# JOBSHEET W08 ABSTRACT CLASS

1. **KOMPETENSI**
   1. Memahami konsep dasar dan tujuan abstract class
   2. Mampu menerapkan abstract class dalam suatu kode program
   3. Mampu membuat subclass yang meng-extend abstract class dengan mengimplementasikan seluruh abstract method-nya

# PENDAHULUAN

Abstract class merupakan class yang **tidak dapat diinstansiasi namun dapat di-extend**. Umumnya abstract class digunakan sebagai **generalisasi** atau **guideline** dari subclass dan hanya bisa digunakan lebih lanjut setelah **di-extend** oleh **concrete class** (class pada umumnya)

Abstract class memiliki karakteristik sebagai berikut:

* 1. Selalu dideklarasikan dengan menggunakan keyword “**abstract class**”
  2. Dapat memiliki atribut dan methods (yang bukan abstract method) seperti concrete class
  3. Umumnya memiliki **abstract method**, yaitu method yang hanya dideklarasikan tetapi tidak memiliki implementasi (body)
     + Abstract method mendifinisikan apa saja yang bisa dilakukan oleh sebuah class namun tidak ada detail bagaimana cara melakukannya
     + Untuk membuat abstract method, hanya menuliskan deklarasi method tanpa body dan menggunakan keyword abstract

Untuk mendeklarasikan abstract class:

# public abstract class <NamaClass>

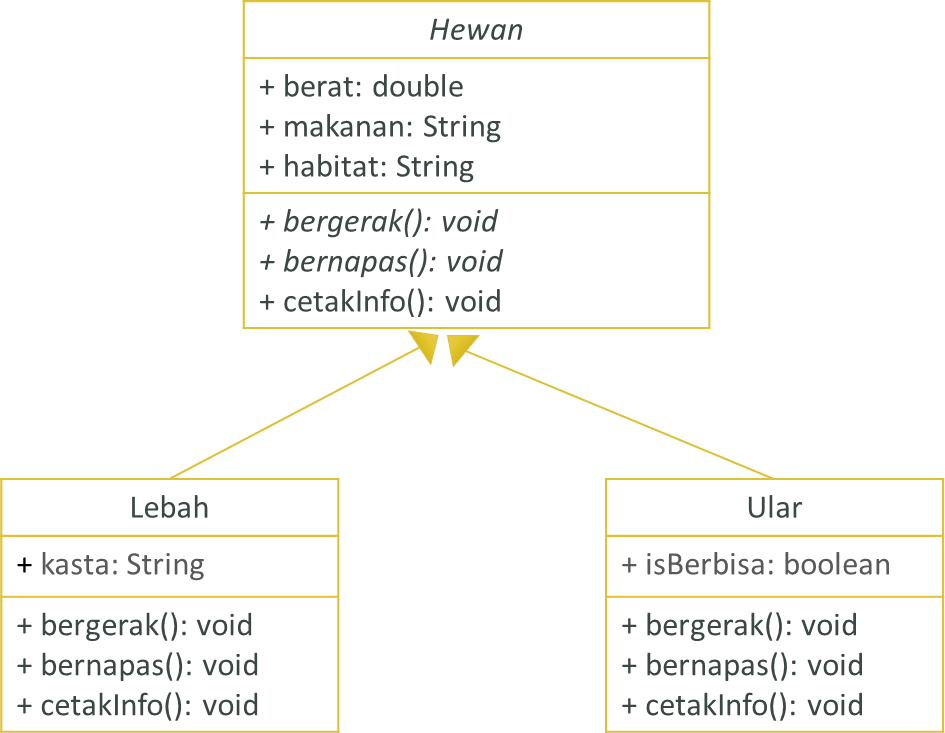
Untuk mendeklarasikan method abstract:

# public abstract <return\_type> <namaMethod>();

Contoh:



Secara umum, notasi class diagram untuk abstract class sama dengan concrete class, namun nama kelas dicetak miring atau ditambah anotasi <<abstract>> di atas nama kelas. Di samping itu abstract method harus dicetak miring juga seperti pada contoh berikut.



