**JOBSHEET 8**

**ABSTRACT CLASS**

Nama : Rizqi Rohmatul Huda

Kelas : 2G – TI

No.Absen : 26

NIM : 2141720264

# 3. PERCOBAAN

# A. PERCOBAAN 1

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

**package** pemrogramanberbasisobjek.pertemuan10.abstractclass;

*public* class Hewan {

*public* *double* berat;

*public* String makanan;

*public* String habitat;

*public* Hewan(*double* **berat**, String **makanan**, String **habitat**){

*this*.*berat* **=** berat;

*this*.*makanan* **=** makanan;

*this*.*habitat* **=** habitat;

    }

*public* *void* bergerak(){

*//*

    }

*public* *void* bernapas(){

*//*

    }

*public* *void* cetakInfo(){

        System.*out*.println("Berat\t: " **+** *this*.*berat*);

        System.*out*.println("Makanan\t: " **+** *this*.*makanan*);

        System.*out*.println("Habitat\t: " **+** *this*.*habitat*);

    }

}

1. Selanjutnya buat class Lebah sebagai subclass dari class Hewan sebagai berikut

**package** pemrogramanberbasisobjek.pertemuan10.abstractclass;

*public* class Lebah *extends* *Hewan* {

*public* String kasta;

*public* Lebah(String **kasta**, *double* **berat**, String **makanan**, String **habitat**){

*super*(berat, makanan, habitat);

*this*.*kasta* **=** kasta;

    }

}

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

**package** pemrogramanberbasisobjek.pertemuan10.abstractclass;

*public* class AbstractClassDemo {

*public* *static* *void* main(String[] **args**) {

        Hewan hewan1 **=** **new** Hewan(10, "Rumput", "Savana");

        hewan1.cetakInfo();

        hewan1.bergerak();

        hewan1.bernapas();

        Lebah lebah1 **=** **new** Lebah("Ratu", 0.05, "Nektar", "Hutan");

        lebah1.cetakInfo();

        lebah1.bergerak();

        lebah1.bernapas();

    }

}

# B. PERTANYAAN

# 

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

Jawab : Iya, masih dapat diinstansiasi karena belum dideklrasikan sebagai abstract class.

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

Jawab : karena atribut dan method diwariskan dari parent class (class Hewan) kepada subclass (class Lebah) .

# C. PERCOBAAN 2

1. Ubah method bergerak dan bernapas menjadi abstract method.

**package** pemrogramanberbasisobjek.pertemuan10.abstractclass;

*public* class Hewan {

*public* *double* berat;

*public* String makanan;

*public* String habitat;

*public* Hewan(*double* **berat**, String **makanan**, String **habitat**){

*this*.*berat* **=** berat;

*this*.*makanan* **=** makanan;

*this*.*habitat* **=** habitat;

    }

*public* *abstract* *void* bergerak();

*public* *abstract* *void* bernapas();

*public* *void* cetakInfo(){

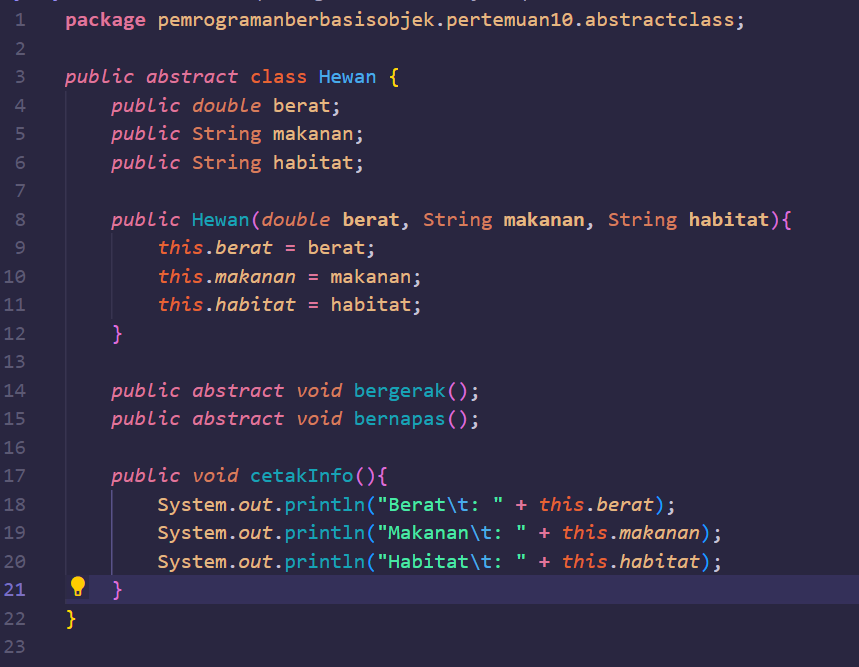
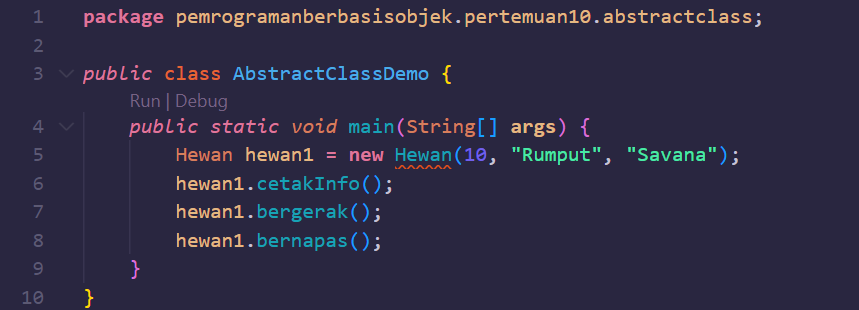
        System.*out*.println("Berat\t: " **+** *this*.*berat*);

        System.*out*.println("Makanan\t: " **+** *this*.*makanan*);

        System.*out*.println("Habitat\t: " **+** *this*.*habitat*);

    }

}



2.

