Perbedaan dan persamaan antara Interface Class dan Abstrak Class

## Abstract Class

Abstract Class merupakan Class yang khusus dibuat untuk keperluan inheritance. Tujuan dari pembuatan abstract class ini ialah untuk membuat definisi umum bagi class-class yang akan menjadi turunan darinya. Abstract Class tidak bisa diinstansiasi. Abstract method tidak mempunyai implementasi, sehingga penulisannya dilakukan hanya dengan mengikutkan semikolon, bukan blok method {} seperti biasanya. Abstract method ini bisa digunakan oleh Class turunannya dengan melakukan override. Abstract class hanya bisa digunakan sebagai super class, tapi juga bisa diturunkan dari class abstract lainnya. Untuk mendeklarasikan sebuah abstract class digunakan keyword abstract, [abstract] class [class\_name] Sebuah abstract class pada dasarnya tidak hauh beda dengan class lainnya, yakni juga berisi method yang menggambarkan carakteristik dari kelas abstract tersebut, bedanya yakni sebuah abstract class bisa berisi method tanpa diimplementasikan artinya sebuah method tanpa body, methid seperti ini disebut method abstract.

Kegunaan Class Abstract

• Class Abstract berisi beberapa method dan beberapa method  abstract. Class Abstract berisi sebagian implementasi, dan subclass yang melengkapi implementasinya. Dengan kata lain Class Abstract memiliki beberapa kesamaan (Bagian yang  diimplementasikan oleh subclass) dan memiliki perbedaan (method yang dimiliki sendiri olehclass abstract)

• Deklarasikan method abstract, jika ada satu atau lebih subclass yang diharapkan mempunyai fungsionalitas yang sama tapi implementasi berbeda.

• Gunakan class abstract untuk mendefinisikan behavior secara umum sebagai superclass, sedangkan subclass menyediakan implementasi detail.

• Jika class abstract semua method merupakan method abstract,  sebaiknya class abstract tersebut diganti menjadi Interface

## Interface

Sedangkan *interface* secara filosofis lebih berfungsi sebagai antarmuka yang membentuk komunikasi dengan code lain. Misalnya membentuk hubungan antara sebuah object dengan object yang lain atau hubungan antara object sebagai penyedia dengan code pengguna. Karena itulah interface ini banyak digunakan dalam dunia design pattern. Sebab kebanyakan dari design pattern ini menekankan hubungan antar object.

Interface adalah jenis khusus dari blok yang hanya berisi method signature (atau constant).  Interface mendefinisikan sebuah(signature) dari sebuah kumpulan method tanpa tubuh. Interface mendefinisikan sebuah cara standar dan umum dalam menetapkan sifat-sifat dari class-class. Mereka menyediakan class-class, tanpa memperhatikan lokasinya dalam hirarki class, untuk mengimplementasikan sifat-sifat yang umum. Dengan catatan bahwa interface-interface juga menunjukkan polimorfisme, dikarenakan program dapat memanggil method interface dan versi yang tepat dari method yang akan dieksekusi tergantung dari tipe object yang melewati pemanggil method interface

|  |  |
| --- | --- |
| **Abstract Class** | **Interface** |
| Bisa berisi abstract dan non-abstract method. | Hanya boleh berisi abstract method. |
| Kita harus menuliskan sendiri modifiernya. | Kita tidak perlu susah2 menulis public abstract di depan nama method. Karena secara implisit, modifier untuk method di interface adalah public dan*abstract*. |
| Bisa mendeklarasikan constant dan instance variable. | Hanya bisa mendeklarasikan constant.  Secara implisit variable yang dideklarasikan di interface bersifatpublic, static dan final. |
| Method boleh bersifat static. | Method tidak boleh bersifat static. |
| Method boleh bersifat final. | Method tidak boleh bersifat final. |
| Suatu abstact class hanya bisa meng-*extend* satu abstract class lainnya. | Suatu interface bisa meng-*extend*  satu atau lebih interface lainnya. |
| Suatu abstract class hanya bisa meng-*extend* satu abstract class dan meng-implement beberapa interface. | Suatu interface hanya bisa meng-*extend* interface lainnya.  Dan tidak bisa meng-*implement* class atau interface lainnya. |