Package machine

ATM
- cash5: int - cash10: int - cash20: int - cash50: int - total: int + date: Date - CADtoUSDRate: double + format: SimpleDateFormat - ATM: ATM
- ATM() + getATM(): ATM + getCADtoUSDRate(): double + checkATMBalance(): boolean + displayAlert(): String - writeFile(): void - withdrawCash(money: int): int[] + withdraw(money: int): void + deposit(cashFive: int, cashTen: int, cashTwenty: int, cashFifty: int): void ~ loadFile(): void

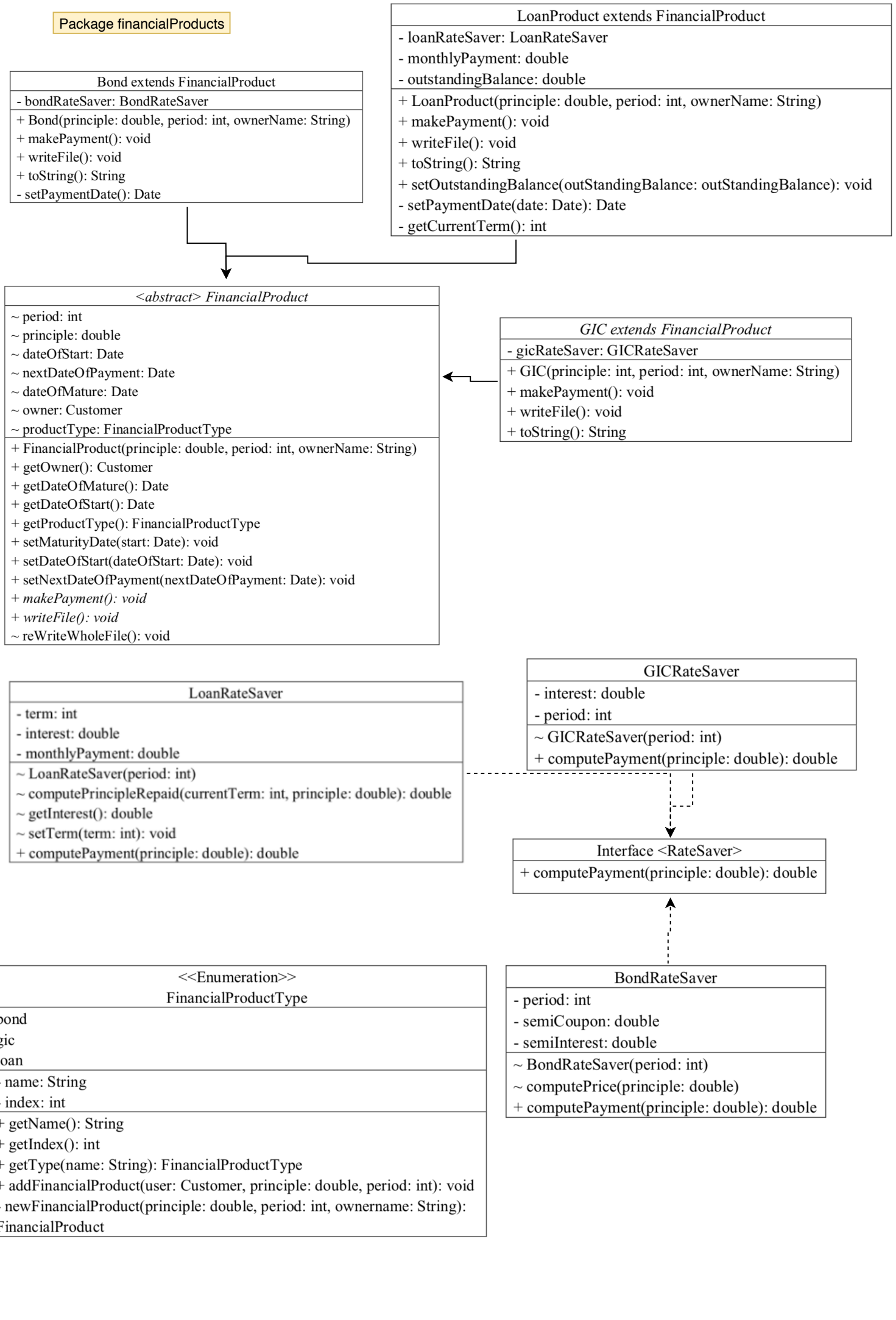
Users
- manager: Manager - customers: List<Customer> - financialAdvisers: List<FinancialAdviser> - financialProducts: List<FinancialProduct> - Users: Users
+ getManager(): Manager + setManager(manager: Manager): void - Users() + getUsers(): Users - constructManager(): void - constructUser(fileContent: ArrayList<String>): Customer - constructFinancialUser(fileContent: ArrayList<String>): FinancialAdviser.FinancialUser - constructHelper(fileContent: ArrayList<String>, customer: Customer): Customer - addCustomer(customer: Customer): void - addFinancialAdviser(financialAdviser: FinancialAdviser): void - addFinancialProduct(financialProduct: FinancialProduct): void - loadCustomer(): void - loadFinancialAdviser(): void - loadFinancialProduct(filename: String): void - loadGICProduct(principle: double, period: int, ownerName: String, startDate: Date): void - loadBondProduct(principle: double, period: int, ownerName: String, startDate: Date, paymentDate: Date): void - loadLoanProduct(principle: double, period: int, ownerName: String, startDate: Date, paymentDate: Date, outstandingBalance: double): void + customerFinder(username: String): Customer + financialAdviserFinder(username: String): FinancialAdviser + getCustomers(): ArrayList<Customer> + getFinancialAdvisers(): ArrayList<FinancialAdviser> + getFinancialProducts(): List<FinancialProduct> ~ loadFile(): void