Akan muncul error sebagai berikut

3.

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

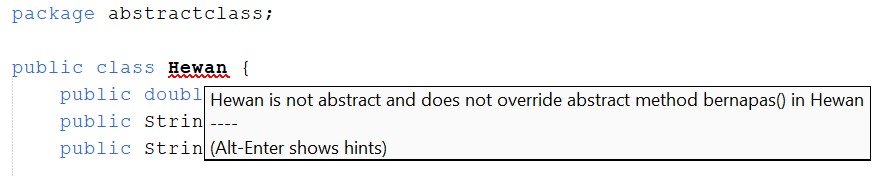
4.

Ubah class demo sebagai berikut.

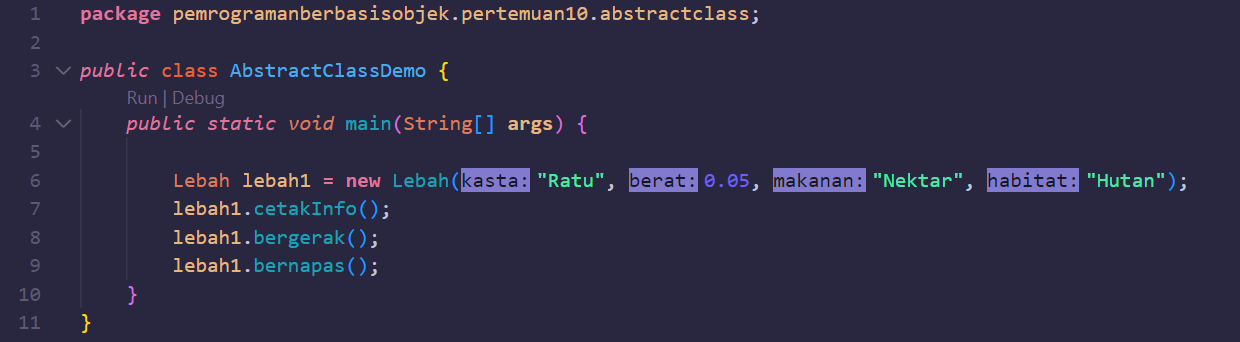
Run

program kemudian amati

hasilnya

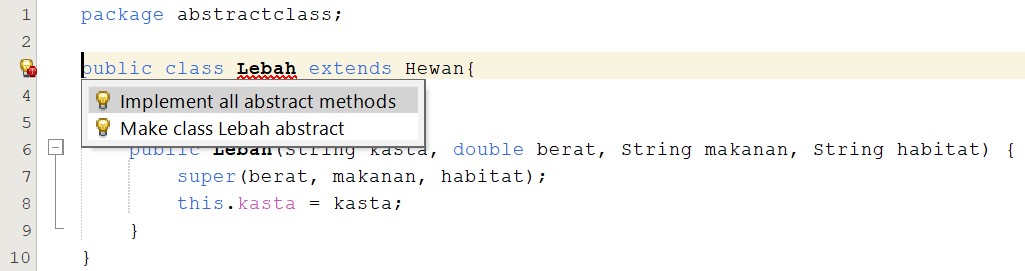


5. Ubah class demo sebagai berikut. Run program kemudian amati hasilnya

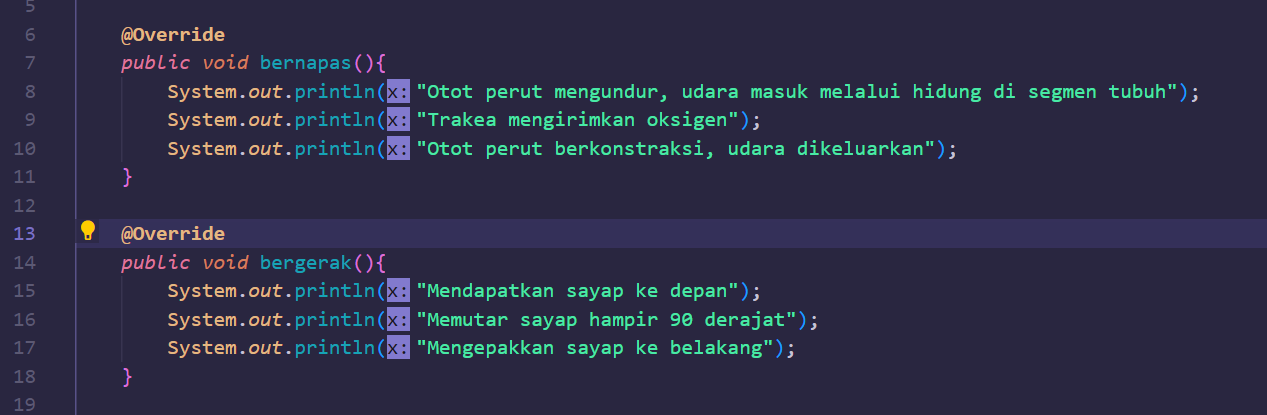


6.

