LAPORAN PRAKTIKUM KE 4

PEMROGRAMAN BERORIENTASI OBJEK

Nama : Malikah Haibatul Mulkiyyah

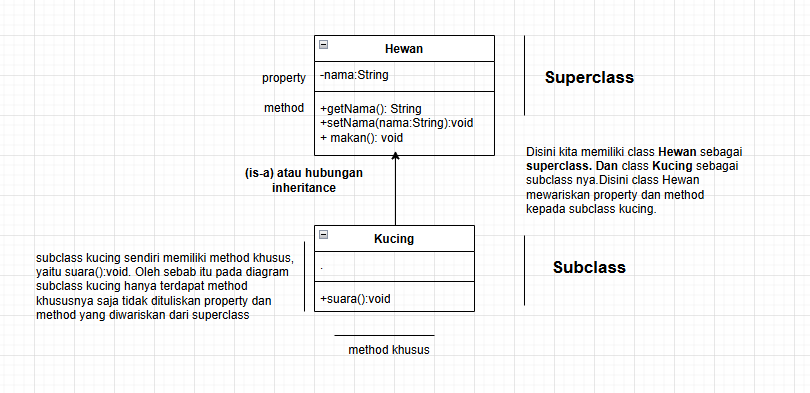
Nim : 2403072

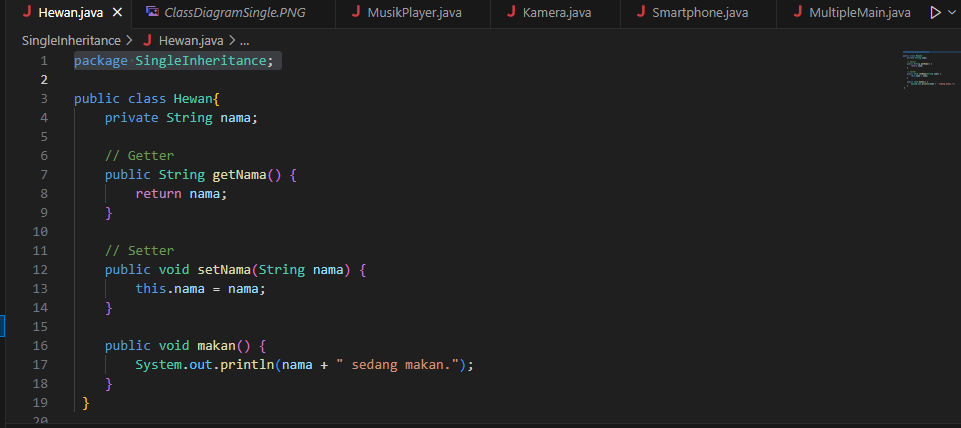
Kelas : TI2C

1. INHERITANCE
2. Single Inheritance

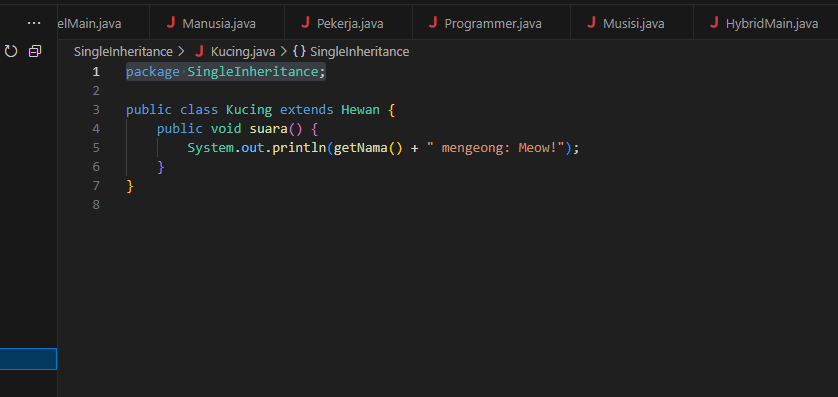
Single inheritance adalah pewarisan dimana satu subclass hanya mewarisi dari satu superclass saja.

Berikut adalah contoh class diagram beserta kode program untuk single inheritance:

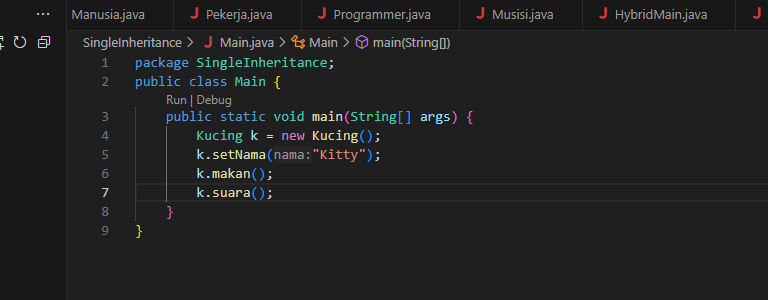




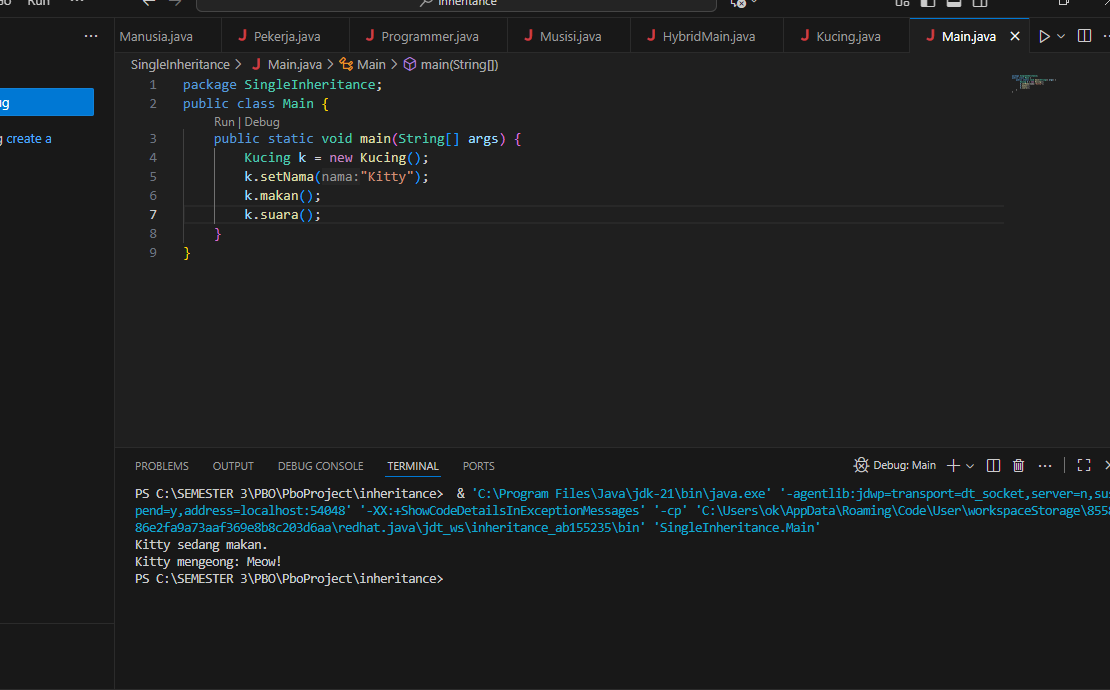
* Package SingleInheritance; = menyatakan bahwa class hewan berada di package tersebut, ini bertujuan agar class lebih terorganisir.
* Kita menggunakan Getter & Setter agar bisa mengakses property nama.



* Class Kucing masih berada di package yang sama dengan class Hewan.
* Keyword extends artinya class Kucing ini adalah subclass yang mewarisi semua property dan method dari clas Hewan (Superclass).



* Membuat objek k dari class Kucing, dan otomatis mewarisi semua property dan method dari class Hewan.
* Kemudian mengisi property nama, memanggil method makan yang ada pada class hewan, serta memanggil method suara yang merupakan method khusus pada class Kucing.

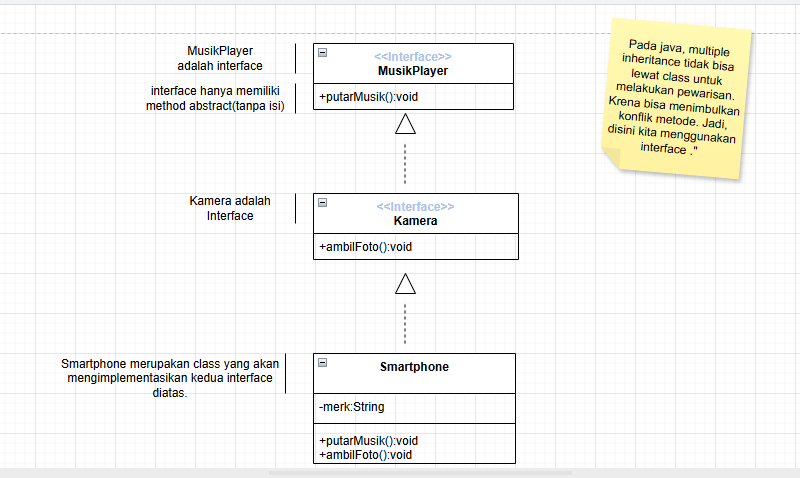


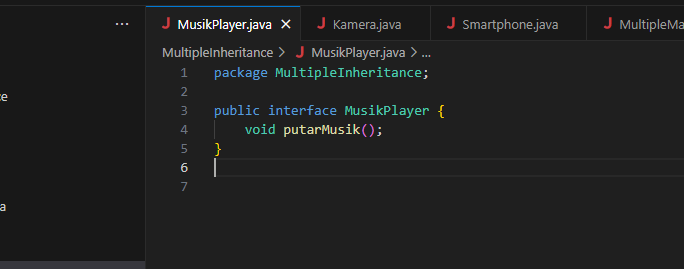
* Ini adalah output dari kode program single inheritance.

