class Architecture Model

ui::BankInterface

- \# actions: Map<String, UIAction> {readOnly}
- \- credentials: Credentials
- \# location: OperationLocation {readOnly}
- \# log: Log {readOnly}

- \# addAction(String, UIAction) : void
- \# BankInterface(OperationLocation)
- \+ *createAndShowUI() : void*
- \# getCredentials() : Credentials
- \+ getOperationLocation() : OperationLocation
- \+ isLoggedIn() : boolean
- \+ login(Credentials) : void
- \+ logout() : void
- \+ *readBranchId() : Long*
- \+ *readCurrentAccountNumber() : Long*
- \# toggleActions() : void

+actions

«interface»
ui::UIAction

- \+ *execute() : void*
- \+ *isEnabled() : boolean*
- \+ *setEnabled(boolean) : void*

«interface»
business::AccountOperationService

- \+ *deposit(long, long, long, long, double) : Deposit*
- \+ *getBalance(long, long) : double*
- \+ *getStatementByDate(long, long, Date, Date) : List<Transaction>*
- \+ *getStatementByMonth(long, long, int, int) : List<Transaction>*
- \+ *login(long, long, String) : CurrentAccount*
- \+ *transfer(long, long, long, long, long, double) : Transfer*
- \+ *withdrawal(long, long, long, double) : Withdrawal*

«interface»
business::AccountManagementService

- \+ *createCurrentAccount(long, String, String, int, Date, double) : CurrentAccount*
- \+ *login(String, String) : Employee*

domain::CurrentAccount

- \- balance: double
- \- client: Client
- \- deposits: List<Deposit>
- \- id: CurrentAccountId
- \- transfers: List<Transfer>
- \- withdrawals: List<Withdrawal>

- \+ CurrentAccount(Branch, long, Client)
- \+ CurrentAccount(Branch, long, Client, double)
- \+ deposit(OperationLocation, long, double) : Deposit
- \- depositAmount(double) : void
- \+ getBalance() : double
- \+ getClient() : Client
- \+ getDeposits() : List<Deposit>
- \+ getId() : CurrentAccountId
- \+ getTransactions() : List<Transaction>
- \+ getTransfers() : List<Transfer>
- \+ getWithdrawals() : List<Withdrawal>
- \- hasEnoughBalance(double) : boolean
- \- isValidAmount(double) : boolean
- \+ transfer(OperationLocation, CurrentAccount, double) : Transfer
- \+ withdrawal(OperationLocation, double) : Withdrawal
- \- withdrawalAmount(double) : void

impl::AccountOperationServiceImpl

- \- database: Database {readOnly}

- \+ AccountOperationServiceImpl(Database)
- \+ deposit(long, long, long, long, double) : Deposit
- \+ getBalance(long, long) : double
- \- getOperationLocation(long) : OperationLocation
- \- getStatementByDate(CurrentAccount, Date, Date) : List<Transaction>
- \+ getStatementByDate(long, long, Date, Date) : List<Transaction>
- \+ getStatementByMonth(long, long, int, int) : List<Transaction>
- \+ login(long, long, String) : CurrentAccount
- \- readCurrentAccount(long, long) : CurrentAccount
- \+ transfer(long, long, long, long, long, double) : Transfer
- \+ withdrawal(long, long, long, double) : Withdrawal

impl::AccountManagementServiceImpl

- \- database: Database {readOnly}
- \- random: RandomString

- \+ AccountManagementServiceImpl(Database)
- \+ createCurrentAccount(long, String, String, int, Date, double) : CurrentAccount
- \+ login(String, String) : Employee