Cara menggunaan abstract class:

* + - Abstract class tidak dapat diinstansiasi (tidak dapat dibuat objectnya). Baris kode berikut akan memunculkan *compilation error* “Hewan is abstract; cannot be instantiated”

Hewan hewan1 = new Hewan();

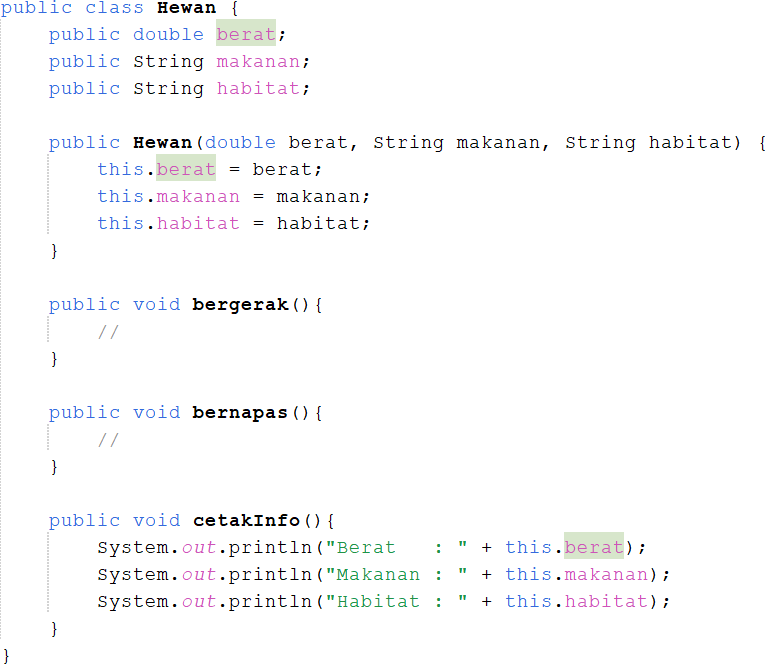
* + - Untuk menggunakan abstract class, dibuat concrete class yang meng-extend abstract class tersebut
      * concrete class menggunakan extends keyword
      * concrete class harus mengimplementasi semua abstract method
    - Class yang menge-extend abstract class tetapi tidak mengimplementasi seluruh abstract method nya maka harus dideklarasikan sebagai abstract class juga

Fungsi abstract class:

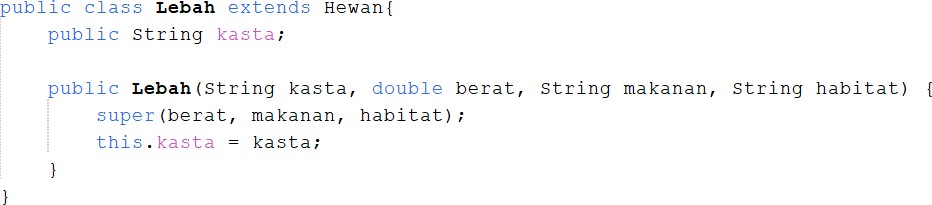
* + - Mencegah suatu class diinstansiasi atau dibuat objeknya
    - Sebagai generalisasi/superclass pada class hierarki
    - Sebagai guideline untuk subclass dengan cara memaksa subclass untuk mengimplementasikan abstract method

# PERCOBAAN

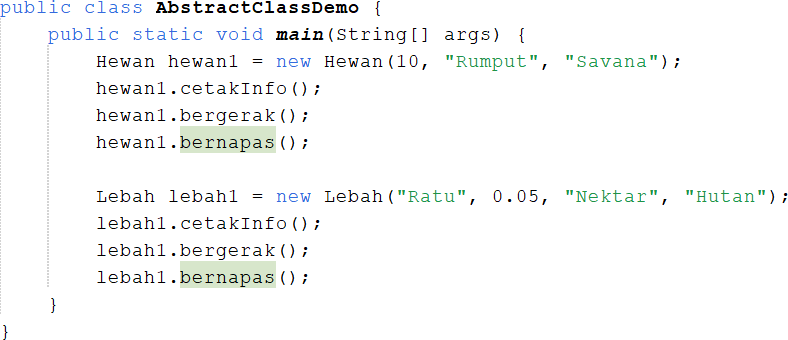
1. **PERCOBAAN 1**
   1. Buatlah project baru dengan nama Praktikum08 kemudian buat class baru dengan nama Hewan. Method bernapas dan bergerak tidak memiliki statement atau baris kode.