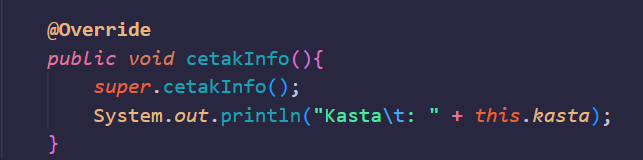
Klik icon lampu pada class Lebah, kemudian pilih option “Implement all abstract method”



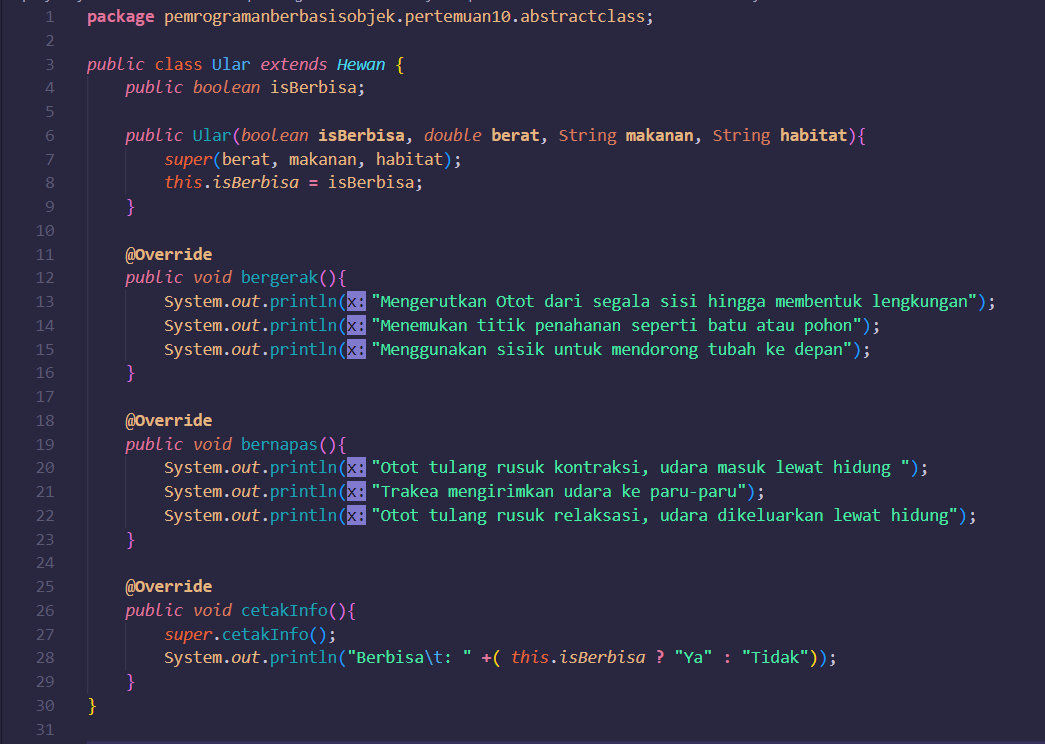
1. Implementasi method bergerak dan bernapas pada class Lebah sebagai berikut. Run program kemudian amati hasilnya.



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.



# D. PERTANYAAN

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

Jawab : Karena belum diketahui secara pasti dan memungkin ada banyak perbedaan mengenai apa yang bisa dilakukan oleh sebuah class (tidak ada detail bagaimana cara melakukannya.

1. Mengapa pada langkah 2 muncul error?

Jawab : Karena kelas belum dideklarasikan sebagai abstract class, maka harus diberi keyword **abstract** untuk mendefisikan abstract method.

1. Bagaimana hasil pada langkah 4? Mengapa demikian?

Jawab : Hasil akan error(“Hewan is abstract; cannot be instantiated”) karena abstract class merupakan class yang tidak dapat diinstansiasi (tidak dapat dibuat objectnya), umumnya abstract class digunakan sebagai generalisasi atau guideline dari subclass karena tidak ada detail bagaimana cara melakukannya.

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

Jawab : Karena method bergerak() dan bernapas() merupakan implementasi dari method abstract maka harus mengimplementasi semua abstract method dari parent class, jadi harus di-override, sedangkan untuk method cetakInfo() adalah method yang dapat didefinisikan pada superclass jadi tidak perlu diset sebagai abstract maka tidak harus mengimplementasi dari method abstract sehingga tidak harus di-override.