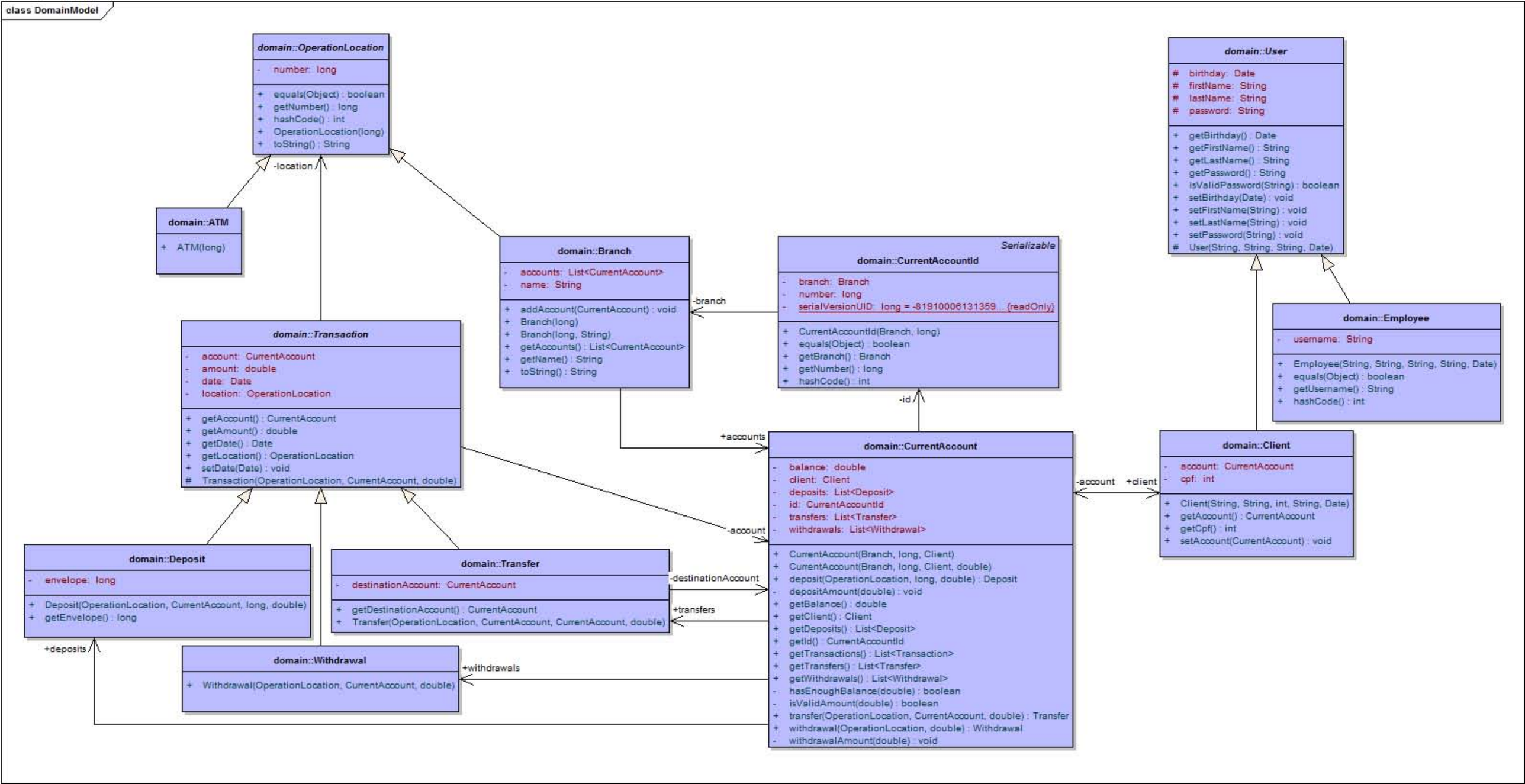
-database

-database

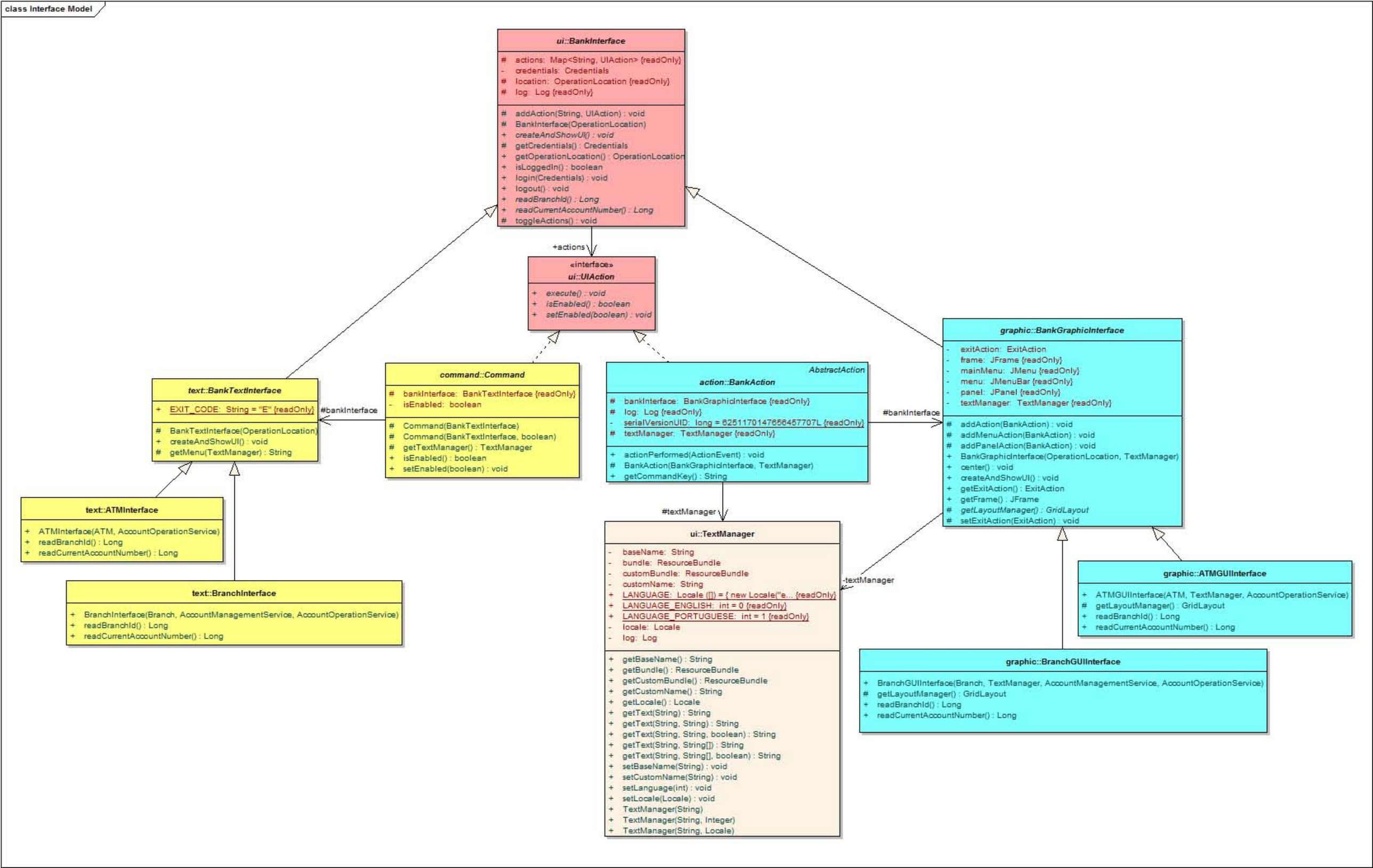
data::Database

- \- currentAccounts: Map<CurrentAccountId, CurrentAccount> {readOnly}
- \- employees: Map<String, Employee> {readOnly}
- \- log: Log {readOnly}
- \- operationLocations: Map<Long, OperationLocation> {readOnly}