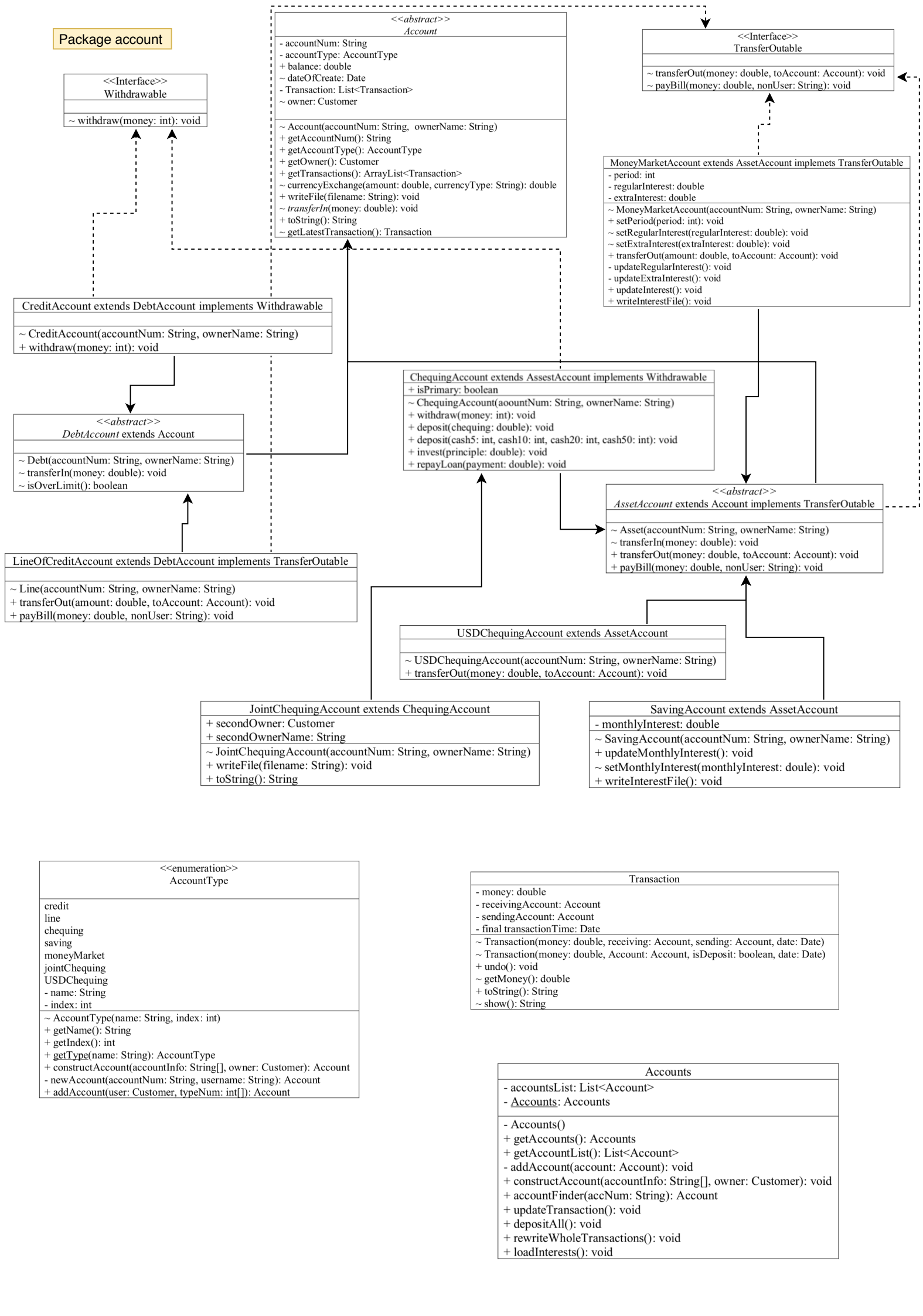
Simulator
- simulator: Simulator
- Simulator() + getSimulator(): Simulator + constructManager(username: String, password: String): void + setSystemDate(): void + initialize(): void + loginPass(username: String, password: String): Person + LoginPassHelper(person: Person, password: String): Person + updateDate(): void - recordInterests(): void

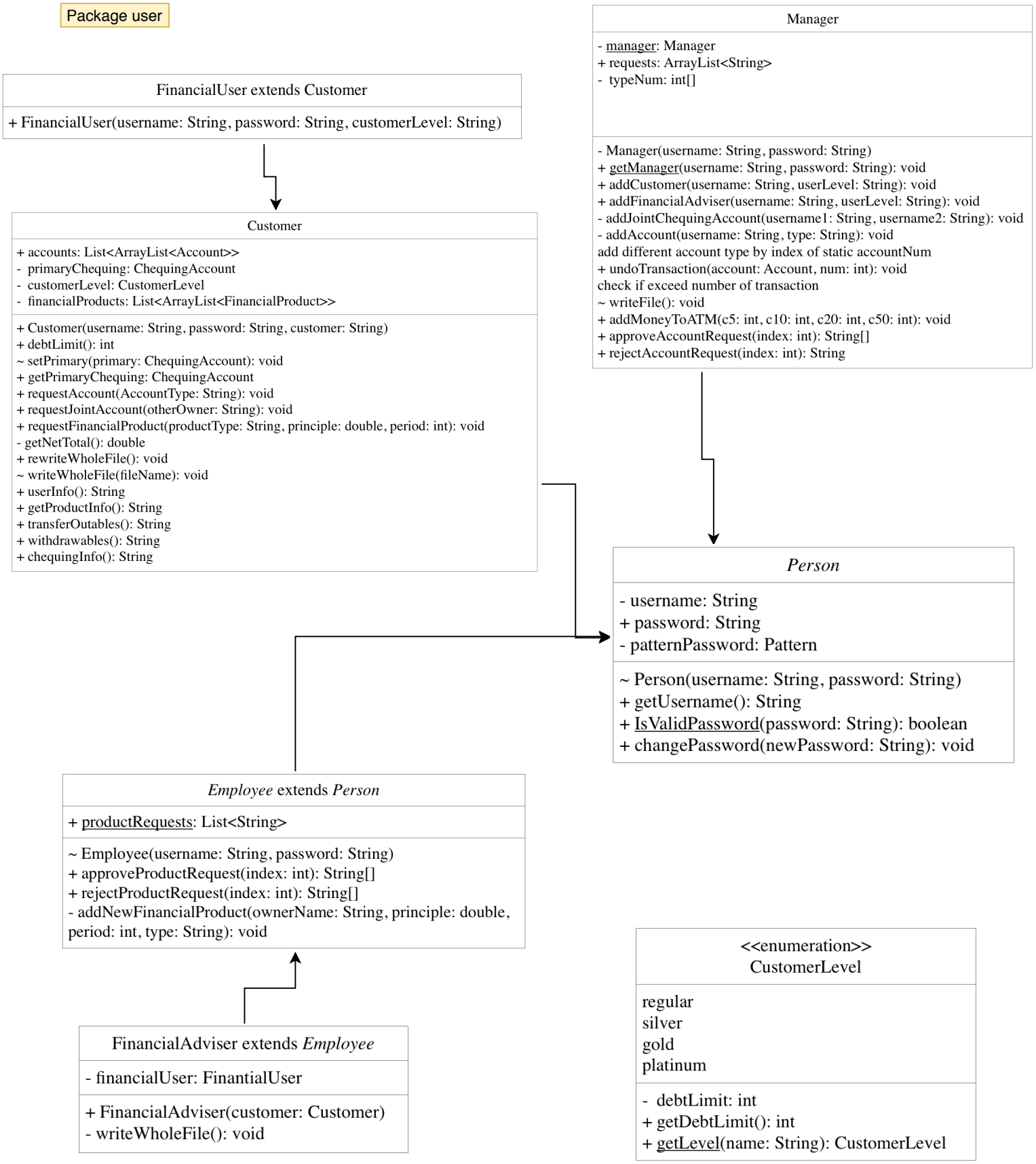
Package manageFile

ReadFile
+ readFile (filename: String): ArrayList<String> + readFolder (folderName: String): ArrayList<ArrayList<String>>

WriteFile
+ write(content: String, filename: String): void + clearInfo(fileName: String): void


Package financialProducts
LoanProduct extends FinancialProduct
- loanRateSaver: LoanRateSaver
- monthlyPayment: double
- outstandingBalance: double
+ LoanProduct(principle: double, period: int, ownerName: String)
+ makePayment(): void
+ writeFile(): void
+ toString(): String
+ setOutstandingBalance(outStandingBalance: outStandingBalance): void
- setPaymentDate(date: Date): Date
- getCurrentTerm(): int
Bond extends FinancialProduct
- bondRateSaver: BondRateSaver
+ Bond(principle: double, period: int, ownerName: String)
+ makePayment(): void
+ writeFile(): void
+ toString(): String
- setPaymentDate(): Date
<abstract> FinancialProduct
~ period: int
~ principle: double
~ dateOfStart: Date
~ nextDateOfPayment: Date
~ dateOfMature: Date
~ owner: Customer
~ productType: FinancialProductType
+ FinancialProduct(principle: double, period: int, ownerName: String)
+ getOwner(): Customer
+ getDateOfMature(): Date
+ getDateOfStart(): Date
+ getProductType(): FinancialProductType
+ setMaturityDate(start: Date): void
+ setDateOfStart(dateOfStart: Date): void
+ setNextDateOfPayment(nextDateOfPayment: Date): void
+ makePayment(): void
+ writeFile(): void
~ reWriteWholeFile(): void
GIC extends FinancialProduct
