**Memahami Keyword Final Dalam Program Java**

**Class Yang Dideklarasikan Sebagai Final**

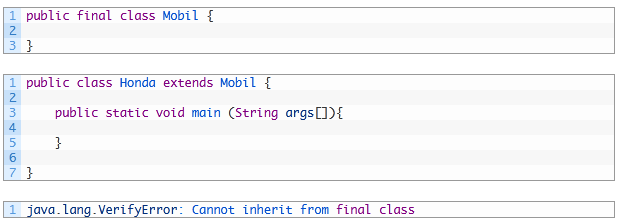
Ketika suatu class dideklarasikan sebagai final maka class tersebut tidak bisa diwariskan. Misalnya pada package java.lang di library Java kita bisa melihat jika [class String](http://bahasajava.com/belajar-java-class-string-fundamental-dan-method/) adalah class final.

Class String tersebut dideklarasikan sebagai:



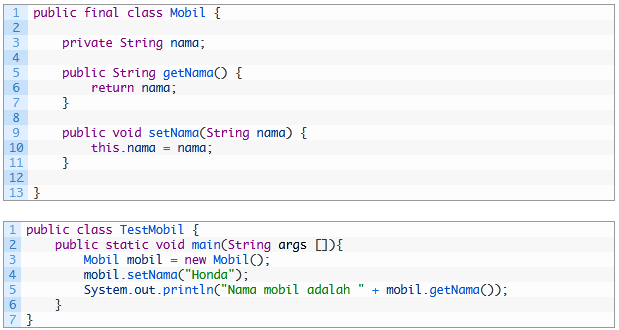
Bisa dibayangkan kalau class String tersebut tidak dideklarasikan sebagai final. Maka Anda bisa saja mengoverride method-method yang dimilikinya. Akibatnya adalah semua instance dari String itu bisa disubstitusi oleh instance yang spesifik dari subclass yang Anda miliki. Hasilnya adalah pengoperasian dari objek-objek String menjadi tidak bisa diprediksi. Karena class String itu banyak digunakan di mana saja, tentu ini adalah sebuah hal yang tidak bisa diterima. Atas dasar inilah class String dideklarasikan sebagai final.

Jadi meng-exetend-kan class final akan menyebabkan error:



Hal yang perlu Anda perhatikan adalah ketika Anda mendeklarasikan class sebagai final, maka objek dari class tersebut tidak immutable. Dalam artian bahwa Anda bisa mengganti field dari class tersebut dengan bebas.

Contoh:



Output:



Jadi final pada class artinya class tersebut hanya tidak bisa diwariskan saja.

Class yang dideklarasikan final biasanya dilakukan karena alasan keamanan. Dalam pemrograman Java mendokumentasikan setiap class yang dibuat dengan baik merupakan *good practice*.

Ketika class dideklarasikan sebagai final maka itu harus berdasarkan pertimbangan yang matang.

Karena ketika suatu class dibuat final maka tidak ada programmer lain yang bisa memperbaikinya.

Contohnya ketika Anda menggunakan sebuah class final dan class tersebut tidak memiliki *source code*. Kemudian ketika terdapat masalah pada salah satu method  maka akan menjadi sulit karena class final tersebut tidak bisa di extend.

Akibatnya method tersebut tidak bisa [dioverride](http://bahasajava.com/method-overriding-java/) untuk memperbaiki masalahnya. Dalam kasus ini kita akan kehilangan ekstensibilitas, sebuah konsep penting yang menguntungkan dari pemrograman berorientasi objek.

**Tujuan dari Class 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 class itu tidak final.
3. Class yang dibuat final sangat berguna ketika akan digunakan untuk tujuan tingkat keamanan tinggi dalam setiap aplikasi.

**Method Final**

Suatu method yang dideklarasikan sebagai final maka method tersebut tidak bisa dioverride. Anda akan menemukan banyak sekali method final di library Java.

Jadi kenapa sebuah method harus dideklarasikan sebagai final? Alasannya utamanya adalah untuk menunjukkan dan memastikan tidak ada seorangpun yang dapat mengubah definisi method tersebut.

