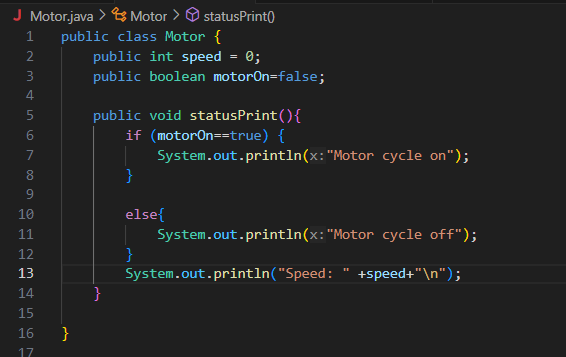
**ENCAPSULATION**

Erwan Majid/08/2i

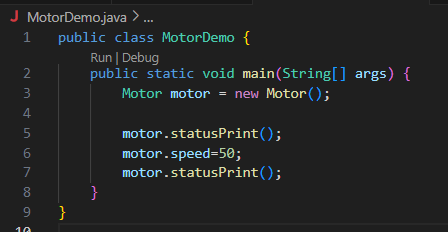
Link github: <https://github.com/Majid5654/Semester-3/tree/Main/JAVA%20OOP/Week3>

* **3.1 Experiment 1 – Encapsulation**

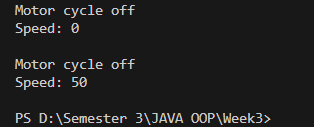
Class Motor:



Class Motordemo:



Result:



From experiment 1 - encapsulation, in your opinion, is there anything strange?

That is, the speed of the motor suddenly changes from 0 to 50. Even more awkward, the motor contact

position is still in the OFF condition. How is it possible for a motor to be blinked from zero to 50, and

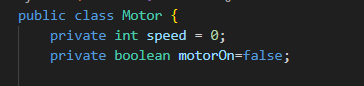
even then the ignition is OFF?

Now in this case, access to motor attributes is apparently not controlled. In fact, objects in the real

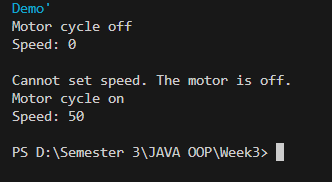
world always have limits and mechanisms for how these objects can be used. Then, how can we

improve the Motor class above so that it can be used properly? We can consider the following:

1. Hiding internal attributes (speed, motorOn) from users (other classes)

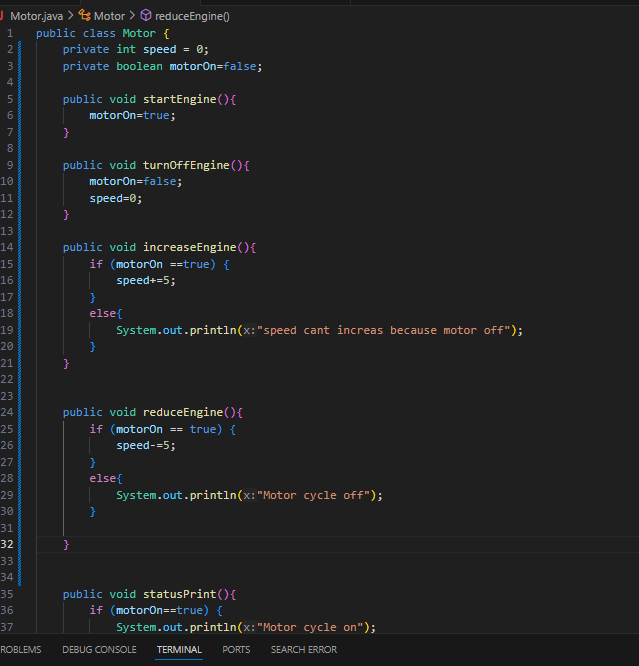


2. Provides a special method for accessing attributes.

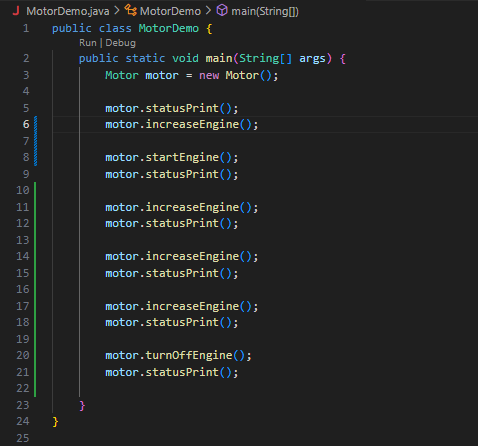
For that, let's continue the next experiment about Access Modifier.