- gicRateSaver: GICRateSaver
+ GIC(principle: int, period: int, ownerName: String)
+ makePayment(): void
+ writeFile(): void
+ toString(): String
LoanRateSaver
- term: int
- interest: double
- monthlyPayment: double
~ LoanRateSaver(period: int)
~ computePrincipleRepaid(currentTerm: int, principle: double): double
~ getInterest(): double
~ setTerm(term: int): void
+ computePayment(principle: double): double
GICRateSaver
- interest: double
- period: int
~ GICRateSaver(period: int)
+ computePayment(principle: double): double
Interface <RateSaver>
+ computePayment(principle: double): double
<<Enumeration>>
FinancialProductType
bond
gic
loan
- name: String
- index: int
+ getName(): String
+ getIndex(): int
+ getType(name: String): FinancialProductType
+ addFinancialProduct(user: Customer, principle: double, period: int): void
- newFinancialProduct(principle: double, period: int, ownername: String): FinancialProduct
BondRateSaver
- period: int
- semiCoupon: double
- semiInterest: double
~ BondRateSaver(period: int)
~ computePrice(principle: double)
+ computePayment(principle: double): double


Package account
<<Interface>>
Withdrawable
~ withdraw(money: int): void
<<abstract>>
Account
- accountNum: String
- accountType: AccountType
+ balance: double
~ dateOfCreate: Date
- Transaction: List<Transaction>
~ owner: Customer
~ Account(accountNum: String, ownerName: String)
+ getAccountNum(): String
+ getAccountType(): AccountType
+ getOwner(): Customer
+ getTransactions(): ArrayList<Transaction>
~ currencyExchange(amount: double, currencyType: String): double
+ writeFile(filename: String): void
~ transferIn(money: double): void
+ toString(): String
~ getLatestTransaction(): Transaction
<<Interface>>
TransferOutable
~ transferOut(money: double, toAccount: Account): void
~ payBill(money: double, nonUser: String): void
MoneyMarketAccount extends AssetAccount implemets TransferOutable