1. Dari percobaan tersebut, simpulkan kegunaan dari abstract class dan abstract method.

Jawab :

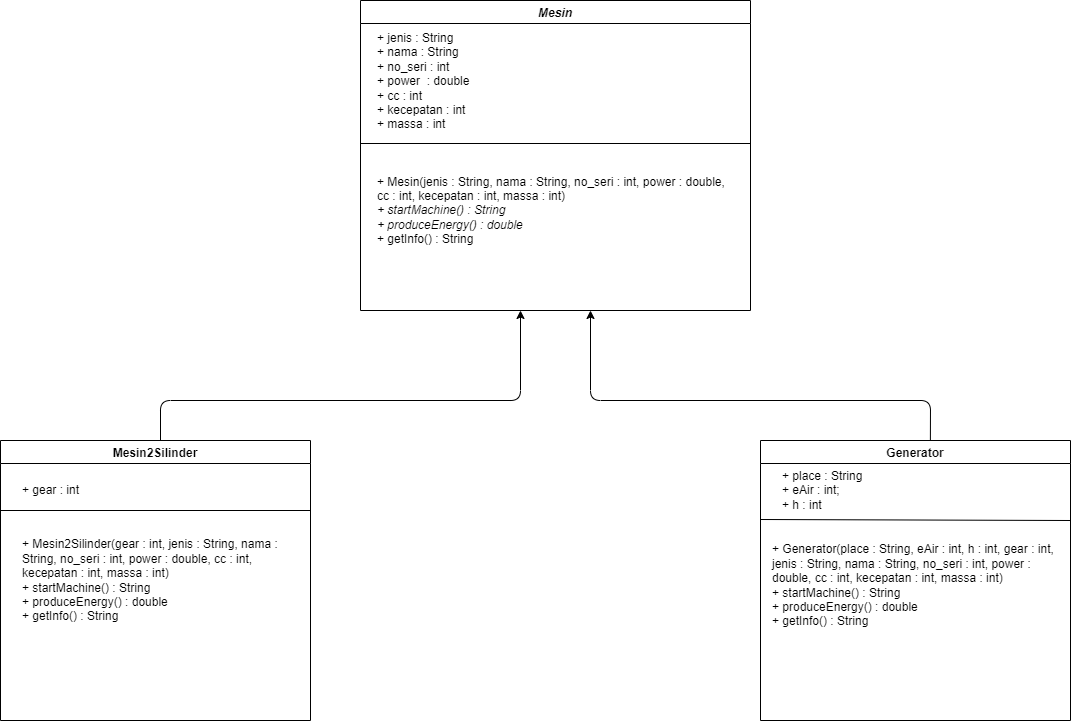
* **kegunaan dari abstract class :**
* Mencegah suatu class diinstansiasi atau dibuat objeknya
* Sebagai generalisasi/guideline pada class hierarki Method yang tidak dapat didefinisikan pada superclass
* Memaksa subclass untuk mengimplementasi seluruh abstract method tersebut.
* **kegunaan dari abstract method :**

Memastikan class turunannya akan membuat kerangka yang sama dengan class parentnya.

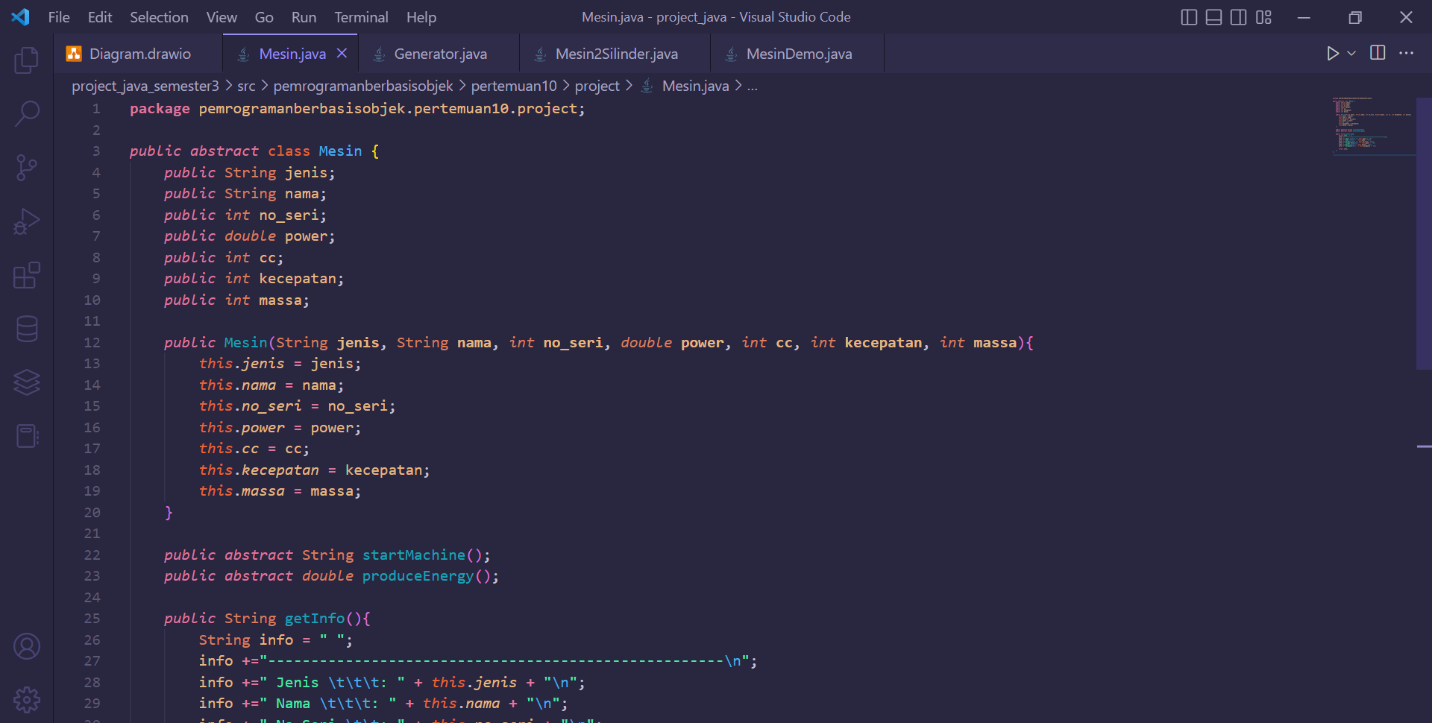
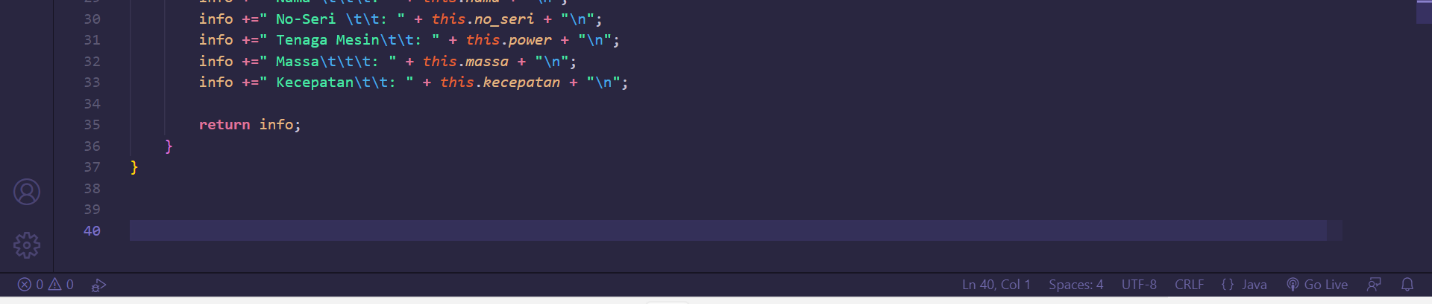
# 4. TUGAS

