. Semua data dan fungsi di dalam paradigm ini dibungkus dalam kelas-kelas atau objek-objek. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya. Pada jaman sekarang, banyak bahasa pemrograman yang mendukung OOP.

OOP adalah paradigma pemrograman yang cukup mendominasi saat ini, karena mampu memberikan solusi kaidah pemrograman modern. Meskipun demikian, bukan berarti bahwa pemrogaman procedural sudah tidak layak lagi. OOP diciptakan karena dirasakan masih adanya keterbatasan pada bahasa pemrograman tradisional. Konsep OOP sendiri adalah semua pemecahan masalah dibagi kedalam objek. Dalam OOP data dan fungsi-fungsi yang akan mengoperasikannya digabungkan menjadi satu kesatuan yang dapat disebut sebagai objek. Pembahasan mengenai orientasi objek tidak akan terlepas dari konsep objek seperti *inheritance* atau pewarisan, *encapsulation* atau pembungkusan, dan *polymorphism* atau kebanyakrupaan. Konsep-konsep ini merupakan fundamental dalam orientasi objek yang perlu dipahami serta digunakan dengan baik, dan menghindari penggunaanya yang tidak tepat.

OOP lebih mudah dipelajari bagi pemula dibandingkan dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

1. **Tujuan Praktikum**
2. Untuk melengkapi nilai tugas mata kuliah Pemrograman Berorientasi Objek
3. Mampu membuat program java pada Netbeans berdasarkan aturan OOP
4. Untuk mengkaji kembali pengetahuan yang telah didapatkan pada kuliah Pemrograman Berorientasi Objek
5. **Manfaat**
6. Dapat melengkapi nilai tugas mata kuliah Pemrograman Berorientasi Objek
7. Dapat membuat program Java pada Netbeans berdasarkan ilmu aturan OOP
8. Dapat mengkaji kembali pengetahuan yang telah didapatkan pada kuliah Pemrograman Berorientasi Objek

**BAB II**

**LANDASAN TEORI**

1. **Java OOP**

OOP adalah singkatan dari Object Oriented Programming (Pemrograman Berorientasi Objek).

Pemrograman procedural adalah tentang prosedur atau metode penulisan yang melakukan operasi pada data, sedangkan pemrograman berorientasi objek adalah tentnag membuat objek yang berisi data dan metode.

Pemrograman berorientasi objek memiliki beberapa keunggulan dibandingkan pemrograman procedural:

1. OOP lebih cepat dan lebih mudah untuk dieksekusi
2. OOP menyediakan struktur yang jelas untuk program-program tersebut
3. OOP membantu menjaga kode java KERING “Don’t Repeat Yourself”, dan membuat kode lebih mudah untuk mempertahankan, memodifikasi dan men-debug
4. OOP memungkinkan untuk membuat aplikasi yang dapat digunakan kembali secara penuh dengan kode yang kebih sedikit dan waktu pengembangan yang lebih singkat

Secara garis besar, suatu bahasa pemrograman dapat dikatakan sebagai bahasa pemrograman berorientasi objek apabila ia mendukung konsep abtraksi (abstraction), enkapsulasi (encapsulation), polimorfisme (polymorphism), dan pewarisan (inheritance). Selain konsep – konsep ini, ada beberapa konsep fundamental lainnya seperti, kelas, objek dan message.

**A.1 Class dan Objects**

Class dan objek adalah dua aspek utama dari pemrograman berorientasi objek. Di dalam sebuah pemrograman pasti terdapat object dan class. Untuk memudahkan perbedaan antara kelas dan objek, mari lihat contoh dibawah ini:

|  |  |
| --- | --- |
| **Kelas** | **Object** |
| Buah | Apel  Pisang  Mangga |

Jadi, kelas adalah tempalate untuk objek, dan objek adalah turunan dari kelas. Ketika objek individu dibuat, mereka mewarisi semua variabel dan metode kelas.

1. **Java Classes / Objects**

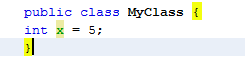
Java adalah bahasa pemrograman berorientasi objek. Segala sesuatu di java dikaitkan dengan kels dari objek, bersama dengan atribut dan metodenya. Contoh , di dalam kehidupan nyata, mobil adalah objek. Mobil memiliki atribut, seperti berat, dan warna, dan metode nya, seperti drive,dan rem.

**B.1 Class**

Class mendefinisikan karakteristik-karakteristik abstrak dari sesuatu (objek), termasuk karakteristik dan perilaku dari objek itu sendiri. Class dapat dilustrasikan sebagai sebuah cetak biru (blueprint), prototype, atau pabrik (factory) yang berfungsi untuk menghasilkan objek-objek.



Untuk membuat class, gunakan kata kunci **class**. Dibawah ini merupakan contoh membuat class baru bernama “**MyClass**” dengan variabel x.



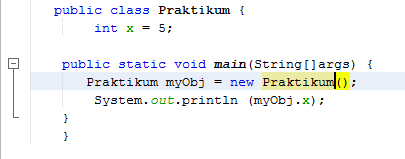
**B.2 Object**

**B.2.1 Buat Object**

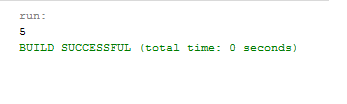
Di java, object dibuat dari class, dalam contoh ini telah dibuat class bernama **Praktikum,** jadi sekarang bisa digunakan untuk membuat object. Untuk membuat object **Praktikum,** tentukan nama class, diikuti nama object, dan gunakan kata kunci **new**:

Contoh:

Buat object yang disebut “**myObj”** dan cetak nilai x:

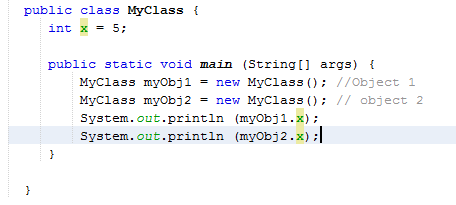


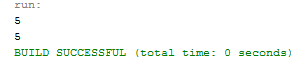
Akan didapat hasil running seperti ini:



**B.2.2 Banyak Object**

Ketika membuat sebuah program, dapat membuat beberapa objek dari satu kelas. Berikut adalah salah satu contoh program yang membuat banyak objek dalam satu kelas:

 Jika menjalankan proses running, akan didapat hasil seperti gambar berikut:

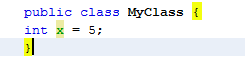


**B.2.3 Menggunakan Multiple Objects**

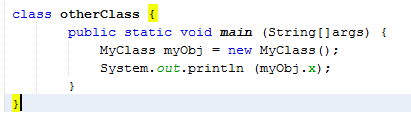
Ketika membuat sebuah objek, disini bisa juga untuk membuat objek kelas dan mengaksesnya di kelas lain. Ini sering digunakan untuk organisasi kelas yang lebih baik (satu kelas memiliki semua atribut dan metode, sedangkan kelas lainnya untuk **main( )** metode (kode yang akan diekseskusi)).

Perlu diingat bahwa nama file java harus sesuai dengan nama kelas. Dalam contoh ini, telah dibuat dua file direktori/pada folder yang sama:

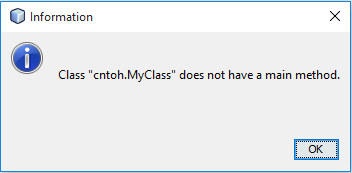
1. MyClass.java
2. otherClass.java
3. **MyClass.java**



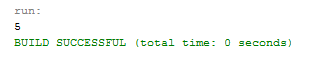
1. **otherClass.java**



Ketika dilakukan proses running pada class MyClass.java, maka akan mucul kotak dialog seperti gambar dibawah ini:



Munculnya kotak dialog tersebut, dikarenakan pada class MyClass.java tidak mempunyai main( ) metode. Namun, berbeda ketika melakukan proses running pada class otherClass.java. berikut adalah gambar dari hasil running class otherClass.java:

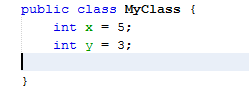


1. **Java Class Attributes**

**C.1 Java Class Attributes**

Pada bab Java Classes and Objects, digunakan istilah variabel untuk x dalam contoh seperti gambar dibawah. Ini sebenarnya adalah atribut dari class atau bisa dikatakan bahwa atribut kelas adalah variabel dalam class.

Untuk membuat class attribute, dibuat class yang disebut MyClass dengan dua atribut x dan y:

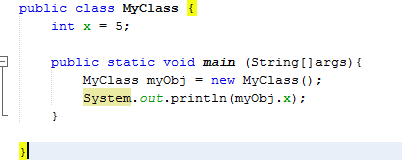


**C.2 Accessing Attributes**

Untuk mengakses atribut, bisa juga dengan membuat object class, dan dengan menggunakan sintaks dot (.):

Contoh berikut akan dibuat object MyClass class, dengan nama myObj. Digunakan “x” atribut pada object untuk mencetak nilaiya:

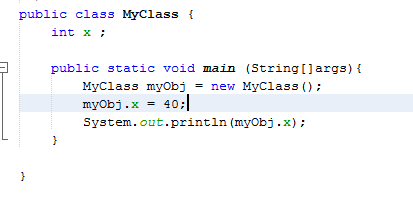
Berikut adalah program untuk membuat object “myObj” dan mencetak nilai x:



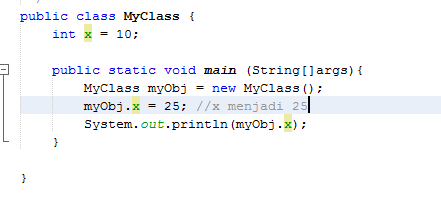
**C.3 Modify Attributes**

Untuk mengubah nilai attribute, bisa juga melihat source code pada contoh dibawah ni:

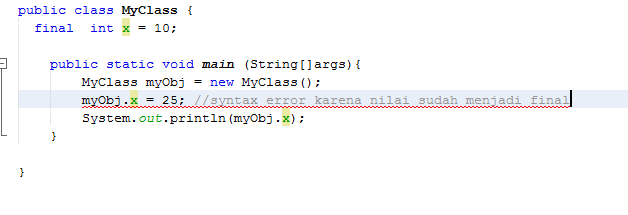
Nilai x = 40



Untuk menimpa nilai yang ada dengan mengubah nilai x menjadi 25:



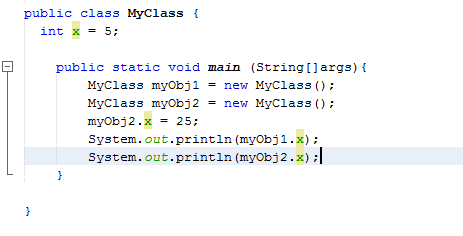
Jika tidak ingin menimpa nilai yang ada, maka nyatakan atribut sebagai final:



**C.4 Multiple Objects**

Ketika akan membuat beberapa object dari satu class, bisa juga mengubah nilai atribut di satu object, tanpa memengaruhi nilai atribut yang lain:

Berikut adalah mengubah nilai x ke 25 di myobj2, dan meninggalkan x di myObj1 berubah menjadi:



1. **Java Class Methods**

Method merupakan suatu operasi berupa fungsi-fungsi yang dapat dikerjakan oleh suatu object. Method didefinisikan pada class akan tetapi dipanggil melalui object. Metode menentukan perilaku objek, yakni apa yang terjadi ketika objek itu dibuat serta berbagai operasi yang dapat dilakukan objek sepanjang hidupnya. Ada 4 (Empat) bagian dasar yang dimiliki metode antara lain:

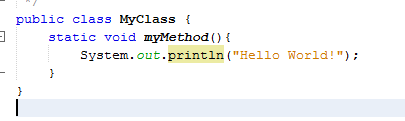
1. Nama Method
2. Tipe Objek atau Tipe primitive yang di kembalikan ke objek
3. Daftar Parameter
4. Badan atau Isi Method

Tiga bagian pertama mengindikasikan informasi penting tentang metode itu sendiri. Dengan kata lain, nama metode tersebut=metode lain dalam program. Dalam java kita dapat memiliki metode-metode berbeda yang memiliki nama sama tetapi berbeda tipe kembalian atau daftar argumennya, sehingga bagian-bagian definisi metode ini menjadi penting. Ini disebut overloading metode(proses yang berlebihan dalam suatu mtehod).

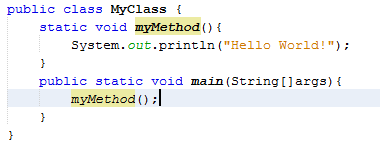
Untuk menjalankan program yang memiliki sifat polymorphism tersebut, diperlukan suatu kemampuan overloading, yaitu suatu kemampuan untuk menentukan fungsi yang mana yang harus digunakan atau dijalankan jika terdapat nama fungsi yang sama. Polimorfisme bisa diartikan seperti kemampuan suatu variable untuk mengubah perangai sesuai dengan objek hasil instansiasi yang digunakan. Polimorfisme membiarkan lebih dari 1 objek dari sub class sub class dan diperlakukan sebagai objek dari super class tunggal.

Metode harus dideklarasikan di dalam kelas, yang didefinisikan dengan nama metode, diikuti oleh tanda kurung (). Java menyediakan beberapa metode yang telah ditentukan, seperti System.out.println(), tetapi juga dapat membuat metode sendiri untuk melakukan tindakan tertentu.

Contoh:



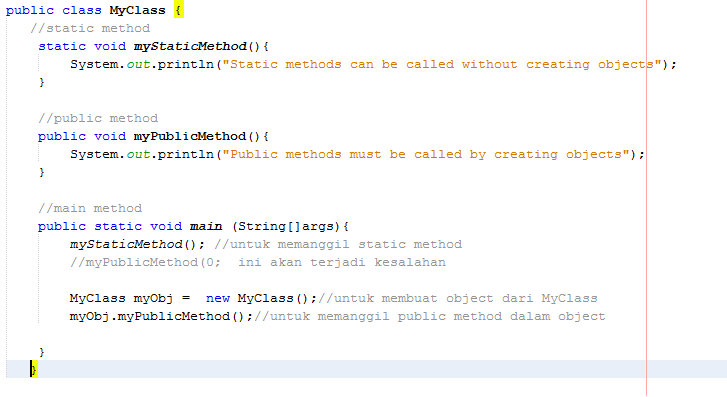
myMethod() mencetak teks (tindakan), ketika dipanggil. Untuk memanggil metode, bisa dituliskan nama metode diikuti oleh dua tanda kurung() dan tanda titik koma ; pada akhir metode. Jika menggunakan **main**  maka metode dipanggil pada class **main**



**D.1 Static vs Non-Static**

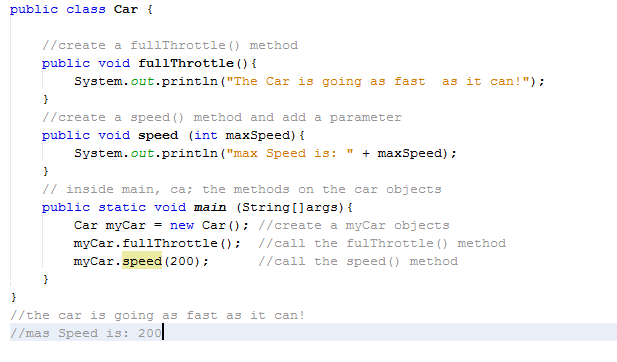
Sering dijumpai pada program java yang memiliki atribut static atau public metode. Dalam contoh diatas, dibuat static metode, yang berarti dapat diakses tanpa membuat object class terlebih dahulu, tidak seperti public metode, yang hanya dapat diakses oleh object.

Contoh untuk menunjukkan perbedaan antara static dan public metode:



**D.2 Access Methods With Object**

Untuk mengakses methods dengan object, bisa dilihat dalam contoh dengan membuat class Car, dan membuat object bernama myCar. Untuk menjalankan programnya maka panggil fullThrottle() dan speed() metode pada object.

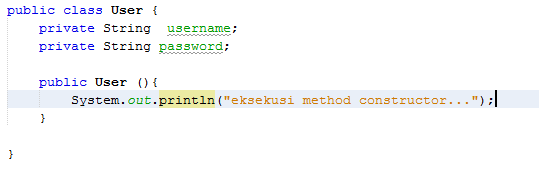


Untuk lebih jelasnya, terdapat keterangan untuk menjelaskan source cpdepada contoh diatas:

1. Dibuat kelas Car khusus dengan menggunakan kata kunci “Class”
2. Untuk program java “Car” diciptakan fullThrottle() dan speed() metode pada class “Car”.
3. Metode fullThorttle() dan speed() akan mencetak beberapa teks, ketika dipanggil.
4. Metode speed() ini menerima int parameter yang disebut dengan max-Speed
5. Untuk menggunakan class ”Car” dan metode nya, perlu dibuat object dari class “Car”
6. Buka main() method, untuk menjalankan program java
7. Dengan meggunakan kata kunci “new”, dibuat class “car” bernama “myCar”.
8. Selanjutnya memanggil metode fullThrottle() dan speed() pada “myCar” object, dan menjalankan program menggunakan nama object “myCar” diikuti oleh titik(.) dan diikuti oleh metode (fullThrottle(); dan speed(200);) . perhatikan bahwa telah ditambahkan int parameter 200 di dalam metode speed().
9. **Java Constructors**

Constructor adalah method khusu yang akan diekseskusi pada saat pembuatan object (instance). Biasanya method ini digunakan untuk inisialisasi atau mempersiapkan data untuk object.

Berikut adalah contoh constructor:



Cara membuat contructor adlah dengan menuliskan nama method constructor yang sama seperti nama class. Pada contoh diatas, dituliskan contructor:

“Public User ( ) {

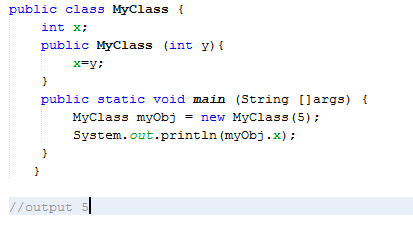
System.out.println(“ekseskusi method constructor…”); }

Pastikan memberikan modifier “public” kepada constructor, karena akan diekseskusi saat pembuatan object (instance).

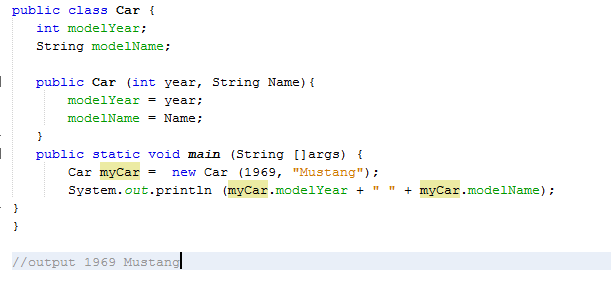
**E.1 Constructors Parameter**

Constructors biasanya digunakan untuk initialize (menyiapkan) data untuk class. Untuk membuat ini, terlebih dahulu membuat parameter sebagai inputan constructor. Contoh berikut menambahkan “int y” parameter ke constructor. Di dalam constructoratur x ke y (x = y). ketika memanggil constructor, dapat juga meneruskan parameter ke constructor (5), yang akan menetapkan nilai x ke 5.

Berikut adalah contoh constructor parameter dengan satu parameter:



Contoh constructors dengan banyak parameter:



1. **Java Modifiers**

Class dalam program java dapat saling berhubungan dengan cara memberikan akses terhadap member mereka. Salah satu hubungan tersebut erring disebut dengan inheritance (pewarisan). Semua yang ada di dalam class (atribut dan method) disebut member. Biasanya aka nada tingkatan akses yang disebut dengan **modifier.**

Pada hubungan inheritance, semua member di dalam class induk bisa diakses oleh class anak (subclass), kecuali member tersebut diberikan modifier **private**.

Modifier tidak hanya bisa diberikan kepada member saja, tetapi juga bisa diberikan kepada interface, enum, atau class itu sendiri.

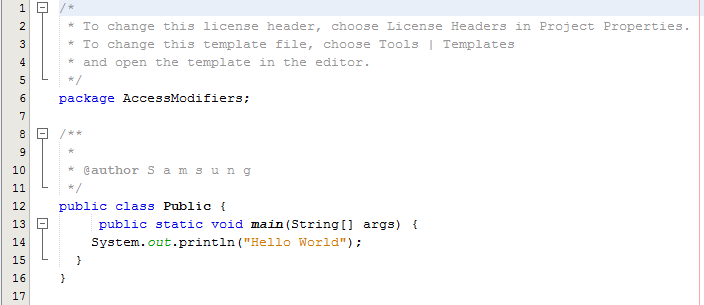
Untuk lebih mudahnya, modifiers dibagi menjadi dua kelompok:

1. Access Modifiers : untuk mengontrol level akses
2. Non-Access Modifiers : tidak mengontrol level akses, tetapi hanya menyediakan fungsionalitas lain.
3. Access Modifiers

Untuk class, dapat meggunakan salah satu dari “public” atau “default”:

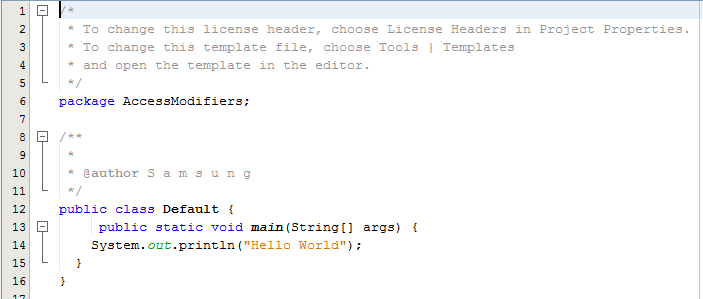
1. Public

Class ini bisa diakses oleh kelas lain. Dibawah ini merupakan contoh dari penggunaan public pada Access Modifiers.



1. Default

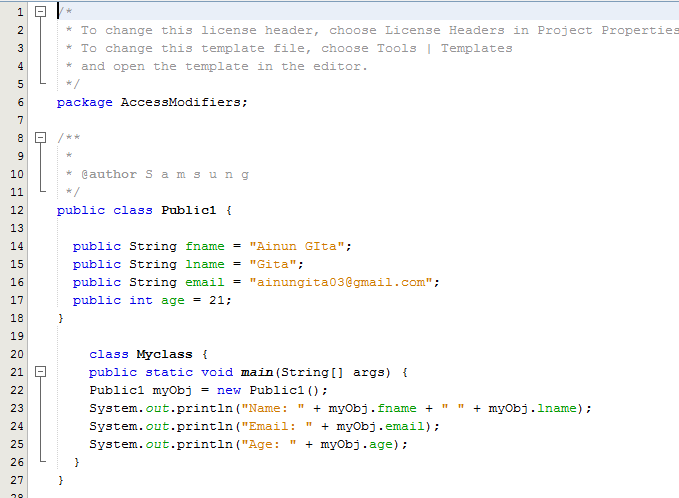
Class ini hanya bisa diakses oleh kelas lain dalam satu paket yang sama. Ini digunakan ketika tidak menetapkan sebuah modifiers. Berikut adalah salah satu contoh default pada Access Modifiers.



Untuk atribut, method, dan constructor,dapat meggunakan modifies seperti public, private, default, protected.

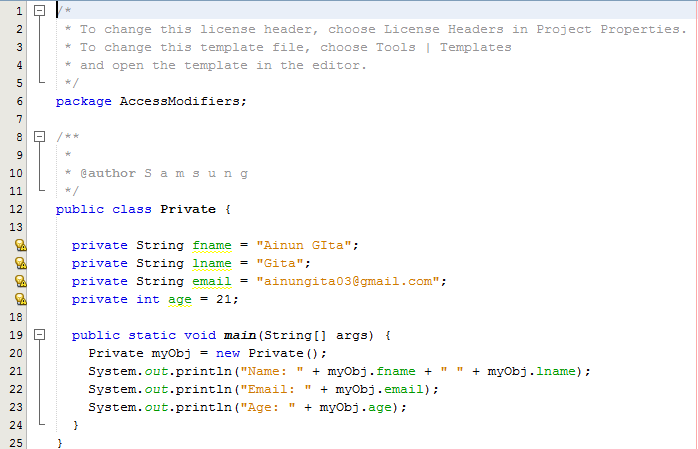
1. Public

Kode ini bisa diakses oleh semua class.