- period: int
- regularInterest: double
- extraInterest: double
~ MoneyMarketAccount(accountNum: String, ownerName: String)
+ setPeriod(period: int): void
~ setRegularInterest(regularInterest: double): void
~ setExtraInterest(extraInterest: double): void
+ transferOut(amount: double, toAccount: Account): void
- updateRegularInterest(): void
- updateExtraInterest(): void
+ updateInterest(): void
+ writeInterestFile(): void
CreditAccount extends DebtAccount implements Withdrawable
~ CreditAccount(accountNum: String, ownerName: String)
+ withdraw(money: int): void
ChequingAccount extends AssestAccount implements Withdrawable
+ isPrimary: boolean
~ ChequingAccount(aoountNum: String, ownerName: String)
+ withdraw(money: int): void
+ deposit(chequing: double): void
+ deposit(cash5: int, cash10: int, cash20: int, cash50: int): void
+ invest(principle: double): void
+ repayLoan(payment: double): void
<<abstract>>
DebtAccount extends Account
~ Debt(accountNum: String, ownerName: String)
~ transferIn(money: double): void
~ isOverLimit(): boolean
<<abstract>>
AssetAccount extends Account implements TransferOutable
~ Asset(accountNum: String, ownerName: String)
~ transferIn(money: double): void
+ transferOut(money: double, toAccount: Account): void
+ payBill(money: double, nonUser: String): void
LineOfCreditAccount extends DebtAccount implements TransferOutable
~ Line(accountNum: String, ownerName: String)
+ transferOut(amount: double, toAccount: Account): void
+ payBill(money: double, nonUser: String): void
USDChequingAccount extends AssetAccount
~ USDChequingAccount(accountNum: String, ownerName: String)
+ transferOut(money: double, toAccount: Account): void
JointChequingAccount extends ChequingAccount
+ secondOwner: Customer
+ secondOwnerName: String
~ JointChequingAccount(accountNum: String, ownerName: String)
+ writeFile(filename: String): void
