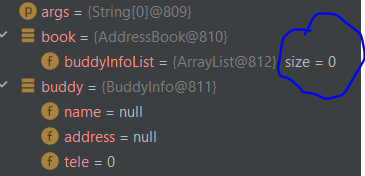
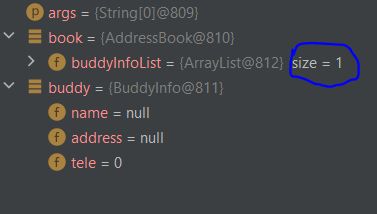
**Lab 3**

**Section 1: Git and Debug**

Part 1

Step 5: size = 0



Step 7: size value has changed, size = 1

Part 2

Step 7: https://github.com/mralifahd/Lab3

Step 11: change is not automatically shown online, you need to push

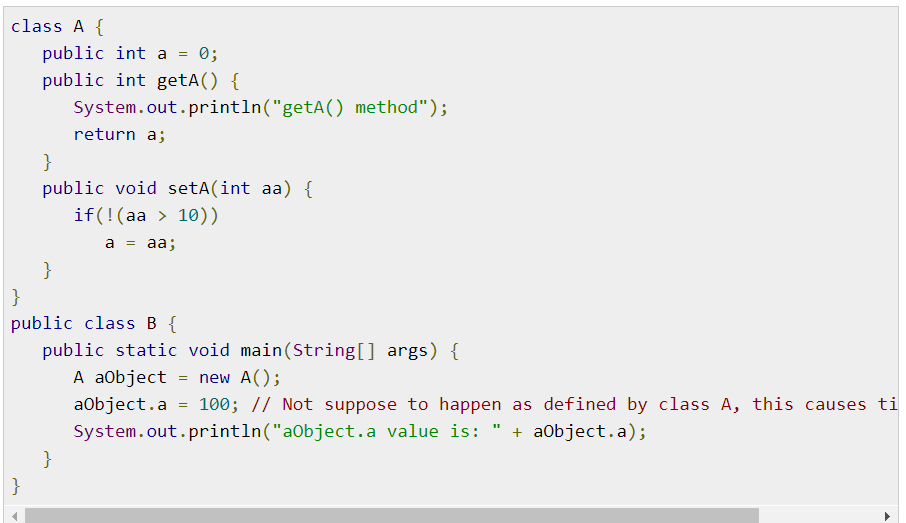
**Section 2: Definitions**

Tight vs loose coupling:

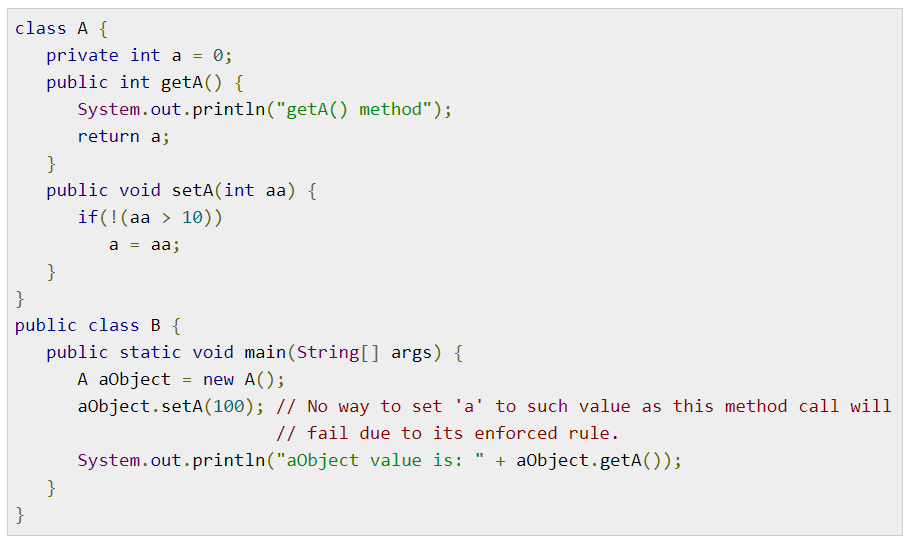
Loose coupling is minimizing the dependencies for a class that uses another class directly.

Tight coupling is when classes and their instances(objects) are dependent on each other.

Source: https://www.interviewsansar.com/loose-coupling-and-tight-coupling-in-java/

Ex: 

If we change variable A then class B breaks, tight coupling



If class A changes internally class B will not break, loose coupling

Source: https://www.tutorialspoint.com/what-are-the-differences-between-tight-coupling-and-loose-coupling-in-java

Encapsulation:

Encapsulation is the wrapping of data (variables) and code acting on the data (methods) together as a single unit. Variables of a class will be hidden from other classes, and can only be accessed through the methods of their current class.

For example, If I have a train system with multiple car objects and a separate class called booking, in order to adjust the setting in the car object from my booking object I would have to call get or set methods of the car class. For example, if I want to book a seat in a car. Seats will be a private field in car class that can only be changed/booked via a set method in the class which I call from my booking object.

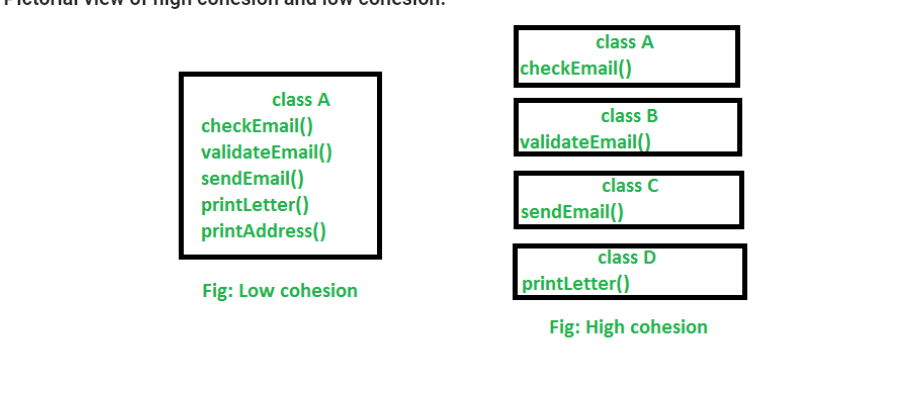
Also called data hiding.

Source: https://www.tutorialspoint.com/java/java\_encapsulation.htm#:~:text=Encapsulation%20in%20Java%20is%20a,methods%20of%20their%20current%20class.

Class cohesion:

Cohesion is making sure that a class is designed with a single, well-focused purpose.  
The more focused a class is, the cohesiveness of that class is more. Classes become easier to maintain and easier to reuse

Ex: like in the image below just specify the different possible methods you would have in their own classes. Different actions in different classes.



Sources: https://www.geeksforgeeks.org/cohesion-in-java/#:~:text=Cohesion%20is%20the%20Object%20Oriented,of%20that%20class%20is%20more.

Law of Demeter:

The Law of Demeter attempts to minimize coupling between classes in any program. The idea is to prevent reaching into an object and gaining access to a third object’s methods or more.

Ex: “Only talk to your immediate friends.” “Don't talk to strangers.”

The idea is to not do code like this:

objectA.getObjectB().doSomething();

Source: https://alvinalexander.com/java/java-law-of-demeter-java-examples/#:~:text=Summary%3A%20The%20Law%20of%20Demeter,using%20Java%20source%20code%20examples.&text=The%20Law%20of%20Demeter%20for,to%20a%20third%20object's%20methods.