NUR AZYYATI BINTI ABU BAKAR | 291560

STIWK3014 REAL TIME PROGRAMMING
Tutorial / Exercise 11: ReentrantReadWriteLock () Total 15 Marks

```
import java.util.concurrent.locks.ReentrantReadWriteLock;
import java.util.concurrent.locks.Lock;

public class BankAccountWithLock {
    private double balance;
    private final ReentrantReadWriteLock lock = new ReentrantReadWriteLock();
    private final Lock readLock = lock.readLock();
    private final Lock writeLock = lock.writeLock();

    public BankAccountWithLock(double initialBalance) {
        this.balance = initialBalance;
    }

    // Read balance (shared lock)
    public double getBalance() {
        readLock.lock();

        try {
    System.out.println(Thread.currentThread().getName() + " reads   balance: " +
    balance);
            return balance;
        } finally {
            readLock.unlock();
        }}

    // Deposit money (exclusive lock)
    public void deposit(double amount) {
        writeLock.lock();

        try {
    System.out.println(Thread.currentThread().getName() + " deposits:   " + amount);

         balance += amount;
        } finally {
            writeLock.unlock();
        } }

    // Withdraw money (exclusive lock)
    public void withdraw(double amount) {
        writeLock.lock();

        try {
            if (balance >= amount) {
    System.out.println(Thread.currentThread().getName() + "  withdraws: " + amount);

         balance -= amount;
            } else {
    System.out.println(Thread.currentThread().getName() + " insufficient funds for: "
    + amount);
            }

        } finally {
            writeLock.unlock();
        } }
}
```

The following questions are based on the above Java program.

a. Create the main class for this program

```
public class MainClass {
    public static void main(String[] args) {
        //create bankacc with RM1000 balance
        BankAccountWithLock myAccount = new BankAccountWithLock( initialBalance: 1000.0);

        //thread to read balance
        Thread readerA = new Thread(() -> {
            myAccount.getBalance();
        }, name: "Reader-A");

        //read balance
        Thread readerB = new Thread(() -> {
            myAccount.getBalance();
        }, name: "Reader-B");

        //Thread to deposit
        Thread depositor = new Thread(() -> {
            myAccount.deposit( amount: 300.0);
        }, name: "Depositor");

        //thread to withdraw
        Thread withdrawer = new Thread(() -> {
            myAccount.withdraw( amount: 200.0);
        }, name: "Withdrawer");
        //start
        readerA.start();
        readerB.start();
        depositor.start();
        withdrawer.start();

        //wait all threads to habis
        try {
            readerA.join();
            readerB.join();
            depositor.join();
            withdrawer.join();
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted.");
        }
    }
}
```

(5 marks)

b. What is the output of this program?

```
"C:\Program Files\Java\jdk-16.0.1\bin\j
Reader-A reads  balance: 1000.0
Reader-B reads  balance: 1000.0
Withdrawer withdraws: 200.0
Depositor deposits: 300.0

Process finished with exit code 0
```

(3 marks)

c. What is the advantage of using `ReentrantReadWriteLock` over `synchronized` method in this program?

- the advantage of using ReentrantReadWriteLock over synchronized method in this program which is ReentrantReadWriteLock allows multiple threads to read the balance at the same time but for synchronized method is only one thread can access at a time even for reading, so using ReentrantReadWriteLock makes the program faster and more efficient.

(2 marks)

d. Explain the difference between `readLock()` and `writeLock()`.

- the difference between readLock() and writeLock() is readlock used when we only want to read the data thus it can many thread use at the same time, but for writelock is used when we want to change the data like deposit and withdraw above and only one thread can use at a time.

(3 marks)

e. Why is `writeLock.unlock()` placed in a `finally` block?

- writeLock.unlock() placed in a finally block is to make sure the lock is always released even have problem. Like example if forget to unlock it other threads cannot continue and it will get stuck. That's why finally is like a safety step to always unlock the lock no matter what happens.

(2 marks)

Submission:

Platform: 1. Online Learning - Sample Coding & Output in PdF form
2. GitHub – Upload the coding file to your GitHub account.

Date: 21 May 2025 (Wednesday, before 23.59 pm)