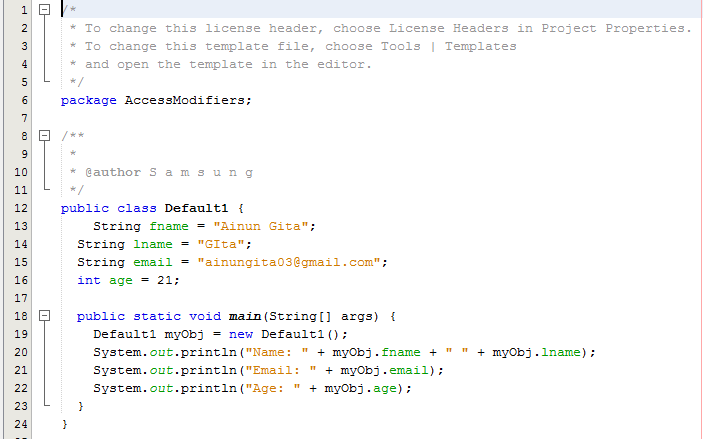
1. Private

Kode ini hanya diakses dalam menyatakan class.



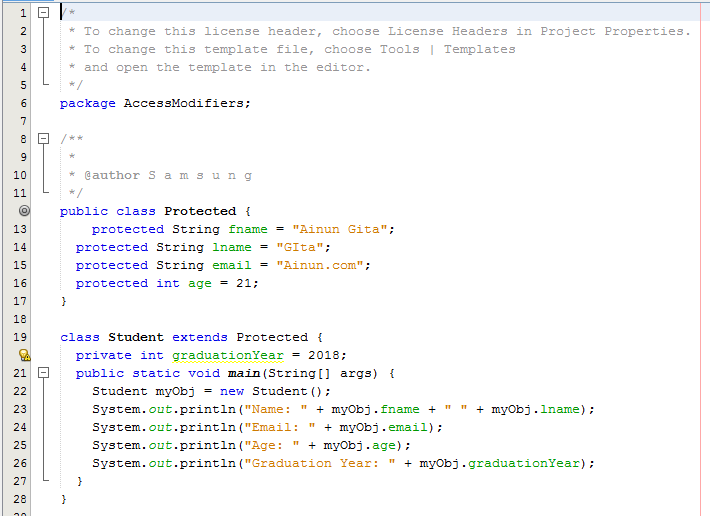
1. Default

Kode ini hanya bisa diakses di dalam paket yang sama. Default digunakan ketika tidak menetapkan sebuah modifier.



1. Protected

Kode ini hanya bisa diakses pada paket yang sama dan pada sub-class.

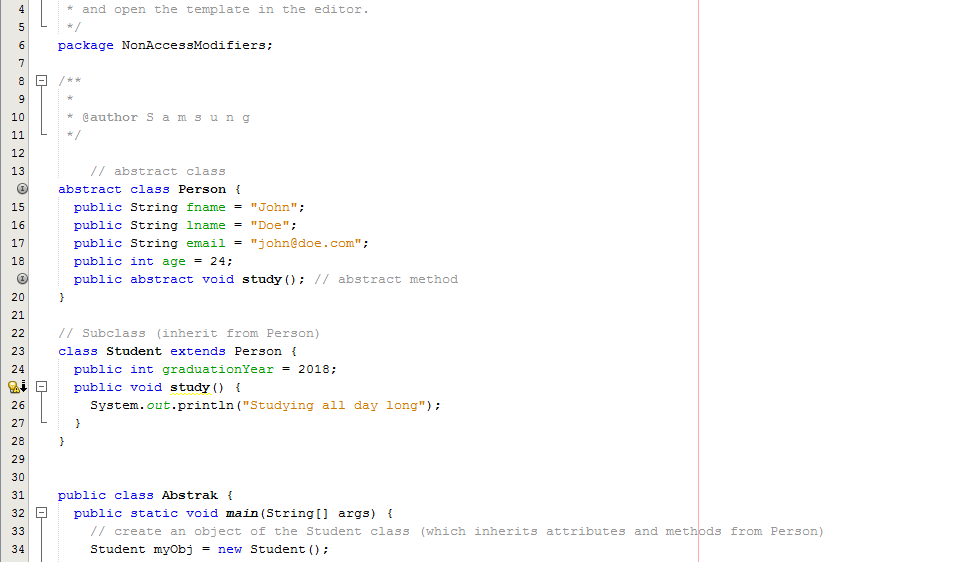
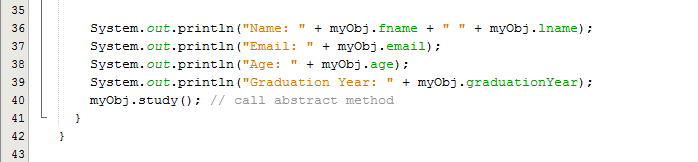


1. Non-Access Modifiers

Untuk modifiers Non-Access pada Class, bisa menggunakan final atau abstrak.

1. Final

Jika menggunakan final, maka Class ini tidak dapat diwarisi oleh kelas lain.

**** 

1. Abstract

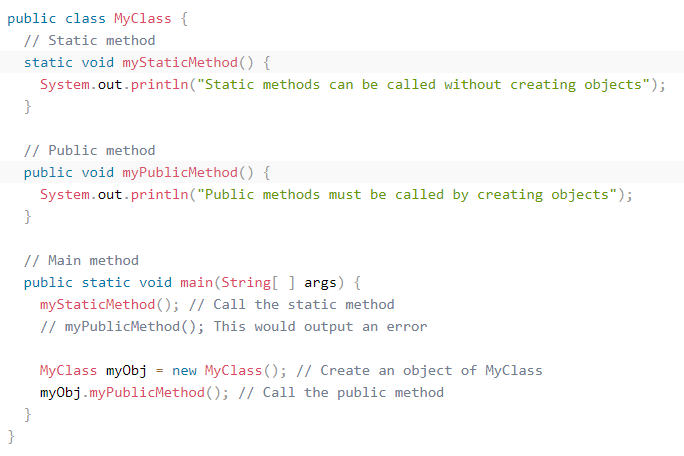
Jika menggunakan abstract, maka class ini tidak dapat membuat object (untuk mengakses class abstrak, harus diwarisi dari class lain).

Untuk atribut dan method dapat menggunakan salah satu dari final, static, abstract, transient, synchronized, atau volatile.

1. Final

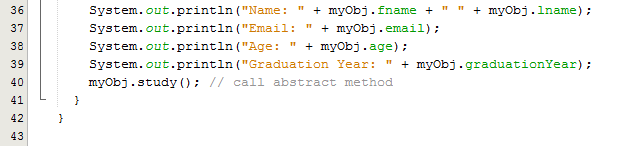
Jika menggunakan final, maka atribut dan method tidak dapat diubah atau diganti.

1. Static

ketika menggunakan static maka aribut dan method itu milik class, bukan dari object.

1. Abstract

Jika menggunakan abstract, maka hanya dapat digunakan dalam class abstract, dan dalam method. Method disini tidak memiliki isi, misalnya abstract void run( );. Isi telah disediakan dari sub-class atau dari pewarisan.

****

1. Transient

Atribut dan method dilewati saat membuat serialisasi dari object yang memuatnya.

1. Synchronized

Metode hanya dapat diakses oleh satu urutan pada satu waktu.

1. Volatile

Nilai atribut tidak disimpan dalam local memory dan selalu dibaca dari main memory.

**F.1 Macam-macam Modifiers dalam Java**

Secara umum ada 3 macam modifier yang digunakan dalam java, yaitu public, rivate, dan protected. Apabila tidak menggunakan tiga kata kunci tersebut, maka class itu tidak menggunakan modifiers. Masing-masing modifiers akan menentukan dimana saja member bisa diakses.

1. Public

Modifier public akan membat member atau Class bisa diakses dari mana saja.

1. Private

Modifier private akan membuat member hanya bisa diakses oleh dari dalam class itu sendiri. Modifier private tidak bisa diberikan kepada class, enum dan interface. Modifier private hanya bisa diberikan kepada member class.

1. Protected

Modifier protected akan membuat member dan class hanya bisa diakses dari class itu sendiri, sub-class atau anak class, package (class yang berada dalam satu package). Modifier protected juga haya boleh digunakan pada member saja.

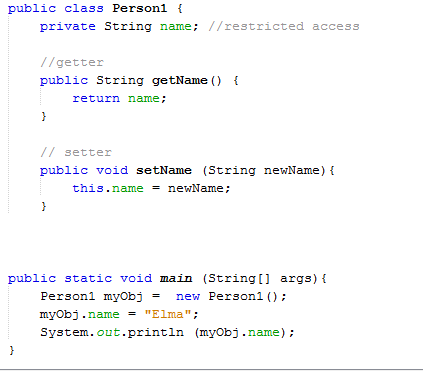
1. **Java Encapsulation**

Enkapsulasi (encapsulation) adalah untuk memastikan bahwa data “sensitive” disembunyikan dari pengguna. Artinya, encapsulation pada java digunakan untuk membungkus class dan menjaga apa saja yang ada dalam class baik itu method maupun attribute agar tidak dapat di akses oleh kelas lain. Untuk mencapai ini maka perlu diperhatikan hal-hal berikut ini :

1. Mendeklarasikan variabel / attribute kelas sebagai **Private**
2. Memberikan metode public **Get** dan **Set**  untuk mengakses dan memperbarui nilai suatu **Private** variabel

**G.1 Get and Set**

Variabel **Private** hanya dapat diakses didalam kelas yang sama (kelas luar tidak memiliki akses). Namun, memungkinkan mengaksesnya jika diberi tambahan metode Public **Get** dan **Set. Get** adalah metode mendapatkan nilai variabel , dan **Set** metode mengatur nilai **syntaks** untuk keduanya adalah bahwa dimulai dengan salah satu **get** atau **set** , diikuti oleh nama variabel , dengan huruf pertama menggunakan huruf kapital. Berikut adalah contoh penggunaan dari **get** and **set:**



Penjelasan dari contoh yaitu method **get** berfungsi untuk mengembalikan nilai variabel **name.** Sedangkan, the **set** method berfungsi untuk mengambil parameter (**newName**) dan penerima ke **name**  variabel. Kata kunci “**this”** digunakan untuk merujuk ke objek saat ini. Namun, ketika variabel **name**  di deklarasikan sebagai **private** , maka tidak dapat diakses dari luar class. Jika tetap mencoba untuk mengakses variabel **private,** maka akan terjadi kesalahan. Sebagai gantinya, maka digunakan method **getName( )** dan **setName ( )**  untuk mengakses dan memeprbarui variable. Tujuan dari Encapsulation antara lain :

1. mengamankan atribut dan metode kelas dengan baik
2. Atribut kelas dapat dibuat hanya-baca (jika hanya menggunakan get method) atau hanya-menulis (jika hanya menggunakan set method)
3. Fleksibel artinya, Programmer dapat mengubah satu bagian kode tanpa mempengaruhi bagian yang lainnya.
4. Peningkatan terhadap keamanan data
5. **Java Packages /API**

Java package digunakan untuk mengelompokkan kelas yang terkait. Anggap saja sebagai folder dalam direktori file. Java package digunakan untuk menghidari kesalahan nama, dan untuk menulis kode yang dapat dikelola dengan lebih baik. Java package dibagi menjadi dua kategori yaitu:

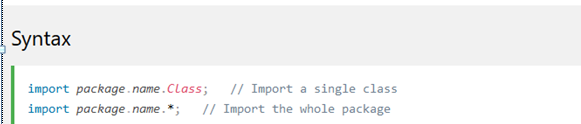
1. Paket Built-in (paket dari java API)
2. Paket buatan pengguna (membuat paket sendiri)

**H.1 Paket built-in**

Java API adalah pustaka class yang sudah ditulis sebelumnya, yang bebas digunakan termasuk dalam Java Development Environment.

Library berisi tentang komponen untuk mengelola input, pemrograman basis data, dan banyak lagi lainnya. Library dibagi mnejadi beberapa **package**  dan **Class,**  yang berarti dapat mengimpor datu class (beserta method dan atributya) atau seluruh paket yang berisi semua class yang termasuk dalam paket yang ditentukan.

Untuk menggunakan class atau package dari library, maka perlu menggunakan kata kunci **import.**



**H.1.1 Import Class**

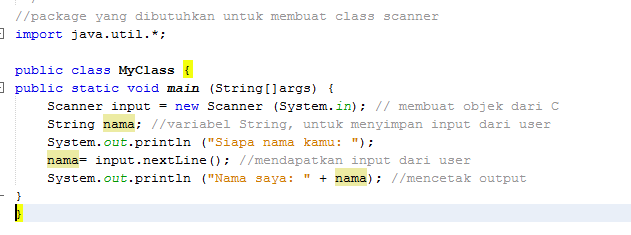
Jika menginginkan untuk mengimpor class dari class yang lain, maka menggunakan Class **Scanner,** untuk mendapatkan input pengguna.

Contoh:

import java.util.Scanner;

Dalam contoh diatas, **java.util** adalah package sedangkan class **Scanner**  dari package **java.util.**

**H.1.1.1 Class Scanner**

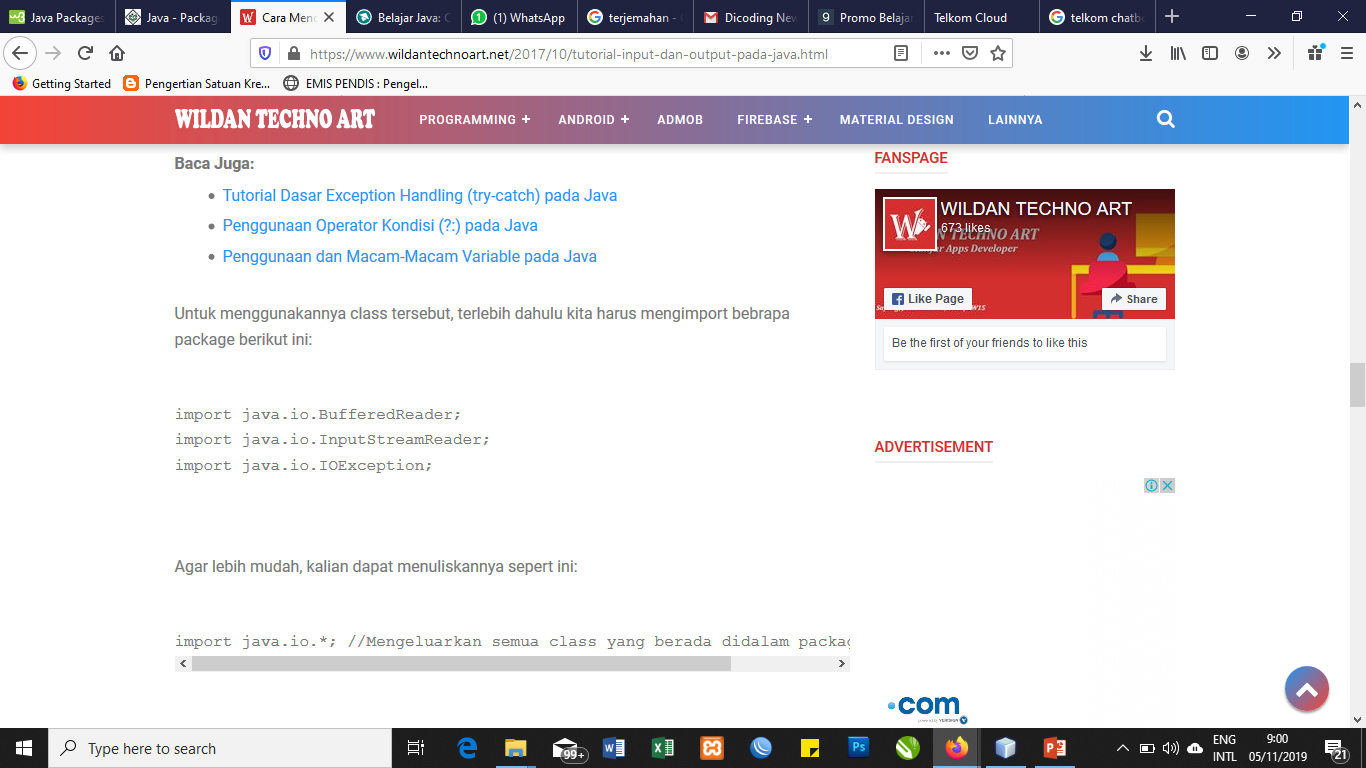
Untuk menggunakan class **Scanner**, maka buat object terlebih dahulu dari class Scanner, class tersebut berada didalam package **java.util.\*;** , jadi perlu untuk mengimpor packagenya terlebih dahulu, setelah package dan object selesai dibuat, selanjutnya membutuhkan sebuah variabel untuk menampung nilai yang di inputkan dari user.

Untuk menyimpan data String, bisa menggunakan method *next()* atau *nextLine()* pada Scanner. Selain tipe data String, ada beberapa method lainnya yang bisa kalian gunakan pada Scanner, diantaranya:

1. **nextInt:**  Menerima input bilangan bulat/Integer
2. **nextBoolean**: Menerima input Boolean (true/false)
3. **nextFloat:** Menerima input pecahan/Float
4. **nextDouble:** Menerima input pecahan/Double
5. **nextByte:** Menerima bilangan bulat/Byte
6. **nextLong:** Menerima bilangan bulat/Long
7. **nextShort:**  Menerima bilangan bulat/Short

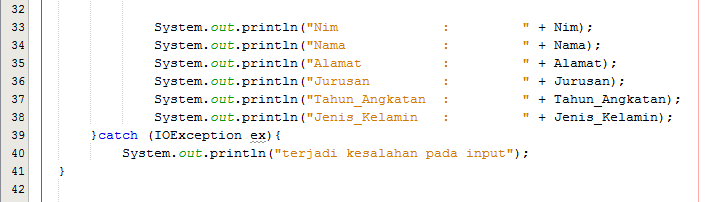
**H.1.1.2 Buffer Reader**

Pada bagian ini akan digunakan class BuffredReader, class tersebut berada di dalam package **java.io,** class ini digunakan untuk mendapatkan input dari user, hampir sama seperti Scanner, yang membedakan hanya cara penulisan syntaxnya saja.



Contoh:

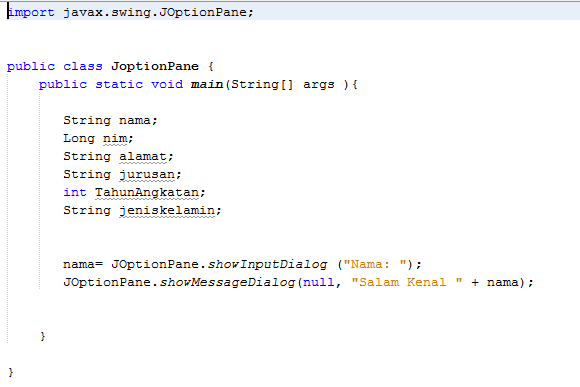
****

****

**H.1.1.3 Class JOptionPane**

Cara terakhir untuk mendapatkan input dari user adalah dengan menggunakan class JOptionPane tersebut terdapat pada package **javax.swing** , berbeda dari class sebelumnya, JOptionPane memiliki tampilan visual atau GUI untuk memudahkan pengguna. Tampilan tersebut berupa dialog box standart yang memfasilitasi kepada user untuk mengkonfirmasi sendiri.

Contoh :

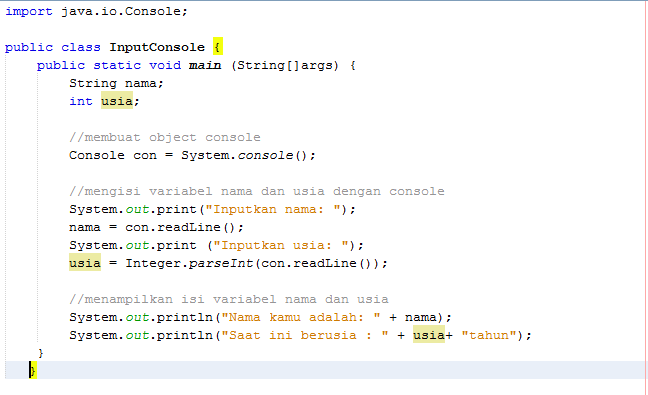


**H.1.1.4 Class Console**

Class console hampir sama dengan BufferReader. Class console menggunakan fungsi readLine( ) untuk mengambil input. Akan tetapi, class ini hanya bisa digunakan di lingkungan console saja, seperti terminal dan CMD.

Class console tidak bisa digunakan langsung pada NetBeans. Maka dari itu, harus dikompilasi secara manual.untuk menggunakan class ini, diperlukan untuk mngimpornya terlebih dahulu.

Contoh: ketik printah javac input Console.java untuk melakukan kompilasi

**H.1.2 Import Package**

Terdapat banyak package untuk dipilih. Pada contoh sebelumnya, percobaan menggunakan class **Scanner**  dari **java.util** package. Package ini berisi fasilitas tanggal dan waktu, generator nomor acak, dan manfaat class lainnya.

Untuk mengimpor seluruh package, maka akhiri kalimat dengan tanda bintang (\*). Contoh berikut akan mengimpor semua class dalam **java.util** package.



**H.2 Daftar Package pada Program Java**

Berikut ini adalah beberapa daftar package java yang sering digunakan pada program java:

1. javax.swing

paket ini berisi kelas-kelas dari interface untuk komponen GUI swing java yang menyediakan dukungan untuk tampilan grafis yang portable.

1. Javax.swing.event

Paket ini berisi kelas-kelas dari interface yang memperbolehkan penanganan event untuk komponen grafis yang terletak di paket javax.swing

1. Java.lang

Paket ini berisi kelas-kelas dari interface yang diperlukan olehbanyak program java. Paket ini diimpor oleh compiler ke semua program java secara otomatis.

1. Java.applet

Paket ini berisi kelas-kelas applet dan juga beberapa interface yang memperbolehkan interakse applet dan browser serta untuk memainkan klip audio. Kelas javax.swing.JApplet digunakan untuk menetapkan applet yang menggunaka komponen java GUI swing.

1. Java.text

Paket ini berisi kelas-kelas dari interface yang memperbolehkan program java untuk memanipuasi angka, tanngal, karakterdan juga string.

1. Java.net

Paket ini berisi kelas yang memperbolehkan program untuk berkomunikasi melalui jaringan.

1. **Java Inheritance (Pewarisan)**

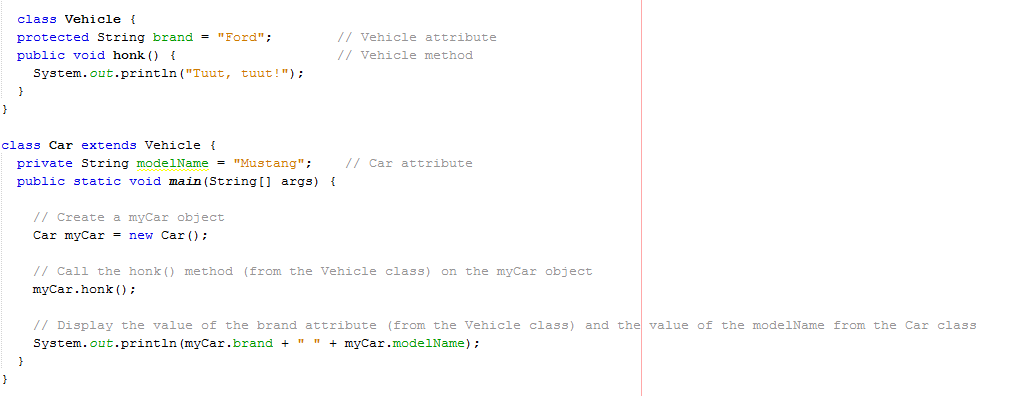
Inheritance  (Pewarisan) adalah salah satu contoh konsep OOP, yang merupakan suatu proses dimana suatu class yang bisa disebut *super class* ,dapat mewarisi sifat atau ciri-ciri seperti atribut dan method ke dalam class turunanya yaitu *sub class*. Super class akan mewarisi nilai dari atribut atau behavior ke Class turunanya atau class lain. Sistem penurunan sifatnya akan dilakukan secara hirarki (bertingkat-tingkat) , semakin ke bawah class, maka akan semakin detail.