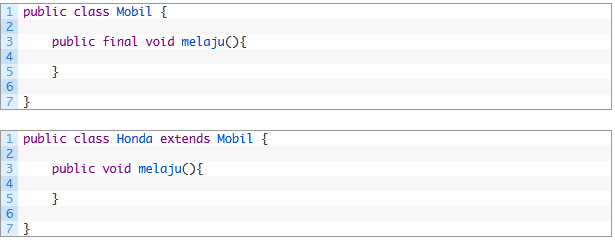
Terutama ketika berurusan dengan method-method yang bersifat sensitif dalam artian berhubungan langsung dengan fungsi kritikal dan keamanan.

Perlu diingat juga bahwa ketika suatu class dideklarasikan sebagai final, maka class tersebut tidak bisa diwariskan, dan tentu saja secara otomatis semua method yang ada di class tersebut tidak bisa dioverride. Artinya ini sama saja dengan membuat semua method tersebut menjadi final.

Ketika membuat sebuah method final, maka lebih baik memberikan dokumentasi yang baik mengenai kenapa melakukannya. Ini akan membantu Anda dan orang lain untuk menelusuri kode program Java yang Anda buat.

Ketika Anda mencoba mengoverride method final maka akan terjadi [error](http://bahasajava.com/perbedaan-error-exception-java/).

Contoh kode di bawah ini akan error karena method melaju() dari class mobil tidak bisa dioverride oleh class Honda.



Tips:

* Jika beberapa method dari class di panggil oleh method lainnya maka pertimbangkan untuk mendeklarasikan method tersebut sebagai final. Karena jika tidak, mengoveridde method tersebut bisa memberikan hasil yang tidak diharapkan.
* Ketika suatu constructor [memanggil method lainnya](http://bahasajava.com/memanggil-method-yang-tidak-mengembalikan-nilai-method-void/) maka secara umum method tersebut akan dideklarasikan sebagai final, karena jika tidak juga bisa memberikan hasil di luar dugaan.

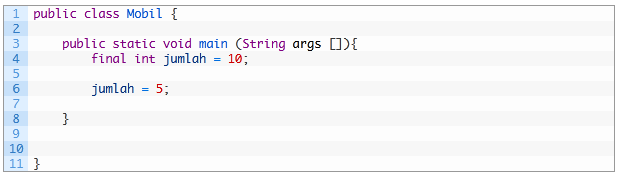
**Variabel Final Pada Program Java**

Mari kita lihat keyword final pada [variabel primitif](http://bahasajava.com/belajar-java-dasar-mendeklarasikan-variabel-untuk-pemula/) dan [variabel instance](http://bahasajava.com/variabel-instance-set-method-dan-get-method/).

**a. Variabel Primitif**

Keyword final pada variabel primitif akan membuat nilai variabel itu tidak bisa ditetapkan ulang. Dengan kata lain ketika variabel tersebut diinisialisasi maka nilainya tidak bisa diubah.

Contoh:



Akan terlontar exception:

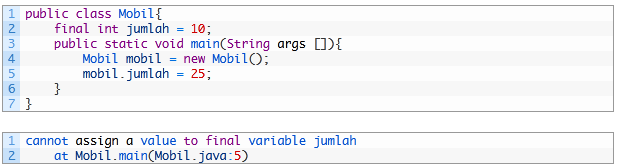


Satu perbedaan antara variabel biasa dengan variabel final adalah pada variabel biasa nilainya bisa ditetapkan ulang sedangkan pada variabel final nilainya tidak bisa di ubah setelah ditetapkan.

**b. Keyword Final Pada** [**Variabel Instance**](http://bahasajava.com/variabel-instance-set-method-dan-get-method/)

Ketika Anda memberikan suatu nilai pada variabel instance maka Anda tidak bisa mengubah nilainya. Error akan muncul ketika mencoba untuk mengubah nilai variabel instance dari suatu class dimana variabel itu telah dideklarasikan sebagai final.

