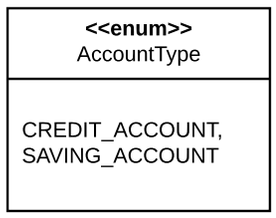
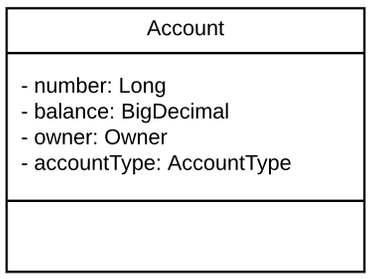
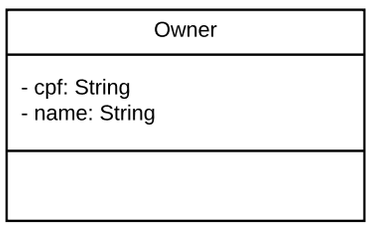
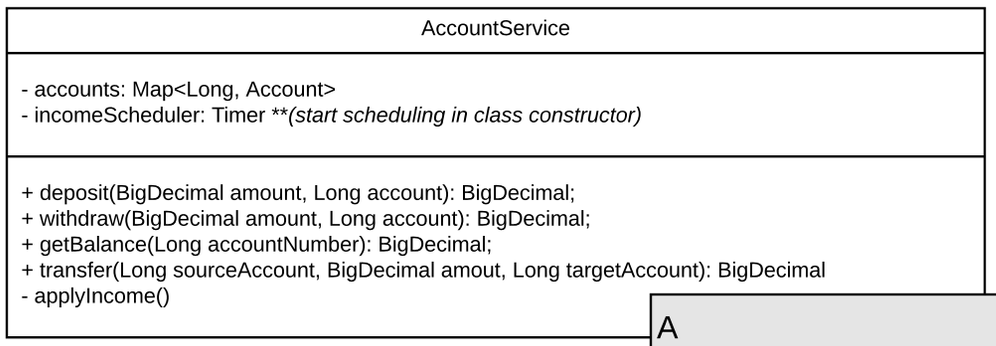
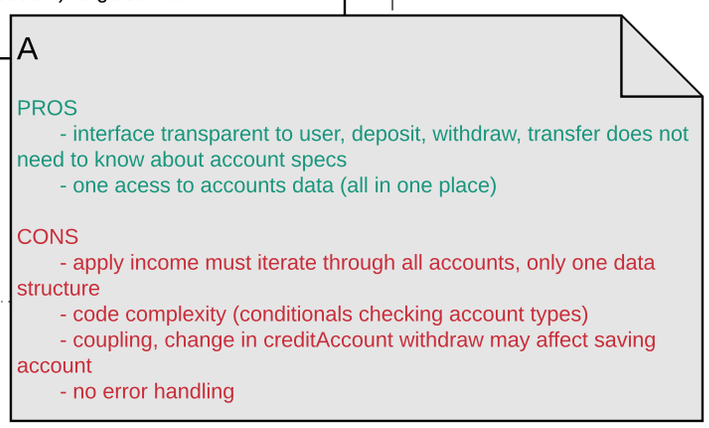
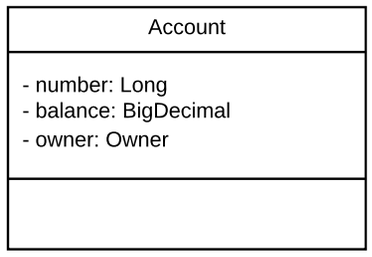

<<enum>>
AccountType
CREDIT_ACCOUNT,
SAVING_ACCOUNT

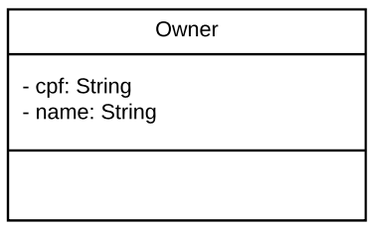

Account
- number: Long
- balance: BigDecimal
- owner: Owner
- accountType: AccountType

