**Shem Thuo SCT212-0529/2022**

**BIT 2203 Advanced Programming**

**Assignment 1; Quiz 3**

1. **Create the** `InsufficientFundsException` **Class:** Define a custom exception class to handle insufficient funds during transactions.

   java

```
public class InsufficientFundsException extends Exception {
    public InsufficientFundsException(String message) {
        super(message);
    }
}
```

2. **Modify the** `WithdrawalTransaction` **Class:** Implement an overloaded `apply()` method that checks the balance and handles insufficient funds.

   java

```
import java.util.Calendar;

public class WithdrawalTransaction extends BaseTransaction {
    private boolean reversed = false;
    private double amountNotWithdrawn;

    public WithdrawalTransaction(double amount, Calendar date, String
transactionID) {
        super(amount, date, transactionID);
    }

    @Override
    public void apply(BankAccount ba) throws InsufficientFundsException
{
        if (amount > ba.getBalance()) {
            throw new InsufficientFundsException("Insufficient funds
for withdrawal");
        }
        ba.withdraw(amount);
        System.out.println("Withdrew: " + amount);
    }

    // Overloaded apply method
    public void apply(BankAccount ba, boolean checkBalance) {
        try {
            if (checkBalance && ba.getBalance() < amount) {
                if (ba.getBalance() > 0 && ba.getBalance() < amount) {
                    amountNotWithdrawn = amount - ba.getBalance();
                    ba.withdraw(ba.getBalance());
                    System.out.println("Partially withdrew: " +
amount);
```

```
                } else {
                    throw new InsufficientFundsException("Insufficient
funds for withdrawal");
                }
            } else {
                ba.withdraw(amount);
                System.out.println("Withdrew: " + amount);
            }
        } catch (InsufficientFundsException e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            System.out.println("Completed the apply method.");
        }
    }

    public boolean reverse(BankAccount ba) {
        if (reversed) {
            System.out.println("Transaction already reversed.");
            return false;
        }
        try {
            ba.deposit(amount);
            reversed = true;
            System.out.println("Withdrawal reversed: " + amount);
            return true;
        } catch (Exception e) {
            System.out.println("Failed to reverse withdrawal: " +
e.getMessage());
            return false;
        }
    }
}
```

3. **Update the** `BankAccount` **Class:** Ensure it has methods for deposits and withdrawals with proper exception handling.

   java

```
public class BankAccount {
    private double balance;

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

    public void deposit(double amount) {
        balance += amount;
    }

    public void withdraw(double amount) throws
InsufficientFundsException {
        if (amount > balance) {
            throw new InsufficientFundsException("Insufficient funds
for withdrawal");
        }
        balance -= amount;
    }

    public double getBalance() {
        return balance;
    }
}
```

4. **Client Code:** Demonstrate the functionality, including handling the insufficient funds scenario.

   java

```
import java.util.Calendar;

public class Main {
    public static void main(String[] args) {
        BankAccount account = new BankAccount(1000);
        Calendar date = Calendar.getInstance();

        DepositTransaction deposit = new DepositTransaction(200, date,
"TXN001");
        WithdrawalTransaction withdrawal = new
WithdrawalTransaction(1500, date, "TXN002");

        try {
            deposit.apply(account);
            System.out.println("Balance after deposit: " +
account.getBalance());

            withdrawal.apply(account);
            System.out.println("Balance after withdrawal: " +
account.getBalance());
        } catch (InsufficientFundsException e) {
            System.out.println("Exception: " + e.getMessage());
        }

        System.out.println("Final Balance: " + account.getBalance());
    }
}
```