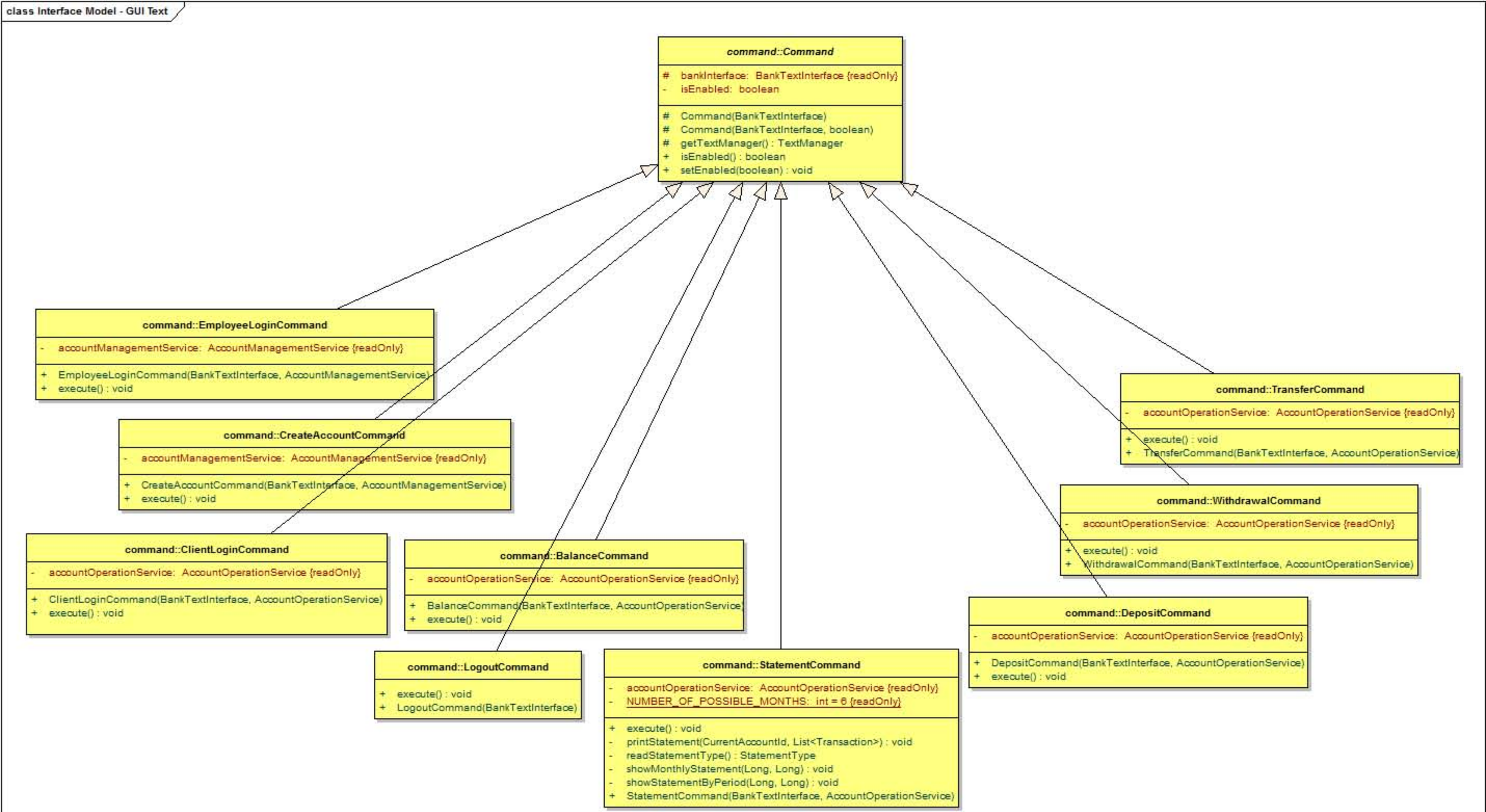
- \- changeDate(Transaction, Random, Calendar) : void
- \+ Database()
- \+ Database(boolean)
- \+ getAllCurrentAccounts() : Collection<CurrentAccount>
- \+ getAllEmployees() : Collection<Employee>
- \+ getAllOperationLocations() : Collection<OperationLocation>
- \+ getCurrentAccount(CurrentAccountId) : CurrentAccount
- \+ getEmployee(String) : Employee
- \+ getNextCurrentAccountNumber() : long
- \+ getOperationLocation(long) : OperationLocation
- \- initData() : void
- \+ save(CurrentAccount) : void
- \+ save(Employee) : void
- \+ save(OperationLocation) : void

