<u>Question</u>

- Run the `main` method on the `Main` class, what is it doing? What is being outputted to the console?

<u>Answer</u>

The main method has been setup so as to perform exactly the same user journey with two different classes. One being DodgyBankAccount, the other being SecureBankAccount.

However, one class has been so badly setup so as to potentially allow the user to perform cart blanche with their account balance. They are able to do this as the fields within the class have not been correctly encapsulated. Thus, they can freely set their account balance to 1000000 (1 million) and, similarly, add financial rewards to further increment their account balance without first having actually deposited any money.

In comparison, the SecureBankAccount class has been setup in such a way as fraud is a lot harder, if not impossible to achieve. The user can therefore not add rewards at will, nor alter the account balance in their favour.

<u>Question</u>

- Look at the `DodgyBankAccount`, this class is not well-encapsulated. Can you note down the problems with how the class is designed, and the ways it is being misused?

<u>Answer</u>

The three fields at the top of the class, accountNumber, accountBalance and rewardAmount have all been declared as variables, rather than constants. Not only that, but they are declared with the public access modifier. Thus meaning that their values can be accessed/altered outside of the methods theoretically, in this case, designed to achieve this. This means that the account balance can be altered at will, as discussed in the previous question. The method at the end of the class, addReward is also declared with the public access modifier, meaning that rewards, like the account balance can be added to at will.

<u>Question</u>

- Compare and contrast the `DodgyBankAccount` and the `SecureBankAccount`, how is the `SecureBankAccount` different to the `DodgyBankAccount`? How is it designed to prevent it from being misused? Are there instances of better method names for clearer abstraction?

<u>Answer</u>

Comparing and contrasting DodgyBankAccount and SecureBankAccount highlights that SecureBankAccount is the complete antithesis of DodgyBankAccount.

For one thing, accountNumber, accountBalance and rewardAmount have all been declared private meaning that they can only be accessed by methods and properties within the SecureBankAccount class. Not only that, but accountNumber and rewardAmount have been additionally declared as constants meaning that once an initial value has been set for these fields they **cannot** be altered. At all.

There is one instance in SecureBankAccount where the method to obtain the account balance has possibly been given the more meaningful name with displayAccountBalance rather than getAccountBalance, as is the case with DodgyBankAccount.

Finally, the method at the end of the class, addReward is also declared with the private access modifier, meaning that the method cannot be called outside of the class as is the case with the equivalent in DodgyBankAccount. Indeed, looking down the script in main, you can see that there is code there to call addReward although it is commented out. However, should you uncomment out those method calls, three compiler errors are thrown straight away. Thus reflecting addReward has been correctly declared with the private access modifier.