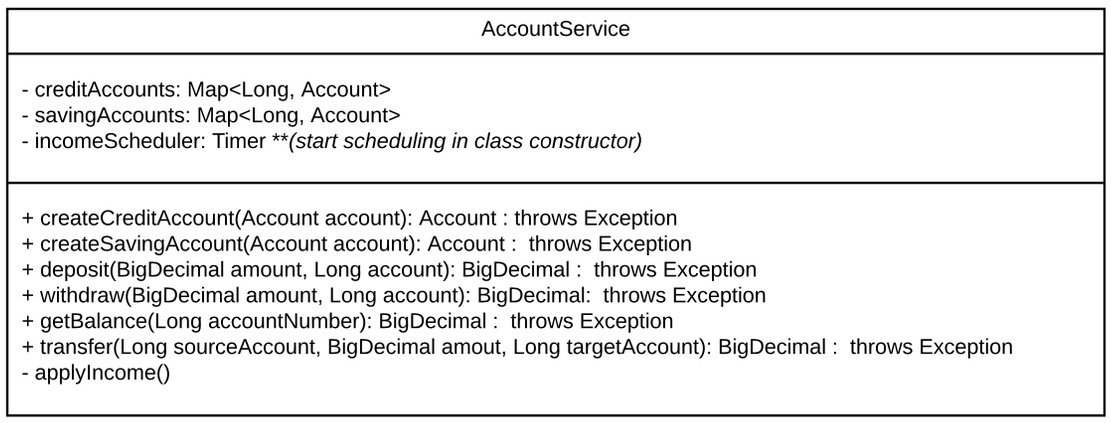

Owner
- cpf: String
- name: String

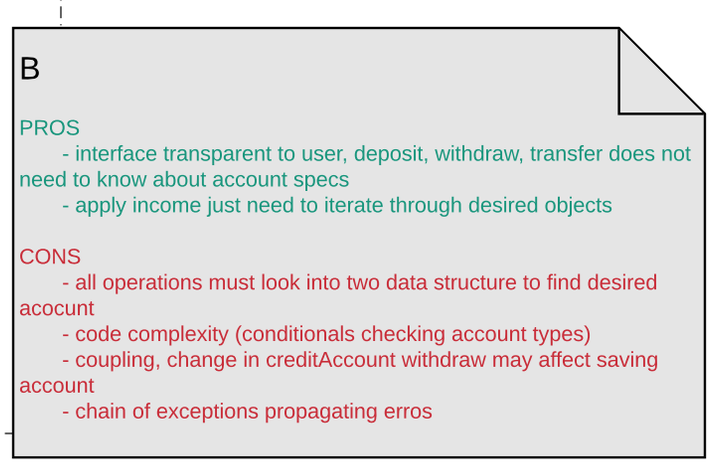

AccountService
- accounts: Map<Long, Account>
- incomeScheduler: Timer **(start scheduling in class constructor)
+ deposit(BigDecimal amount, Long account): BigDecimal;
+ withdraw(BigDecimal amount, Long account): BigDecimal;
+ getBalance(Long accountNumber): BigDecimal;
+ transfer(Long sourceAccount, BigDecimal amout, Long targetAccount): BigDecimal
- applyIncome()

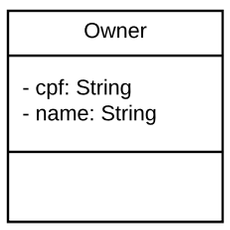

A
PROS
- interface transparent to user, deposit, withdraw, transfer does not need to know about account specs
- one acess to accounts data (all in one place)
CONS
- apply income must iterate through all accounts, only one data structure
- code complexity (conditionals checking account types)
- coupling, change in creditAccount withdraw may affect saving account
- no error handling