Contoh:

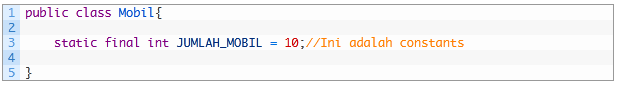


Untuk field instance final dapat diinisialisasi dengan tiga cara, yaitu:

* Ketika deklarasi dilakukan
* Dilakukan di blok penginisialisasi instance
* Dilakukan pada constructor

Sesuai dengan konvensi penamaan coding Java, setiap nama [constants](http://bahasajava.com/belajar-java-mendeklarasikan-dan-menggunakan-constants/) harus ditulis dengan huruf  kapital dan jika terdiri lebih dari satu suku kata harus dipisahkan dengan tanda “\_”.

Contoh:

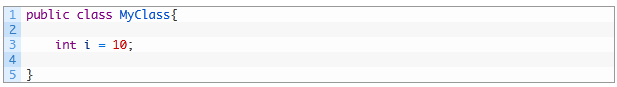


Variable static final dapat diinisialisasi dengan dua cara, yaitu:

* Ketika deklarasi langsung, misalnya seperti pada contoh di atas
* Dilakukan di blok penginisialisasi static

**c. Variabel referensi objek**

Ketika suatu variabel referensi dinyatakan sebagai final maka Anda tidak dapat menetapkan ulang objek baru pada referensi itu setelah variabel referensi tersebut merujuk ke objek. Tapi, Anda bisa mengubah state objek di mana variabel referensi itu mengarah.



public class MenggunakanFinalKeyword{  
 public static void main(String args []){  
  
 final MyClass a = new MyClass(); //variabel referensi final  
 a.i = 25;  
 //Anda bisa mengubah state dari objek dimana variabel referensi itu mengarah pada objek tersebut  
  
 MyClass b = new MyClass();  
  
 a = b; //compile time error karena tidak bisa menetapkan objek baru ke variabel referensi final  
 }  
}

**Hal-hal Penting Mengenai Keyword Final Pada Java**

Berikut ini adalah beberapa poin penting untuk diingat dari keyword final pada pemrograman Java.

* [Suatu constructor](http://bahasajava.com/9-aturan-contoh-penggunaan-constructor-pada-program-java/) tidak bisa dideklarasikan sebagai final
* Variabel final lokal harus diinisialisasi ketika dideklarasikan
* Dalam [interface](http://bahasajava.com/belajar-java-memahami-interface-dengan-mudah/), semua variabel yang dideklarasikan adalah final secara default
* Nilai dari suatu variabel final tidak dapat diubah
* Suatu method final tidak bisa dioverride
* Class final tidak bisa diwariskan
* Ketika suatu parameter dari method dideklarasikan final maka nilai dari parameter tersebut tidak bisa diubah
* Gunakan huruf kapital untuk penamaan constants pada Java
* Keyword final, finally, dan finalize pada Java adalah berbeda. Dimana [finally digunakan pada exception handling](http://bahasajava.com/contoh-penggunaan-blok-finally-pada-exception-handling-program-java/) sedangkan finalize digunakan pada garbage collection.

**Fungsi Garbage Collection pada Java**

Pada Java, *Garbage Collection* adalah mekanisme Java untuk menghapus suatu objek dari memori tanpa perlu dideklarasikan secara eksplisit dalam program. Berfungsi untuk meningkatkan *Management Memory* yaitu kita dapat menghemat penggunaan memori, java akan menghapus sebuah objek yang tidak diperlukan atau tidak di referensikan lagi, ruang memori pada objek tersebut akan dimanfaatkan untuk keperluan lain sehingga tidak ada pemborosan memori. *Garbage Collection* merupakan salah satu mekanisme dari fitur JVM (Java Virtual Machine). Ini adalah salah satu kelebihan dari bahasa pemrograman Java.

**Garbage Collection pada Java**

Sebagai seorang programmer yang handal, aplikasi atau program yang kita buat harus memiliki penggunaan memori yang kecil atau seminimal mungkin, karena dengan begitu, program yang kita buatpun menjadi lebih handal juga.  
  
Salah satu kelebihan dari pemrograman Java adalah kita dapat menigkatkan efisiensi pada memori, dengan membuang objek yang tidak terpakai atau tidak direferensikan, ini akan secara otomatis dikontrol oleh JVM (Java Virtual Machine).  
  