* **Experiment 2**

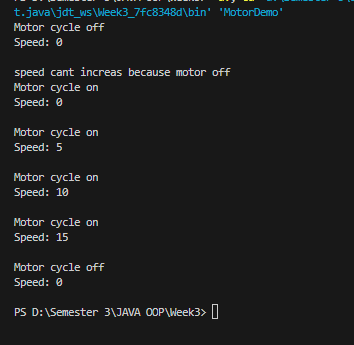
Class motor:

****

Class motordemo:



Result:



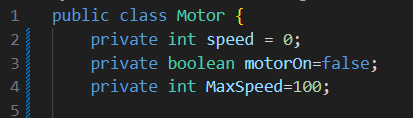
3.3 Questions

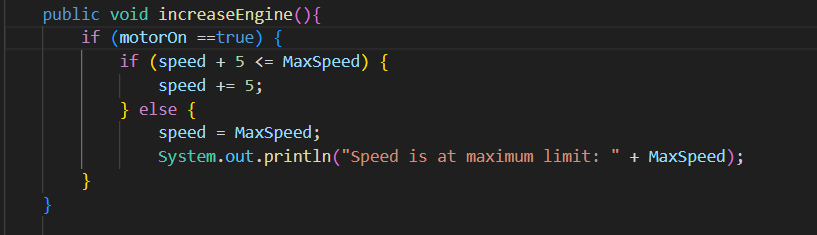
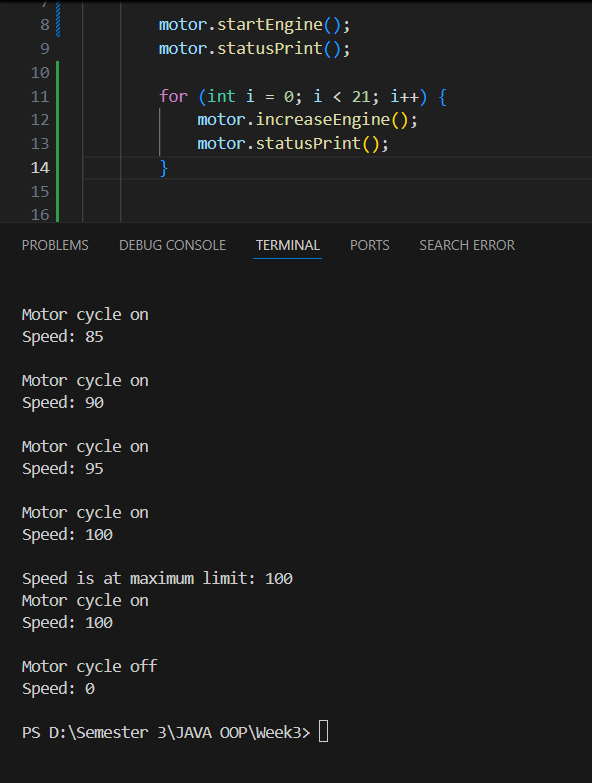
1. In the MotorDemo class, when we add speed for the first time, why does the warning "Speed cannot appear because the engine is off!"