1. **Subclass** (anak) - kelas yang mewarisi dari kelas lain
2. **Superclass** (orang tua) - kelas yang diwarisi dari

Untuk mewarisi dari kelas, gunakan keyword extends yaitu sebagai kata kunci.

Keyword **extends** adalah salah satu jenis keyword di dalam bahasa pemrograman java yang digunakan untuk mewarisi sifat-sifat dari class parent atau super, seperti atribut dan method, ke dalam suatu class turunan. Cara untuk mendeklarasikan inheritance adalah dengan memberikan keyword **extends** setelah nama class diikuti dengan nama parent class. Kata kunci extends memberikan informasi pada compiler java bahwa ingin melakukan perluasan class.

Contoh :



**KATA KUNCI FINAL**

Ketika suatu class dideklarasikan sebagai final maka class tersebut tidak bisa diwariskan atau mengakibatkan output menjadi eror. Beberapa tujuan dari kelas **final** :

Ada tiga tujuan utama kenapa suatu class dibuat final. Tujuan tersebut adalah:

1. Untuk mencegah inheritance karena class final tidak bisa di extends
2. Untuk membuat suatu class immutable. Class tidak akan bisa dibuat immutable jika sebelumnya belum menggunakan final.
3. Class yang dibuat final sangat berguna ketika akan digunakan untuk tujuan tingkat keamanan tinggi dalam setiap aplikasi.

Contohnya:

Final class Vehicle {

…

}

Class Car extends Vehicle {

…

}

Maka akan menghasilkan output error atau terjadi kesalahan.

1. **Java Polymorphism**

Polymorphism merupakan konsep sederhana dalam bahasa pemrograman berorientasi obyek yang berarti kemampuan sebuah obyek untuk menghasilkan aksi yang berbeda, bila method yang sama dipanggil, dimana aksi method yang dikerjakan tergantung dari tipe obyeknya. Polymorphism merupakan suatu konsep yang menyatakan sesuatu yang sama dapat memiliki berbagai bentuk dan perilaku berbeda. Jadi dapat disimpulkan bahwa polymorphism adalah satu objek yang memiliki banyak bentuk.

Konsep dari Polymorphism menggunakan 2 metode. Metode pertama yaitu Pewarisan yang menggunakan konsep overriding yaitu membuat method yang bernama dan berstruktur sama akan tetapi isi method berbeda pada class induk dan class anak. Metode yang kedua adalah overloading. Overloading adalah method bernama yang sama dengan method lain pada suatu class tetapi dengan parameter da nisi yang berbeda.

Keuntungan Polymorphism:

1. Kita dapat menggunakan kelas-kelas yang kita buat (sebagai super kelas) dan membuat kelas kelas baru berdasar superkelas tersebut dengan karakteristik yang lebih khusus dari behaviour umum yang dimiliki superkelas.
2. Kita dapat membuat super kelas yang hanya mendefinisikan behaviuor namun tidak memberikan implementasidari metode-metode yang ada.Hal ini berguna jika kita ingin membuat semacam template kelas,kelas semacam ini disebut kelas abstrak karena behaviournya masih abstrak dan belum diimplementasikan.subkelas-subkelas dari kelas semacam ini yang disebut kelas konkret,mengimplementasikan behaviuor abstrak tersebut sesuai dengan kebutuhan masing-masing.
3. Menghindari duplikasi object,kita dapat menciptakan class baru dari class yang sudah ada,sehingga tidak perlu menuliskan code dari nol ataupun mengulangnya, namun tetap bisa menambahkan attribute dan atau method unik dari class itu sendiri. Dalam konsep yang lebih umum sering kali polymorphism disebut dalam istilah satu interface banyak aksi.

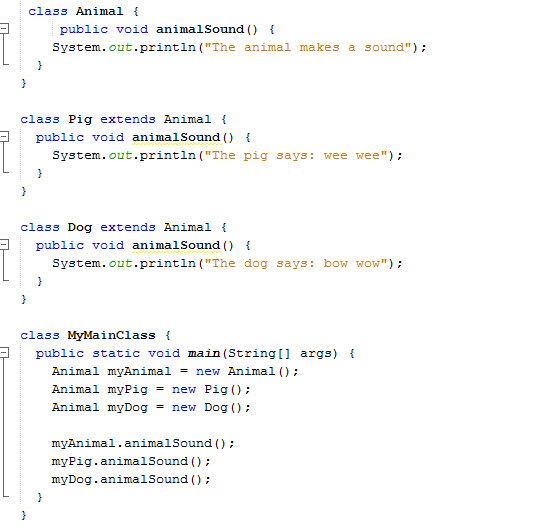
Contoh:

class Animal {

public void animalSound() {

System.out.println("The animal makes a sound"); }}

Sebagai contoh, pikirkan superclass bernama Animal yang memiliki metode yang disebut animalSound (). Subkelas Hewan dapat berupa Babi, Kucing, Anjing, Burung - Dan mereka juga memiliki implementasi sendiri dari suara binatang (babi oinks, dan kucing mengeong, dll):



1. **Java Inner Class**

Inner class di definisikan di dalam isi kelas lain. Inner class dapat dideklarasikan private, public, protected, atau access default sedangkan kelas luar hanya memiliki akses public atau standar. Kelas java Nested dibaggi menjadi dua jenis yaitu :

**K.1 Static Nested Class (Kelas Bersarang Statis)**

Jika kelas bersarang adalah statis, maka disebut dengan kelas bersarang statis. Kelas bersarang statis dapat di akses oleh anggota satis dari kelas luar. Kelas bersaranng statis sama dengan kelas tingkat atas lainnya da bersarang hanya untuk kenyamanan dan pengemasan suatu kode pada java.

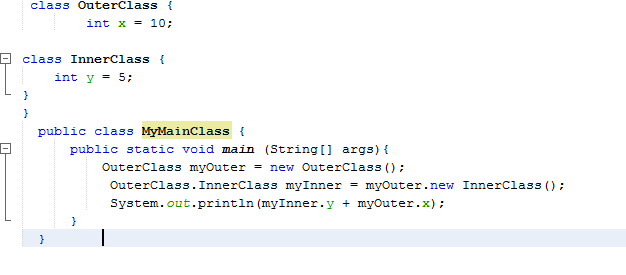
**K.2 Java Inner Class**

Setiap kelas bertingkat non-statis dikenal sebagai inner class di java. Inner class java dikaitkan dengan objek kelas dan mereka dapat mengakses semua variabel dan method kelas luar. Karena kelas dalam dikaitkan dengan instance, maka tidak dapat memiliki variabel statis di dalamnya.

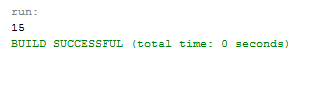
Object kelasdalam java adalah bagian dari objek kelas luar dan untuk membuat turunan dari kelas dalam, pertama-tama perlu dibuat urutan dari kelas luar.

Di dalam java, inner class merupakan kelas besarang (kelas dalam kelas). Tujuan dari inner class untuk mengelompokkan kelas-kelas yang termasuk bersama, yang membuat sebuah source code lebih mudah dibaca dan di pelihara.

Untuk mengakses inner class, buatlah object dari kelas luar, dan kemudian buat objek dari kelas dalam. Berikut adalah contohnya:



Akan didapatkan hasil running seperti ini :



Terdapat dua jenis dari java inner class, yaitu:

1. Local Inner Class

Ketika kelas didefinisikan dalam tubuh isi method, itu disebut dengan local inner class. Karena local inner class tidak terkait dengan object, maka tidak dapat menggunakan private, public, atau protected. Satu-satunya pengubah yang diizinkan adalah abstract atau final.

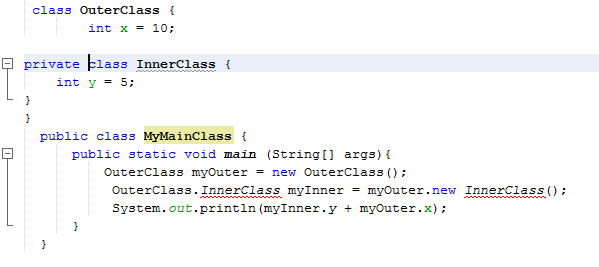
Local inner class dapat mengkases semua anggota kelas penutup dan variabel akhir local dalam lingkup yang ditentukan. Selain itu, local inner class juga dapat mengakses variabel local non-final dari metode yang didefinisikan, tetapi tidak dapat memodifikasinya. Jadi, jika mnecoba untuk mencetak nilai variabel local non-final akan diizinkan tetapi jika mencoba untuk merubah nilainya dari metode inner class, maka akan mendapatkan kesalahan waktu kompilasi.

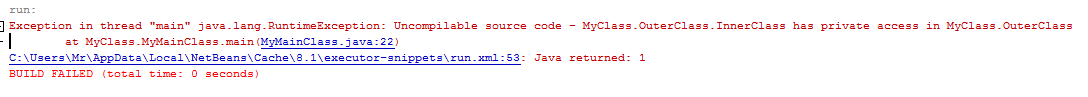
1. Anonymous Inner Class

Local inner class tanpa nama dikenal dengan anonymous inner class. Kelas anonym didefinisikan dan dipakai dalam satu pernyataan. Anonymous inner class selalu memperluas kelas atau mengimplementasikan interface. Karena kelas anonym tidak memiliki nama, tidak mungkin untuk mendefinisikan konstruktor untuk kelas anonym.

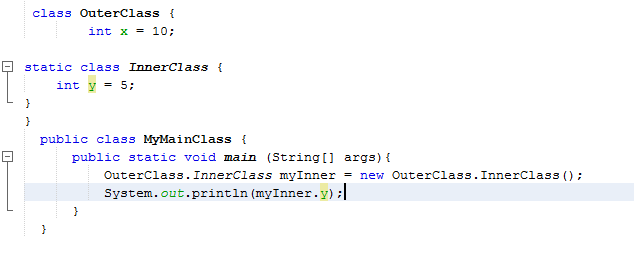
**K.2.1 Private Inner Class**

Tidak seperti kelas regular, kelas dalam bisa **private**  atau **protected.** Jika tidak ingin object luar mengakses kelas dalam, maka nyatakan dalam kelas sebagai **private**. Seperti contoh dibawah ini:

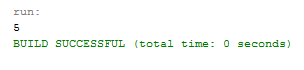


Jika dilihat dalam gambar, maka muncul garis merah pada syntax yang salah. Dan jika tetap ingin mengakses private inner class dari kelas luar (MyMainClass), maka terdapat kesalahan (error):

**K.2.2 Static Inner Class**

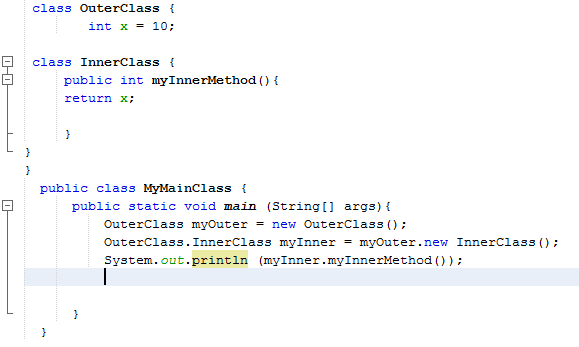
Kelas dalam juga bisa menggunakan static, yang berarti bahwa dapat mengaksesnya tanpa membuat objek dari kelas luar. Berikut adalah contoh dari source kode menggunakan static:

Dan akan memperoleh hasil running seperti gambar dibawah ini:

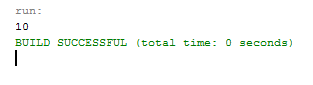


Perlu diingat bahwa, sama seperti **static** atribut dan metode, **static**  kelas dalam tidak memiliki akses ke anggota kelas kelas luar.

**K.2.3 Access Outer Class dari Inner Class**

Terdapat satu keuntungan dari adanya kelas dalam, adalah mereka dapat mengakses atribut dan metode dari kelas luar, contoh gambar dibawah ini merupakan source code yang dapat mengakses kelas luar dari kelas dalam:

Akan didapatkan hasil dari running program tersebut:



1. **Java Abstraksi**

Kelas abstrak (*abstract class*) adalah kelas yang mengandung komsep abstrak, dan tidak alan pernah bisa diinstansiasi. Kelas abstrak didefinisikan dengan tujuan untuk digunakan dan diperluas oleh kelas lain. Dengan demikian, kelas ini merupakan cikal bakal dari superkelas.

Abstraction Class adalah sebuah kelas abstrak yang memiliki metod dan atribut. Abstraction class adalah hirarki tertinggi dari sub-sub classnya. Kelas abstrak digunakan untuk membuat sebah kelas yang memiliki method yang belum jelas implementasinya.

Berikut ini aturannya:

1. Jika sebuah kelas memiliki method abstrak maka kels itu harus menjadi kelas abstrak.
2. Sebuah kelas abstrak dapat saja memiliki method yang tidak abstrak.
3. Jika sebuah kelas abstrak diturunkan menjadi kelas konkret, maka semua method abstrak dari kelas abstrak haruslah ditulis ulang/ dibuat ulang di kelas konkretnya dan diberi detail dari methodnya.
4. Jika method abstrak di turunkan dan kelas turunannya adalah kelas abstrak, maka tidak perlu menulis ulang method abstraknya.

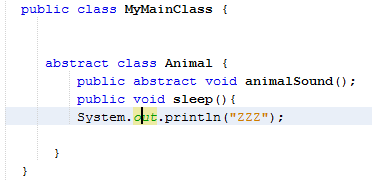
**L.1 Java Abstract Classes dan Methods**

Abstraksi data adalah proses menyembunyikan detail tertentu dan hanya menampilkan informasi penting kepada pengguna. Abstraksi dapat dicapai dengan kelas atau interface abstrak.

Kata kunci abstract adalah pengubah non-akses, digunakan untuk kelas dan metode:

1. Kelas abstrak : adalah kelas terbatasyang tidak dapat digunakan untuk membuat objek (untuk mengaksesnya, itu harus diwarisi dari kelas lain).
2. Metode abstrak : hanya dapat digunakan dalam kelas abstrak, dan tidak memiliki tubuh, tubuh disediakan dari subclass(pewarisan dari).

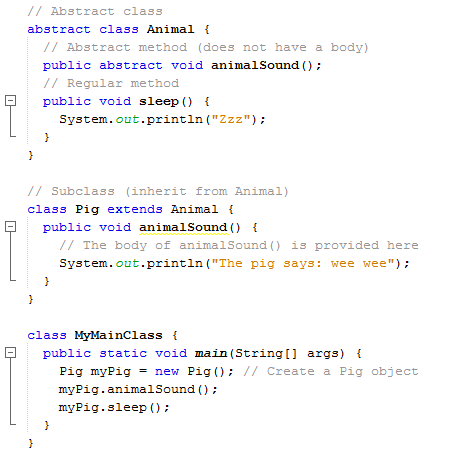
Kelas abstrak dapat memiliki metode abstrak dan regular, seperti gambar dibawah ini :



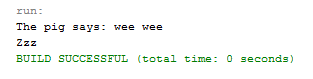
Dari contoh diatas, tidak mungkin membuat objek dari kelas Animal:



Untuk mengakses elas abstrak, harus diwarisi dari kelas lain. Maka dari itu, konversikan kelas Animal pada bab polymorphism menjadi kelas abstrak. Kata kunci yang digunakan pada bab inheritance saat mewarisi dari kelas adalah **extends.**



Hasil dari running diperoleh seperti gambar berikut:



1. **Java Interface**

Interface merupakan kumpulan deklarasi fungsi tanpa implementasi yang mendefinisikan aturan pemanggilan fungsi oleh sembarang class lain. Interface mendefinisikan sebuah cara standar dan umum dalam menetapkan sifat-sifat dari class-class. Interface memungkinkan pemrogram untuk berbagi konstanta atau menentukan bentuk metode yang dapat digunakan oleh sejumlah class.

Interface digunakan jika kita ingin class yang tidak berhubungan mengimplementasikan method yang sama. Melalui interface-interface, kita dapat menangkap kemiripan diantara class yang tidak berhubungan tanpa membuatnya seolah-olah class yang berhubungan.

Ciri-ciri dari interface:

1. Method interface tidak punya tubuh, sebuah interface hanya dapat mendefinisikan konstanta dan interface tidak langsung mewariskan hubungan dengan class istimewa lainnya, mereka didefinisikan secara independent.
2. Tidak bisa membuat instance atau objek baru dari sebuah interface.

3, Hirarki antara class interface dan class yang mengimplementasinya adalah independen atau tidak saling berelasi

1. Interface diimplementasikan ke dalam suatu class dengan menggunakan kata kunci implements, dan sebuah kelas dapat mengimplementasikan lebih dari satu interface.
2. Ciri umum lain adalah baik interface maupun class dapat mendefinisikan method. Interface tidak punya sebuah kode implementasi sedangkan class abstrak memiliki salah satunya.

Cara lain untuk mencapai abstraksi di Java, adalah dengan interface. Interface adalah "kelas abstrak" yang sepenuhnya digunakan untuk mengelompokkan metode terkait dengan benda kosong.

Contoh :

// interface

interface Animal {

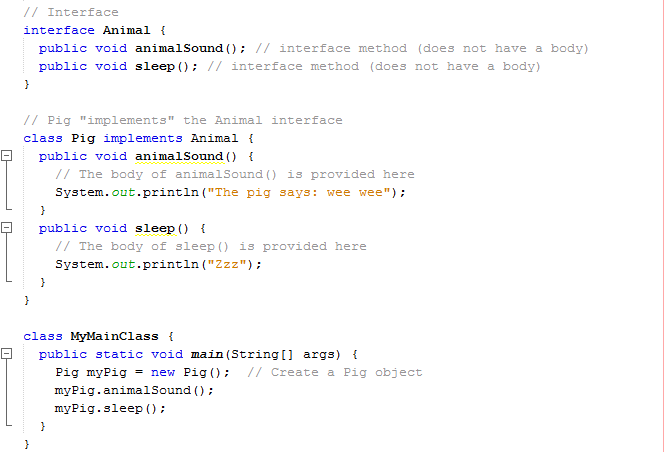
public void animalSound(); // interface method (does not have a body)

public void run(); // interface method (does not have a body)

}

Untuk mengakses metode interface, interface harus "diimplementasikan" (agak seperti diwariskan) oleh kelas lain dengan kata kunci implement (alih-alih meluas). Isi metode interface disediakan oleh kelas "implement".

Contoh:



Catatan tentang Interface:

1. Seperti kelas abstrak, interface tidak dapat digunakan untuk membuat objek (dalam contoh di atas, tidak mungkin membuat objek "Hewan" di MyMainClass)
2. Metode interface tidak memiliki body, body disediakan oleh kelas "implement"
3. Pada implementasi interface, kita harus mengganti semua metodenya
4. Metode interface secara default abstrak dan public
5. Atribut interface secara default publik, statis dan final
6. Interface tidak dapat berisi konstruktor (karena tidak dapat digunakan untuk membuat objek)

Mengapa Dan Kapan Menggunakan Interface?

1. Untuk mencapai keamanan - sembunyikan detail tertentu dan hanya tampilkan detail penting dari suatu objek (interface).
2. Java tidak mendukung "multiple inheritance" (sebuah kelas hanya dapat diwarisi dari satu superclass). Namun, ini dapat dicapai dengan interface, karena kelas dapat mengimplementasikan banyak interface. Catatan: Untuk mengimplementasikan beberapa interface, pisahkan dengan koma (lihat contoh di bawah).

Contoh:

package Interface;

/\*\*

\*

\* @author Putri

\*/

interface FirstInterface {

public void myMethod(); // interface method

}

interface SecondInterface {

public void myOtherMethod(); // interface method

}

// DemoClass "implements" FirstInterface and SecondInterface

class DemoClass implements FirstInterface, SecondInterface {

public void myMethod() {

System.out.println("Some text..");

}

public void myOtherMethod() {

System.out.println("Some other text...");

}

}

class MultipleInterfaces {

public static void main(String[] args) {

DemoClass myObj = new DemoClass();

myObj.myMethod();

myObj.myOtherMethod();

}

}

1. Java Enum

Enum (enumeration)adalah salah satu kelas library di J2SE (versi 5), yang dapat kita gunakan untuk membuat kumpulan variable yang telah didefinisikan menjadi tipe data **konstanta** (variabel tidak bisa diubah, seperti final variabel).setiap objek baru yang didefinisikan berdasarkan tipe data Enum, objek tersebut dapat mengakses atribut atau method (Enum), lalu mengaplikasikan nilai atau value yang ada di dalamnya. Enum bisa juga disebut array, namun dalam bentuk konstanta, dan juga bertipe class. Jika sebelumnya mendefinisikan nilai pada variable bertipe data konstanta dengan baris per baris dengan menggunakan keyword **enum**, kita dapat mengklasifikan variable-variabel bertipe data konstanta tersebut didalam **block{}**. Untuk membuat enum, gunakan enum kata kunci (bukan kelas atau interface), dan memisahkan konstanta dengan koma. Catatan bahwa mereka harus dalam huruf besar:

Contoh:

enum Level {

LOW;

MEDIUM;

HIGH;

}

Dapat juga mengakses enum konstanta dengan sintaks dot:

Level myVar = Level.MEDIUM;

**N.1 Enum dalam Pernyataan SWITCH-CASE**

Merupakan pernyataan yang dirancang khusus untuk menangani suatu keputusan atau suatu hasil akhir dalam membuat project, switch-case memiliki kegunaan yang sama seperti if-else. Tetapi switch-case memiliki suatu batasan yaitu

1. Data yang diperiksa harus bertipe integer(INT) atau karakter (CHR).
2. Range data yang diperiksa bernilai 0 s/d 255.

Bentuk Penulisannya :

|  |
| --- |
| switch (value)  {  case constanta 1 : statement 1;  break;  case constanta 2 : statement 2;  break;  ……  ….etc  default : statement x;  } |

Keteterangan:

* **Constanta 1, constanta 2,** dst adalah yang akan dicocokkan dengan isi value.
* **Statement** adalah pernyataan yang akan di kerjakan jika **value (bertipe Integer atau Karakter)**cocok dengan salah satu data dari **constanta 1 , constanta 2 atau yang lainnya**.
* **Break** adalah perintah untuk mengakhiri **statement**. Apabila tidak ada break, komputer akan mengeksekusi intruksi yang berada di bawahnya walaupun berada di case yang berbeda.
* **Default**, bersifat optional. dieksekusi jika **value** tidak cocok dengan salah satu **constanta**yang tersedia.

**N.2 Loop melalui enum**

Yaitu untuk mengulang kelas enum di atas atau mengembalikan sebuah array dari semua konstanta enum. Metode values() ini berguna ketika Anda ingin loop melalui konstanta enum.

 Ada dua method yang sering digunakan pada enum atara lain

* + 1. Values ();

Method ini mengembalikan keseluruhan dari enumeration

* + 1. ValueOf(String);

Method ini mengembalikan isi dari enumeration yang sesuai dengan parameter string

bentuk penulisannya:

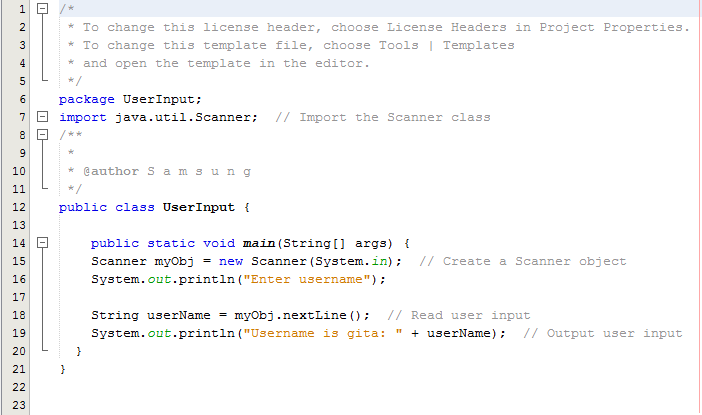
|  |
| --- |
| for (EnumClass obj : EnumClass.values()){  System.out.println (obj); |

1. **Java User Input (Scanner)**

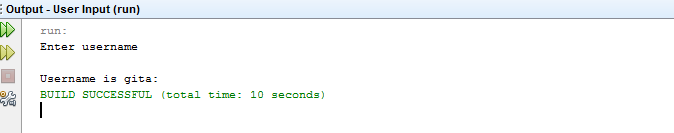
Kelas **Scanner** digunakan untuk mendapatkan input pengguna, dan ditemukan dalam paket java.util .