1. Mutiple inheritance

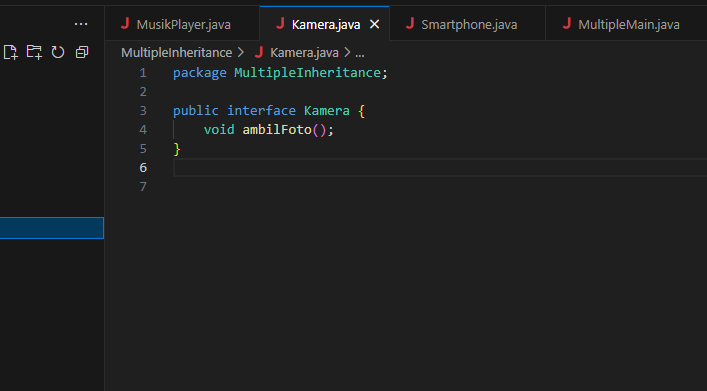
Dalam java multiple inheritance tidak bisa mewarisi antar class langsung, artinya sebuah class di java hanya boleh extend satu superclass saja, ini bertujuan untuk menghindari diamond konflik. Java hanya mendukung multiple inheritance melalui interface, karena satu kelas bisa implements lebih dari satu interface. Satu class bisa extends satu class induk **dan sekaligus** implements banyak interface. Inilah cara Java mengakomodasi kebutuhan multiple inheritance.

Berikut adalah contoh class diagram dan kode program multiple inheritance :

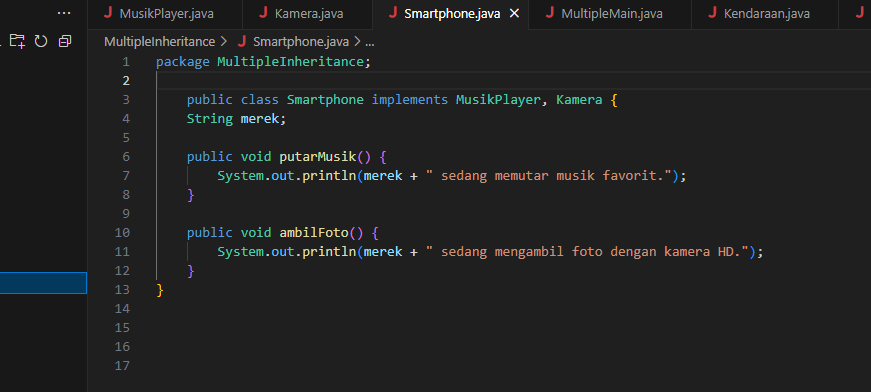




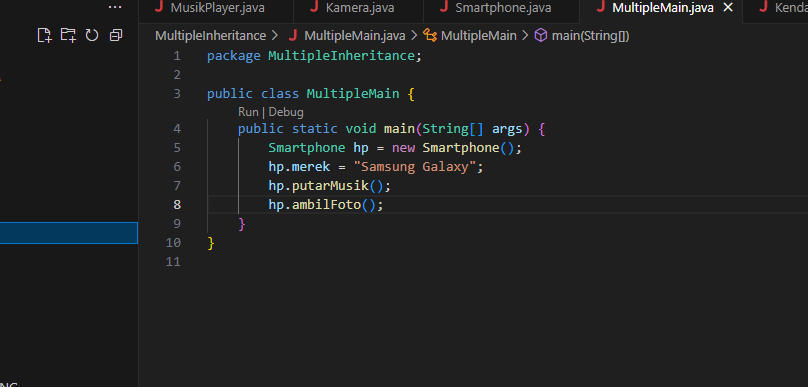
* MusikPlayer merupakan interface yang berada di package MultipleInheritance.
* Interface MusikPlayer hanya memiliki method putarMusik.