* 1. Buat class Lebah sebagai subclass dari class Hewan sebagai berikut



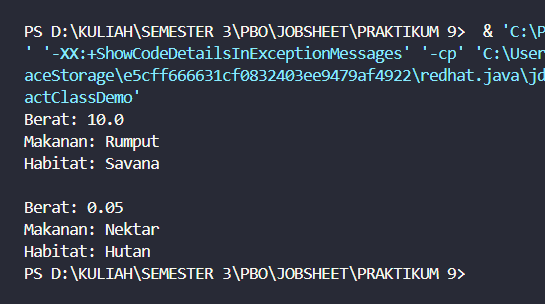
* 1. Buat class main dengan nama AbstractClassDemo lalu instansiasi objek dari class Hewan dan class Lebah. Run program kemudian amati hasilnya.



# PERTANYAAN

* 1. Bagaimana hasil pada langkah 3? Apakah objek hewan1 dan lebah1 berhasil diinstansiasi?

Jawab : berhasil diinstansiasi karena class tersebut bukan class konkret



* 1. Menurut Anda, mengapa tidak ada baris program pada method bergerak() dan bernapas() pada class Hewan()?

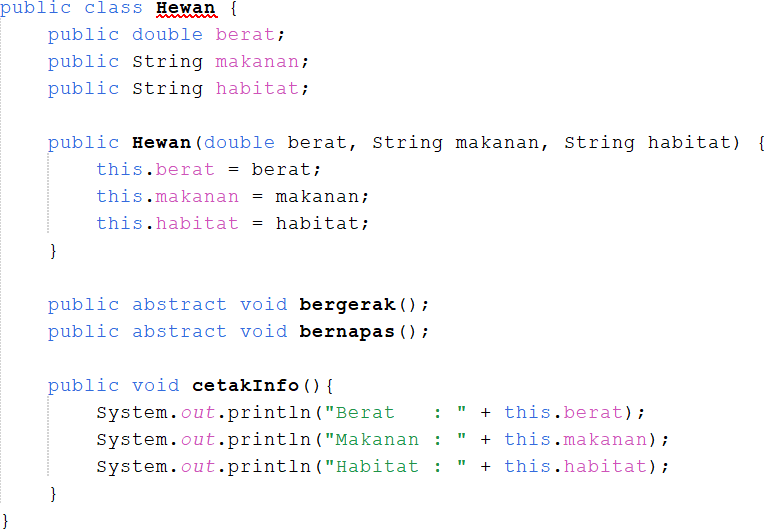
Jawab : karena pada method bergerak() dan bernapas() pada class Hewan tidak memiliki baris program atau implementasinya karena sebagai superclass yang dimana pada subclass nantinya memberikan implementasinya sendiri sesuai dengan cara unik mereka dalam bergerak dan bernapas.

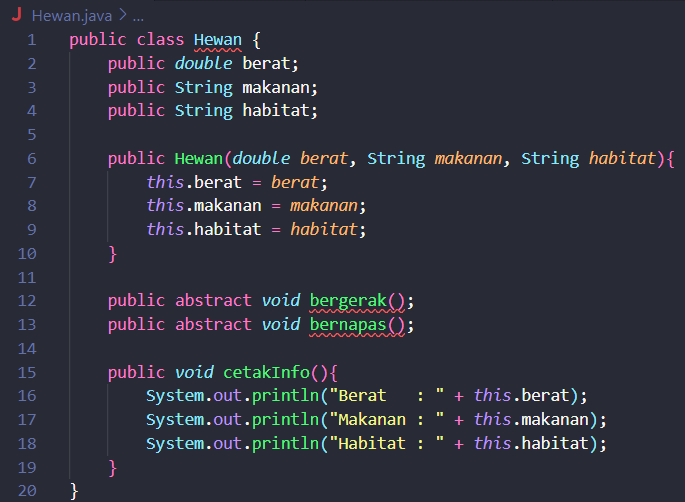
* 1. Class Lebah tidak memiliki method bergerak(), bernapas(), dan cetakInfo(), mengapa tidak terjadi error pada AbstractClassDemo?

Jawab : tidak terjadi eror karena class Lebah extends dari class Hewan. Jadi semua method yang didefinisikan di class Hewan juga akan tersedia di class Lebah.