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

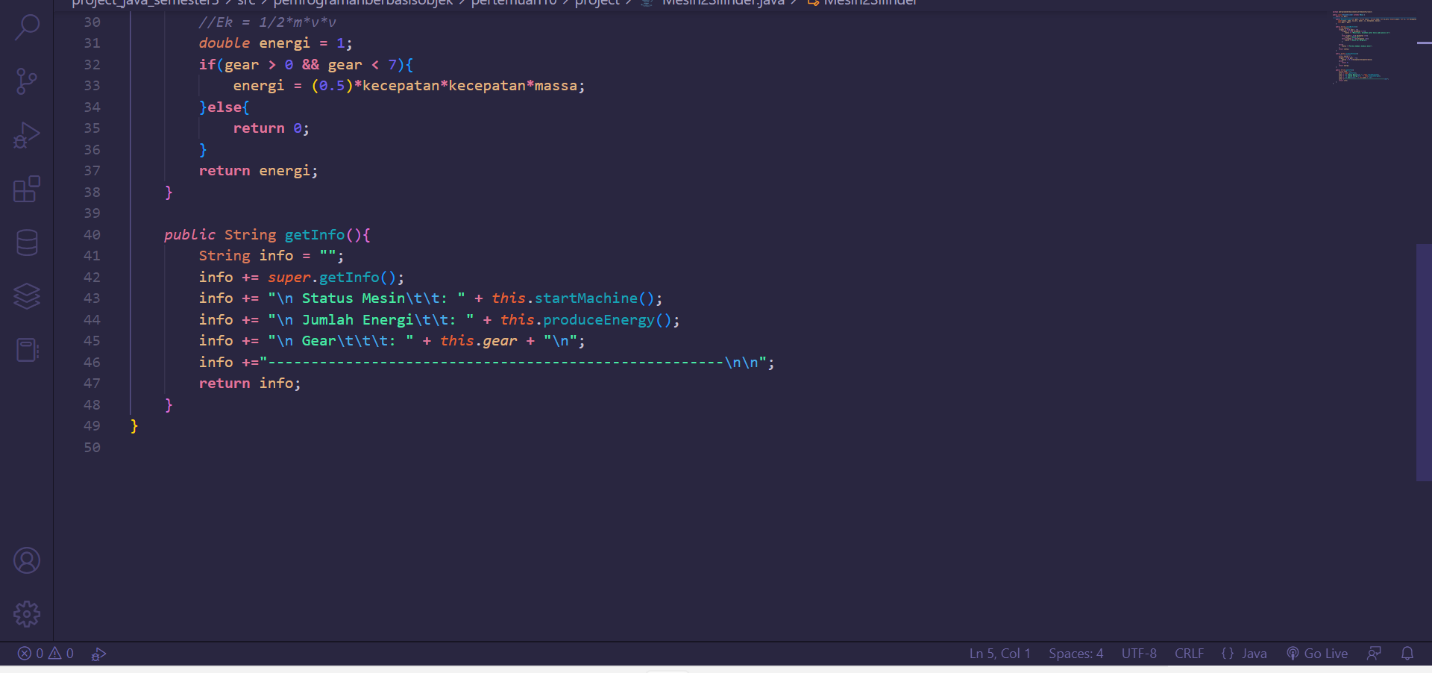
**Revisi Diagram