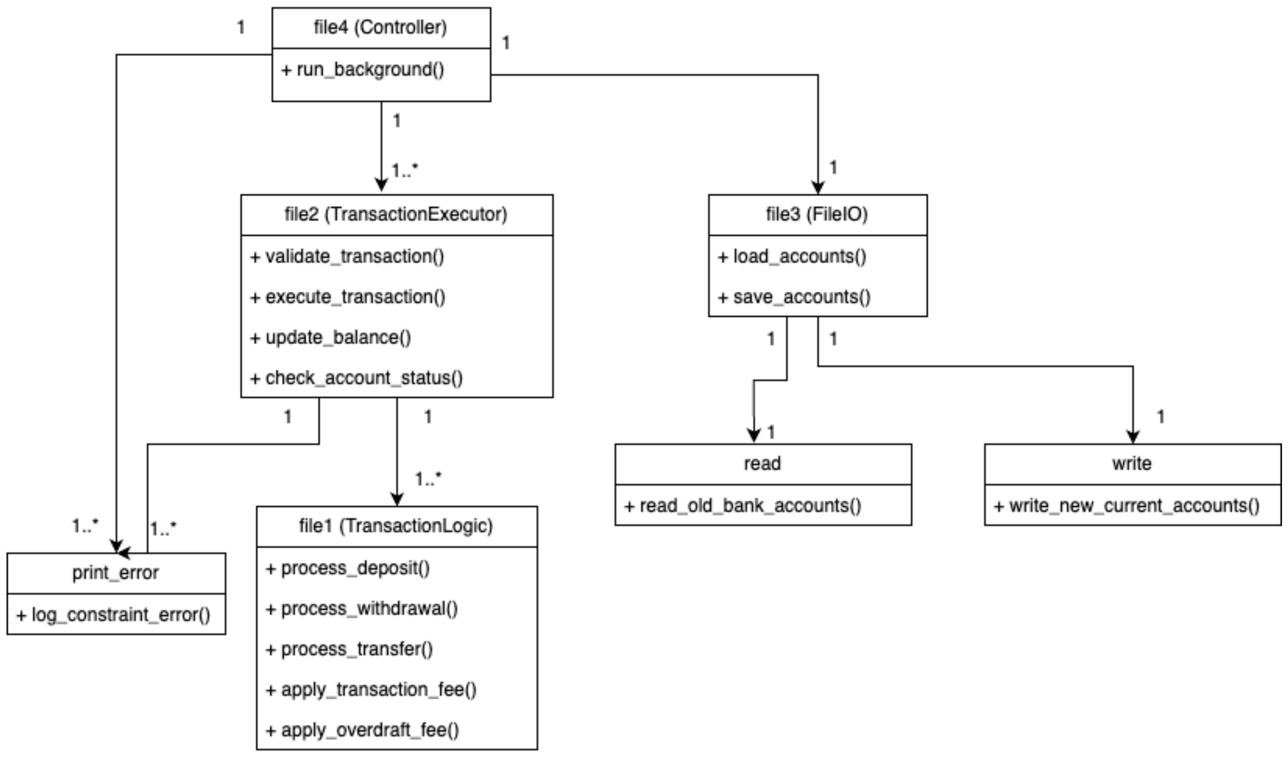
Backend Design Document

UML Diagram



Class/Module Descriptions

Module/File	Description
print_error.py	Provides a standardized way to log errors with specific constraint types.
file1.py (TransactionLogic)	Handles the core logic for processing different types of bank transactions.
file2.py (TransactionExecutor)	Validates and executes transactions using logic from file1.
file3.py (FileIO)	Handles file input/output for account data using read/write utilities.
file4.py (Controller)	Acts as the main backend controller to coordinate reading, processing, and saving.
read.py	Parses and validates old-format account records from an input file.
write.py	Validates and writes updated account records to a new file format.

Class & Method Descriptions

Module/File	Function Name	Description
print_error.py	log_constraint_error	Prints standardized error messages with type and description.
file1.py	process_deposit	Processes deposit if amount is valid.
file1.py	process_withdrawal	Processes withdrawal if sufficient balance.
file1.py	process_transfer	Transfers money from one account to another if valid.
file1.py	apply_transaction_fee	Applies a fee to a transaction based on type.
file1.py	apply_overdraft_fee	Deducts overdraft fee if account goes negative.
file2.py	validate_transaction	Validates if a transaction is permissible on a given account.
file2.py	execute_transaction	Orchestrates the correct transaction logic depending on the type.
file2.py	update_balance	Updates account balance and increments transaction counter.
file2.py	check_account_status	Verifies that account is active.
file3.py	load_accounts	Loads account data from input file.
file3.py	save_accounts	Saves modified accounts to output file.
file4.py	run_backend	Main controller that reads accounts, processes transactions, writes output.
read.py	read_old_bank_accounts	Parses and validates account data from old input format.
write.py	write_new_current_accounts	Outputs current account data in required format, ensuring validation.