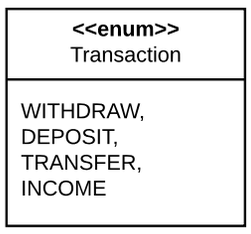

Account
- number: Long
- balance: BigDecimal
- owner: Owner

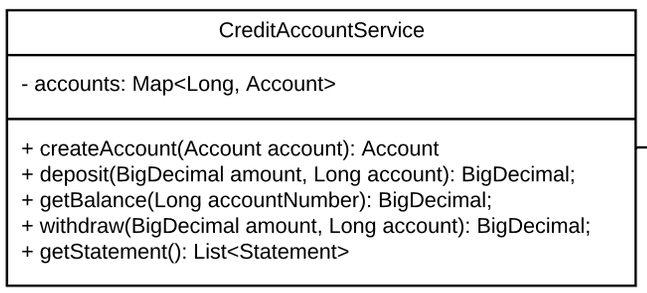

Owner
- cpf: String
- name: String

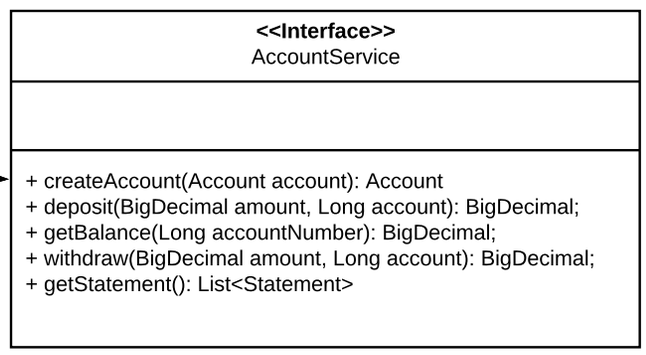

AccountService
- creditAccounts: Map<Long, Account>
- savingAccounts: Map<Long, Account>
- incomeScheduler: Timer **(start scheduling in class constructor)
+ createCreditAccount(Account account): Account : throws Exception
+ createSavingAccount(Account account): Account : throws Exception
+ deposit(BigDecimal amount, Long account): BigDecimal : throws Exception
+ withdraw(BigDecimal amount, Long account): BigDecimal: throws Exception
+ getBalance(Long accountNumber): BigDecimal : throws Exception
+ transfer(Long sourceAccount, BigDecimal amout, Long targetAccount): BigDecimal : throws Exception
- applyIncome()

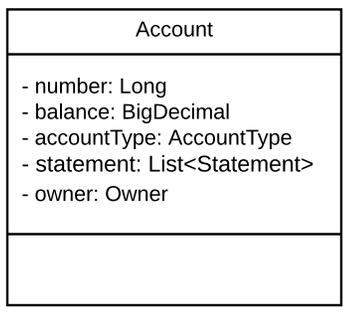

B
PROS
- interface transparent to user, deposit, withdraw, transfer does not need to know about account specs
- apply income just need to iterate through desired objects
CONS
- all operations must look into two data structure to find desired acocunt
- code complexity (conditionals checking account types)
- coupling, change in creditAccount withdraw may affect saving account
- chain of exceptions propagating erros

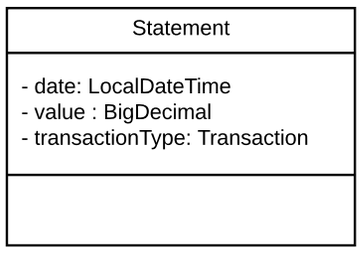

Owner
- cpf: String
- name: String

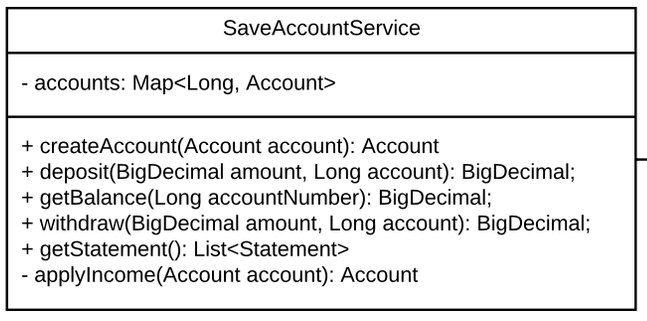

<<enum>>
Transaction
WITHDRAW,
DEPOSIT,
TRANSFER,
INCOME

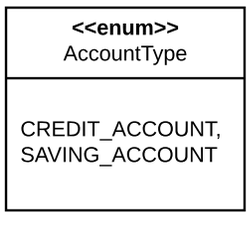

CreditAccountService
- accounts: Map<Long, Account>
+ createAccount(Account account): Account
+ deposit(BigDecimal amount, Long account): BigDecimal;
+ getBalance(Long accountNumber): BigDecimal;
+ withdraw(BigDecimal amount, Long account): BigDecimal;
+ getStatement(): List<Statement>