class**

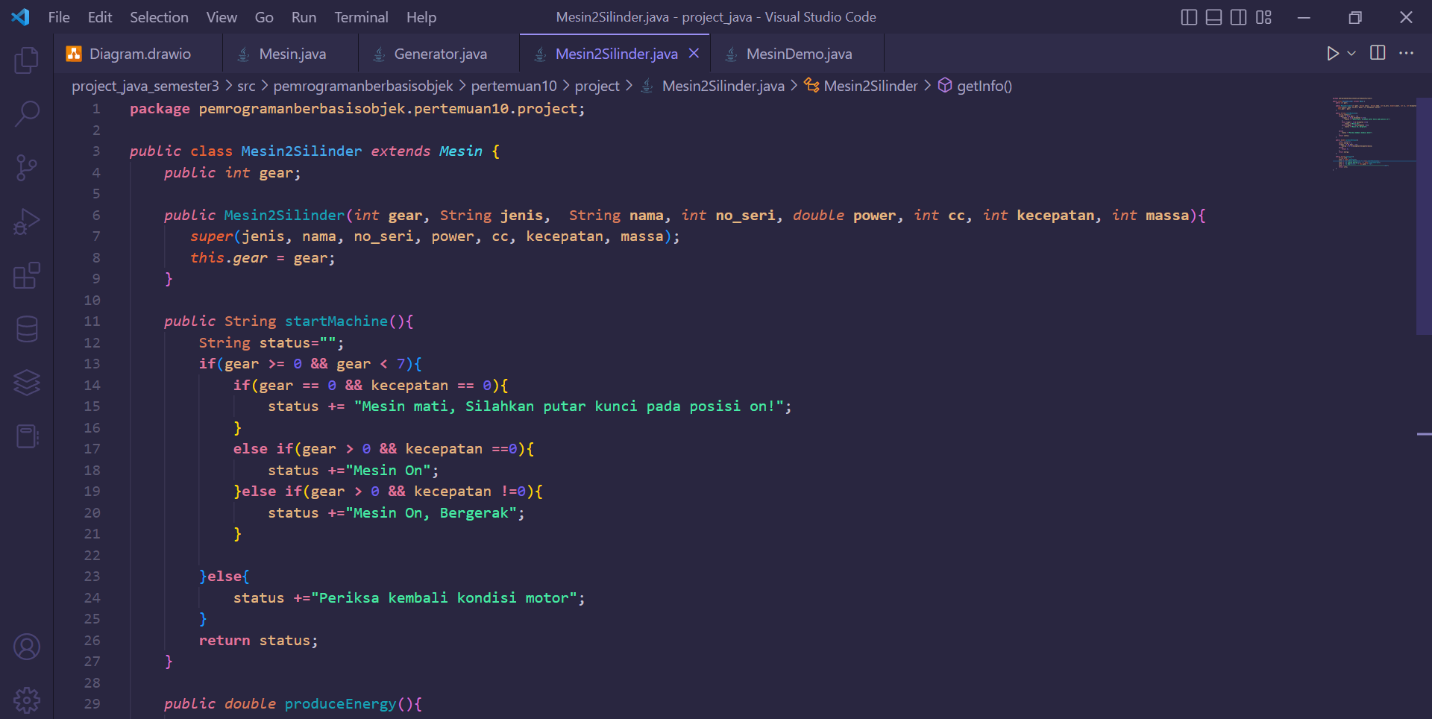


**Code class Mesin**

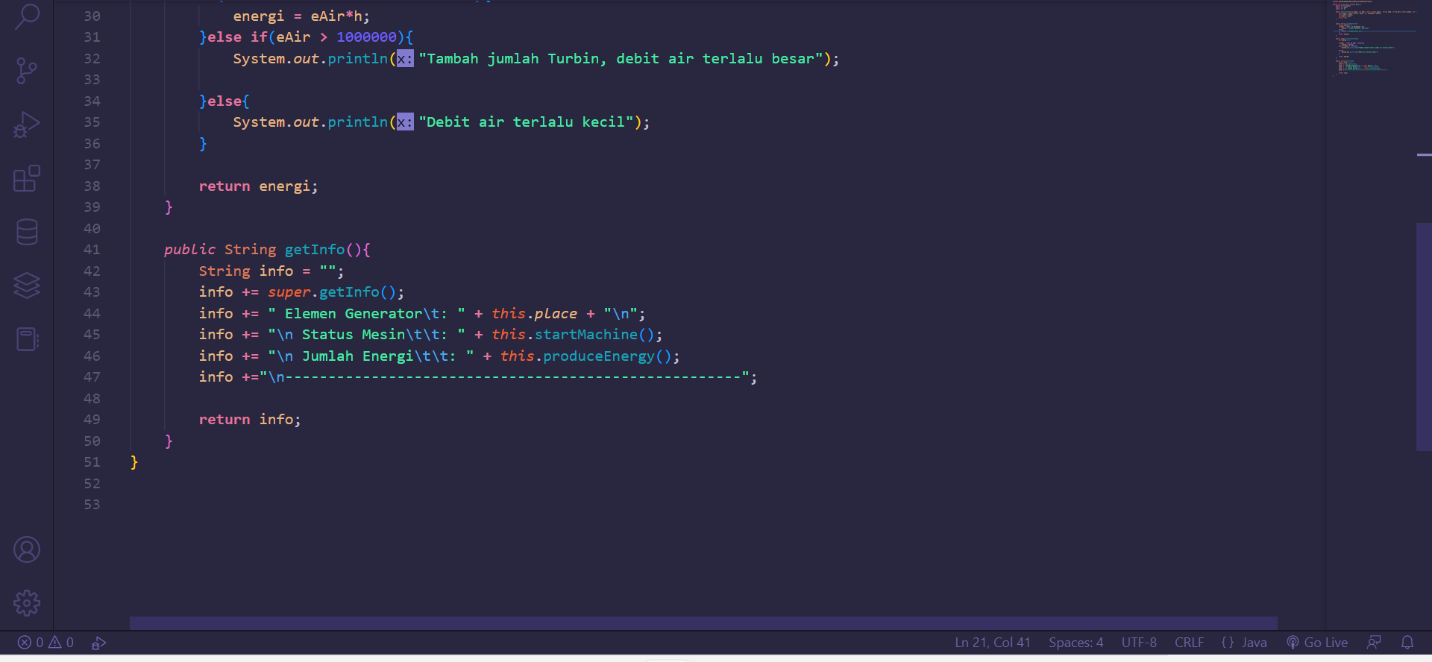


