Project Four
Object-Oriented Programming – Bank Account System
GitHub: *https://github.com/SereneAura2/BankAccountJavaProject*
Objectives of this Lab:

- Understand and demonstrate writing Java classes
- Instantiate and manipulate objects of custom classes
- Apply object-oriented concepts like constructors, methods, and encapsulation
- Use exception handling and static variables
- Demonstrate inter-class communication (via a `Financial` helper class)

Project Name: Project 4 – Bank Account System
Group Members: Ashton Pleickhardt, Rhamseys Garcia
Project Description:
Develop a bank account management system using three Java classes: `BankAccount.java`, `BankAccountDemo.java`, and `Financial.java`.
Date: 05/23/2025

Task	Completed By	Communication	Notes
Create Task Division Table	Ashton Pleickhardt	N/A	Created on 05/27/2025
Write `BankAccount.java`	Ashton Pleickhardt	GitHub/Email	Includes constructors, methods, and exception handling
Write `BankAccountDemo.java`	Rhamseys Garcia	GitHub/Email	Handles testing and outputs with method demonstrations
Write `Financial.java`	Rhamseys Garcia	GitHub/Email	Contains static method `percentOf()`
Add Javadoc Comments	Ashton Pleickhardt	Internal Review	Applied to all public methods

Implement `addInterest()` Logic	Ashton Pleickhardt	GitHub/Email	Uses `Financial.percentOf()` and checks account type
Test `withdraw` and `transfer`	Rhamseys Garcia	Slack	Tested with insufficient funds - used try/catch
Capture Screenshots for Word Doc	Ashton Pleickhardt	N/A	Execution + exception snapshots
Format Word Document for Submission	Ashton Pleickhardt	N/A	Task table, snapshots, and backup code included

Execution Snapshot

```
mac@macbooks-MacBook-Pro ProjectFour %  java BankAccountDemo.java
Initial Account Information:
Account Information:
Name: Unknown
Account Number: 1
Account Type: Checking
Balance: $0.00
Interest Rate: 3.0%
-------------------------
Account Information:
Name: Sam
Account Number: 2
Account Type: Savings
Balance: $1000.00
Interest Rate: 3.0%
-------------------------
Account Information:
Name: Leila
Account Number: 3
Account Type: Savings
Balance: $2000.00
Interest Rate: 3.0%
-------------------------

Testing deposit method:
Account Information:
Name: Unknown
Account Number: 1
Account Type: Checking
Balance: $500.00
Interest Rate: 3.0%
-------------------------

Testing withdraw method:
Account Information:
Name: Unknown
Account Number: 1
Account Type: Checking
Balance: $300.00
Interest Rate: 3.0%
-------------------------

Testing transfer method:
After transfer:
Account Information:
Name: Sam
Account Number: 2
Account Type: Savings
Balance: $700.00
Interest Rate: 3.0%
-------------------------
Account Information:
Name: Leila
Account Number: 3
Account Type: Savings
Balance: $2300.00
Interest Rate: 3.0%
-------------------------

Testing addInterest method:
Before adding interest:
Account Information:
Name: Sam
Account Number: 2
Account Type: Savings
Balance: $700.00
Interest Rate: 3.0%
-------------------------
Account Information:
Name: Leila
Account Number: 3
Account Type: Savings
Balance: $2300.00
Interest Rate: 3.0%
-------------------------

After adding interest:
Account Information:
Name: Sam
Account Number: 2
Account Type: Savings
Balance: $721.00
Interest Rate: 3.0%
-------------------------
Account Information:
Name: Leila
Account Number: 3
Account Type: Savings
Balance: $2369.00
Interest Rate: 3.0%
-------------------------

Testing exception handling for insufficient funds:
Caught exception: insufficient funds
Caught exception: insufficient funds

Testing getter and setter methods:
Updated account information:
Account Information:
Name: Adam Smith
Account Number: 1
Account Type: Checking
Balance: $300.00
Interest Rate: 4.0%
-------------------------
```

Backup Code

```
        try {
            AdamsAcc.withdraw(1000.0); // This should throw an exception
        } catch (IllegalArgumentException e) {
            System.out.println("Caught exception: " + e.getMessage());
        }

        try {
            SamsAcc.transfer(LeilasAcc, 2000.0); // This should throw an exception
        } catch (IllegalArgumentException e) {
            System.out.println("Caught exception: " + e.getMessage());
        }
```

```
    public void withdraw(double amount) {
        if (amount > balance) {
            throw new IllegalArgumentException("insufficient funds");
        }
        balance -= amount;
    }
```

```
    public void transfer(BankAccount otherAccount, double amount) {
        if (amount > balance) {
            throw new IllegalArgumentException("insufficient funds");
        }
        this.withdraw(amount);
        otherAccount.deposit(amount);
    }
```