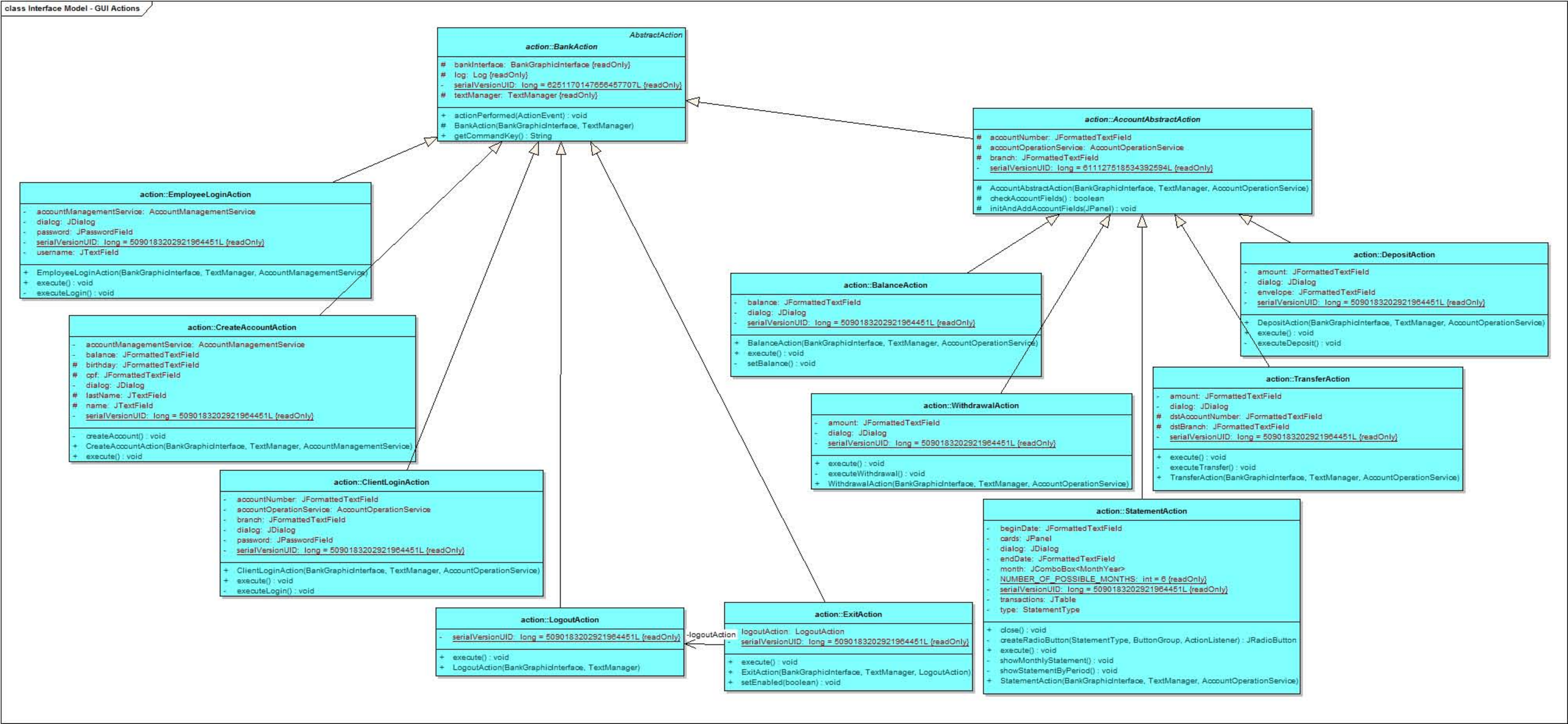

class DomainModel
domain::OperationLocation
- number: long
+ equals(Object) : boolean
+ getNumber() : long
+ hashCode() : int
+ OperationLocation(long)
+ toString() : String
-location
domain::ATM
+ ATM(long)
domain::Branch
- accounts: List<CurrentAccount>
- name: String
+ addAccount(CurrentAccount) : void
+ Branch(long)
+ Branch(long, String)
+ getAccounts() : List<CurrentAccount>
+ getName() : String
+ toString() : String
-branch
Serializable
domain::CurrentAccountId
- branch: Branch
- number: long
- serialVersionUID: long = -81910006131359... {readOnly}
+ CurrentAccountId(Branch, long)
+ equals(Object) : boolean
+ getBranch() : Branch
+ getNumber() : long
+ hashCode() : int
-id
domain::User
birthday: Date
firstName: String
lastName: String
password: String
+ getBirthday() : Date
+ getFirstName() : String
+ getLastName() : String
+ getPassword() : String
+ isValidPassword(String) : boolean
+ setBirthday(Date) : void
+ setFirstName(String) : void
+ setLastName(String) : void
+ setPassword(String) : void
# User(String, String, String, Date)
domain::Employee
- username: String
+ Employee(String, String, String, String, Date)
+ equals(Object) : boolean
+ getUsername() : String
+ hashCode() : int
domain::Transaction
- account: CurrentAccount
- amount: double
- date: Date
- location: OperationLocation
+ getAccount() : CurrentAccount
+ getAmount() : double
+ getDate() : Date
+ getLocation() : OperationLocation