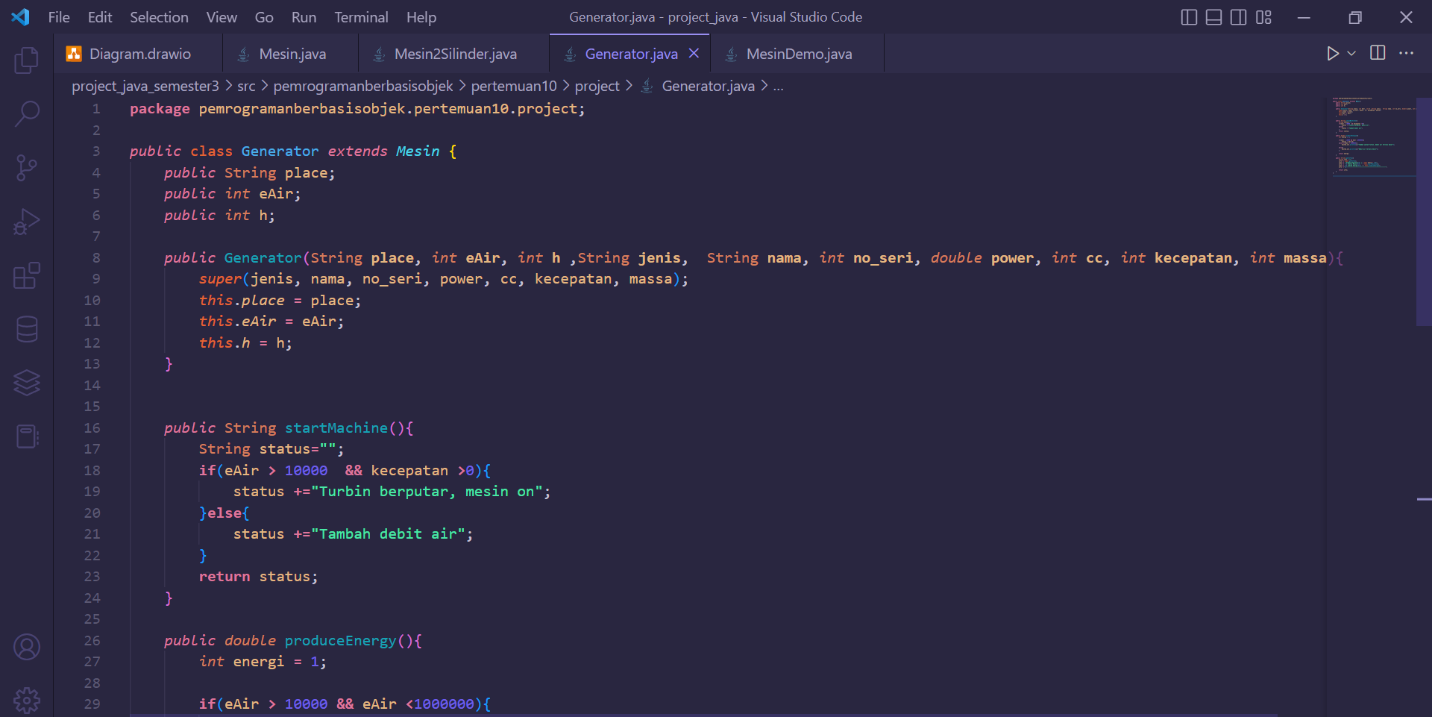
**Code class Mesin2Silinder**



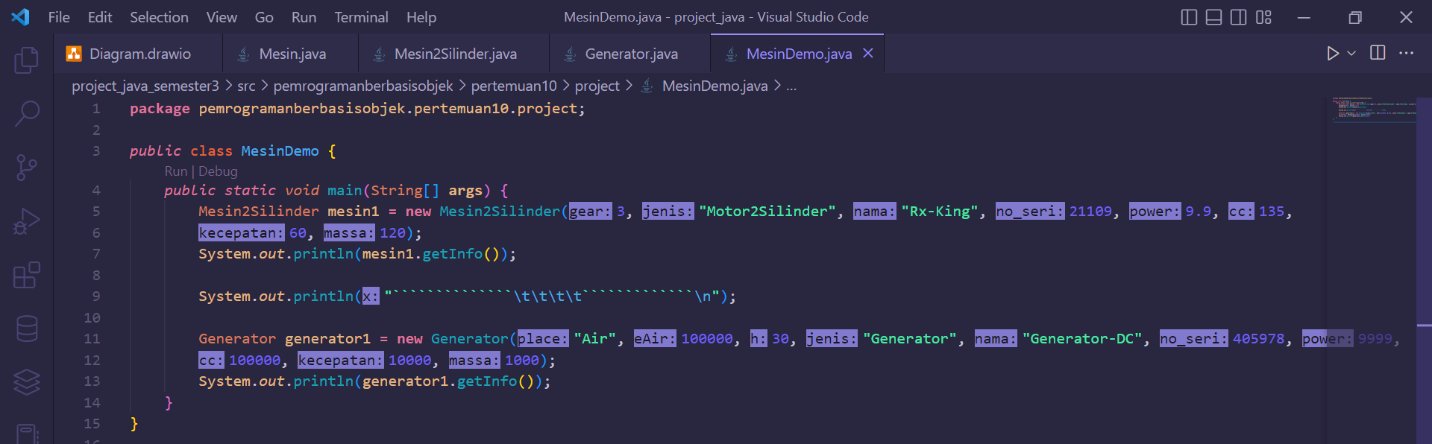


