QAP1 - Documentation

Clean Code Practices

My code follows clean code practices for many reasons. Three examples of this are meaningful names, error handling, and consistent formatting and readability.

- My meaningful names are BankAccount.java, AccountTest.java, etc. These make it clear which folder is which when looking at the code. They clearly describe the purpose of the class.
- The error handling represents clean code practices because it prevents invalid operations and throws an error when necessary. When you look at the deposit or withdraw validation for example, you will see that I used if/else to determine if an error should be presented or not.
- Consistent formatting and readability is important when coding. Not only for myself, but others who look at the code, it is good to have it clean and organized so it can be easily dissected. Proper indentation and spacing is something we learned early on in our coding journey.

Explaining my Project

My code sets up a basic Java application using IntelliJ, Junit 5, and GitHub Actions. It includes BankAccount.java and AccountTest.java.

- BankAccount.java is the main class for account creation, deposit and withdraw, and balance inquiry. It includes the constructor and then methods for whatever I'm trying to accomplish in the end. Constructor initializes an account, ensures the money is not less than 0, and adds a transaction list. Deposit method deposits money into the account, with a validation rejecting invalid amounts and then adds the transaction to the list. The withdraw method withdraws money from the account. It has 2 validations which mean you can't withdraw less then 0, and if you try to withdraw more than what's in your balance, it will present an error.
- AccountTest.java is the class that does the testing for the main class. My imports on the top are based on my dependencies from pom.xml, and it verifies that it all works. We were asked to write three tests and one has to be a negative scenario, so I tested deposit, withdraw, and insufficient funds. For each one, I made an account called "TylerAcc" with an initial deposit of $100. TestDeposit() tests that when I deposit $50, the total balance is then $150, TestWithdraw() tests that when I go to take out $30, the updated balance will be $70, and TestInsufficientFunds() tests that if I wanted to take out $200, I would not be able to because my initial balance was $100. I used an exception for this test as it was a part of the TestWithdraw() method.

Dependencies

When you initialize a Maven project on IntelliJ, you also created a pom.xml package at the same time. This is where we keep the dependencies. I got the dependencies from an example we did in class. When you add the dependencies in pom.xml, IntelliJ automatically downloads them (I’m pretty sure).

Problems?

No problems. With class lectures, examples, and research, this project went fairly smooth.