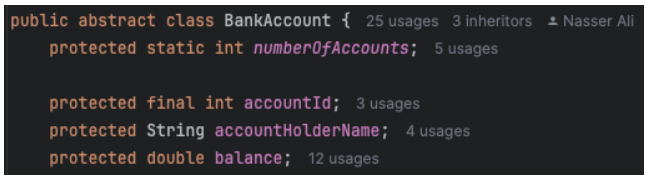
Dev Ops + Software Development Architecture and Testing – QAP #1

Link to GitHub Repo: https://github.com/Nasser-A-Ali/DevOps-SDAT-QAP1

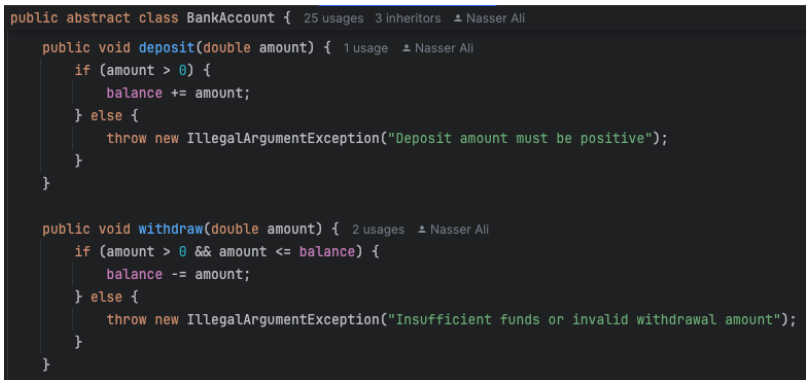
1. **Explain how your code meets clean code practices by using at least 3 examples of your own code. Screenshots should be used.**

While creating the code I kept many of the clean code principles which I believe my code meets including using meaningful names, following the single responsibility principle, small functions, avoiding repetition (DRY principle), readable code, and using comments only when necessary. Here are some specific examples:

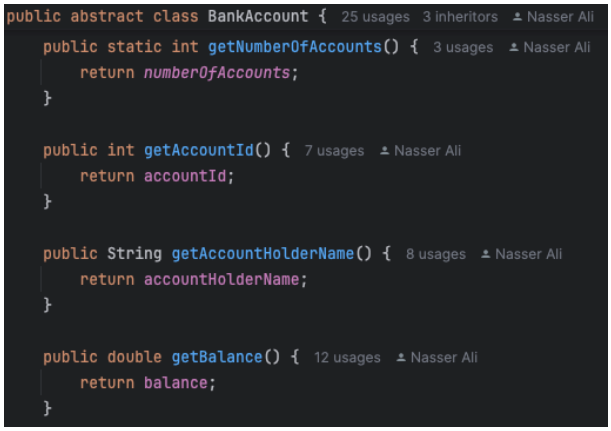
Meaningful names:



Functions Should Only One Thing (Single Responsibility Principle):



Avoid Repetition (DRY – Don't Repeat Yourself):



2. **Explain your project. What it does, how it works. Explain the test cases you used.**

   The project is based on an abstract BankAccount base class that extends into ChequingAccount, SavingsAccount, and CreditAccount classes. A BankAccountService class is used to handle common functionality of the various account instances.

   Finally, there is a BankAccountServiceTest class that contains five test suites using Junit 5. The test cases are as follows:

   1. testCreateAccount(): creates a chequing account and then uses the assertEquals assertion to ensure the attributes of the instance are equal to the parameters used to create it.
   2. testDeposit(): creates a savings account with a balance of $800.50 and then uses an instance of the BankAccountService to deposit $350.00. The test uses the assertEquals assertion to ensure the updated balance is equal to $1150.50.
   3. testWithdraw(): creates a savings account with a balance of $500.00 and then uses an instance of the BankAccountService to withdraw $150.50. The test uses the assertEquals assertion to ensure the updated balance is equal to $349.50.
   4. testChargeAccountFees(): creates a chequing account with a balance of $500.00 and then uses an instance of the BankAccountService to run the chargeAccountFees method which should subtract $16.95 from the balance. The test uses the assertEquals assertion to ensure the updated balance is equal to $483.05.
   5. testWithdrawLargerThanBalance(): creates a chequing account with a balance of $500.00 and then uses the assertThrows assertion to ensure that running a lambda function that withdraws $501.00 throws an exception that matches the IllegalArgumentException class.

3. **Outline the needed dependencies. Where did you get them from?**

   The only dependency used was the JUnit 5 dependency, it can be found on Maven Repository by following this link: https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-engine

4. **If you had any problems with the QAP please explain what happened.**

   No issues of significance were encountered during the completion of this QAP.