Untuk menggunakan kelas **Scanner ,** buat objek kelas dan gunakan salah satu metode yang tersedia yang ditemukan dalam dokumentasi kelas **Scanner** . Dalam contoh kita, kita akan menggunakan metode **nextLine()** , yang digunakan untuk membaca String

**Contoh:**



Hasil dari outputan/run:



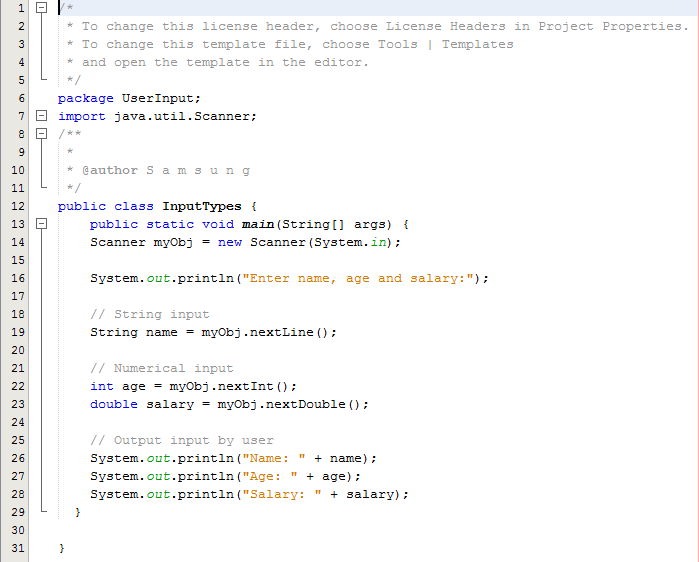
## O.1 Jenis Input

## Pada contoh di atas, kami menggunakan metode nextLine() , yang digunakan untuk membaca Strings. Untuk membaca jenis lain, lihat tabel di bawah ini:

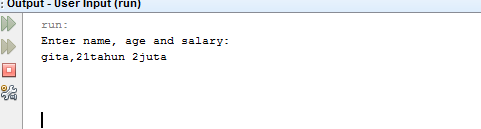
|  |  |
| --- | --- |
| **Method** | **Description** |
| nextBoolean() | Reads a boolean value from the user |
| nextByte() | Reads a byte value from the user |
| nextDouble() | Reads a double value from the user |
| nextFloat() | Reads a float value from the user |
| nextInt() | Reads a int value from the user |
| nextLine() | Reads a String value from the user |
| nextLong() | Reads a long value from the user |
| nextShort() | Reads a short value from the user |

Dalam contoh di bawah ini, kami menggunakan metode berbeda untuk membaca data dari berbagai jenis:

Contoh:



Hasil dari outputan/run



**Catatan:** Jika Anda memasukkan input yang salah (mis. Teks dalam input numerik), Anda akan mendapatkan pesan pengecualian / kesalahan (seperti "InputMismatchException").

1. **Java Date**

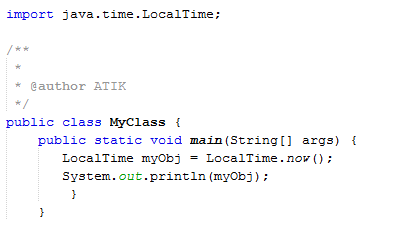
Java tidak memiliki kelas tanggal bawaan , tetapi dapat mengimpor paket java.time agar berfungsi dengan API tanggal dan waktu. Paket terdiri atas beberapa kelas tanggal dan waktu. Sebagai contoh :

|  |  |
| --- | --- |
| Class | Deskripsi |
| Local Date | Menjelaskan tentang tanggal (tahun,bulan,hari  (yyyy-MM-dd)) |
| Local Time | Menjelaskan tentang waktu (jam,menit,detik dan seperinu detik (HH-mm-ss-zzz)) |
| Local Date Time | Menjelaskan tentang tanggal dan waktu (yyyy-MM-dd-HH-mm-ss.zzz) |
| Date Time Formatter | Menampilkan dan menguraikan objek tanggal dan waktu |

**P.1 Menampilkan Tanggal saat ini**

Untuk menampilkan tanggal saat ini, impor kelas java.time.LocalDate dan gunakan metode now().

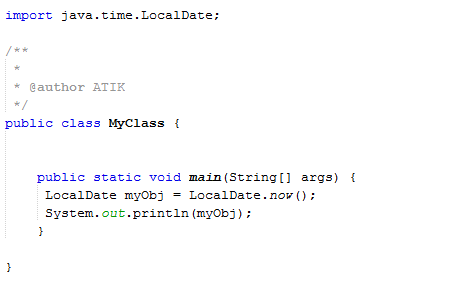
Contoh:



**P.2 Menampilkan Waktu Saat Ini**

Untuk menampilkan waktu saat ini (jam, menit, detik, dan milidetik), impor kelas **java.time.LocalTime** dan gunakan method **now( )** :

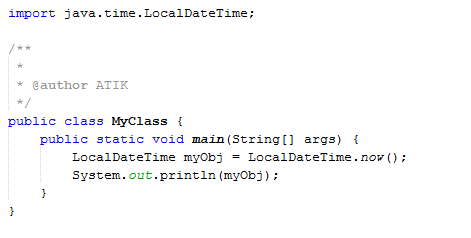
Contoh:



**P.3 Menampilkan Tanggal dan Waktu Saat Ini**

Untuk menampilkan tanggal dan waktu saat ini, impor kelas **java.time.LocalDateTime** dan gunakan method **now( ):**

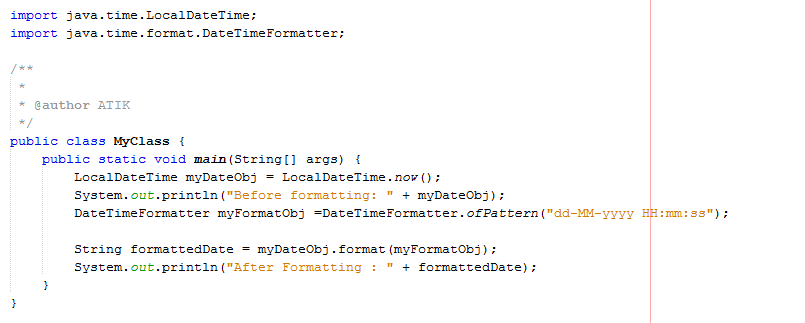
Contoh:



**P.4 Memformat Tanggal dan Waktu**

‘T’ pada contoh diatas digunakan untuk memisahkan tanggal dan waktu . bisa juga menggunakan kelas DateTimeFormatter dengan metode ofPattern() dalam paket yang sama untuk memformat atau menguraikan objek waktu . Contoh berikut akan menghapus ‘T’ dan milidetik dari tanggal-waktu :

Contoh:

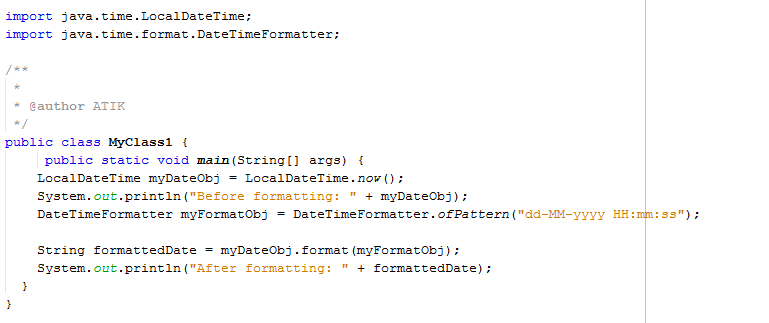


The ofPattern() Metode menerima segala macam nilai , jika ingin menampilkan tanggal dan waktu dalam format yang berbeda. Sebagai contoh :

1. Value: yyy – MM – dd

Example: 1998 – 09 – 29

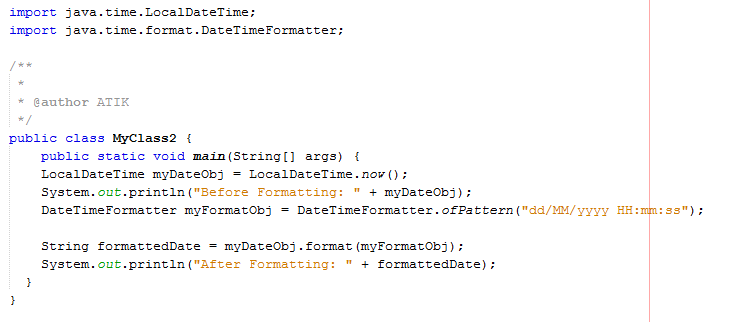
Contoh:



1. Value: dd/MM/yyy

Example: 29/09/1988

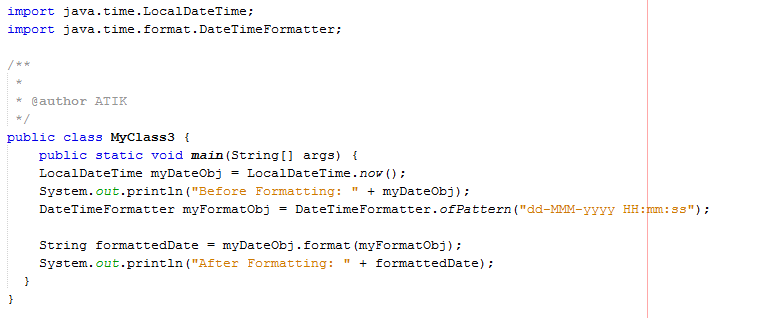
Contoh:



1. Value: dd – MMM – yyyy

Example: 29 – sep – 1988

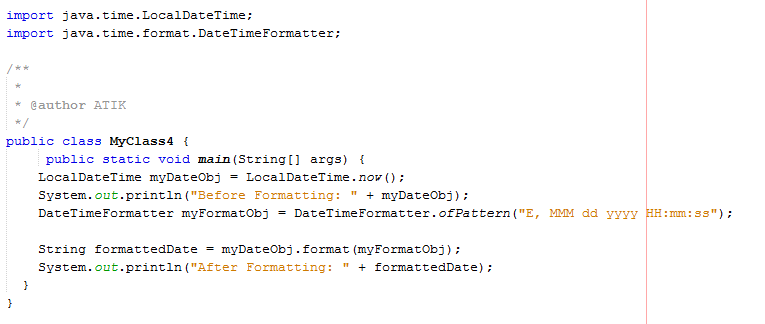
Contoh:



1. Value: E, MMM dd yyyy

Example: Thu, sep 29 1988

Contoh:



**Q. Java Array List**

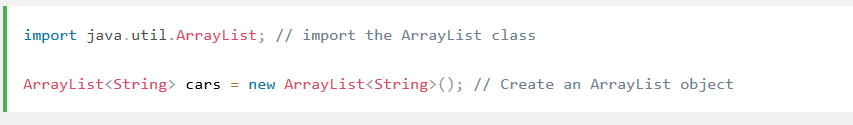
ArrayList di java digunakan untuk menyimpan koleksi elemen yang berukurran dinamis. Berlawanan dengan Array yang memiliki ukuran tetap, ArrayList memperbesar ukurannya secara otomatis ketika elemen baru ditambahkan padanya. Perbedaan antara array bawaan dan ArrayList di Java, adalah bahwa ukuran array tidak dapat dimodifikasi.

Berikut adalah beberapa poin penting tentang ArrayList di Java:

1. ArrayList adalah array yang cukup besar, sehingga disebut array dinamis.
2. arrayList secara internal menggunakan array untuk menyimpan elemen. Sama seperti array, ini memungkinkan untuk mengambil elemen dengan indeks merka.
3. Java arrayList memungkinkan nilai duplikat dan nol
4. Java arraylist adalah koleksi yang diurutkan.

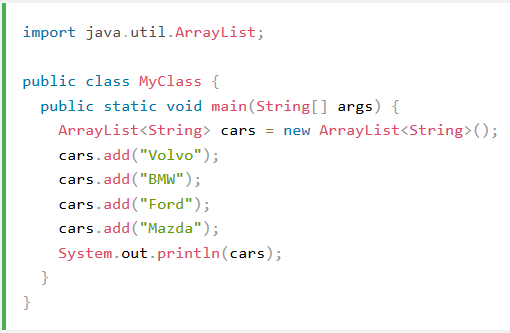
Contoh dari penggunaan array adalah:

Buat objek ArrayList disebut **mobil** yang akan menyimpan string:

****

**Q.1 Membuat Array List dan Menambahkan Elemen Baru ke Dalamnya**

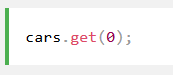
Contoh dibawah ini menunjukkan cara membuat ArrayList menggunakan **ArrayList( )** konstrukstor, dengan menambahkan elemen baru ke ArrayList menggunakan method **add( ).**



**Q.2 Mengakses Elemen dari ArrayList**

Contoh ini menunjukkan mengakses elemen di ArrayList , gunakan metode **get()** dan lihat nomor indeks:

Contoh :



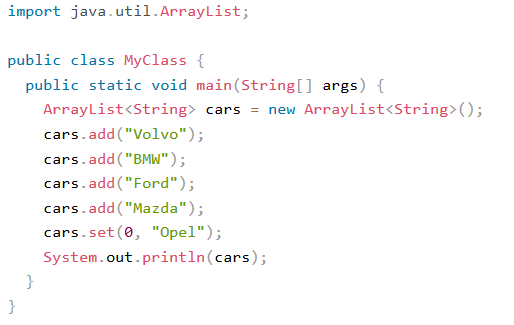


**Q.3 Mengubah Elemen pada ArrayList**

## Untuk memodifikasi elemen, gunakan metode set() dan lihat nomor indeks:

Contoh :

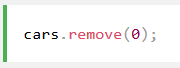


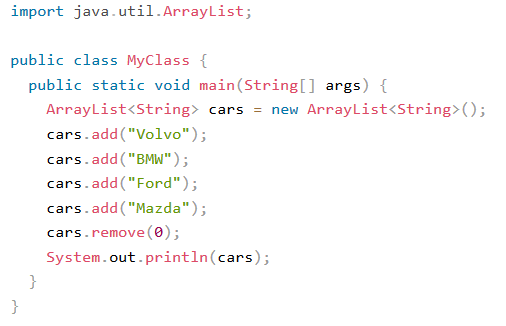


**Q.4 Menghapus Sebuah Elemen pada ArrayList**

Untuk menghapus elemen, gunakan metode **remove()** dan lihat nomor indeks:

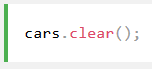
Contoh :

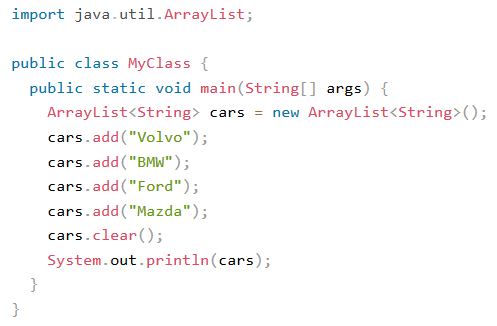




Untuk menghapus semua elemen di ArrayList , gunakan metode **clear()** :

Contoh :

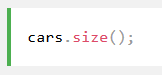


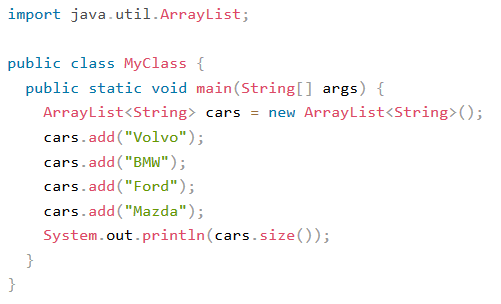


**Q.5 Mengetahui Ukuran ArrayList**

Untuk mengetahui berapa banyak elemen yang dimiliki ArrayList, gunakan metode **size**( ) :

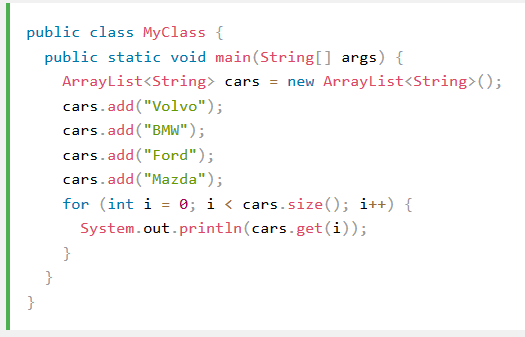
Contoh :





## Q.6 Loop Through ArrayList

Ulangi elemen ArrayList dengan for loop, dan gunakan metode size() untuk menentukan berapa kali loop harus dijalankan. Contoh:



## Q.7 Tipe yang lain

Elemen dalam ArrayList sebenarnya adalah objek. Dalam contoh di atas, telah dibuat elemen (objek) dari tipe "String". Ingat bahwa String di Java adalah objek (bukan tipe primitif). Untuk menggunakan tipe lain, seperti int, harus menentukan [kelas pembungkus yang](https://translate.googleusercontent.com/translate_c?depth=1&hl=id&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://www.w3schools.com/java/java_wrapper_classes.asp&xid=25657,15700021,15700186,15700191,15700256,15700259,15700262,15700265,15700271,15700283&usg=ALkJrhgVuCoieuJyi6KxJTUJaORwg-c7ZQ) setara: Integer . Untuk jenis primitif lainnya, gunakan: Boolean untuk boolean, Character untuk char, Double untuk double, dll:

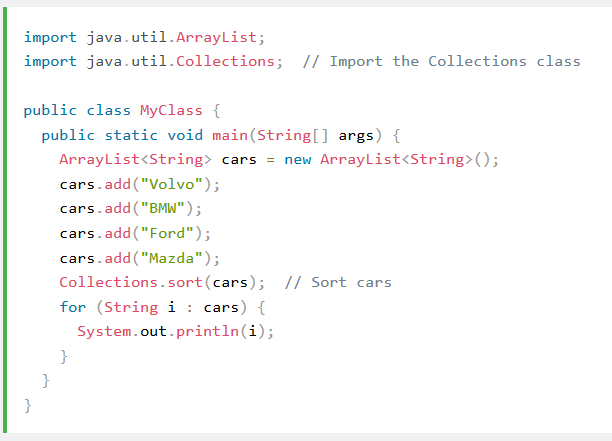
Contoh: buat ArrayList untuk menyimpan angka (untuk menggunakan tipe data integer)



**Q.8 Sortir Daftar Array**

Kelas lain yang berguna dalam paket java.util adalah kelas Collections , yang mencakup metode sort() untuk menyortir daftar berdasarkan abjad atau angka:

Contoh: sortir ArrayList of String



Contoh: Sortir ArrayList of Integer



1. **Java HashMap**

Class HashMap merupakan class turunan dari class AbstractMap dan implementasi dari InterfaceMap. HashMap adalah sebuah class yang berisi sekumpulan pasangan nilai (value) dan kunci (key). Nilai bisa dalam bentuk String, Integer, Boolean, float, double, dan objek. Sedangkan untuk key biasanya dalam bentuk string dan integer. HashMap bisa dibilang seperti Array asosiatif dalam Java.

Cara membuat HashMap

Sebelum menggunakan HashMap, terlebih dahulu harus mengimporya:



Kemudian, membuat objeknya, objek dari HashMap harus dibuat dengan kata kunci **new( ).** Namun, ada sedikir tambahan untuk menentukan tipe data untuk key dan value yang akan disimpan.

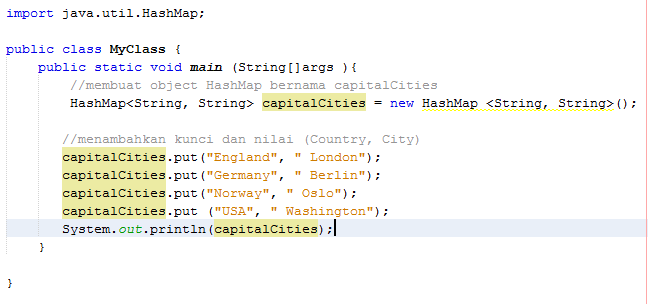


Pada contoh diatas, digunakan objek HashMap bernama **capitalCities.** Objek ini dapat digunakan untuk menyimpan koleksi data.

Tipe data yang digunakan untuk key adalah String, dan value adlah String. Artinya, key harus bertipe String dan nilai yang tersimpan harus dalam bertipe String.

**R.1 Menambahkan Item pada HashMap**

Class HashMap memiliki banyak method yang berguna, misalnya untuk menambahkan item kedalamnya menggunakan method **put( ).**



**R.2 Mengakses Item pada HashMap**

Untuk mengakses nilai dalam HashMap, gunakan method **get( ).**



**R.3 Menghapus Item pada HashMap**

Untuk menghapus sebuah item, gunakan method **remove( ).**



Untuk menghapus semua item, maka gunakan method **clear( ).**



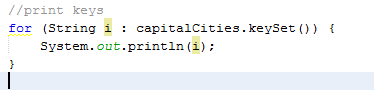
**R.4 Mengetahui Ukuran HashMap**

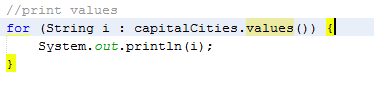
Untuk mengetahui banyak item yang ada pada HashMap, maka gunakan method **size( ).**

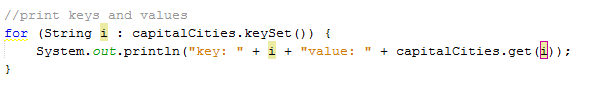


**R.5 Loop Through HashMap**

Peulangan melalui item HashMap dengan masing-masing untuk perulangan. Method **keyset( )** digunakan jika hanya menginginkan kunci, sedangkan method **values( )**  digunakan jika hanya menginginkan nilai.







1. **Java Wrapper Classes**

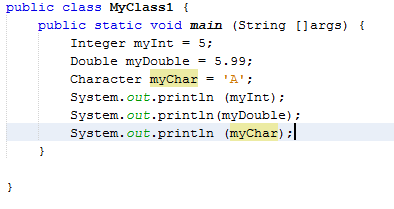
Di dalam bahasa pemrograman Java, *Wrapper Class* (Kelas Pembungkus) adalah suatu mekanisme yang digunakan untuk mengconvert atau mengubah suatu nilai yang di definisikan, dari tipe data **primitif**menjadi sebuah nilai dengan tipe data **referensi**(Objek). Selain itu, Wrapper Class mendukung method dari tipe data primitif, contohnya seperti Boolean, Character, Integer, Long, Float, dsb.

Pada bahasa pemrograman Java, terdapat 2 jenis tipe data, yaitu primitif dan referensi (objek), Tipe Data *Primitif*adalah tipe data yang tidak memiliki method, hanya memiliki data saja. Tipe data ini bukanlah sebuah object, sedangkan Tipe Data *Referensi*adalah tipe data yang digunakan untuk memegang referensi dari sebuah object (instance dari kelas). Dan berikut ini merupakan table yang menjelaskan macam-macam tipe data primitif dan referensi pada Java:

|  |  |
| --- | --- |
| **Tipe Data Primitif** | **Tipe Data Referensi** |
| char | Character |
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| boolean | Boolean |
| float | Float |
| double | Double |

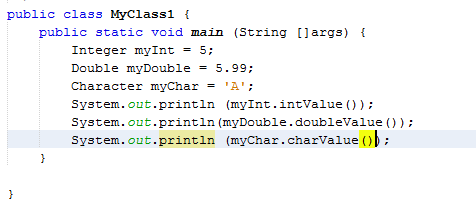
**S.1 Membuat Object Wrapper**

Untuk membuat object wrapper, gunakan kelas wrapper alih-alih tipe primitive.