**Code class Generator**

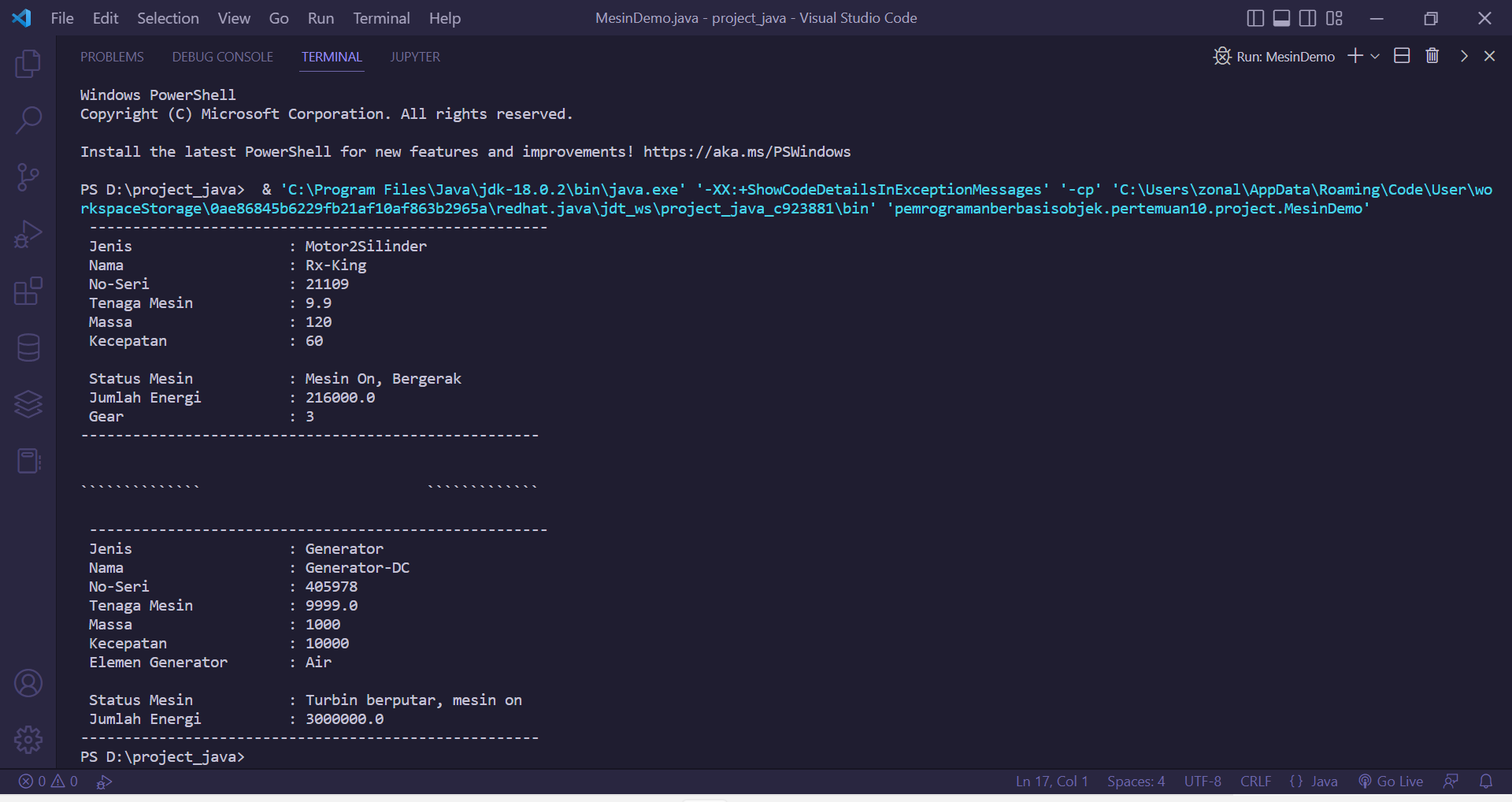




**Class MesinDemo**

****

**Hasil running**

****