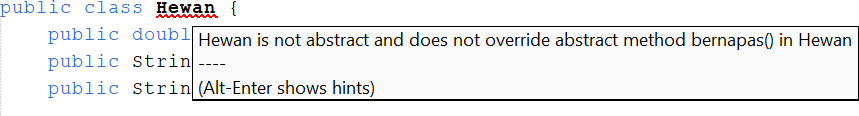
# PERCOBAAN 2

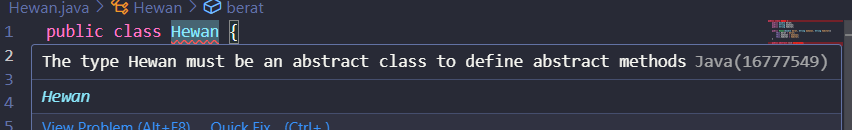
* 1. Ubah method bergerak dan bernapas menjadi abstract method.



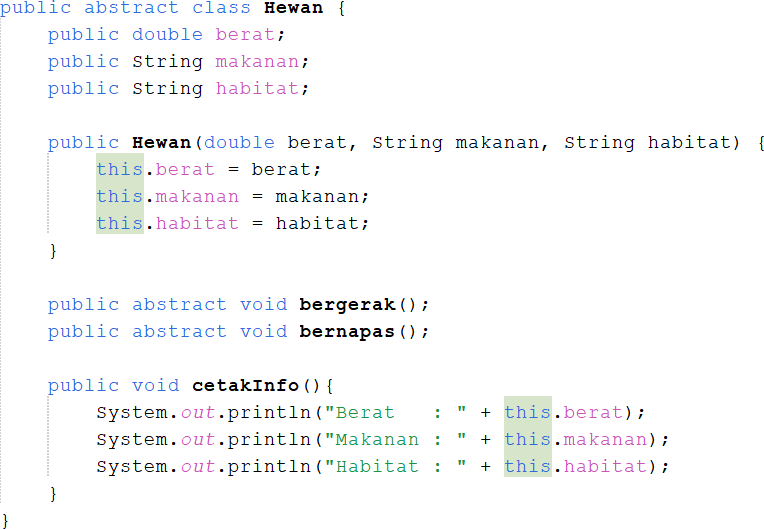


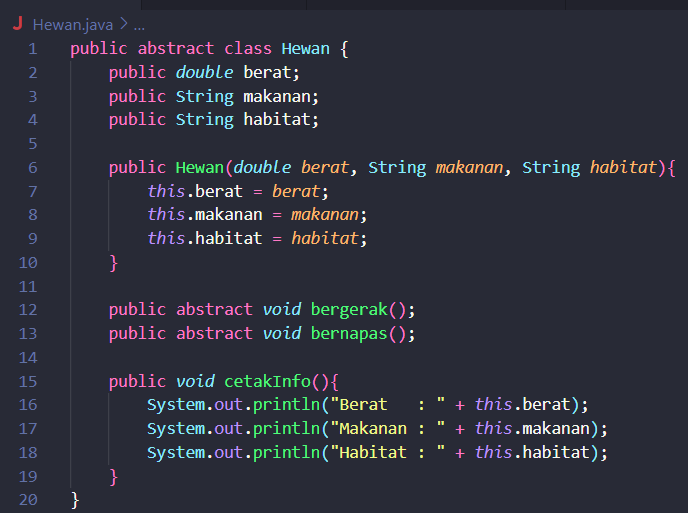
* 1. Akan muncul error sebagai berikut



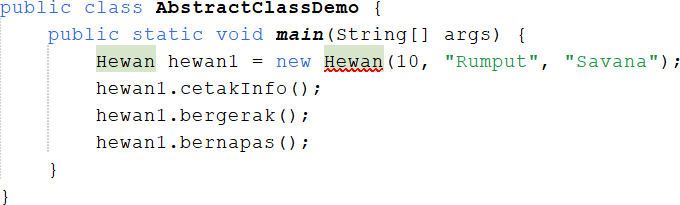


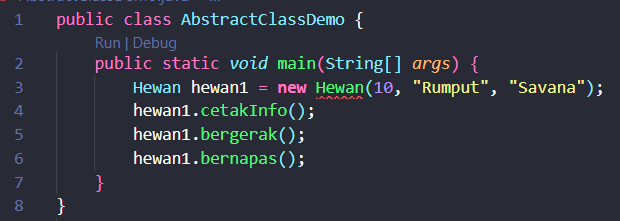
* 1. Ubah class Hewan menjadi abstract Class. Jalankan program kemudian amati hasilnya.