-because the motor still off,and the other reason is not using motor.startEngine(); method first,so in the program Boolean engine is off

2. Can the speed and contact attributes be set private?

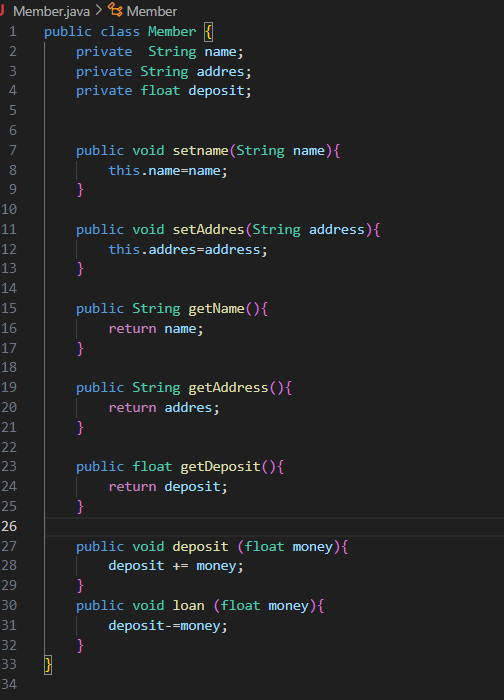
-it can these internal attributes cannot be accessed directly from outside the class, and can only be modified through specific methods (like startEngine, increaseEngine, etc.)

3. Change the Motor class so that the maximum speed is 100! 



* **3.4 Experiment 3 - Getter and Setter**

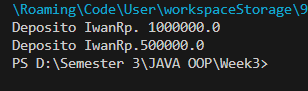
Class member:



Class MemberDemo:



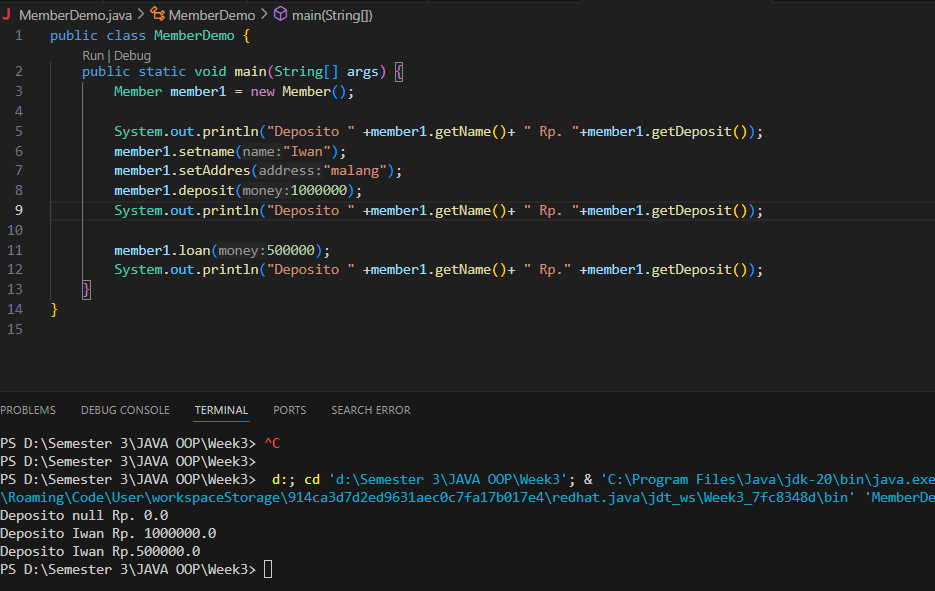
Result:



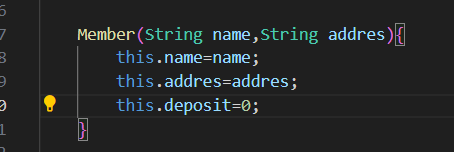
* **Percobaan 4 - Construktor, Instantiation**

1. The first step of experiment 4 is to change the Demo class as follows

Class demo:

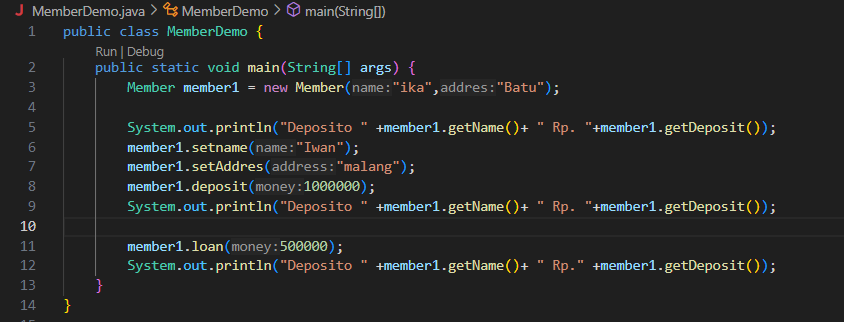
****

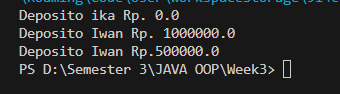
Class member:



In the Anggota class, a constructor is created with a default access modifier that has 2 nama and alamat parameters. And in the constructor, the pinjam value for the first time is Rp. 0

5. Change class Demo:



6. The result:

* **3.6 Question - Experiments 3 and 4**

1. What are getters and setters?

- to access and modify the private attributes of a class. It help to encapsulate the internal representation of an object.

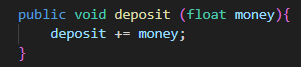
2. What is the use of the getDeposit () method?

- It allows access to the private deposit field in a controlled manner

- getDeposit() is used to retrieve and print the current deposit value

3. What method is used to add balance?

- the deposit() method.



4. What does the constructor mean?

- special method in a class that is called when an instance (object) of the class is created

5. Mention the rules in making a constructor?

- Name Matching: The constructor's name must exactly match the name of the class. For example, if the class is named Member, the constructor must also be named Member

- No Return Type: Constructors do not have a return type, not even void. They are used solely to initialize the object.

- Constructor Chaining: Constructors can call other constructors in the same class using the this() keyword

6. Can the constructor be private?

- Yes, a constructor can be private. Using a private constructor restricts the instantiation of the class from outside its own class

7. When to use parameters with passsing parameters?

- To Provide Input Data: Parameters allow to pass data into a method or constructor so that it can perform operations based on that input.

- To Customize Behavior:Parameters allow to customize the behavior of methods and constructors, making them more flexible

- To Pass Data Between Methods: Parameters can be used to pass data between methods within the same class or across different classes.

8. What is the difference between class attribute and instance attribute?

- Class Attributes:

Scope: Shared among all instances of the class.

Defined: At the class level (outside methods).

Access: Can be accessed via the class name or any instance.

Modification: Changes affect all instances.

Instance Attributes:

Scope: Unique to each instance of the class.

Defined: Inside constructors or methods.

Access: Accessed only through the instance.

Modification: Changes affect only that specific instance.

9. What is the difference between the class method and the method instance?

- Class Methods:

Bound to: The class itself.

Access: Class attributes.

Defined with: static keyword (Java) or @classmethod (Python).

Called by: Class name.

Instance Methods:

Bound to: Instances of the class.

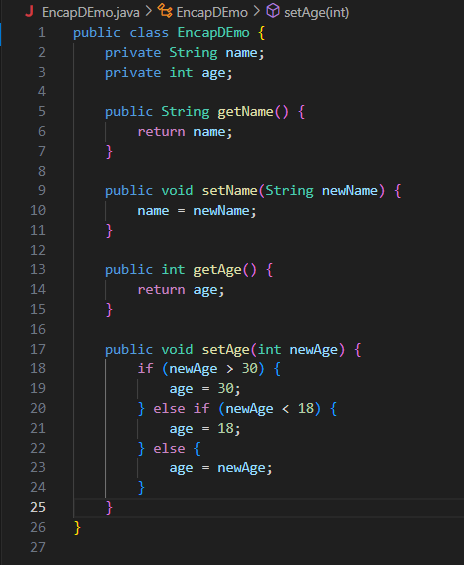
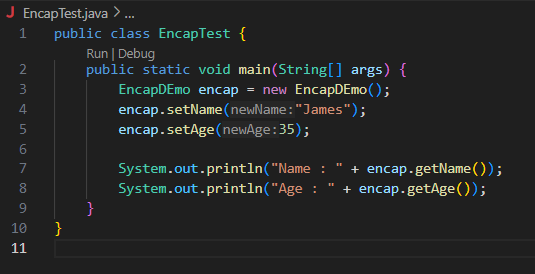
Access: Both instance and class attributes.

