# COMS-280 Final Project

## Challenges and Solutions in the Class Hierarchy and Polymorphic Behavior

1. Handling polymorphism in AccountManager:

Challenge: The AccountManager class is templated, but it is instantiated with BankAccount, which is an abstract base class. This results in difficulties when dealing with polymorphic behavior while storing SavingAccount and CheckingAccount objects together.

Solution: Instead of templating AccountManager<T>, store unique\_ptr<BankAccount> directly. This allows different derived types (SavingsAccount and CheckingAccount) to coexist in a single AccountManager instance.

1. Virtual Destructor in BankAccount:

Challenge: BankAccount has virtual functions but lacks a proper virtual destructor, which may lead to memory leaks if objects are deleted through a BankAccount\* pointer.

Solutions: Ensure BankAccount has a virtual destructor (virtual ~BankAccount() = default😉 to propertly clean up derived objects.

1. Exception Handling for withdrawals:

Challenge: The withdrawal operation can throw exceptions due to insufficient funds or overdraft limits. If not handled properly, it may crash the program.

Solution: A try-catch block is already implemented in performBankingOperations, but it should also validate user input before attempting a transaction to avoid unnecessary exceptions.

1. Avoiding code duplication in SavingsAccount and CheckingAccount:

Challenge: Both SavingsAccount and CheckingAccount define the same formatAmount() function.

Solution: Move formatAmount() into the base class BanAccount so it can be reused by all derived classes.

1. Accessing BankAccount methods in AccountManager<T>:

Challenge: getAccount() in AccountManager<T> attempts to return a pointer of type T\*. Since AccountManager is instantiated with BankAccount, the function should return a BankAccount\*, allowing polymorphic behavior.

Solution: Modify AccountManager to explicity store unique\_ptr<BankAccount> BankAccount\* from getAccount().