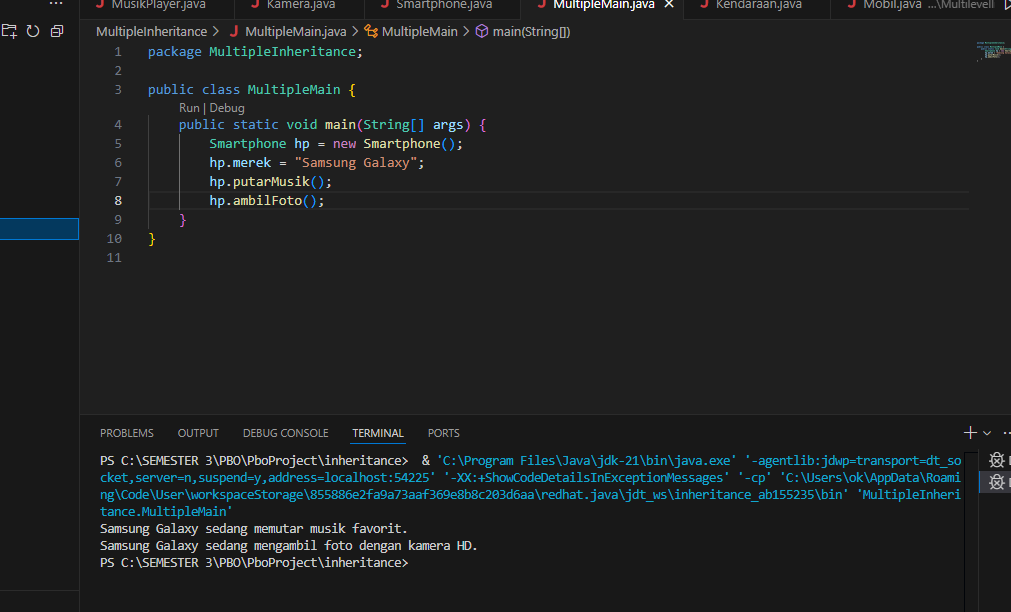
* Interface Kamera juga sama berada di package multiple inheritance.
* Disini interface Kamera hanya memiliki satu method saja yaitu ambilFoto.



* Smartphone merupakan class yang akan mengimplements method dari interface MusikPlayer dan Kamera.
* Disini class Smartphone mengimplements method putarMusik dan ambilFoto.



* Kita buat objek h dari class Smartphone
* Lalu mengisi property merk milik class Smartphone itu sendiri.
* Kemudian memanggil method yang dimiliki oleh interface MusikPlayer dan Kamera.

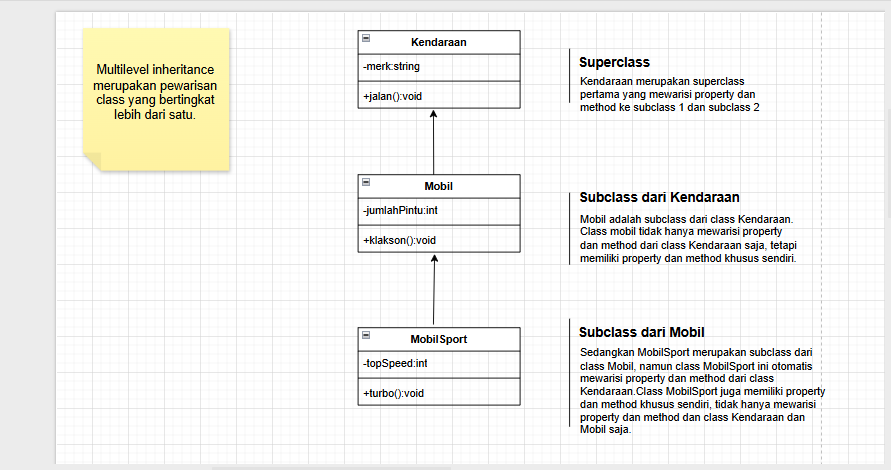


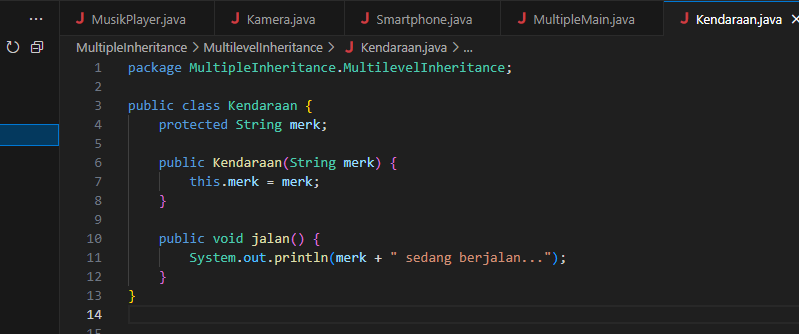
* Dan gambar diatas adalah hasil output dari multiple inheritance.

1. Multilevel inheritance

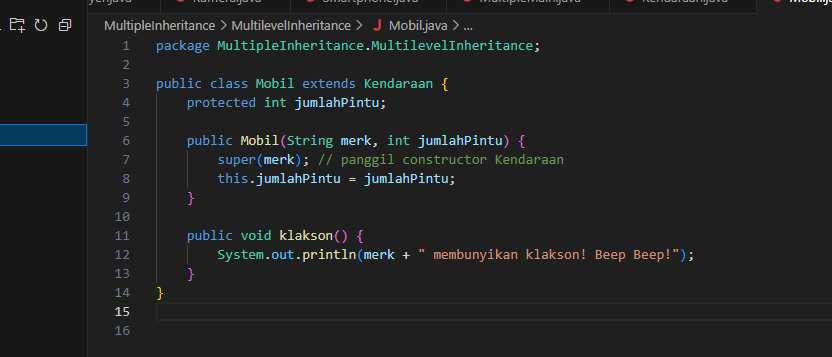
Multilevel inheritance adalah pewarisan yang terjadi bertingkat, yaitu satu superclass diwarisi oleh sebuah subcla**ss**, lalu subclass tersebut menjadi supercla**ss** lagi untuk subclass berikutnya.