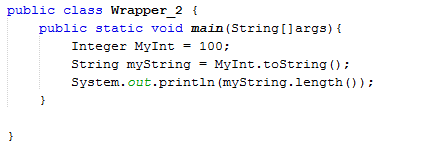


Karena sedang membuat object, maka dapat menggunakan method tertentu untuk mendapatkan informasi tentang suatu object tertentu. Sebagai contoh, method berikut digunakan untuk mendapatkan nilai yang terkait dengan objek pembungkus yang sesuai: **intValue( ), byteValue( ), shortValue( ), longValue ( ), floatValue( ), doubleValue( ), charValue( ), booleanValue( ).**

Contoh:



Terdapat method lain seperti **toString( )**  yang digunakan untuk mengubah objk pembungkus menjadi string. Dalam contoh. Dikonversikan sebuah object dari integer ke string dan menggunakan method **length( )** dan kelas **String** untuk menampilkan panjang String:



1. **Java Exceptions**

Saat menjalankan kode java, kesalahan yang berbeda dapat terjadi seperti kesalahan pengkodean yang dibuat oleh programmer, kesalahan karena input yang salah, atau hal-hal yang tidak terduga lainnya.

Ketika kesalahan terjadi, java biasanya akan berhenti dan menghasilkan pesan kesalahan.

**T.1 Java Try and Catch**

Pernyataan **the try** memungkinkan untuk menentukan blok kode yang akan di uji untuk kesalahan ketika sedang dijalankan.

Pernyataan **the catch**  memungkinkan untuk menentukan blok kode yang akan dieksekusi, jika terjadi kesalahan dalam blok try. Kata kunci **try** dan **catch** berpasangan:

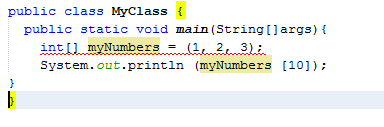
try {

//block of code to try

Catch (Exception e) {

//block of code to handle errors

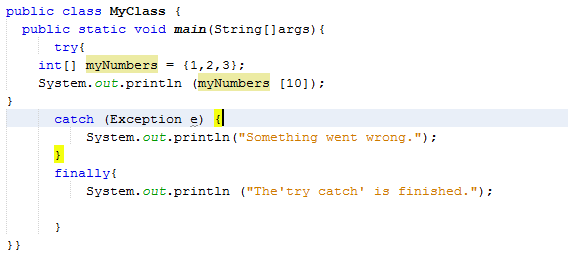
Contoh:



Ini akan menghasilkan hasil yang salah, karena myNumbers (10) tidak ada. Jika terjadi kesalahan, dapat digunakan **try…catch** untuk menangkap kesalahan dan menjalankan beberapa kode untuk menanganinya.

**T.2 Finally**

Pernyataan final memungkinkan untuk mengekseskusi kode setelah **try…catch** terlepas dari hasilnya.



**T.3 Kata Kunci Throw**

Pernyataan the throw memungkinkan untuk membuat kesalahan kustom. Penyataaan the throw digunakan bersama-sama dengan jenis Exceptions. Ada banyak jenis Exceptions yang tersedia di java seperti **AritmeticException, FileNotFoundException, ArrayIndexOutOfBoundsException, SecurityException, dll.**

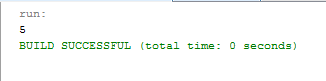
**BAB III**

**HASIL PERCOBAAN PRAKTIKUM**

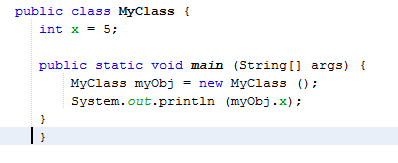
1. **Java Classes and Object**

**A.1 Membuat Object**

Output dari proses running:

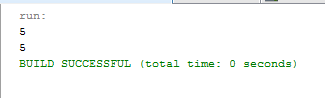


Source code : code program pada java classes and object untuk membuat object

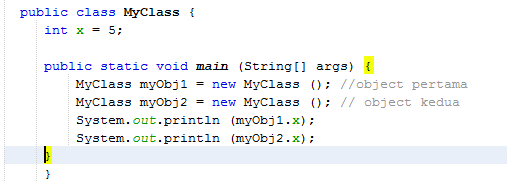


**A.2 Membuat Banyak Object**

Output dari proses running:

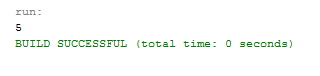


Source code : code program java classes and object untuk membuat banyak ibject dari satu kelas



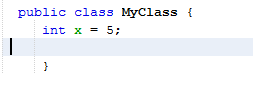
**A.3 Membuat Banyak Kelas**

Output dari hasil kompilasi dua class untuk satu object

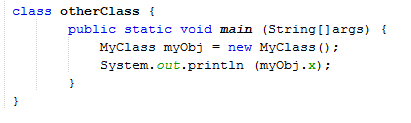


Source code : code membuat object dari banyak class

1. MyClass.java



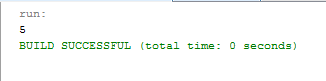
1. otherClass.java



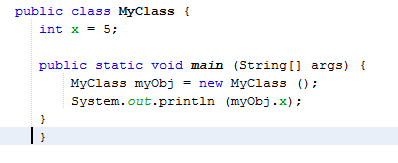
1. **Java Class Attributes**

**B.1 Accessing Attributes**

Output dari proses running:

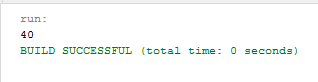


Source code : code program pada java class attributes untuk mengakses atribut

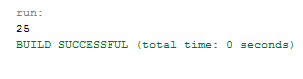


**B.2 Modify Attributes**

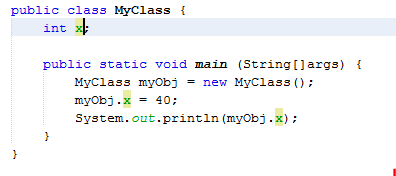
Output : nilai x = 40



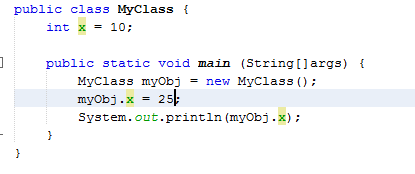
Output: nilai x = 25



Source code : code program java class attributes untuk mengubah nilai x menjadi 40

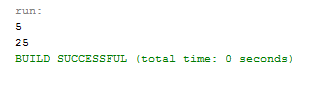


Source code : code program java class attributes untuk megubah nilai x menjadi 25

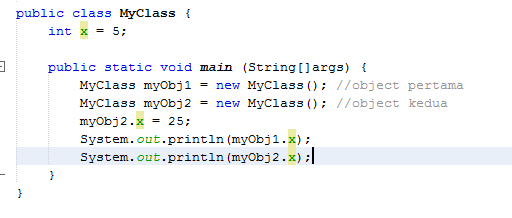


**B.3 Multiple Objects**

Output:

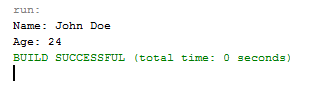


Source code: code program java class attribute untuk mengubah nilai pada myObj2

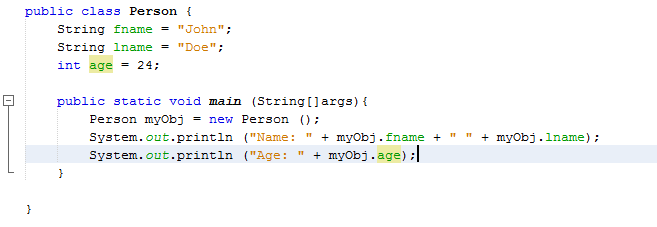


**B.4 Multiple Attributtes**

Output:



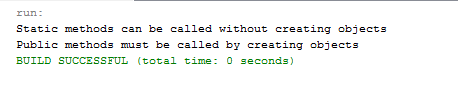
Source code: code program java attribute untuk memperbanyak attribute yang diiginkan



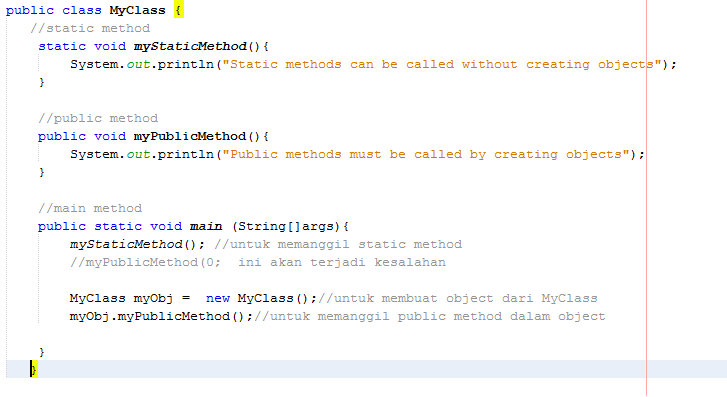
1. **Java Class Methods**

**C.1 Static Vs Non-Static**

Output dari proses running:

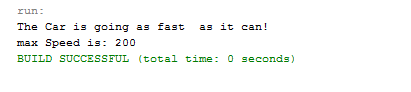


Source code: code program java pada java class methods untuk menunjukkan perbedaan static dan non-static

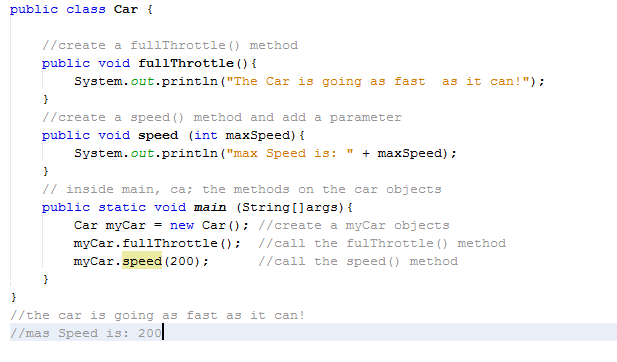


**C.2 Access Method With Object**

Output dari proses running:



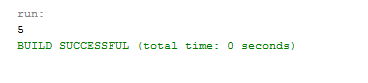
Source code: code program java pada java class methods untuk mengakses sebuah method dengan object



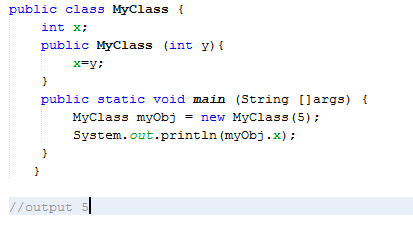
1. **Java Constructor**

**D.1 Constructors Parameters**

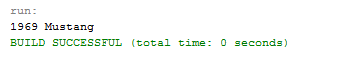
Output dari proses running:



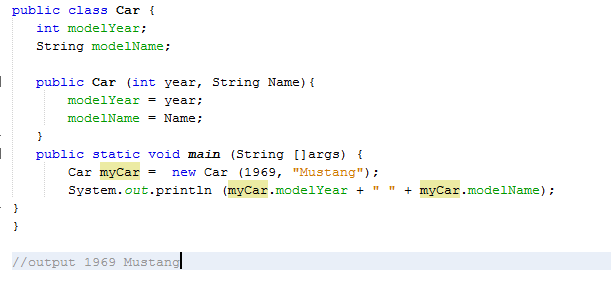
Source code: code program java pada java constructor untuk parameter dengan satu constructor



Output dari proses running:



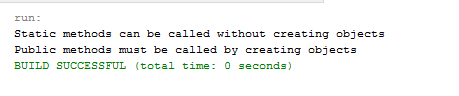
Source code: code program java pada java constructor untuk parameter dengan banyak constructor



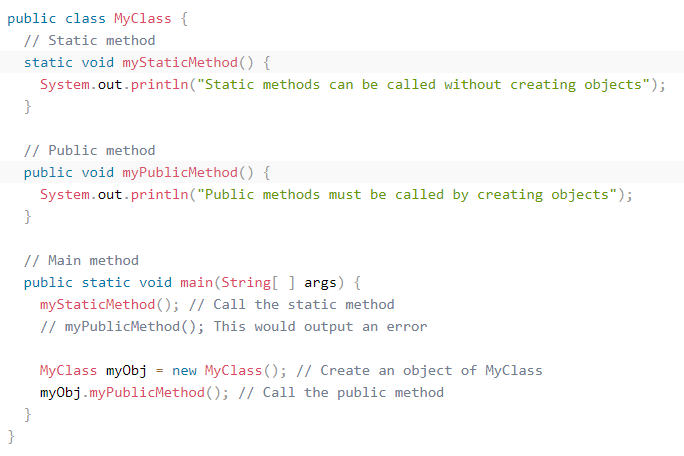
1. **Java Modifiers**

**E.1 Static**

Output dari proses running:

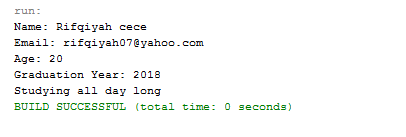


Source code: code program java pada java modifiers untuk menunjukkan perbedaan antara method static dan public



**E.2 Abstract**

Output dari proses running:



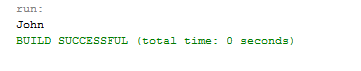
Source code: code program java pada java modifiers untuk sebuah abstract



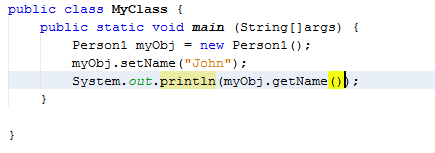
1. **Java Encapsulation**

**F.1 Get and Set**

Output dari proses running:



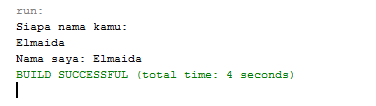
Source code: code program java pada java encapsulation untuk mengambil object dengan menggunakan method Get and Set



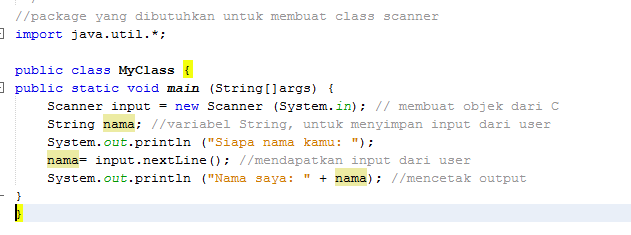
1. **Java Package / API**

**G.1 Class Scanner**

Output dari proses running:

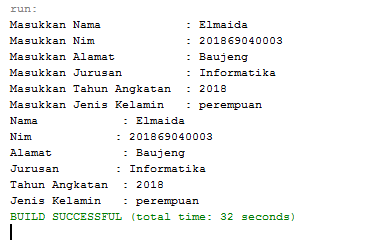


Source code: code program java pada java Package/API untuk class scanner



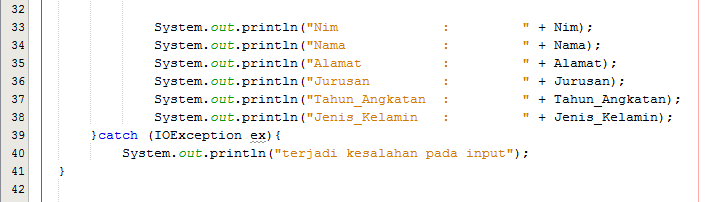
**G.2 Buffer Reader**

Output dari proses running:



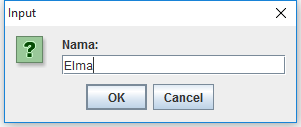
Source code: code program java pada java package/API untuk class buffer reader

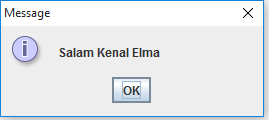
****

****

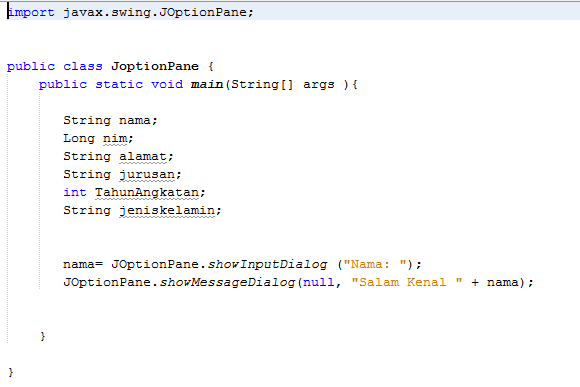
**G.3 JOptionPane**

Output dari proses running:



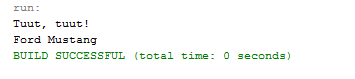


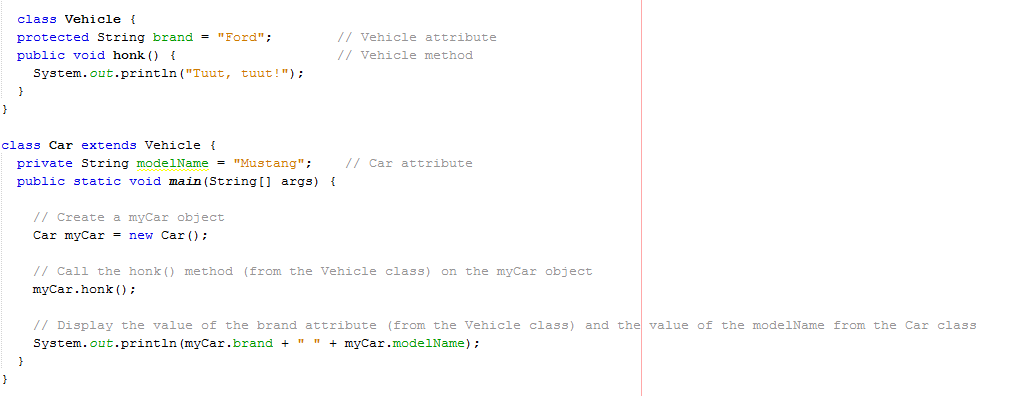
Source code: code program java pada java package / API untuk class JOption Pane



1. **Java Inheritance**

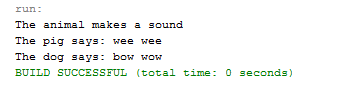
Output dari proses running:



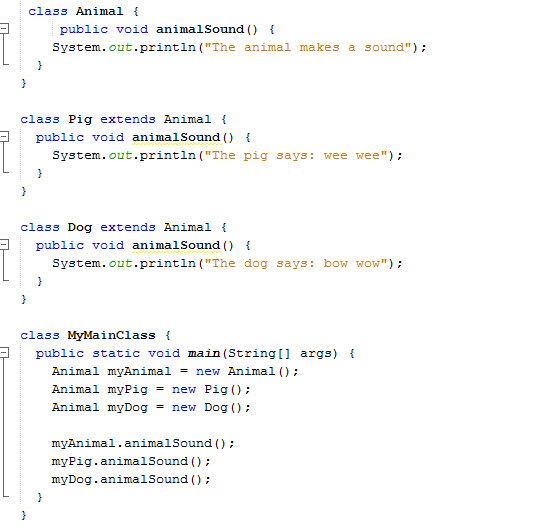
Source code: code program java pada java inheritance untuk kelas car yang mewarisi atribut dan method dari kelas vehicle

1. **Java Polymorphism**

Output dari proses running:



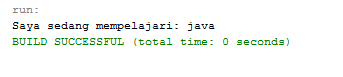
Source code: code progam java pada java polymorphism yang membuat object pig and dog dengan memanggil method animalSound() pada keduanya



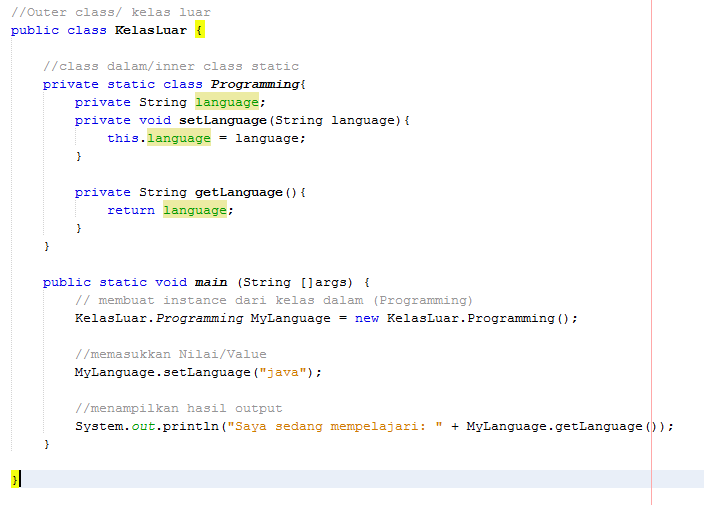
1. **Java Inner Class**

**J.1 Inner Class Static**

Output dari proses running :

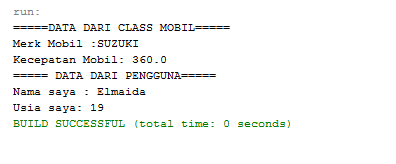


Source code: code program java pada jaa inner class untuk inner class static

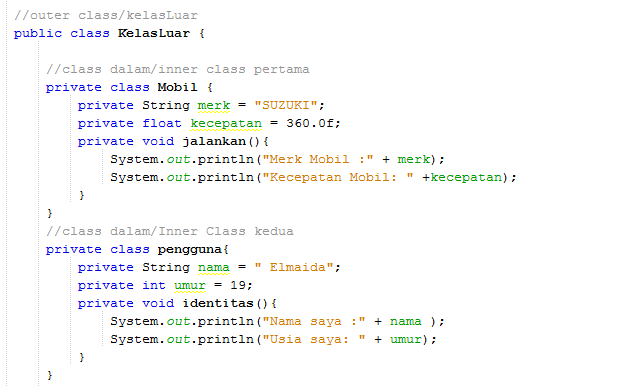


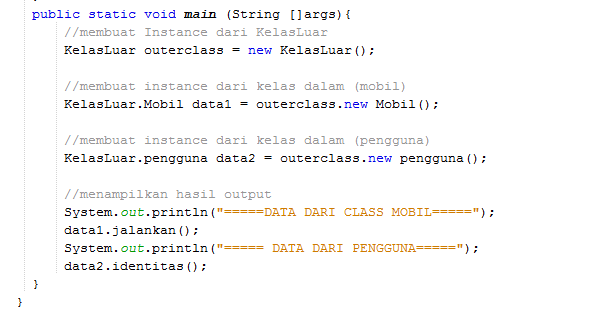
**J.2 Inner Class Non-Static**

Output dari proses running



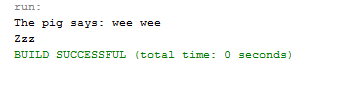
Source code: code program java pada java inner class untuk inner class non-static



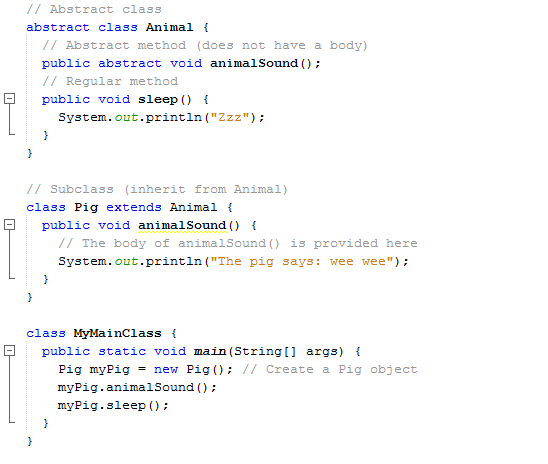


1. **Java Abstraksi**

Output dari proses running:

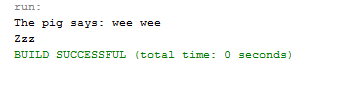


Source code: code program java pada java abstraksi untuk contoh suara binatang

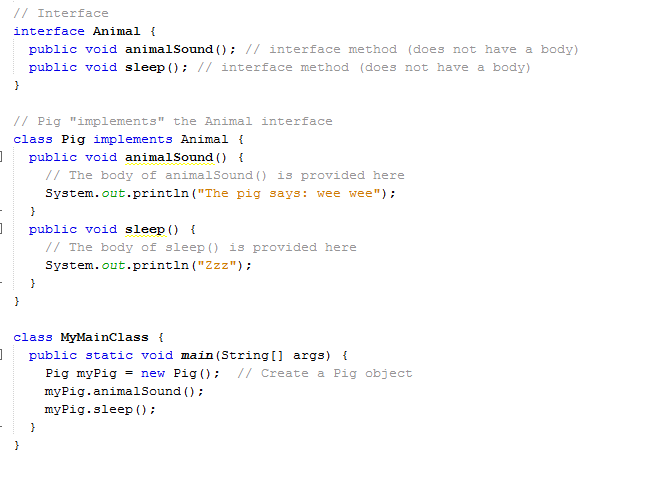


1. **Java Interface**

Output dari proses running:

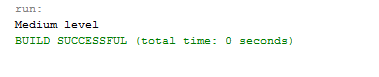


Source code: code program java pada java interface dalam contoh suara binatang

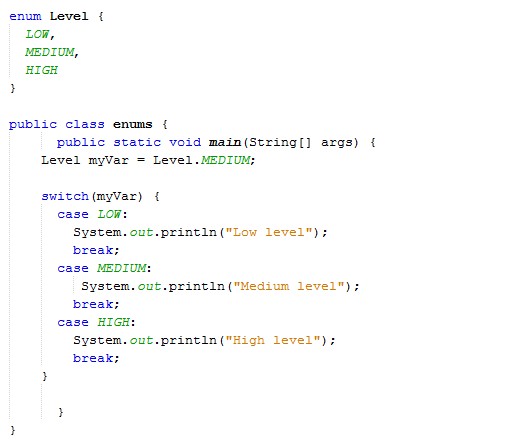


1. **Java Enum**

Output dari proses running:

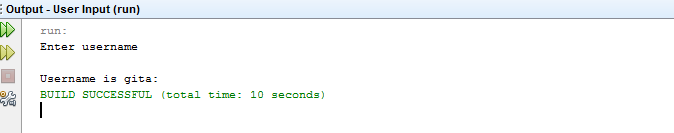


Source code: code program java pada percobaan java enum

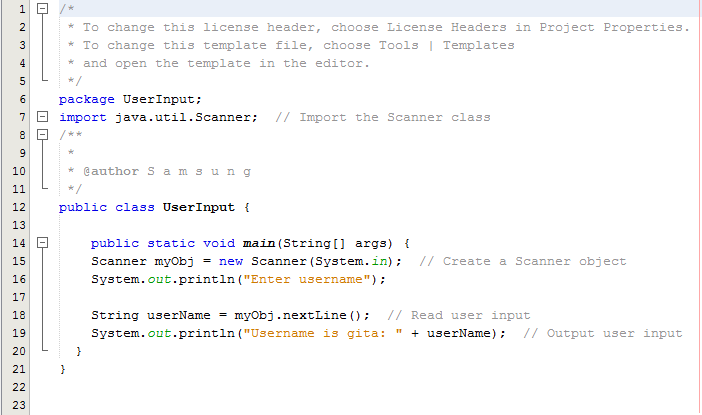


1. **Java User Input**

Output dari proses running:



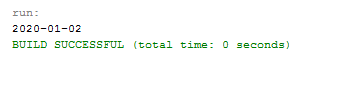
Source code code program java pada java user input dengan menggunakan class scanner



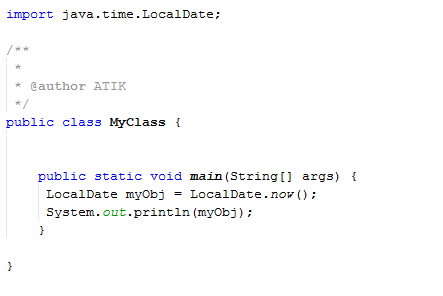
1. **Java Date**

**O.1 Menampilkan Tanggal Saat ini**

Output dari proses running:

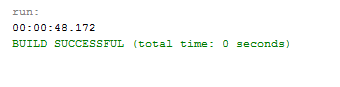


Source code: code program java pada java date untuk menampilkan tanggal saat ini

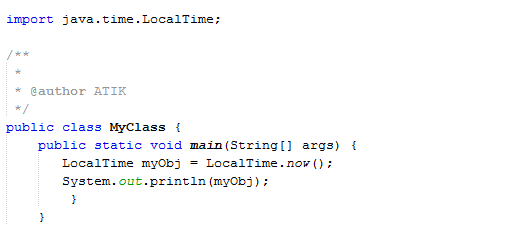


**O.2 Menampilkan Waktu Saat Ini**

Output dari proses running:

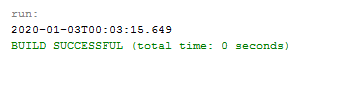


Source code: code program java pada java date untuk menampilkan waktu saat ini

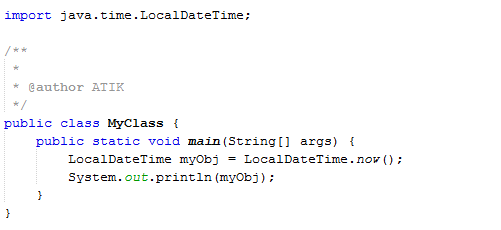


**O.3 Menampilkan Waktu dan Tanggal Saat Ini**

Output dari proses running:



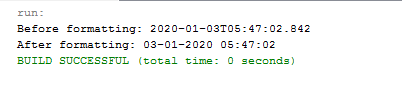
Source code: code program java pada java date untuk menampilkan waktu dan tanggal saat ini



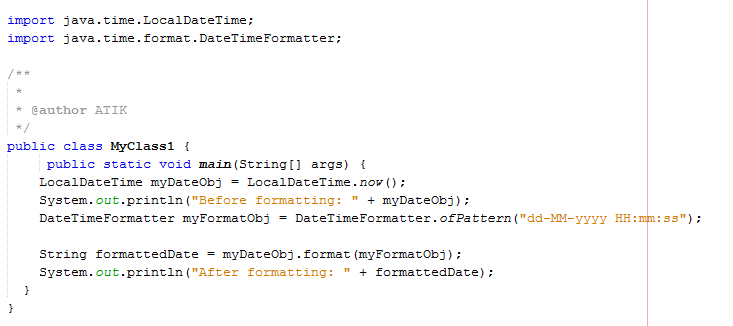
**O.4 Memformat Tanggal dan Waktu**

1. Value: yyy – MM – dd

Output dari proses running:

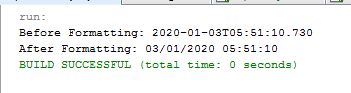


Source code: code program java pada java date untuk memformat tangggal dan waktu dengan format yyyy – MM – dd

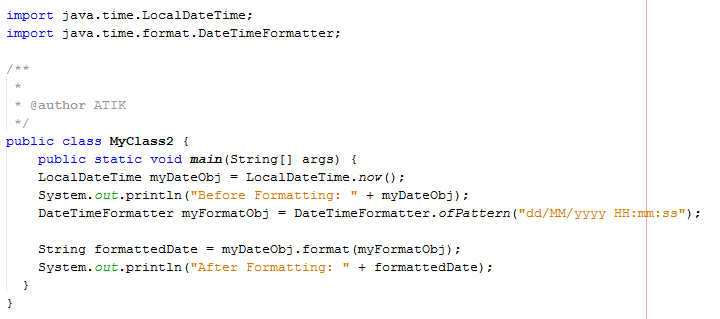


1. Value: dd/MM/yyy

Output dari proses running:

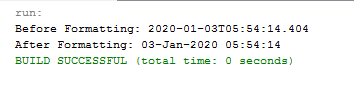


Source code: code program java pada java date untuk memformat tanggal dan waktu dengan format dd/MM/yyyy

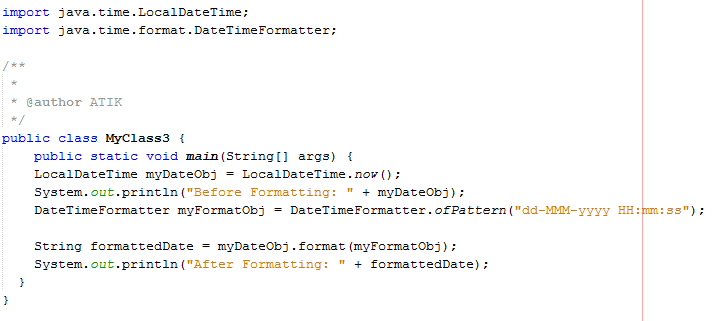


1. Value: dd – MMM – yyyy

Output dari prose running:

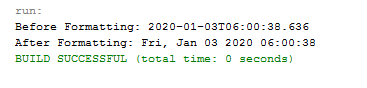


Source code: code program java pada java date untuk memformat tanggal dan waktu dengan format dd – MM – yyyy

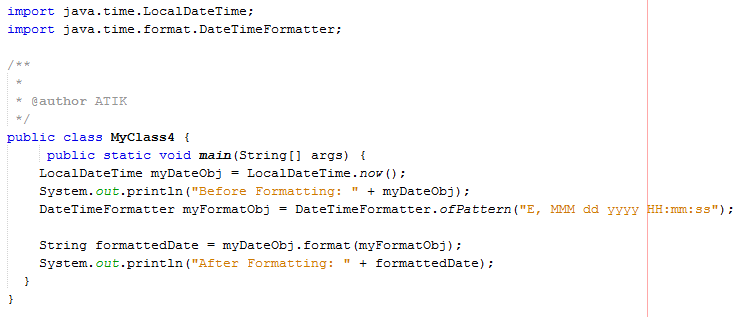


1. Value: E, MMM dd yyyy

Output dari proses running:



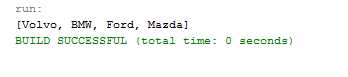
Source code : code program java pada java date untuk memformat tanggal dan waktu saat ini dengan format E, MMM dd yyyy.



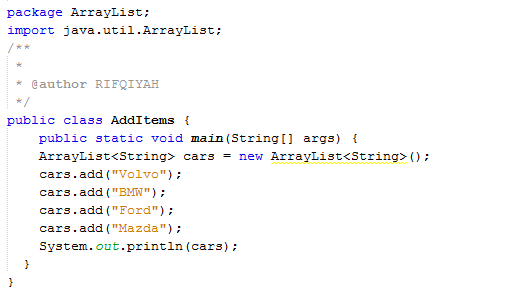
1. **Java ArrayList**

**P.1 Menambahkan Item**

Output dari proses running:

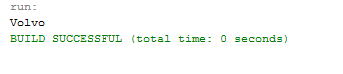


Source code: code program java pada java arraylist untuk menambahkan item

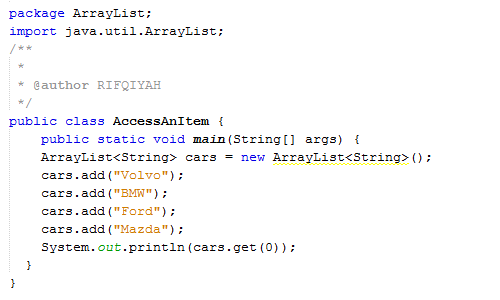


**P.2 Mengakses Item**

Output dari proses running:

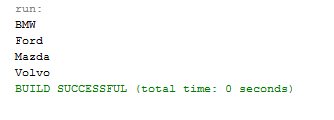


Source code: code program java pada java arraylist untuk mengakses sebuah item

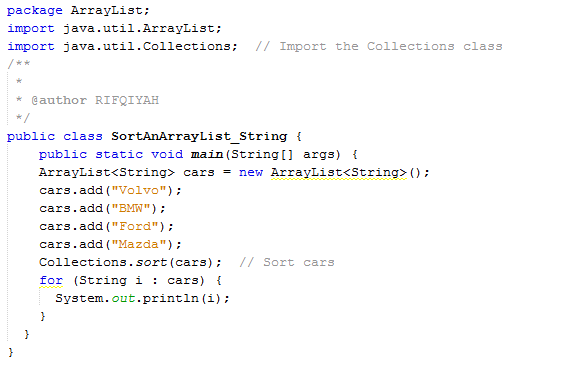


**P.3 Mensortir Daftar Array**

Output dari proses running:



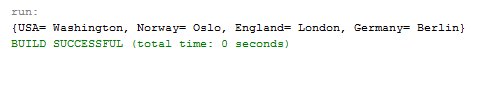
Source code: code program java pada java arraylist untuk mensortir daftar array dengan tipe data string.



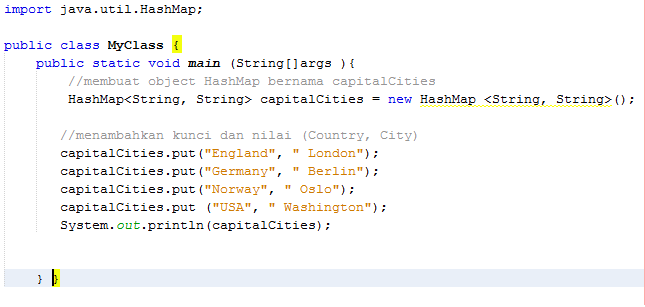
1. **Java HashMap**

**Q.1 Menambahkan Item**

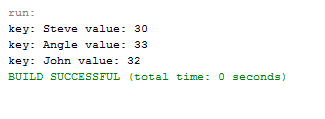
Output dari proses running:



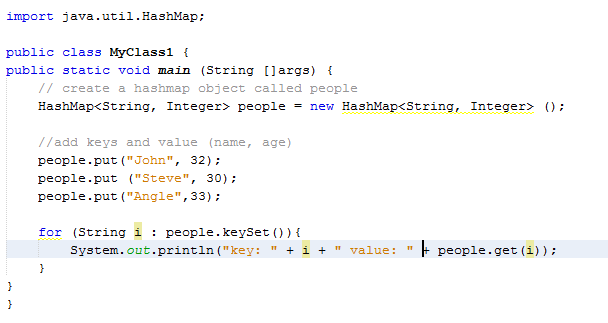
Source code: code program java pada java HashMap untuk menambahkan item dengan kata kunci string dan nilai string



Output dari proses running:



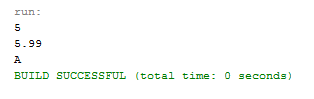
Source code: code program java pada java HashMap untuuk menambahkan item dengan kunci String, dan nilai Integer



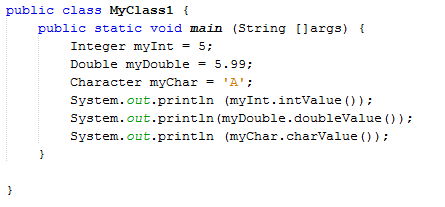
1. **Java Wrapper**

**R.1 Membuat Objek**

Output dari proses running:

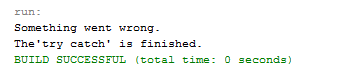


Source code: code program java pada java wrapper class untuk membuat objek

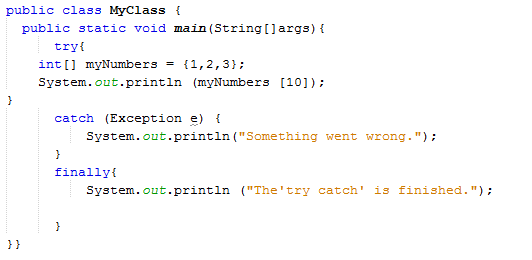


1. **Java Exceptions**

Output dari proses running:



Source code: code program java pada java exceptions dengan menggunaan kata kunci try and catch

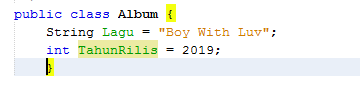


**BAB IV**

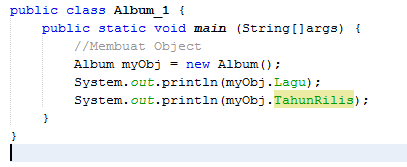
**STUDI KASUS**

1. **Java Class and Objects**

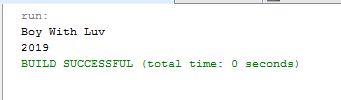
Untuk membuat sebuah studi kasus tentang java class and object, maka buatlah 2 class, bernama Album, dan Album\_1. Pada class Album dibuat sebuah variabel Lagu dan TahunRilis beserta tipe variabelnya. Pembuatan sebuah variabel digunakan sebagai penampung nilai yang telah dibuat.



Dalam class Album\_1 telah dibuat sebuah object bernama Album dan memberikan 2 object di dalamnya, karena pada class Album terdapat 2 variabel. Jika dituliskan kodenya seperti berikut:



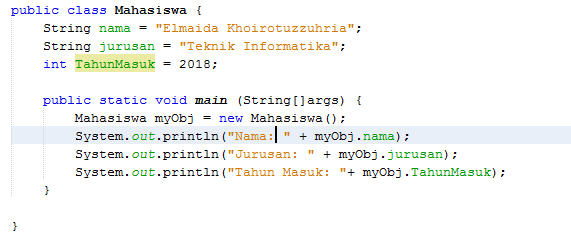
Untuk menjalankan sebuah program, hanya dapat menjalankan class yang memiliki contains main class.



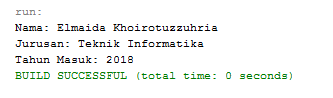
Di dalam tampilan output diatas, telah menampilkan object yang dipanggil.

1. **Java Class Attributes**

Pada studi kasus java class attribute, telah dibuat class bernama mahasiswa yang didalamnya terdapat 3 variabel masing-masing bertipe String dan Integer. String untuk data berupa teks, dan integer untuk data berupa bilangan bulat. Di dalam class mahasiswa, dibuat Objek bernama mahasiswa, dan source code untuk memanggil atribut.



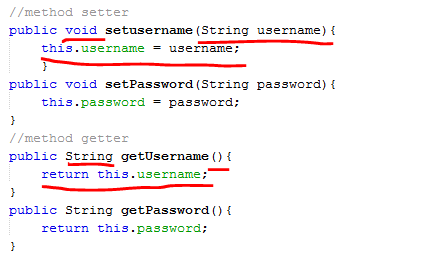
Berikut adalah tampilan outputnya:



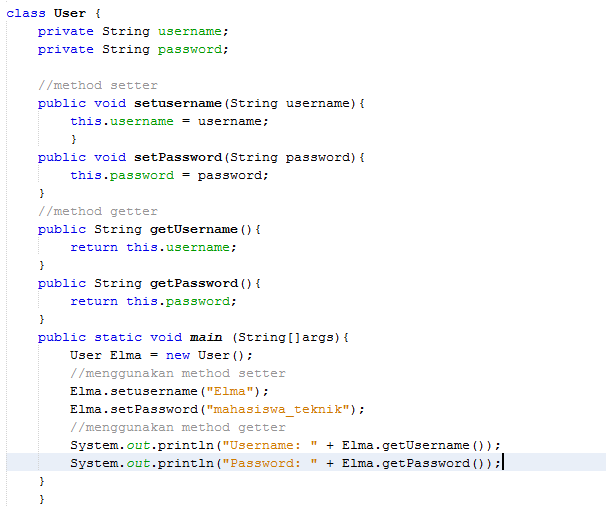
1. **Java Class Methods**

Didalam studi kasus java class method, telah digunakan method berupa getter and setter untuk enkapsulasi. Method setter dan getter harus diberikan modifier public, karena method ini akan diakses dari luar class.

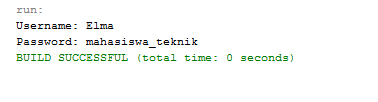
Perbedaan method setter dan getter terletak pada nilai kembalian, parameter, dan isi dari methodnya. Method setter tidak memiliki nilai kembalian void (kosong), karena hanya bertugas untuk mengisi data ke atribut. Sedangkan method getter memiliki niali kembalian sesuai dengan tipe data yang akan diambil.



Dapat dilihat perbedaan dari method getter dan setter pada gambar. Berikut adalah source code lengkapnya untuk studi kasus java class method

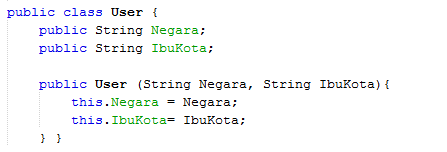


Tampilan output:

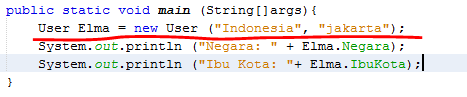


1. **Java Constructors**

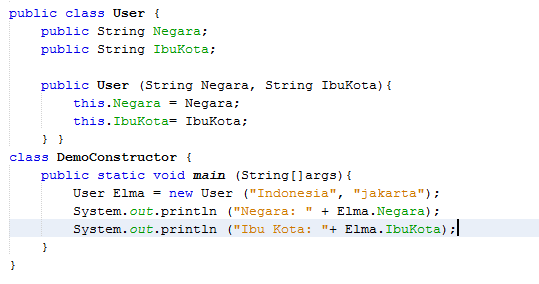
Pada studi kasus untuk java constructor digunakan constructor dengan parameter yang biasanya digunakan untuk menyiapkan data untuk class. Untuk melakukan ini, telah dibuat parameter sebagai inputan untuk constructor.



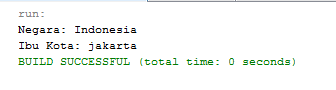
Pada kode class User diatas, telah ditambahkan parameter **Negara** dan **IbuKota** ke dalam constructor. Berarti saat membuat object, harus menambahkan nilai parameter seperti berikut:



Code pada program java secara lengkap:



Hasil outputnya:

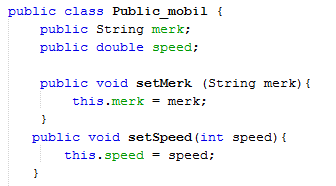


1. **Java Modifiers**

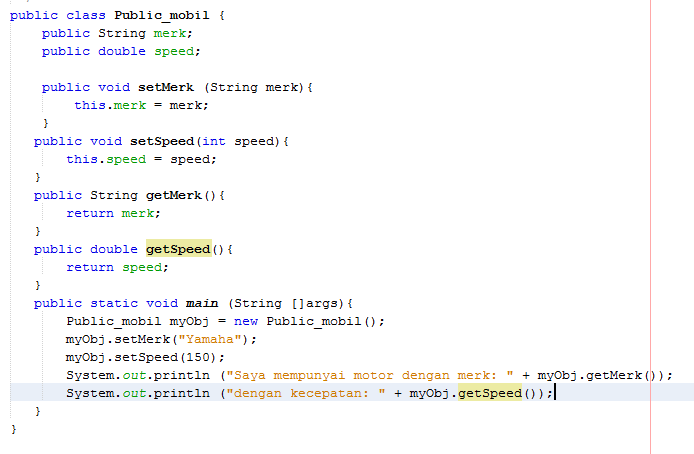
**E.1 Public**

Java modifier public dapat diakses dari mana saja. Pada studi kasus kali ini, hanya dibuat satu class saja agar memudahkan untuk dipahami. Jika seandainya dibuat 2 class yang berbeda dengan package yang berbeda juga bisa, dikarenakan menggunakan modifiers public.

Contoh source code untuk modifiers public:

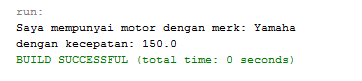


Telah dibuat class bernama public\_mobil dengan memiliki 2 variabel masing-masing data bertipe String dan Integer. Berikut ini adalah source code lengkapnya:



Untuk memanggil atributnya, maka menggunakan method dari getter dan setter, juga telah dibuat object baru dengan nama **myObj.**

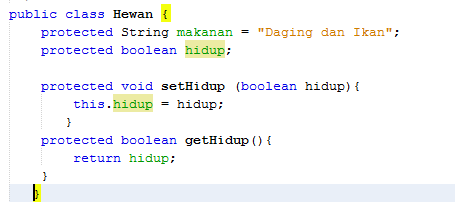
Tampilan output:



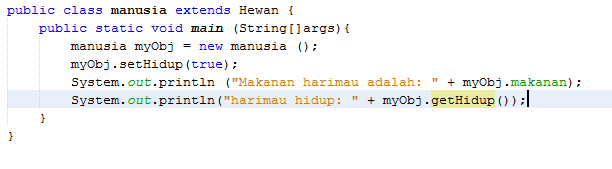
**E.2 Protected**

Variabel, method atau konstruktor yang di deklarasikan protected dapat diakses oleh subclass atau class lain yang berada dalam satu package.

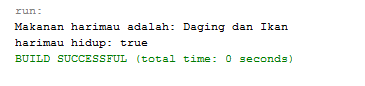
Pada program java ini, dibuat 2 class bernama Hewan dan manusia. Pada class hewan diisikan atribut seperti gambar diawah ini dengan menggunakan method get and set, juga Boolean.