+ toString(): String
SavingAccount extends AssetAccount
- monthlyInterest: double
~ SavingAccount(accountNum: String, ownerName: String)
+ updateMonthlyInterest(): void
~ setMonthlyInterest(monthlyInterest: doule): void
+ writeInterestFile(): void
<<enumeration>>
AccountType
credit
line
chequing
saving
moneyMarket
jointChequing
USDChequing
- name: String
- index: int
~ AccountType(name: String, index: int)
+ getName(): String
+ getIndex(): int
+ getType(name: String): AccountType
+ constructAccount(accountInfo: String[], owner: Customer): Account
- newAccount(accountNum: String, username: String): Account
+ addAccount(user: Customer, typeNum: int[]): Account
Transaction
- money: double
- receivingAccount: Account
- sendingAccount: Account
- final transactionTime: Date
~ Transaction(money: double, receiving: Account, sending: Account, date: Date)
~ Transaction(money: double, Account: Account, isDeposit: boolean, date: Date)
+ undo(): void
~ getMoney(): double
+ toString(): String
~ show(): String
Accounts
- accountsList: List<Account>
- Accounts: Accounts
- Accounts()
+ getAccounts(): Accounts
+ getAccountList(): List<Account>
- addAccount(account: Account): void
+ constructAccount(accountInfo: String[], owner: Customer): void
+ accountFinder(accNum: String): Account
+ updateTransaction(): void
+ depositAll(): void
+ rewriteWholeTransactions(): void
+ loadInterests(): void


Package user
Manager
- manager: Manager
+ requests: ArrayList<String>
- typeNum: int[]
- Manager(username: String, password: String)
+ getManager(username: String, password: String): void
+ addCustomer(username: String, userLevel: String): void
+ addFinancialAdviser(username: String, userLevel: String): void
- addJointChequingAccount(username1: String, username2: String): void
- addAccount(username: String, type: String): void
add different account type by index of static accountNum