Berikut adalah contoh class diagram beserta kode program multilevel inheritance :

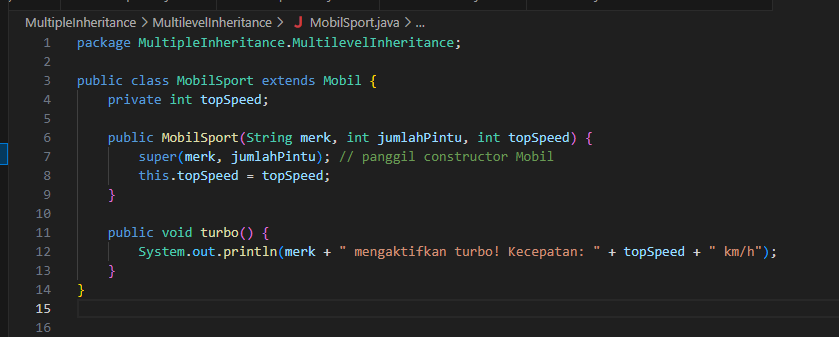


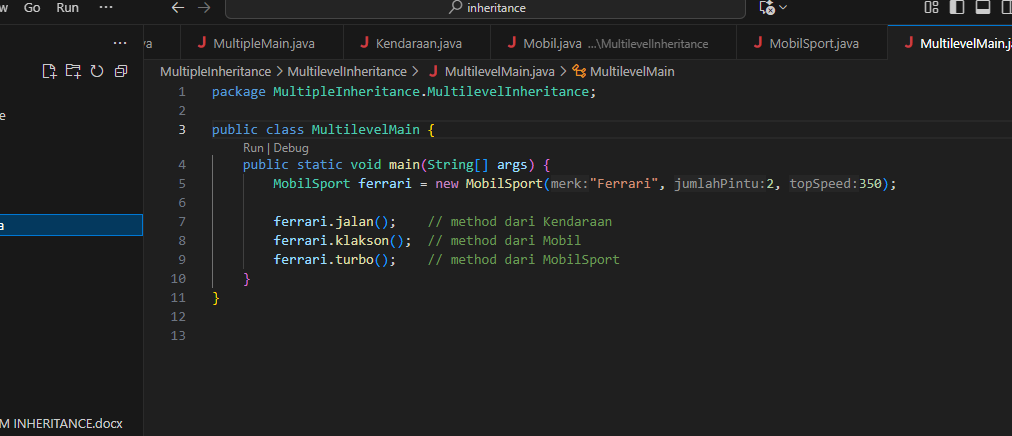


* Class Kendaraan berada di sub-package MultilevelInheritance, yang berada di package MultipleInheritance.
* Dalam class Kendaraan property merk memiliki akses protected, yang artinya property merk ini bisa diakses oleh class itu sendiri, atau subclass, serta kelas lain dalam package yang sama.

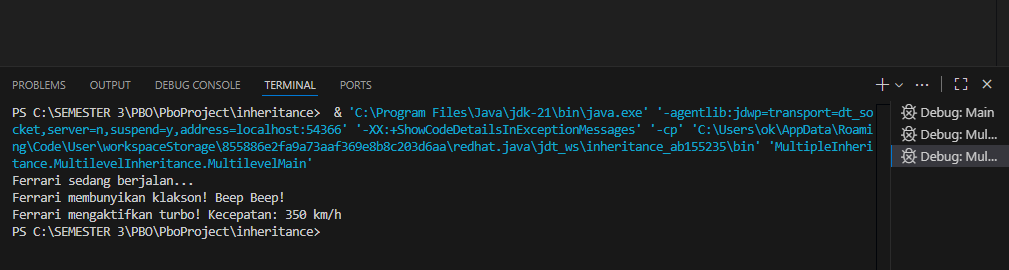


* Keyword extend artinya class Mobil mewarisi semua property dan method yang ada pada class Kendaraan.