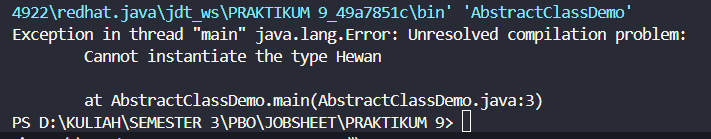




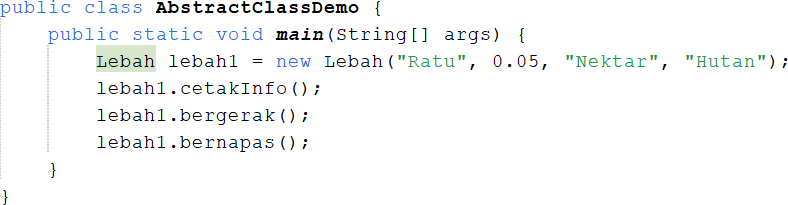
* 1. Ubah class demo sebagai berikut. Run program kemudian amati hasilnya



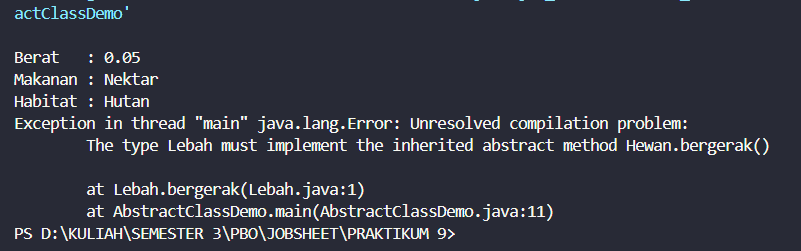




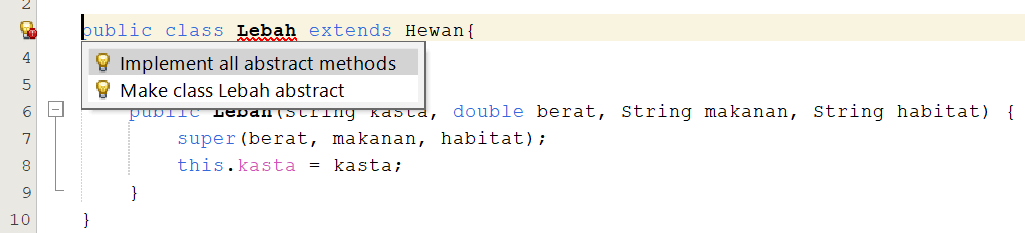
* 1. Ubah class demo sebagai berikut. Run program kemudian amati hasilnya