+ undoTransaction(account: Account, num: int): void
check if exceed number of transaction
~ writeFile(): void
+ addMoneyToATM(c5: int, c10: int, c20: int, c50: int): void
+ approveAccountRequest(index: int): String[]
+ rejectAccountRequest(index: int): String
FinancialUser extends Customer
+ FinancialUser(username: String, password: String, customerLevel: String)
Customer
+ accounts: List<ArrayList<Account>>
- primaryChequing: ChequingAccount
- customerLevel: CustomerLevel
- financialProducts: List<ArrayList<FinancialProduct>>
+ Customer(username: String, password: String, customer: String)
+ debtLimit(): int
~ setPrimary(primary: ChequingAccount): void
+ getPrimaryChequing: ChequingAccount
+ requestAccount(AccountType: String): void
+ requestJointAccount(otherOwner: String): void
+ requestFinancialProduct(productType: String, principle: double, period: int): void
- getNetTotal(): double
+ rewriteWholeFile(): void
~ writeWholeFile(fileName): void
+ userInfo(): String
+ getProductInfo(): String
+ transferOutables(): String
+ withdrawables(): String
+ chequingInfo(): String
Person
- username: String
+ password: String
- patternPassword: Pattern
~ Person(username: String, password: String)
+ getUsername(): String
+ IsValidPassword(password: String): boolean
+ changePassword(newPassword: String): void
Employee extends Person
+ productRequests: List<String>
~ Employee(username: String, password: String)
+ approveProductRequest(index: int): String[]
+ rejectProductRequest(index: int): String[]
- addNewFinancialProduct(ownerName: String, principle: double, period: int, type: String): void
FinancialAdviser extends Employee
- financialUser: FinantialUser
+ FinancialAdviser(customer: Customer)
- writeWholeFile(): void
<<enumeration>>
CustomerLevel
regular
silver
gold
platinum
- debtLimit: int
+ getDebtLimit(): int
+ getLevel(name: String): CustomerLevel