

#### Task 4: Synchronized Blocks and Methods

Write a program that simulates a bank account being accessed by multiple threads to perform deposits and withdrawals using synchronized methods to prevent race conditions.

```
package Day_18;

class BankAccount {
    private double balance;

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

    // Synchronized method for depositing money into the account
    public synchronized void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: " + amount + ", New Balance: " + balance);
    }

    // Synchronized method for withdrawing money from the account
    public synchronized void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + ", New Balance: " + balance);
        } else {
            System.out.println("Insufficient funds for withdrawal: " + amount);
        }
    }

    // Method to get the current balance
    public synchronized double getBalance() {
        return balance;
    }
}

public class Task4 {
    public static void main(String[] args) {
        BankAccount account = new BankAccount(1000);

        // Create and start multiple threads for depositing and withdrawing
        Thread[] threads = new Thread[10];
        for (int i = 0; i < threads.length; i++) {
            threads[i] = new Thread(() -> {
                // Simulate random deposits and withdrawals
                for (int j = 0; j < 5; j++) {
                    double amount = Math.random() * 500;
                    if (Math.random() < 0.5) {
                        account.deposit(amount);
                    } else {
                        account.withdraw(amount);
                    }
                }
                try {
                    Thread.sleep((long) (Math.random() * 100));
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                }
            });
        }
    }
}
```

```

    });
    threads[i].start();
}

// Wait for all threads to finish
for (Thread thread : threads) {
    try {
        thread.join();
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
    }
}

// Print the final balance
System.out.println("Final Balance: " + account.getBalance());
}
}

```

The screenshot shows the Eclipse IDE with a Java project named 'Day\_18'. The main editor displays the source code for the 'BankAccount' class. The class has a private 'balance' field, a constructor, and three public methods: 'deposit', 'withdraw', and 'getBalance'. The 'deposit' and 'withdraw' methods are synchronized. The 'Console' view at the bottom shows the output of the program, which includes several deposit and withdrawal transactions, followed by the final balance.

```

1 package Day_18;
2
3 class BankAccount {
4     private double balance;
5
6     public BankAccount(double initialBalance) {
7         this.balance = initialBalance;
8     }
9
10    // Synchronized method for depositing money into the account
11    public synchronized void deposit(double amount) {
12        balance += amount;
13        System.out.println("Deposited: " + amount + ", New Balance: " + balance);
14    }
15
16    // Synchronized method for withdrawing money from the account
17    public synchronized void withdraw(double amount) {
18        if (balance >= amount) {
19            balance -= amount;
20            System.out.println("Withdrawn: " + amount + ", New Balance: " + balance);
21        } else {
22            System.out.println("Insufficient funds for withdrawal: " + amount);
23        }
24    }
25
26    // Method to get the current balance
27    public synchronized double getBalance() {
28        return balance;
29    }
30 }

```

Console Output:

```

<terminated> Task4 [Java Application] C:\Users\Nikita\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149\jre\bin\javaw.exe (Jun 4, 2024, 5:43
Deposited: 37.145875640596294, New Balance: 2682.22908277302
Withdrawn: 39.376103359808624, New Balance: 2642.8529794132114
Deposited: 181.93903993578115, New Balance: 2824.7920193489927
Withdrawn: 265.5409006831104, New Balance: 2559.251118665882
Deposited: 426.2137589302638, New Balance: 2985.464877596146
Withdrawn: 334.80828129845855, New Balance: 2650.6565962976874
Withdrawn: 98.49323313690145, New Balance: 2552.163363160786
Final Balance: 2552.163363160786

```