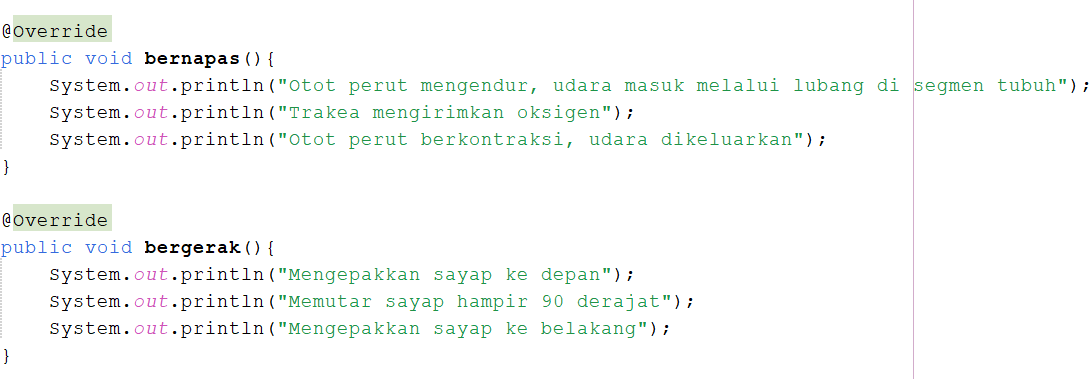


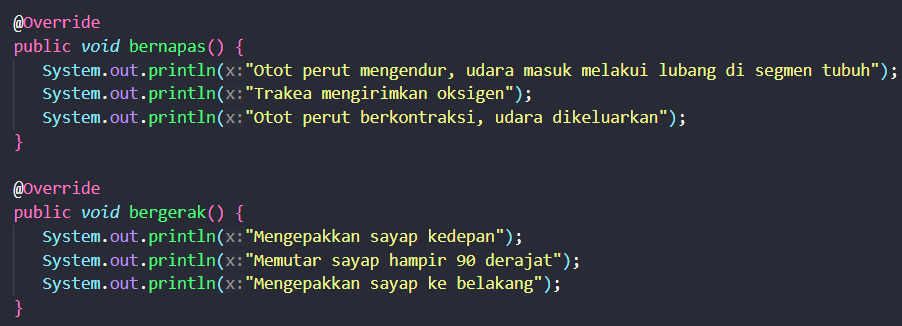


* 1. Klik icon lampu pada class Lebah, kemudian pilih option “Implement all abstract method”

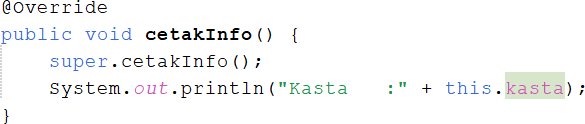


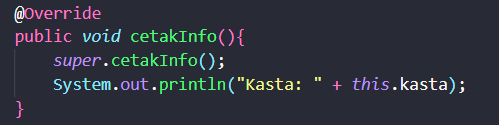
* 1. Implementasi method bergerak dan bernapas pada class Lebah sebagai berikut. Run program kemudian amati hasilnya.





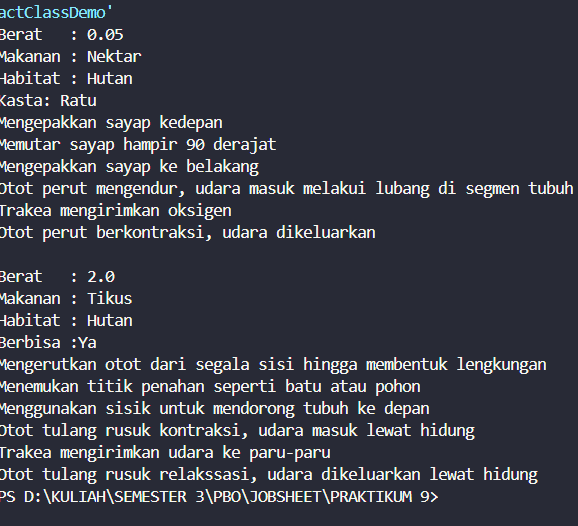
* 1. Tambahkan method cetakInfo() pada class Lebah. Run program kemudian amati hasilnya.





* 1. Buat class Ular kemudian sebagai berikut. Instansiasi objek bertipe Ular pada class AbstractClassDemo. Ekseksusi ketiga method untuk object tersebut.





# PERTANYAAN

* 1. Pada langkah 1, mengapa sebaiknya method bergerak() dan bernapas() dideklarasikan sebagai abstract method?

Jawab : karena bergerak dan bernapas setiap jenis hewan berbeda, dan kedua method ini tergantung pada jenis hewan tertentu. Supaya subclass mendefinisikan perilaku sesuai dengan karateristiknya

* 1. Mengapa pada langkah 2 muncul error?

Jawab : karena pada method bergerak() dan bernapas() dideklarasikan sebagai abstract namun pada class Hewan tidak dideklarasikan sebagai abstract class.

* 1. Apakah sebuah class yang memiliki abstract method harus dideklarasikan sebagai abstract class?

Jawab : iya, jika tidak dideklarasikan sebagai abstract class, compiler akan memberikan eror

* 1. Sebaliknya, apakah abstract class harus memiliki abstract method?

Jawab : tidak,

* 1. Mengapa muncul error pada langkah 4?

Jawab :

* 1. Apakah abstract class dapat memiliki constructor?

Jawab :

* 1. Apakah constructor abstract class dapat dipanggil?

Jawab :

* 1. Pada langkah 6-8, mengapa method bergerak() dan bernapas() **harus** di-override, namun method cetakInfo() **tidak harus** di-override?