Untuk melihat bagaimana cara kerja Garbage Collection pada java, kita dapat menambahkan codingan, seperti method *finalize()* dan juga *System.gc()*. Contoh Sederhananya seperti berikut ini:

package WILDAN\_TECHNO\_ART;

public class Balok {

//Method ini akan ditampilkan jika Garbage Collection bekerja

public void finalize(){

System.out.println("Objek Yang Tidak Terpakai Sudah Dibersihkan:");

}

public static void main(String[] args){

Runtime RT = Runtime.getRuntime();

System.out.println("Jumlah Memori Awal: "+RT.totalMemory());

Block objek1 = new Block(); //Block Satu

Block objek2 = new Block(); //Block Dua

Block objek3 = objek1; //Objek3 Mengembalikan Nilai Objek1

objek1 = null;

objek2 = null;

System.out.println("Jumlah Memori Yeng Tersedia Sebelum di GC: "+RT.freeMemory());

System.gc();

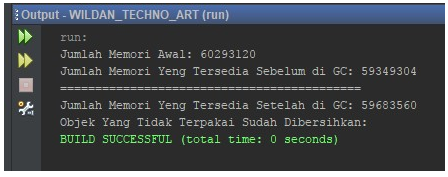
System.out.println("===========================================");

System.out.println("Jumlah Memori Yeng Tersedia Setelah di GC: "+RT.freeMemory());

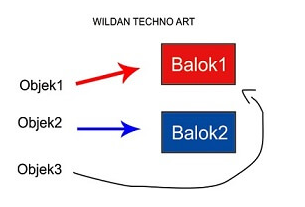
}

}

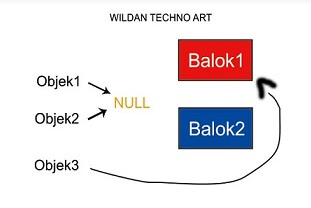
**Demo:**



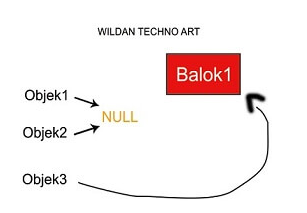
**Penjelasan:**  
  
Pada program tersebut, disana kita membuat 3 buah objek atau variable dari Class Balok, objek1 dan objek2 diinisialisasikan kedalam Balok1 dan Balok2, dan objek3 mempunyai nilai kembali dari objek Balok1.



Pada baris selanjutnya, objek1 dan objek2 kita ubah nilainnya menjadi *null* yang berarti kosong, sedangkan objek3 masih mengembalikan nilainya objek1.



Karena pada Balok2 tidak ada yang direferensikan nilainya, maka secara otomatis, Balok2 tersebut akan di bawa oleh System Garbage Collection pada JVM.

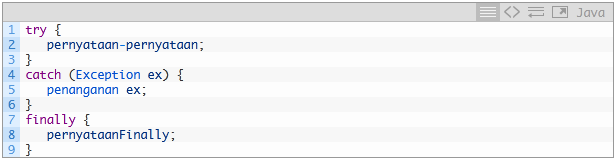


*System.gc()*, berfungsi untuk menjalankan Garbage Collection pada java. lalu Method *finalize()*, digunakan untuk memastikan bahwa objek telah bersih dan Garbage Collection telah bekerja, Di dalam metode *finalize()*, kita dapat menentukan tindakan yang harus dilakukan sebelum suatu objek hancur.  
  
Dan terakhir, kita menambahkan sebuah variable *Runtime*, yang digunakan untuk mengecek Total Memori serta Jumlah Memori yang tersedia sebelum dan sesudah di Garbage Collection.

**Penggunaan Blok Finally Pada Exception Handling Program Java**

Jadi ketika anda menginginkan blok kode yang akan selalu dijalankan oleh program Java terlepas dari ada atau tidak adanya exception, maka blok finally adalah tempat yang tepat bagi anda untuk melakukan tujuan tersebut.