+ setDate(Date) : void
# Transaction(OperationLocation, CurrentAccount, double)
+accounts
domain::CurrentAccount
- balance: double
- client: Client
- deposits: List<Deposit>
- id: CurrentAccountId
- transfers: List<Transfer>
- withdrawals: List<Withdrawal>
+ CurrentAccount(Branch, long, Client)
+ CurrentAccount(Branch, long, Client, double)
+ deposit(OperationLocation, long, double) : Deposit
- depositAmount(double) : void
+ getBalance() : double
+ getClient() : Client
+ getDeposits() : List<Deposit>
+ getId() : CurrentAccountId
+ getTransactions() : List<Transaction>
+ getTransfers() : List<Transfer>
+ getWithdrawals() : List<Withdrawal>
- hasEnoughBalance(double) : boolean
- isValidAmount(double) : boolean
+ transfer(OperationLocation, CurrentAccount, double) : Transfer
+ withdrawal(OperationLocation, double) : Withdrawal
- withdrawalAmount(double) : void
-account
-account
+client
domain::Client
- account: CurrentAccount
- cpf: int
+ Client(String, String, int, String, Date)
+ getAccount() : CurrentAccount
+ getCpf() : int
+ setAccount(CurrentAccount) : void
domain::Deposit
- envelope: long
+ Deposit(OperationLocation, CurrentAccount, long, double)
+ getEnvelope() : long
+deposits
domain::Transfer
- destinationAccount: CurrentAccount
+ getDestinationAccount() : CurrentAccount
+ Transfer(OperationLocation, CurrentAccount, CurrentAccount, double)
-destinationAccount
+transfers
domain::Withdrawal
+ Withdrawal(OperationLocation, CurrentAccount, double)
+withdrawals

