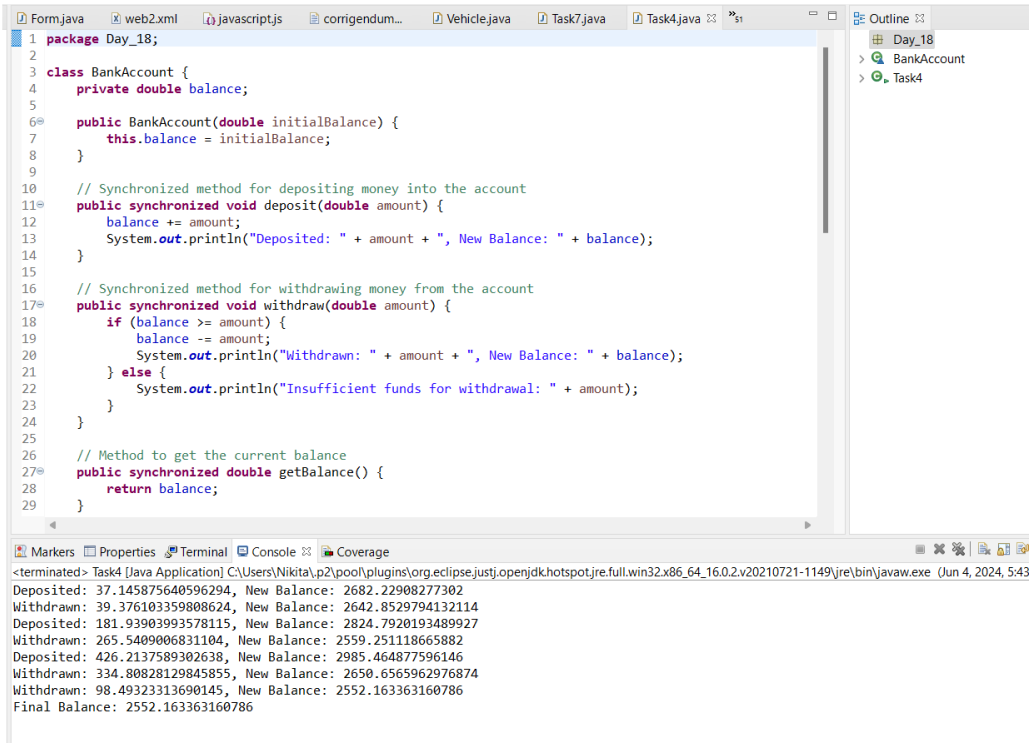
Task 4: Synchronized Blocks and Methods

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.

```java
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            System.out.println("Withdrawn: " + amount + ", New Balance: " + balance);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());
    }
}
```



```
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;
    }
```