Secara sederhana, syntax dari finally adalah sebagai berikut:



**5 Hal Penting Yang Harus Diketahui Dari Blok finally**

Terdapat lima poin penting yang harus anda pahami dari blok finally ini, diantaranya adalah:

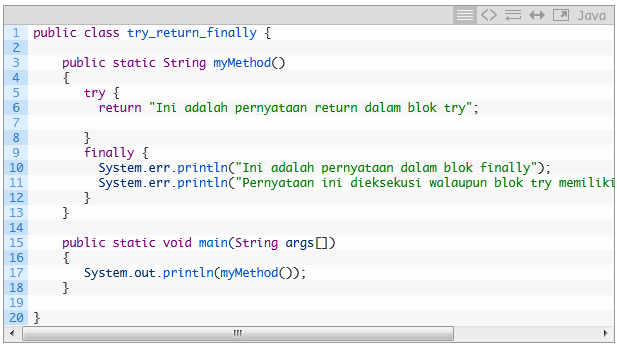
* 1. Ketika anda menggunakan blok finally maka anda juga harus menggunakan blok try. Ini adalah pasangan yang tidak dapat dipisahkan.
  2. Penggunaan blok finally tidak wajib pada exception handling, tapi ini merupakan pilihan bila blok try-catch tidak cukup untuk menyelesaikan masalah. Pernyataan pada blok finally akan selalu dijalankan setelah eksekusi dari blok try.
  3. Blok catch akan dieksekusi lebih dahulu bila terjadi exception pada blok try. Namun, bila tidak ada exception maka blok catch akan diabaikan dan pernyataan pada blok finally akan dieksekusi.
  4. Exception pada blok finally dapat ditindaklanjuti sama persis dengan exception lainnya.
  5. Bila blok try memiliki pernyataan seperti break, continue atau return maka pernyataan dalam blok finally akan tetap dieksekusi.

Jadi, ketika pernyataan-pernyataan dalam blok try melemparkan exception dan kemudian ditangkap oleh blok catch, maka kode dalam blok finally akan dieksekusi. Tidak hanya itu, meskipun tidak ada exception yang terlempar atau tertangkap, kode di blok finally akan tetap dieksekusi.

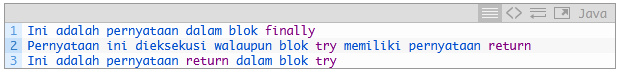
**Blok finally Dengan Blok try Yang Memiliki Pernyataan return**

Finally juga akan mengeksekusi pernyataan di dalam bloknya bahkan ketika terdapat pernyataan return sebelum mencapai blok finally tersebut.

Contoh:



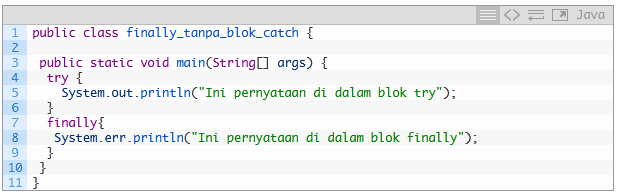
Output:



**Blok try dan finally Tanpa Menggunakan Blok catch**

Berikut ini adalah contoh ketika class hanya menggunakan blok try dan finally saja.

Contoh:



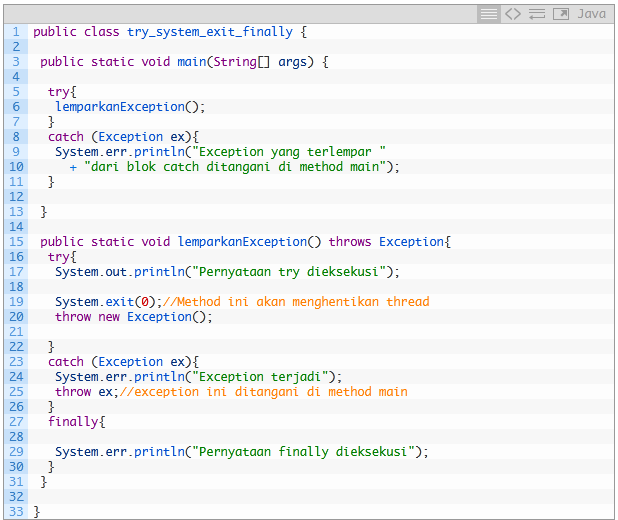
Output:



**Contoh Blok try, catch, finally dan Method System.exit()**

Method System.exit() akan menghentikan thread program, sehingga pernyataan setelahnya dan juga pernyataan dalam blok finally tidak akan dieksekusi.