class Interface Model



class Interface Model - GUI Text




class Interface Model - GUI Actions
AbstractAction
action::BankAction
bankInterface: BankGraphicInterface {readOnly}
log: Log {readOnly}
- serialVersionUID: long = 6251170147656457707L {readOnly}
textManager: TextManager {readOnly}
+ actionPerformed(ActionEvent) : void
# BankAction(BankGraphicInterface, TextManager)
+ getCommandKey() : String
action::AccountAbstractAction
accountNumber: JFormattedTextField
accountOperationService: AccountOperationService
branch: JFormattedTextField
- serialVersionUID: long = 61112751853439259 4L {readOnly}
AccountAbstractAction(BankGraphicInterface, TextManager, AccountOperationService)
checkAccountFields() : boolean
initAndAddAccountFields(JPanel) : void
action::EmployeeLoginAction
- accountManagementService: AccountManagementService
- dialog: JDialog
- password: JPasswordField
- serialVersionUID: long = 5090183202921964451L {readOnly}
- username: JTextField
+ EmployeeLoginAction(BankGraphicInterface, TextManager, AccountManagementService)
+ execute() : void
- executeLogin() : void
action::CreateAccountAction
- accountManagementService: AccountManagementService
- balance: JFormattedTextField
birthday: JFormattedTextField
cpf: JFormattedTextField
- dialog: JDialog
lastName: JTextField
name: JTextField
- serialVersionUID: long = 5090183202921964451L {readOnly}
- createAccount() : void
+ CreateAccountAction(BankGraphicInterface, TextManager, AccountManagementService)
+ execute() : void
action::ClientLoginAction
- accountNumber: JFormattedTextField
- accountOperationService: AccountOperationService
- branch: JFormattedTextField
- dialog: JDialog
- password: JPasswordField
- serialVersionUID: long = 5090183202921964451L {readOnly}
+ ClientLoginAction(BankGraphicInterface, TextManager, AccountOperationService)
+ execute() : void
- executeLogin() : void
action::LogoutAction
- serialVersionUID: long = 5090183202921964451L {readOnly}
+ execute() : void
+ LogoutAction(BankGraphicInterface, TextManager)
-logoutAction
action::ExitAction
logoutAction: LogoutAction
- serialVersionUID: long = 5090183202921964451L {readOnly}
+ execute() : void
+ ExitAction(BankGraphicInterface, TextManager, LogoutAction)
+ setEnabled(boolean) : void
action::BalanceAction
- balance: JFormattedTextField
- dialog: JDialog
- serialVersionUID: long = 5090183202921964451L {readOnly}
+ BalanceAction(BankGraphicInterface, TextManager, AccountOperationService)
+ execute() : void
- setBalance() : void
action::WithdrawalAction
- amount: JFormattedTextField
- dialog: JDialog
- serialVersionUID: long = 5090183202921964451L {readOnly}
+ execute() : void
- executeWithdrawal() : void
+ WithdrawalAction(BankGraphicInterface, TextManager, AccountOperationService)
action::DepositAction
- amount: JFormattedTextField
- dialog: JDialog
- envelope: JFormattedTextField
- serialVersionUID: long = 5090183202921964451L {readOnly}
+ DepositAction(BankGraphicInterface, TextManager, AccountOperationService)
+ execute() : void
- executeDeposit() : void
action::TransferAction
- amount: JFormattedTextField
- dialog: JDialog
dstAccountNumber: JFormattedTextField
dstBranch: JFormattedTextField
- serialVersionUID: long = 5090183202921964451L {readOnly}
+ execute() : void
- executeTransfer() : void
+ TransferAction(BankGraphicInterface, TextManager, AccountOperationService)
action::StatementAction
- beginDate: JFormattedTextField
- cards: JPanel
- dialog: JDialog
- endDate: JFormattedTextField
- month: JComboBox<MonthYear>
- NUMBER_OF_POSSIBLE_MONTHS: int = 6 {readOnly}
- serialVersionUID: long = 5090183202921964451L {readOnly}
- transactions: JTable
- type: StatementType
+ close() : void
- createRadioButton(StatementType, ButtonGroup, ActionListener) : JRadioButton
+ execute() : void
- showMonthlyStatement() : void
- showStatementByPeriod() : void
+ StatementAction(BankGraphicInterface, TextManager, AccountOperationService)