# COMP 1406Z – Fall 2023
# Introduction to Computer Science II
# Midterm – Wednesday, November 22nd 2023

## Midterm Details

The midterm will be submitted through Brightspace. The midterm materials contained in the 1406-midterm-materials.zip include an IntelliJ code project and a text file for short answer questions. To submit your midterm solution, create a single .zip file with all your midterm materials (IntelliJ project and short answer file) and submit it to the '1406 Midterm' assignment submission on Brightspace. The official end time for the midterm is 2:30pm (Eastern Time) but you will have until 2:45pm (Eastern Time) to submit your midterm. **Do not wait until the last minute – no late submissions will be accepted**. After submitting, you should download your own zip file, extract its contents, and ensure the files have the correct content. You are not allowed to consult other students, TAs, or anybody else while completing the midterm. You may use any IDE or text editor you generally use to write your code but must submit a valid IntelliJ project. You can execute your programs as many times as you want to debug and fix your code.

## Problem 1 (20 marks)

**Write your answers for this problem in the shortanswer.txt file**. Explain each of the following terms in your own words: abstraction, encapsulation, inheritance, polymorphism. Discuss the benefits/drawbacks each can provide when developing OOP software, especially in larger, longer-running projects involving multiple developers. You should include comments regarding how each could affect the maintainability, extensibility, and robustness of the software. You should also include arguments or examples based on your own experience within 1405/1406.

## Problem 2 (30 marks)

**Write the code for this part in the Bank.java class within the provided IntelliJ project**. The code included within the provided IntelliJ project defines three types of bank accounts: BankAccount, SavingsAccount, ChequingAccount. You should look through these classes to familiarize yourself with their attributes/methods and class hierarchy. You must implement the Bank class within this project to follow the specifications below. You cannot modify any of the provided BankAccount, SavingsAccount, or ChequingAccount classes.

A Bank will have a name (String) and store an array of savings account and chequing account objects (you must use an array, not a list). The array must have a fixed size of

10, though the size of the array should be easy to change later if necessary. The Bank class must support the following constructors/methods.

public Bank(String name):
A constructor for the bank. The String argument will specify the name of the bank. No bank account objects will be stored in the bank initially.

public String toString():
Returns a string representation of the bank. For the purposes of this midterm, the string representation of a bank should include the bank's name and the total balance of all accounts the bank currently has (i.e., the sum of the balance for all accounts).

public boolean addAccount(BankAccount account):
Adds the given BankAccount object to the bank, if adding that account would be valid. This method must also update the institution attribute of the given BankAccount to the calling Bank object. It should not be possible to add two accounts with the same account number into the bank. It should not be possible to add an account if the array is full. The method returns true to indicate the account was added successfully and false otherwise.

public double getLargestBalance():
Returns the largest balance of any account within the calling bank.

public void performMonthlyUpkeep():
Must perform monthly upkeep for each account in the bank by calling the monthlyUpkeep() method for each account.

public boolean transfer(int source, int destination, double amount):
Withdraws the specified amount from the account at index source in the accounts array, deposits the amount into the account at index dest, and returns true. If there is not enough money in the source account to make the withdrawal, the amount given is negative, the source and destination indices are identical, or either of the specified account indices are not valid (i.e., there is no account at that index), the transfer should not take place and false should be returned.

public static void main(String[] args):
Provide testing code for your Bank methods within this method.