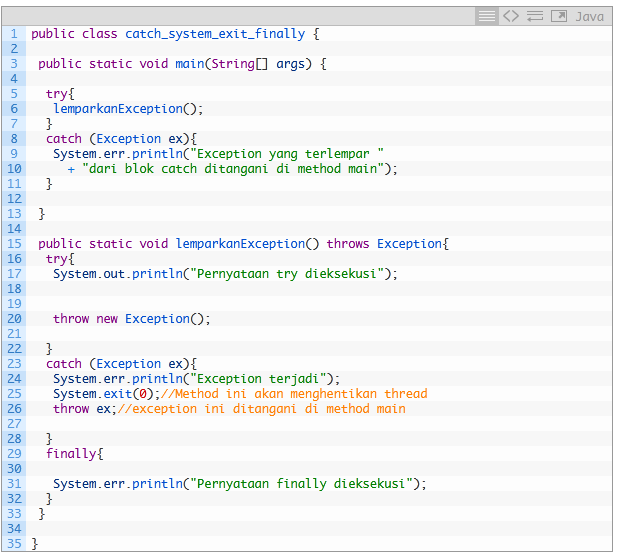
a) Contoh method System.exit() terdapat pada blok try



Output:



b) Contoh method System.exit() terdapat pada blok catch



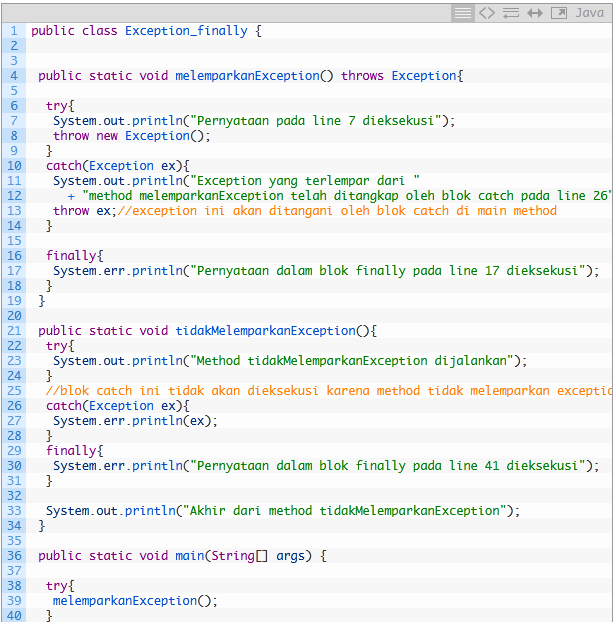
Output:

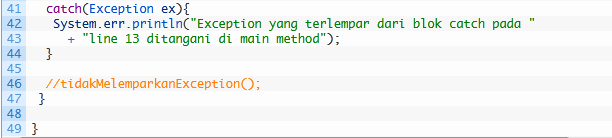


**Contoh Program try, catch, finally, Pada Method Yang Melemparkan Dan Tidak Melemparkan Exception**

Berikut ini adalah contoh ketika method yang melemparkan dan tidak melemparkan exception menggunakan blok try, catch, dan finally.

Contoh:





Contoh output:

Pernyataan pada line 7 dieksekusi

Exception yang terlempar dari method melemparkanException telah ditangkap oleh blok catch pada line 26

Pernyataan dalam blok finally pada line 17 dieksekusi

Exception yang terlempar dari blok catch pada line 13 ditangani di main method

Method tidakMelemparkanException dijalankan

Pernyataan dalam blok finally pada line 30 dieksekusi

Akhir dari method tidakMelemparkanException

Referensi :

<http://bahasajava.com/keyword-final-dalam-program-java/>

<https://www.wildantechnoart.net/2017/11/fungsi-garbage-collection-pada-java.html>

<http://bahasajava.com/contoh-penggunaan-blok-finally-pada-exception-handling-program-java/>