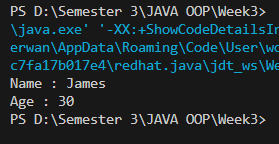
Defined without: static keyword or @classmethod.

Called by: Instance of the class

* **5. Task**

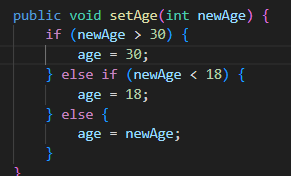
1. Try the program below and write the output results





2. In the above program, in the EncapTest class we set age with a value of 35, but when

displayed on the screen the value is 30, explain why.

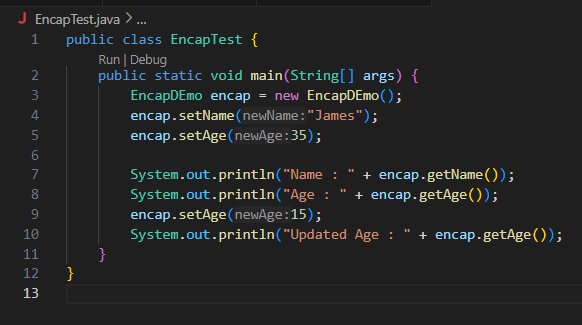
 -

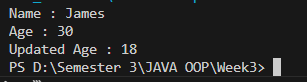
because the setAge() method limited the age to 30.

and when we type 35 it will run in first if newage greater than 30 ,and declare age is 30

3. Change the program above so that the age attribute can be given a maximum value of 30

and a minimum of 18

-



Indeed ,in first program run it set maximum age value 30 and a minimum 18

4. In a savings and loan information system, there are class Members who have attributes

including KTP number, name, loan limit, and loan amount. Members can borrow money within

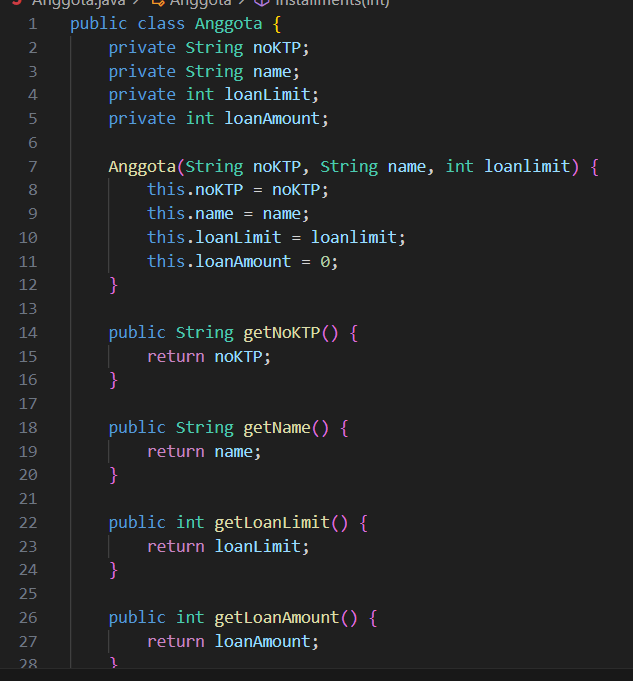
the specified lending limits. Members can also repay loans. When the Member repays the

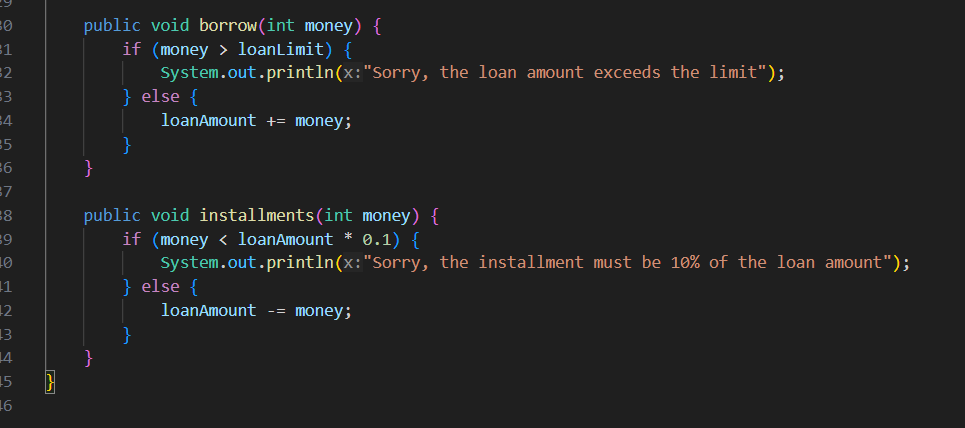
loan, the loan amount will decrease according to the nominal installment. Create a Member