Selanjutnya, pada class manusia, akan mewarisi sifat-sifat dari class hewan. Dan pada class manusia berisi main method. Telah dibuat juga object baru dengan nama myObj.



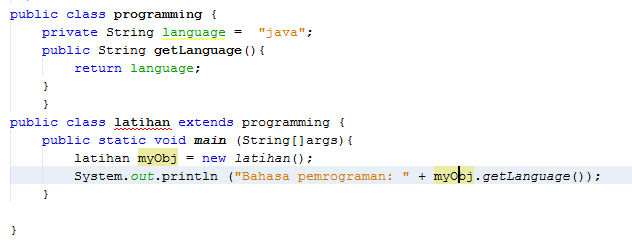
Tampilan output:



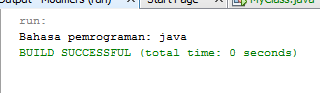
**E.3 Private**

Variabel dan method yang diberikan hak akses private hanya bisa diakses oleh class itu sendiri, data-data tersebut tidak bisa diwariskan pada subclass atau class lainnya.

Agar class lain dapat mengaksesnya maka perlu dibuat method yang mempunyai hak akses public (package yang berbeda) atau protected (package yang sama). Berikut adalah source code lengkapnya:



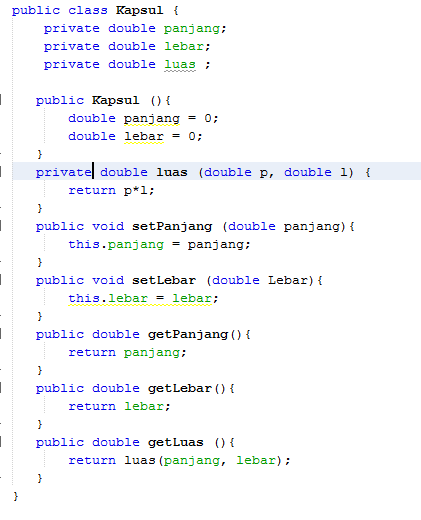
tampilan output:



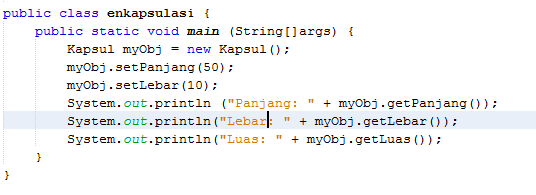
Jika diperhatikan, untuk mengakses variabel language memerlukan sebuah method public yang mengembalikan nilai dari variabel tersebut. Jika mengakses variabel language secara langsung, maka akan terjadi kesalahan/error.

1. **Java Encapsulation**

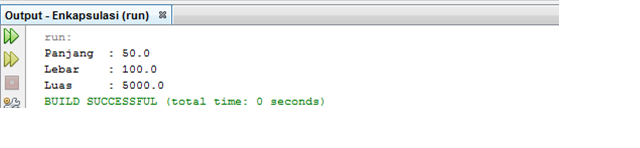
Untuk studi kasus pada java encapsulation,digunakan kode untuk menghitung luas persegi dengan bilangan yang sudah ditetapkan. Telah dibuat dua class dengan nama Kapsul dan enkapsulasi. Pada class kapsul diisikan dengan variabel dan method get and set agar dapat dipanggil dalam class enkapsulasi.



Pada class enkapsulasi berisi main method serta atribut untuk memanggil variabel pada class kapsul, dan telah dibuat object baru dengan nama myObj.



Tampilan output:



**G. Java Package/API**

**G.1 Class Scanner**

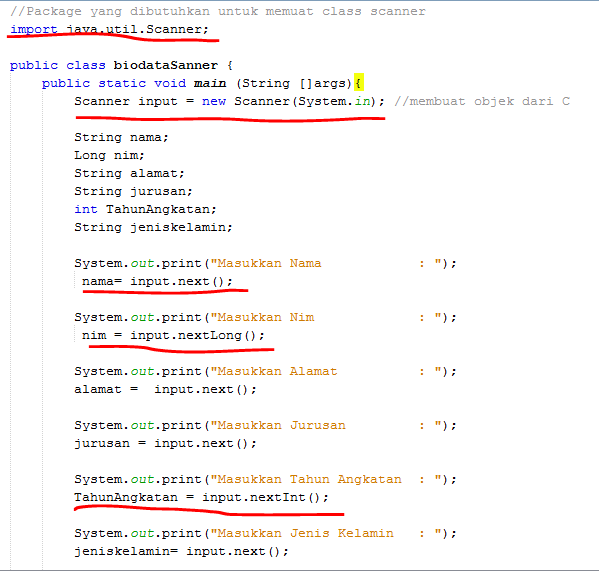
Pada studi kasus java package/API class scanner, akan dibuat contoh menginputkan biodata mahasiswa. Untuk membuat class scanner diperlukan mengimport dulu.

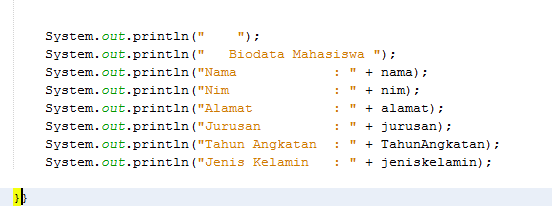


Pada penggunaan scanner untuk menerima input, perlu di deklarasikan sebuah variabel sebagai untuk mewakili scanner pada system.in, perhatikan gambar berikut ini. Jadi, pada saat membuat sebuah fungsi input, hanya perlu memanggil variabel “input” untuk mewakili fungsi system.in



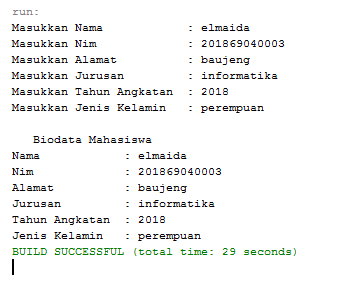
Berikut ini adalah source code lengkapnya:





Perlu diketahui tentang code yang diberi garis merah dibwahnya. Untuk class scanner di haruskan untuk mengimport scanner terlebih dahulu, dan juga membuat object scanner yang berguna untuk membuat object. Untuk menginputkan data pada class scanner terbagi menjadi beberapa tipe data, yaitu untuk tipe data String maka menggunakan **input.next();** , jika tipe data Long (untuk angka berkapasitas besar) maka menggunakan **input.nextLong();** , dan untuk tipe data integer maka menggunakan **input.nextInt();**

Tampilan output:



**G.2 Buffer Reader**

**BufferedReader** adalah class yang berada pada oaket java.io , io adalah singkatan dari input output. Class BufferedReader juga mampu menerima input dari file dari sebuah jaringan. Sehingga untuk penggunaan BufferedReader di java harus mengimportkan 3 class pada java.io sekaligus yaitu:

**import java.io.BufferedReader;**

**import java.io.IOException;**

**import java.io.inputStreamReader;**

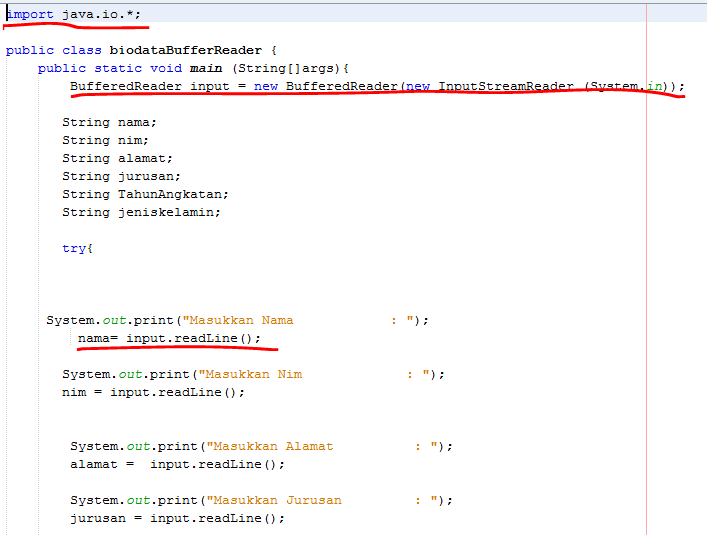
atau untuk lebih mempersingkat pengkodean dapat menggunakan langsung dari clas io, karena dapat memanggil 3 sekaligus class pada java paket io. Untuk membuat java package/API dalam buffer reader diperlukan untuk mengimport terlebih dahulu pada package.

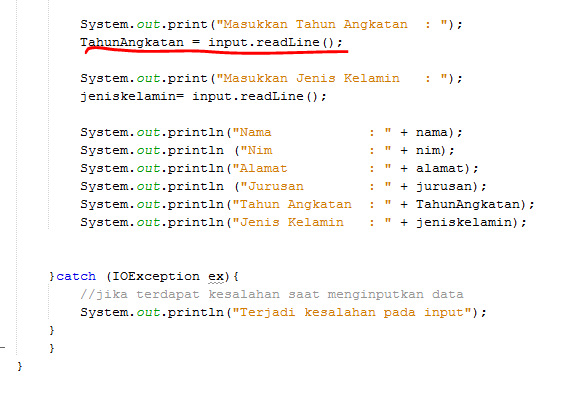


Dan membuat object baru pada class agar dapat memuat object yang nantinya akan di inputkkan.



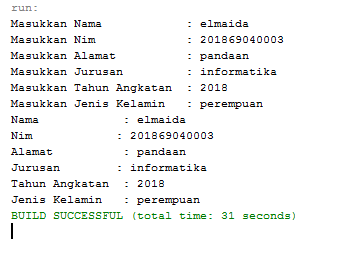
Untuk lebih jelasnya mengetahui perbedaan java package/API pada class scanner, buffer reader, console, dan JOption Pane. Perhatikan source code yan digaris bawahi.





Terdapat beberapa perbedaan dengan class scanner yaitu mengimport di package, membuat object baru, dan untuk menginputkan datanya. Untuk membaca input data tipe string maka menggunakan readLine(), dan untuk tipe data integer menggunakan read().

Tampilan output:



**G.3 Class Console**

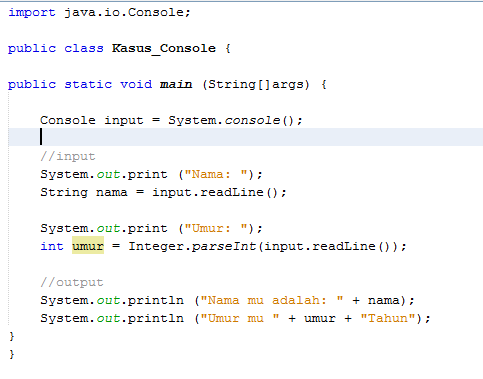
Class console merupakan class yang berada pada paket .io, sama dengan bufferedReader. Sebagai pembedanya adalah jjka bufferedReader berjalan pada console netbeans langsung atau biasa disebut bagian output, namun class console berjalan pada terminalnya sendiri. Dengan kata lain, class console tidak bisa dijalankan langsunng pada netbeans. Untuk penggunaan class console di java harus mengimportkan class pada java.io, yaitu:



Untuk menerima input dengan console, perlu di deklarasikan sebuah variabel untuk mewakili console.



Berikut ini adalah kode lengkap pada penggunaan class console:



Integer.parseInt berguna untuk merubah angka/ numeric yang bertipe String, menjadi bertipe integer.

**G.4 JOptionPane**

JOptionPane merupakan sebuah class yang menyediakan jendela dialog. Sebelum menggunakan JOptionPane, harus mengimpor terlebih dahulu ke dalam program, yaitu:

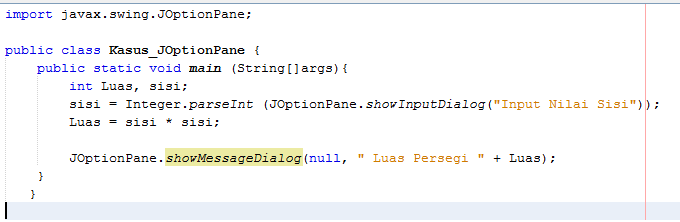


Class JOptionPane dapat digunakan lansgung tanpa harus membuat object, seperti gambar dibawah ini:

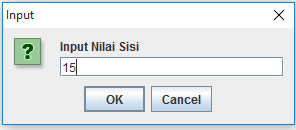


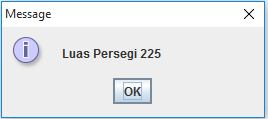
Null adalah parameteruntuk komponen induknya, bisa diberikan null atau this.

Untuk lebih jelasnya, dapat dilihat pada studi kasus program java JOption Pane untuk menghitung luas persegi.



Hasil output:

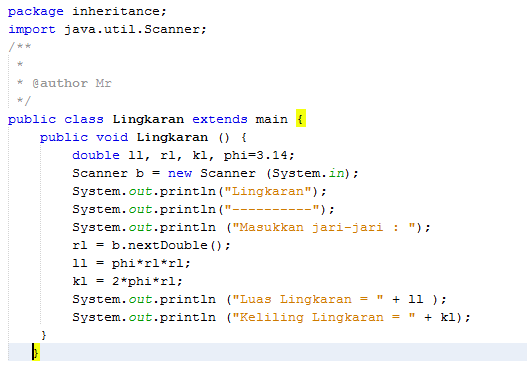




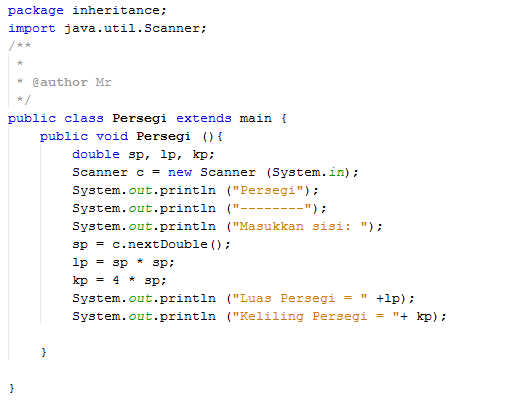
1. **Java Inheritance**

Program java yang dibuat untuk studi kasus java inheritance ini adalah perhitungan luas bangun datar persegi,segitiga, lingkaran, dan persegi panjang dengan menginputkan data sendiri, maka dari itu harus mengimport class scanner. Buat sebuah project dalam satu package bernama **inheritance** dan class yang terdiri dari 5 class, masing-masing bernama segitiga, lingkaran, persegi, persegi panjang, dan main.

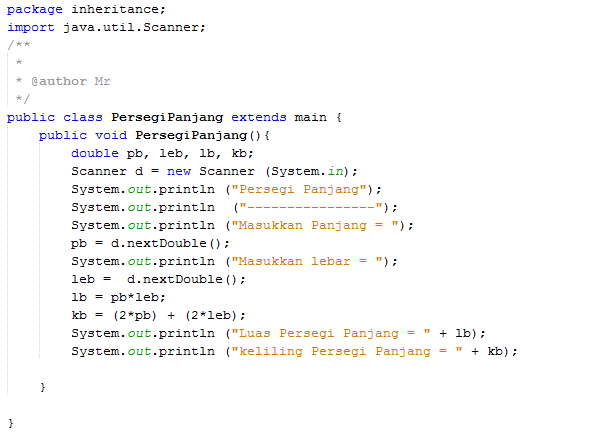
File: inheritance/Lingkaran.java



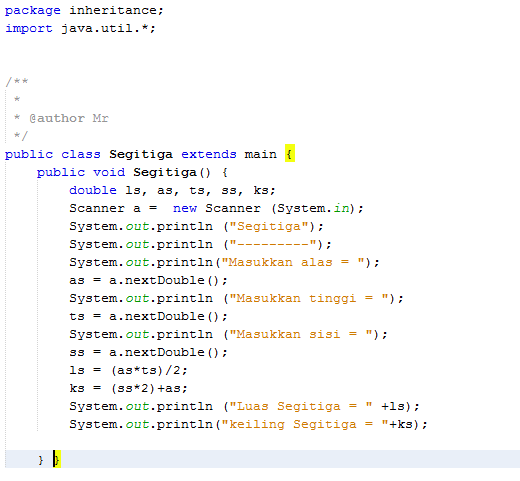
File: inheritance/Persegi.java



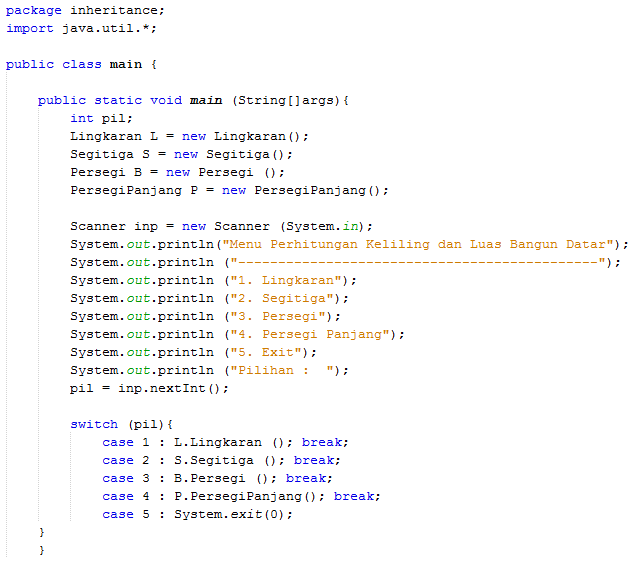
File: inheritance/PersegiPanjang.java



File: inheritance/ Segitiga.java

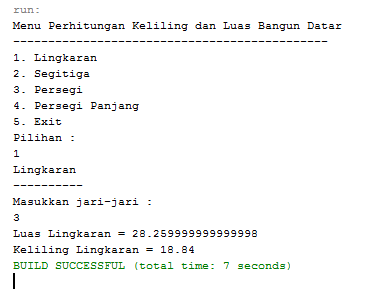


File: inheritance / Main.java

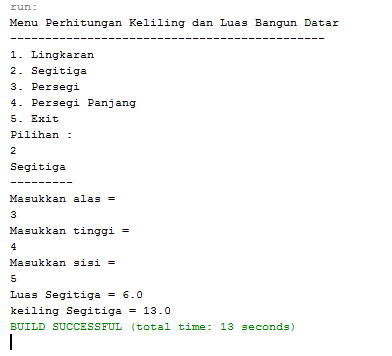


Hasil output:

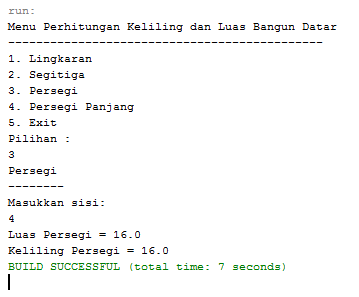
File: inheritance / Lingkaran.java



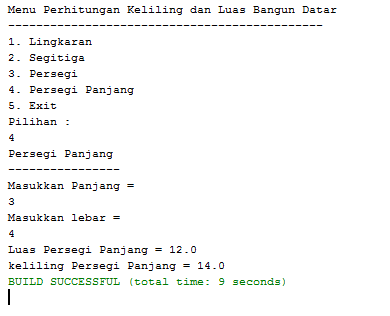
File: inheritance /Segitiga.java



File: inheritance / Persegi.java

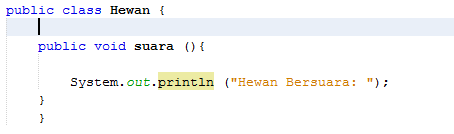


File: inheritance / PersegiPanjang.java



1. **Java Polymorphism**

Polymorphism artinya memiliki banyak bentuk.

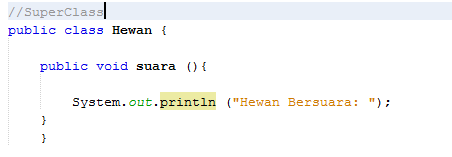


Pada kode diatas, objek hewan hanya dapat melakukan satu jenis aktifitas, yaitu suara. Namun, perl diingat aktifitas itu bisa dilakukak oleh objek sub class dari class hewan atau class apa saja yang diatasnya terdapat hewan yang terkait dalam hirarki inheritance.

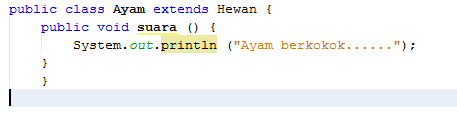


Berikut adalah kode lengkap dari beberapa class

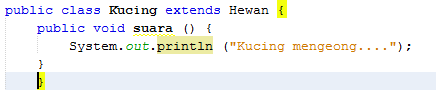
File: Studi\_Kasus / Hewan.java



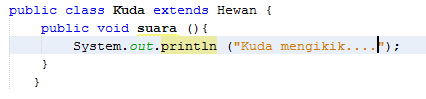
File: Studi\_Kasus / Ayam.java



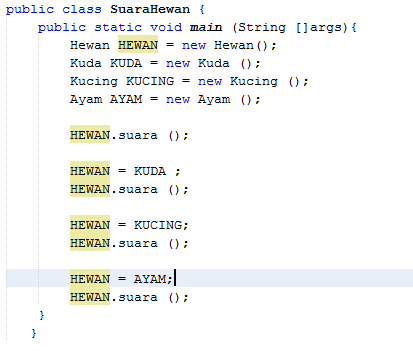
File: Studi\_Kasus / Kucing.java



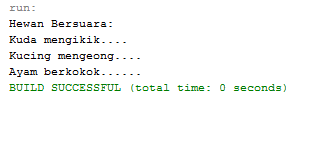
File: Studi\_Kasus / Kuda.java



File: Studi\_Kasus / SuaraHewan.java



Hasil output:



1. **Java Inner Class**

**J.1 Static**

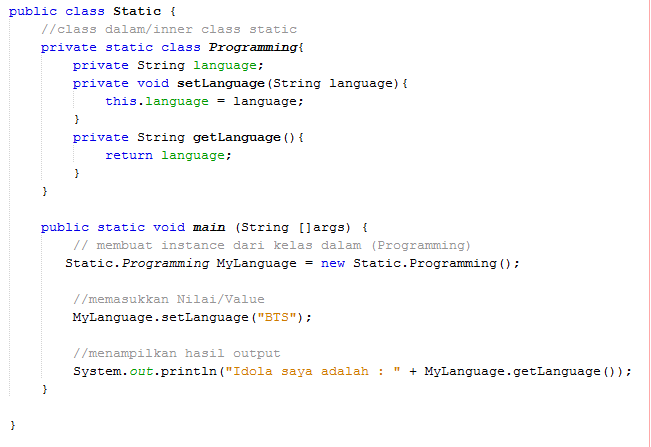
Untuk mengubah sebuah Inner Class menjadi Static, hanya perlu menambahkan kata kunci static di belakang class.



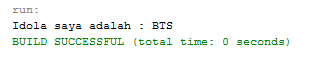
Inner class static hanya membuat instance dari class dalam saja.



Berikut adalah kode lengkap dari studi kasus inner class static

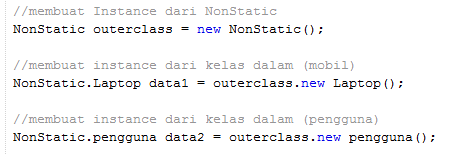


Hasil output:

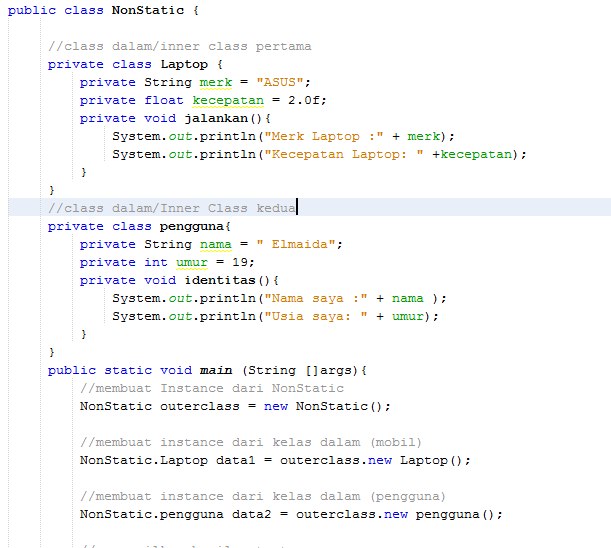


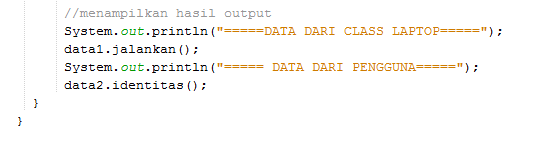
**J.2 Non-Static**

Untuk mengakses variabel atau method pada class luar, perlu membuat instance/objek class luar di dalam inner class.