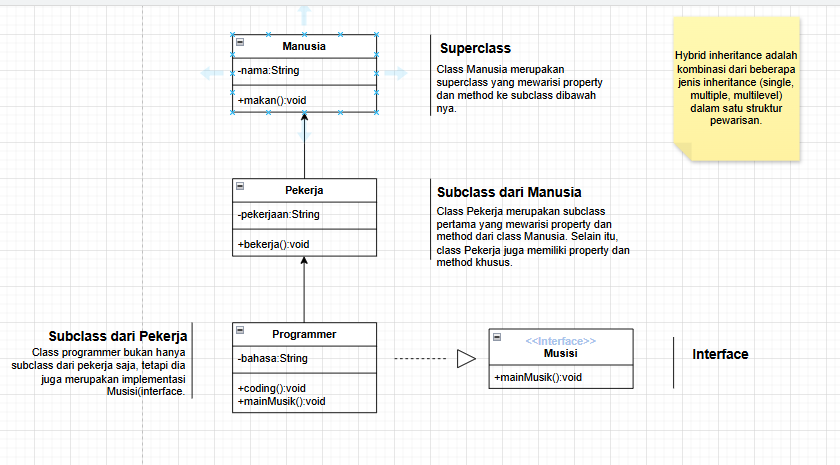
* Di dalam class MobilSport kita buat objek ferrari
* Kemudian objek tersebut akan menjalankan method yang sudah diwarisi oleh class Kendaraan, class Mobil, serta menjalankan method khusus yang dimiliki class MobilSport itu sendiri.
* Berikut adalah hasil outputnya

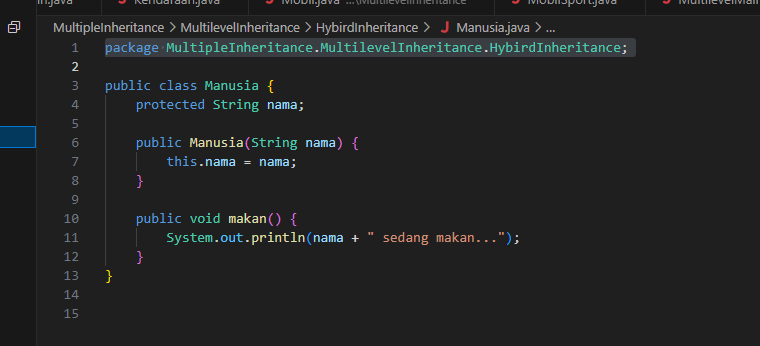


1. Hybrid Inheritance

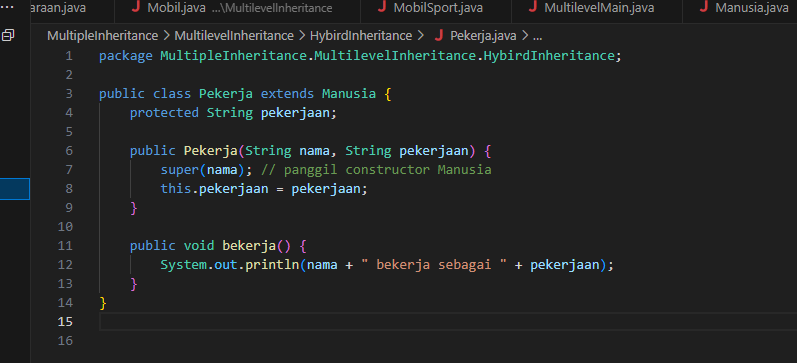
Hybrid inheritance merupakan pewarisan yang menggabungkan beberapa inheritance (single,multiple,multilevel). Dalam java, hybrid inheritance hanya bisa dibuat dengan kombinasi class dan interface.

Berikut adalah contoh dari class diagram dan kode program hybrid inheritance:

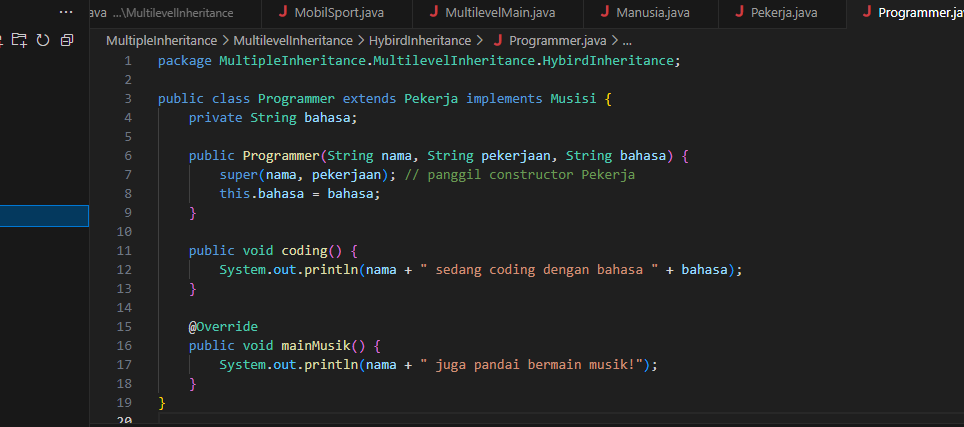




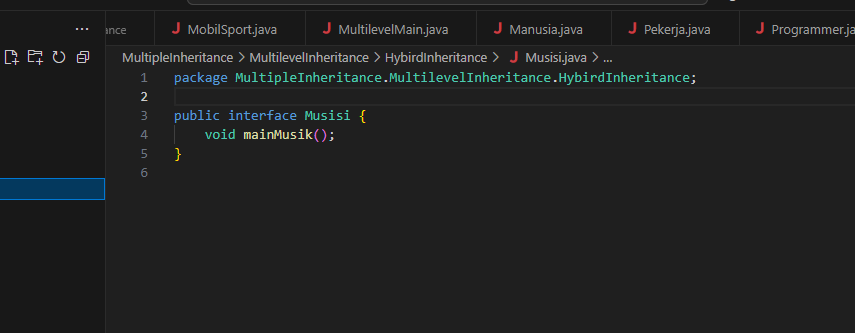
* Manusia merupakan superclass
* Dalam class Manusia property nama memiliki akses protected, artinya property ini bisa diakses oleh class itu sendiri, atau subclass, serta kelas lain dalam package yang sama.



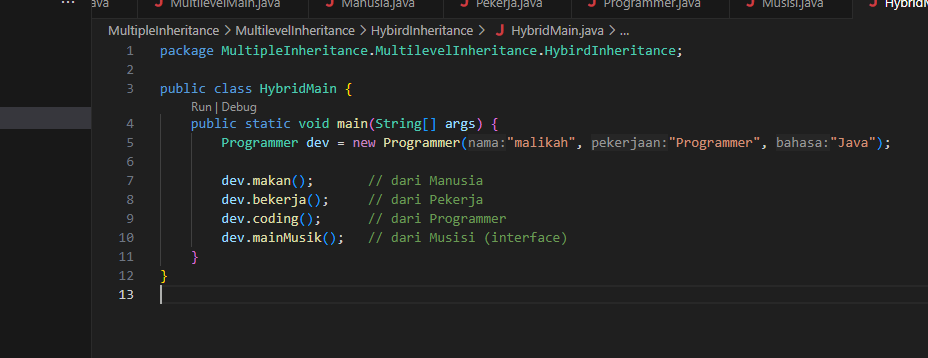
* Pekerja merupakan subclass dari class Manusia
* Class Pekerja mewarisi semua property dan method dari class Manusia.



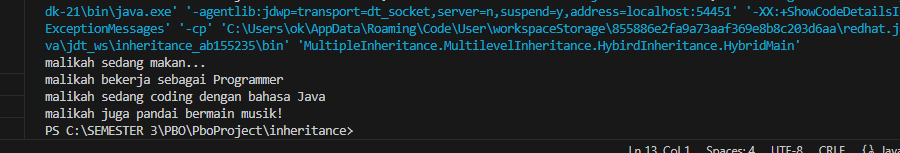
* Programmer merupakan subclass dari class Pekerja (Jadi multilevel inheritance)
* Class Programmer mewarisi property dan method dari class Pekerja, serta mengimplement method yang ada pada interface Musisi.



* Musisi merupakan interface yang memiliki method abstrak berupa mainMusik.



* Kita buat objek dev dalam class Programmer, kemudian kita memanggil constractor dari clas Programmer untuk membuat objek dengan parameter.
* Malikah adalah nilai untuk property nama yang ada pada class Manusia.
* Programmer adalah nilai untuk property pekerjaan yang ada pada class Pekerja.
* Java adalah nilai untuk property bahasa yang ada pada class Programmer.
* Setelah itu, objek memanggil semua method yang diwarisi dari setiap kelas, serta method dari interface yang akan di implements.
* Berikut adalah hasil output dari contoh kode program Hybrid inheritance