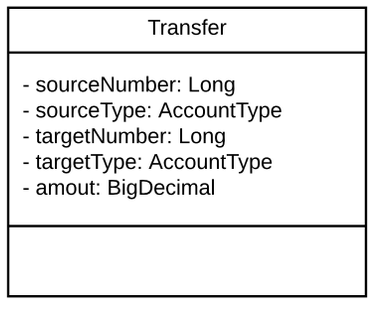

<<Interface>>
AccountService
+ createAccount(Account account): Account
+ deposit(BigDecimal amount, Long account): BigDecimal;
+ getBalance(Long accountNumber): BigDecimal;
+ withdraw(BigDecimal amount, Long account): BigDecimal;
+ getStatement(): List<Statement>

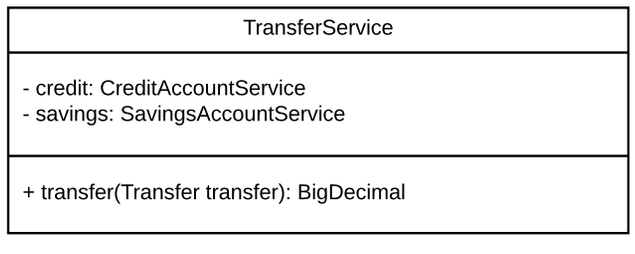

Account
- number: Long
- balance: BigDecimal
- accountType: AccountType
- statement: List<Statement>
- owner: Owner

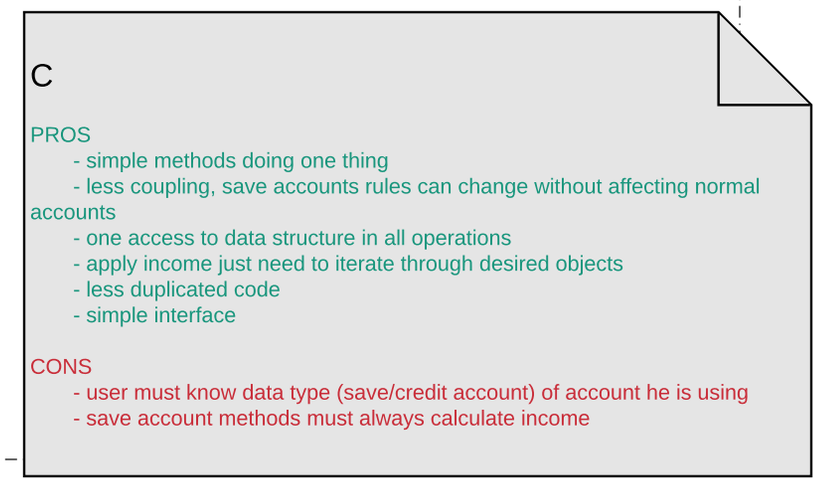

Statement
- date: LocalDateTime
- value : BigDecimal
- transactionType: Transaction


SaveAccountService
- accounts: Map<Long, Account>
+ createAccount(Account account): Account
+ deposit(BigDecimal amount, Long account): BigDecimal;
+ getBalance(Long accountNumber): BigDecimal;
+ withdraw(BigDecimal amount, Long account): BigDecimal;
+ getStatement(): List<Statement>
- applyIncome(Account account): Account


<<enum>>
AccountType
CREDIT_ACCOUNT,
SAVING_ACCOUNT


Transfer
- sourceNumber: Long
- sourceType: AccountType
- targetNumber: Long
- targetType: AccountType
- amout: BigDecimal


TransferService
- credit: CreditAccountService
- savings: SavingsAccountService
+ transfer(Transfer transfer): BigDecimal


C
PROS
- simple methods doing one thing
- less coupling, save accounts rules can change without affecting normal accounts
- one access to data structure in all operations
- apply income just need to iterate through desired objects
- less duplicated code
- simple interface
CONS
- user must know data type (save/credit account) of account he is using
- save account methods must always calculate income