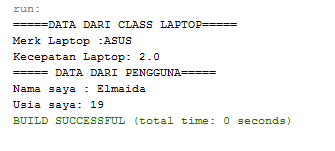


Source code lengkapnya seperti dibawah ini:





Hasil output:

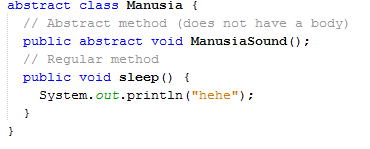


1. **Java Abstraksi**

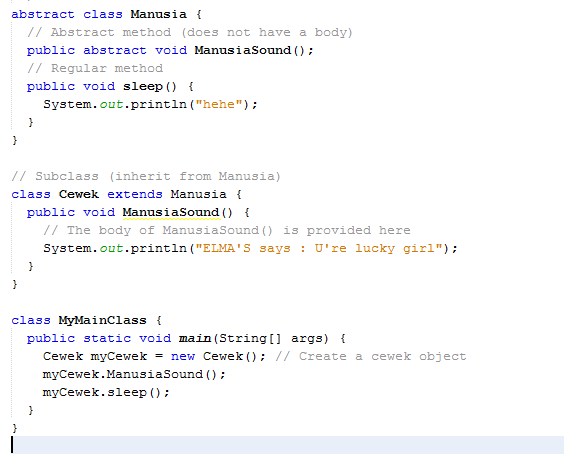
Untuk java abstraksi, digunakan tambahan kata kunci abstract pada classnya.



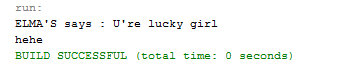
Pada abstract class tidak memiliki body atau method didalamnya.



Berikut adalah code lengkapnya:

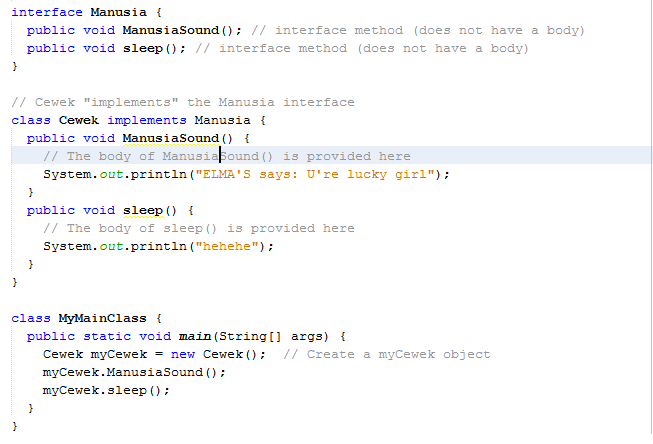


Hasil output:



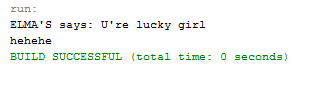
1. **Java Interface**

Berikut adalah studi kasus dari java interface, dengan membuat class bernama Manusia.



Class MyMainClass menggunakan interface Manusia dengan keyword implements. Program diatas, menggunakan interface sebagai induknya, yang membedakan dengan class induk yaitu interface hanya mendeklarasikan method-method nya. Kemudian method tersebut di implementasikan oleh class-class yang mengimplementasikan interface tersebut.

Hasil output:

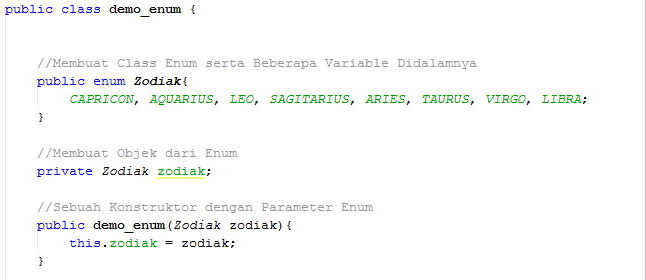


1. **Java Enum**

Enum merupakan salah satu class yang dapat digunakan untuk memuat kumpulan variabel yang telah didefinisikan menjadi tipe data konstanta. Setiap objek baru yang didefinisikan berdasarkan tipe data enum, objek tersebut dapat mengakses atribut atau method (Enum) lalu mengaplikasikan nilai atau value yang ada di dalamnya.

Pada studi kasus java enum, telah dibuat class enum bernama demo\_enum, di dalam Main class, yang terdiri dari nama-nama zodiac, dimulai dari zodiac aquarius.

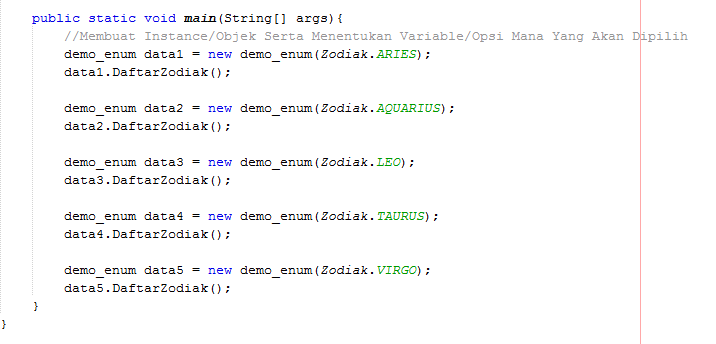
Pada baris code, dibuat variabel refernsi zodiac, lalu membuat constructor milik Main Class.



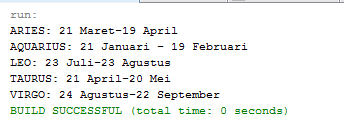
Untuk menentukan nilai output, berdasarkan variabel yang dideklarasikan pada enum zodiac, disini menggunakan switch case yang didefinisikan di dalam method yang bernama DaftarZodiak.



Setelah itu, pada method lain, akan di eksekusi program java dengan cara membuat dan mengintalasi objek baru, llau memanggil method DaftarZodiak().



Hasil output:



1. **Java User Input (Scanner)**

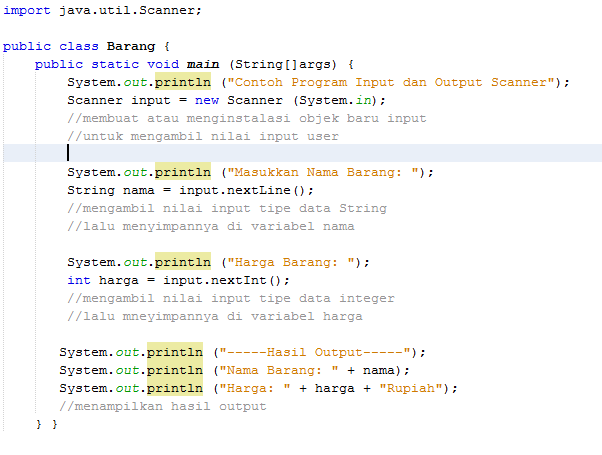
Pada studi kasus java user input, pertama akan dibuat atau menginstalasi objek baru kedalam class scanner yaitu input.



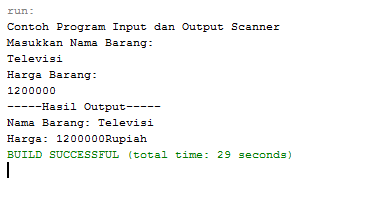
Disini, digunakan system.in agar user atau pengguna dapat menginputkan nilai lewat keyboard. Pada code in, merupakan objek dari class InputStream yang didefinisikan pada class System.

Kemudian objek tersebut digunakan untuk mengakses method input.nextLine(), yang berfungsi untuk mengambil nilai input tipe data string, dan method input.nextInt() befungsi untuk mengambil nilai input tipe data integer.

Berikut ini adalah kode lengkap dari studi kasus java user input menggunakan scanner.



Hasil output :



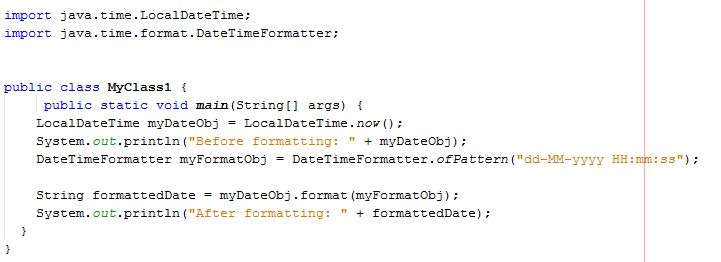
1. **Java Date**

Pada studi kasus java date untuk menampilkan tanggal dan waktu saat ini, buat class dengan nama MyClass. Date untuk menampilkan tanggal sekarang dengan format yang diinginkan, sedangkan time untuk menampilkan waktu sekarang, waktu yang akan datang, waktu yang akan ditampilkan sesuai dengan inisiasi output waktu yang akan ditampilkan.

Berikut adalah source code lengkap untuk membuat tampilan tanggal dan waktu sekarang dengan format yang berbeda.

1. Tanggal: dd-MM-yyyy

Waktu: HH:mm:ss



Berikut penjelasan source code diatas:



Source code untuk menampilkan tanggal dan waktu sekarang, juga untuk memformat tanggal sesuai yang diinginkan.



Source code membuat object baru untuk menampilkan tanggal dan waktu saat ini sebelum di format.

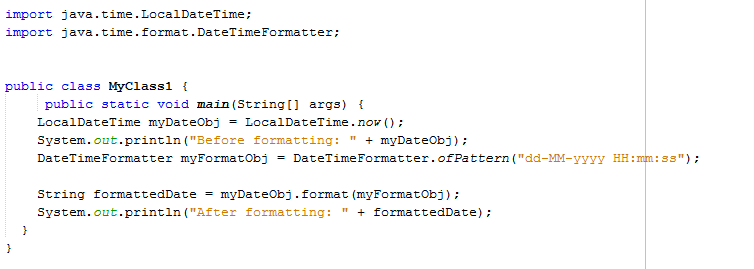


Source code membuat object baru untuk memformat tanggal dan waktu.

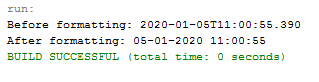


Source code untuk memanggil object baru yang telah di format.

Jadi, source codelengkapnya yaitu:



Hasil output:



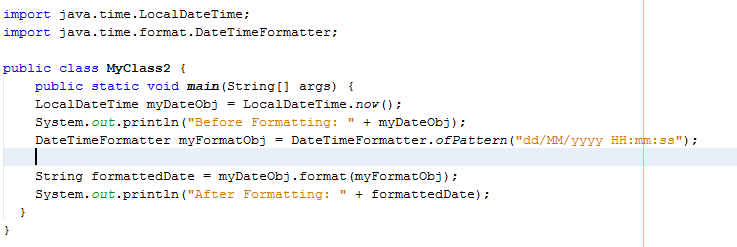
1. Tanggal: dd/MM/yyyy

Waktu: HH:mm:ss

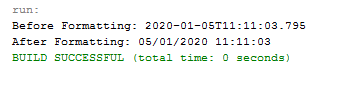
Source code untuk menampilkan format tanggal dan waktu saat ini hampir sama dengan sebelumnya,namun ada perbedaan sedikit pada source code untuk format tanggal dan waktunya.



Source code lengkapnya:



Hasil output:



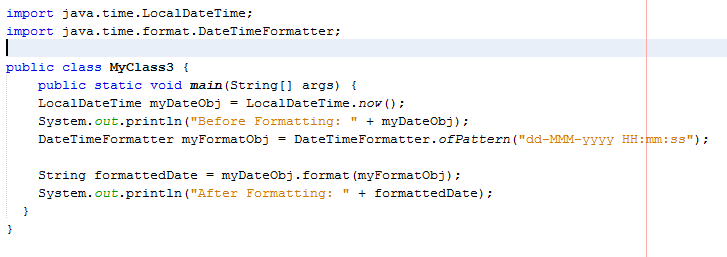
1. Tanggal: dd-MMM-yyyy

Waktu: HH:mm:ss

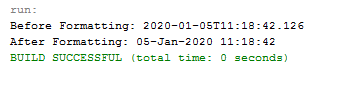
Source code untuk memformat tanggal dan waktu



Source code lengkapnya yaitu:



Hasil output:



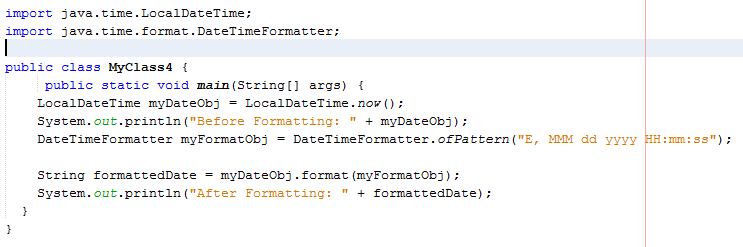
1. Tanggal: E, MMM dd yyyy

Waktu: HH:mm:ss

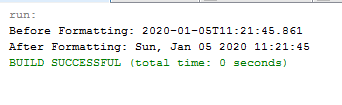
Source code untuk memformat tanggal dan waktu sesuai yang dituliskan.



Source code lengkap untuk menampilkan dan memformat tanggal dan waktu sesuai yang dituliskan.



Hasil output:



1. **Java ArrayList**

ArrayList merupakan class yang memungkinkan membuat sebuag object untuk menampung apapun.

Untuk menggunakan ArrayList, harus mengimpor terlebih dahulu.



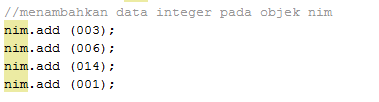
Kemudian, membuat object sesuai dengan tipe data.



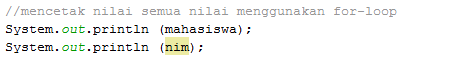
Source code diatas untuk membuat object baru dengan tipe data String. Gambar dibawah ini merupakan source code untuk membuat object dengan tipe data Integer.



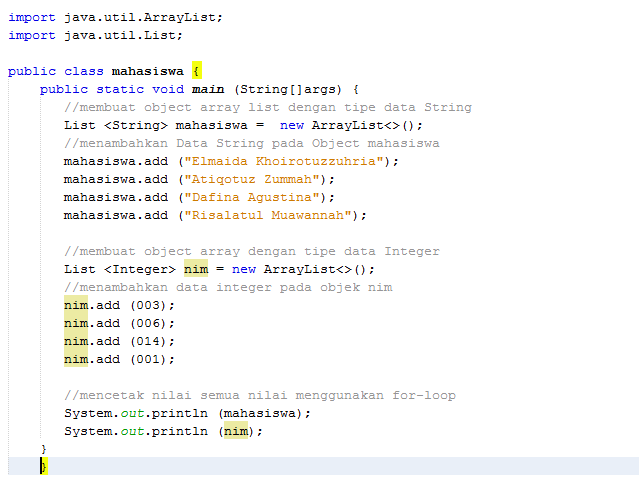
Source code untuk menambahkan data pada tipe data String menggunakan method put().



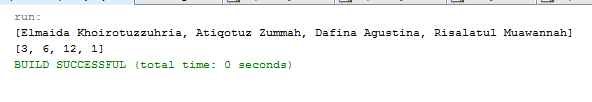
Source code untuk menambahkan data pada tipe data Integer menggunakan method put(). Setelah menambahkan data pada masing-masing object, kemudian panggil semua object menggunakan method for-loop.



Source code lengkapnya yaitu:



Hasil output:



1. **Java HashMap**

Sebelum menggunakan HashMap, terlebih dahulu harus mengimport HashMap pada package.

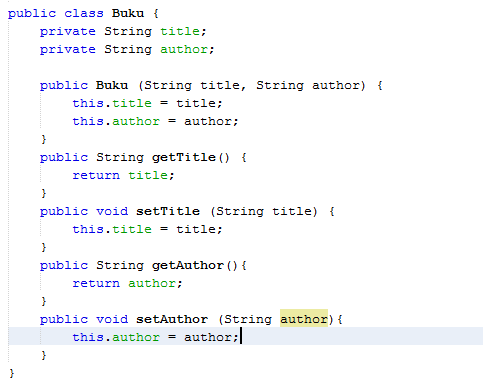


Unntuk menggunakan HashMap, diharuskan untuk membuat objeknya terlebih dahulu. Object dari hashMap dapat dibuat dengan kata kunci **new.**

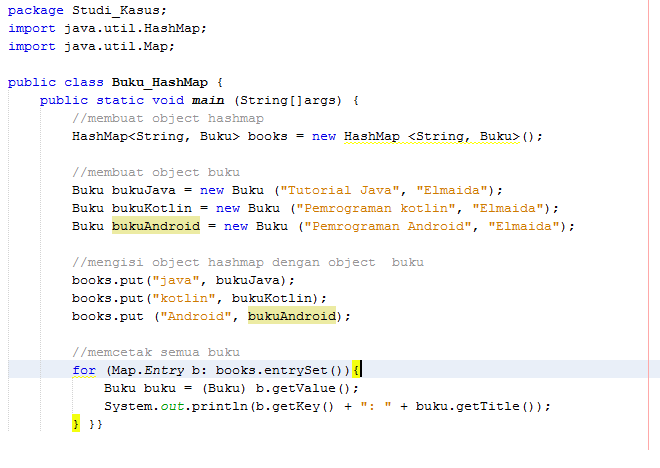


Pada source code diatas, telah dibuat object HashMap bernama books. Object ini dapat digunakan untuk menyimpan koleksi data. Untuk studi kasus java HashMap,dibuat dua class, yaitu Buku.java dan Buku\_HashMap.java dalam satu package.

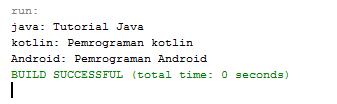
Berikut isi dari class Buku.java



Berikut isi class dari Buku\_HashMap.java



Hasil output:



1. **Java Wrapper**

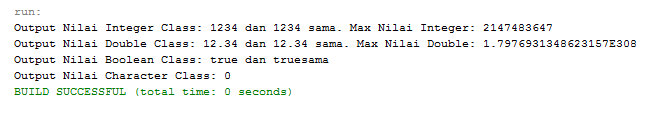
Source code untuk program java wrapper class yaitu:



Pada studi kasus diatas digunakan wrapper class seperti integer, double, Boolean, dan character. Kemudian menginstalasi objek tersebut dan menggunakan constructor untuk mendefinisikan nilainya pada pernyataan.

Kecuali pada character class, di dalam wrapper class dapat juga digunakan dua tipe pernyataan dengan menggunakan tipe data primitive pada umumnya dan juga bertipe data string. Untuk mengakses method dapat juga dilakukan secara langsung pada wrapper class dengan method keyword static yang telah didefinisikan di dalam wrapper class. Seperti contohnya **Integer.MAX\_VALUE,** dan juga **Double.MAX\_VALUE** yang digunakan untuk menampilkan output hasil, nilai maximal dari tipe data integer dan juga double.

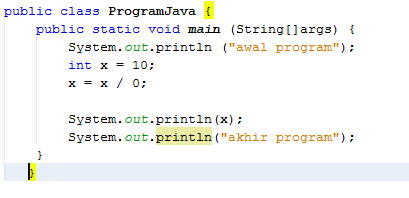
Hasil output:



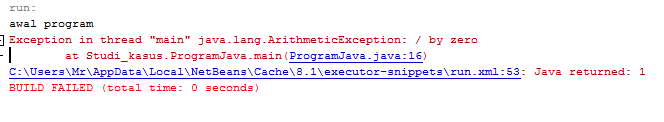
1. **Java Exceptions**

Untuk menangani error pada java,digunakan sebuah statement yang bernama try..catch. statement tersebut digunakan untuk mengurung eksekusi yang menampilkan error dan dapat membuat program tetap berjalan tanpa dihentikan secara langsung. Error yang ditangani oleh try..catch biasa disebut dengan exception.

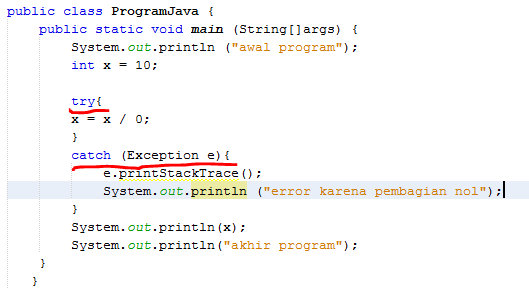
Untuk java exception, dibuat sebuah studi kasus yang mengandung error ketika dijalankan. Berikut adalah source code nya:



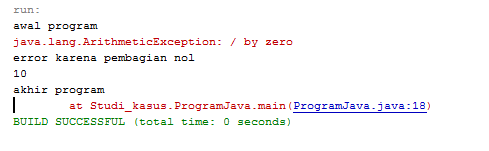
Ketika dijalankan maka akan mendapat hasil output sebagai berikut:



Berbeda ketika operasi pembagian nol diatas dikurung dengan try..catch, maka hasil eksekusi program akan sedikit berbeda.

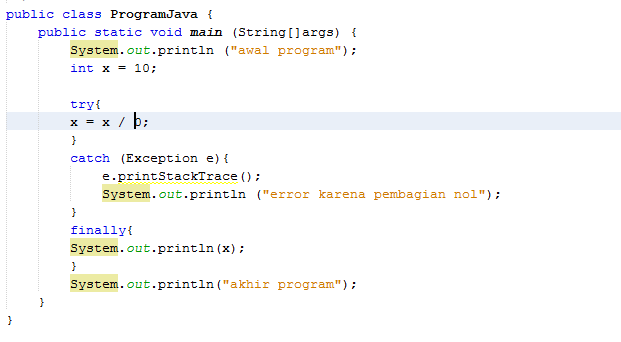


Hasil output:

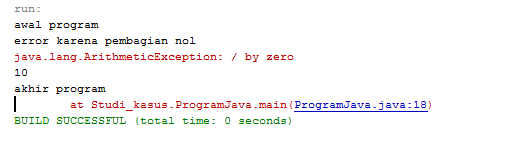


Ketika program di eksekusi kembali maka program akan dijalankan bagian akhir program walaupun terjadi error di tengah-tengah eksekusi.

Untuk menghindari error, akan digunakan statemen finally. Perhatikan source code berikut:



Ketika menjalankan program tersebut maka akan didapat:



Tetap diperoleh hasil error karena pembagi tidak boleh nol.

**BAB V**

**PENUTUP**

1. **KESIMPULAN**

Pemrograman Berorientasi Objek (*Object Oriented Programming* atau OOP) merupakan paradigma pemrograman yang berorientasikan kepada objek. Objek adalah struktur data yang terdiri dari bidang data dan metode bersama dengan interaksi mereka untu merancang aplikasi dan program computer. Dalam OOP data dan fungsi-fungsi yang akan mengoperasikannya digabungkan menjadi satu kesatuan yang dapat disebut sebagai objek. Pembahasan mengenai orientasi objek tidak akan terlepas dari konsep objek seperti *inheritance* atau pewarisan, *encapsulation* atau pembungkusan, dan *polymorphism* atau lainnya. Konsep-konsep ini merupakan fundamental dalam orientasi objek yang perlu dipahami serta digunakan dengan baik, dan menghindari penggunaanya yang tidak tepat. Dalam laporan praktikum ini dilengkapi dengan studi kasus serta hasil percobaan pada setiap babnya.

1. **SARAN**
2. Laporan praktikum ini dapat dikembangkan lebih lanjut sehingga memajukan pendidikan teknologi informasi dan komunikasi Indonesia, terutama pada bidang studi pemrograman brorientasi objek menggunakan bahasa pemrograman java.

**DAFTAR PUSTAKA**

W3Schools. 2019. *Java*.

<https://www.w3schools.com/java/default.asp>

(diakses tanggal 24 Desember 2019 pukul 10.09 WIB)