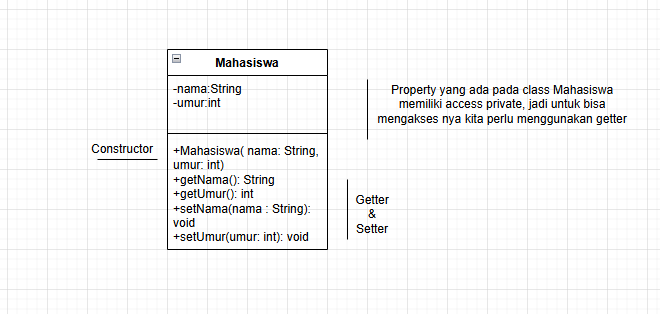


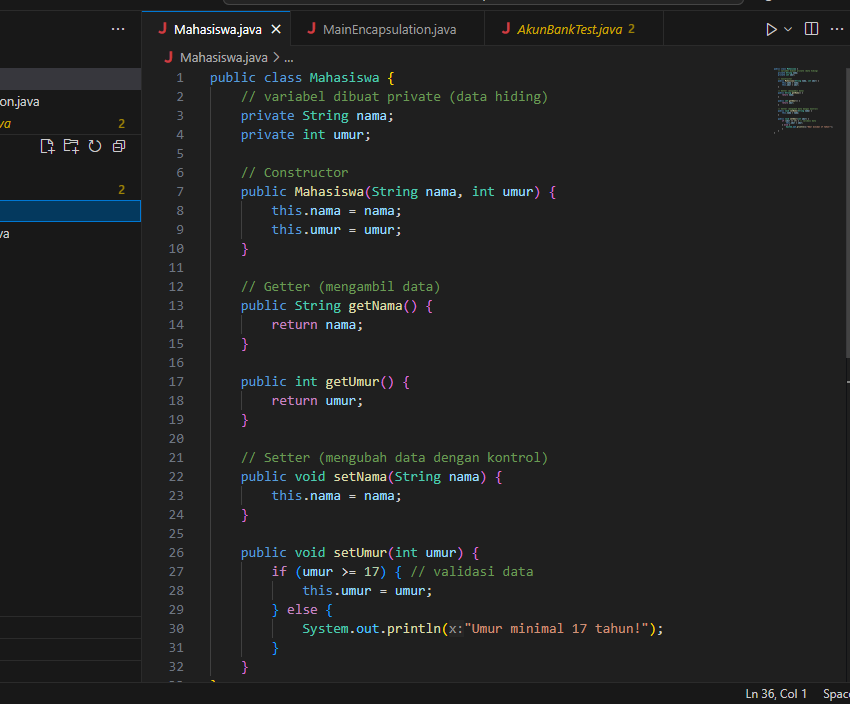
1. ENCAPSULATION

Secara sederhana, encapsulation berarti membungkus data (variabel) dan perilaku (method) dalam satu kesatuan (class), sekaligus membatasi akses langsung ke data tersebut dengan access modifier, tujuannya untuk melindungi data dari akses langsung, memberikan kontrol penuh melalui method khusus ( getter & setter), serta membuat kode lebih aman, rapi, dan mudah dipelihara. Encapsulation memiliki 4 access modifier, yaitu :

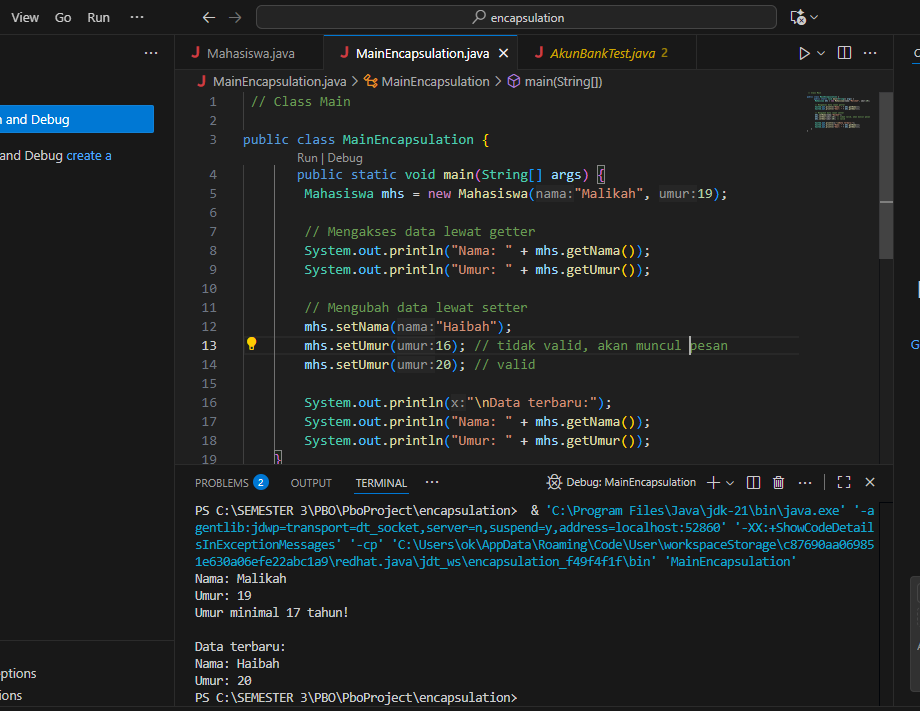
* Privat : Hanya bisa diakses oleh class itu sendiri
* Public : Bisa diakses dari mana saja
* Protected : Bisa diakses oleh subclass/kelas turunan
* Default/tanpa modifier : Hanya bisa diakses di package yang sama

Sebagai contoh : Kita akan buat class Mahasiswa yang memiliki akses property private, jadi harus di akses menggunakan getter. Serta method mengubah isi property menggunakan setter.





* Mahasiswa merupakan nama class yang kita buat
* Class Mahasiswa memiliki property (- nama:String & - umur:int)
* Karena property pada class mahasiswa memiliki access private jadi kita memerlukan getter dan setter untuk mengaksesnya.



* Kita buat objek mhs pada class mahasiswa
* Kemudian kita akan mengakses data lewat getter, dan mengubah data lewat setter
* Lalu menampilkan data yang sudah diubah