Jawab :

* 1. Simpulkan kegunaan dari abstract method

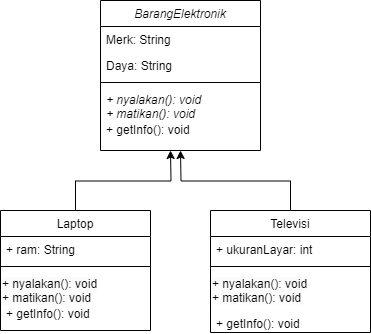
Jawab :

* 1. Simpulkan kegunaan dari abstract class

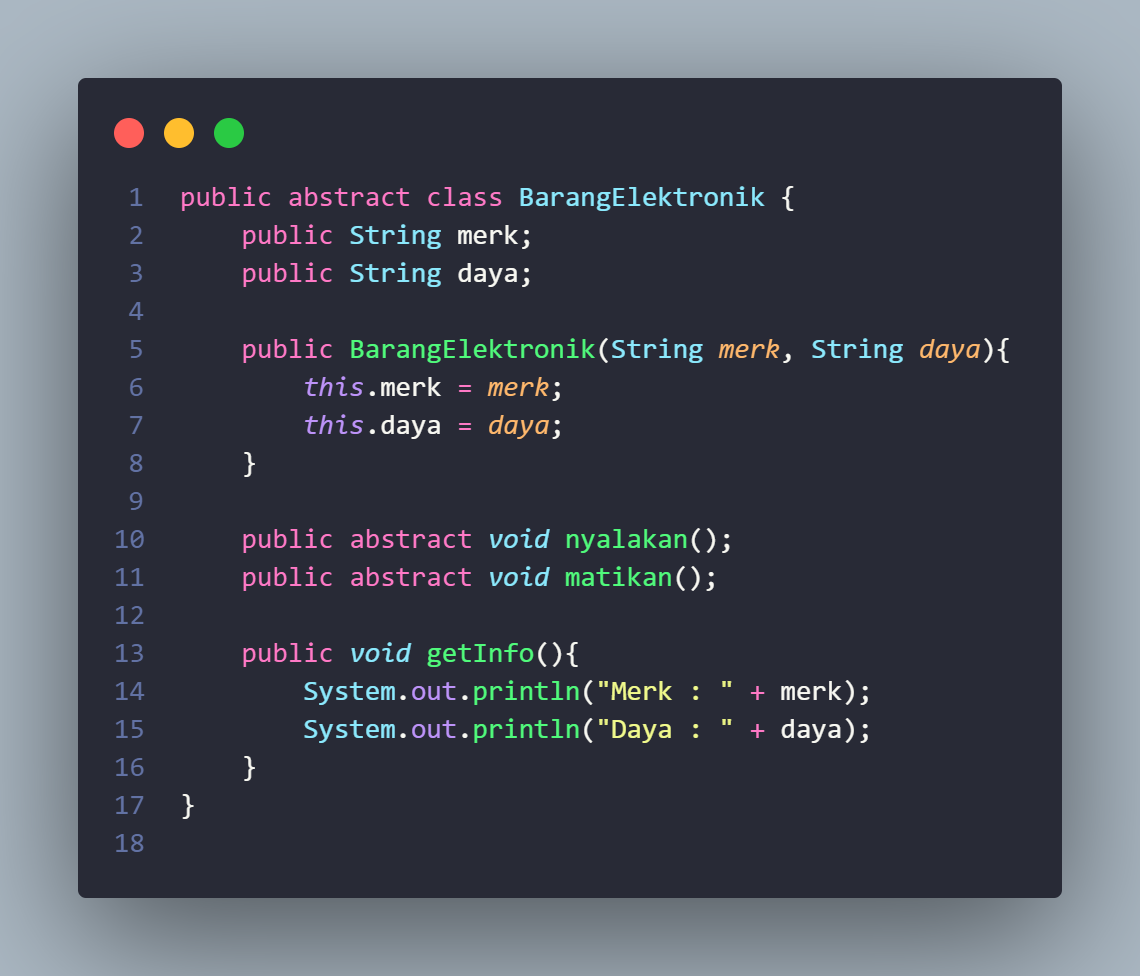
Jawab :

# 4. TUGAS

Implementasikan class diagram yang telah dirancang pada tugas PBO Teori ke dalam kode program. Selanjutnya buatlah instansiasi objek dari masing-masing subclass kemudian coba eksekusi method-method yang dimiliki.



Berikut adalah kode programnya









Berikut ini adalah hasilnya