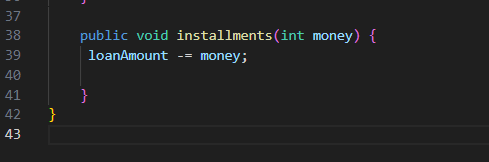
class, provide attributes, methods and constructors as needed. Test with the following

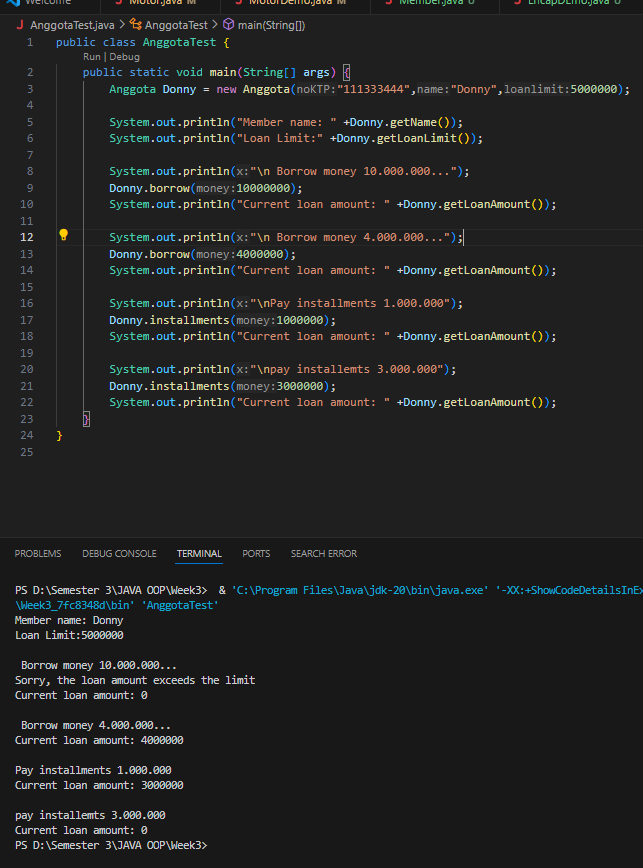
TestKoperasi to check whether the class of the Member you created is as expected

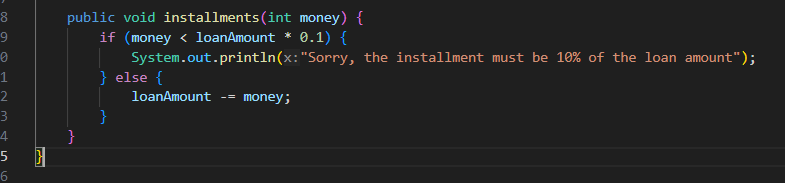
Class Anggota:



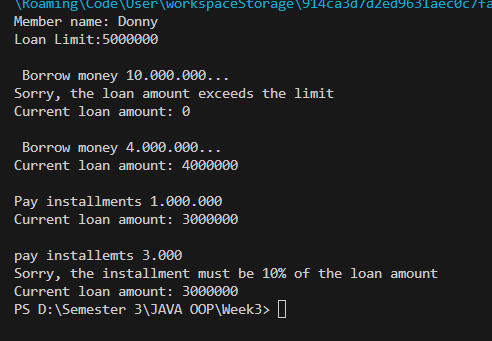




Output:

5. Modification of question no. 4 so that the minimum nominal installment is 10% of the current loan amount. If the installments are less than that, then the warning "Sorry, installments must be 10% of the loan amount".-

Output:



6. Modify the TestKoperasi class, so that the loan amount and installments can receive input from the console.

- 