PROGRAMOWANIE OBIEKTOWE 2019/2020

Projekt: Bank – Konta

JAKUB PIETRAS 224404

BARTŁOMIEJ GRACZYK 224301

v1.8

Spis treści

Projekt: Bank – Konta 1

Client 3

Account 3

Card 4

Loan 4

Repository 5

Manager 5

Client and Account Builders 6

Card and Loan Builders 6

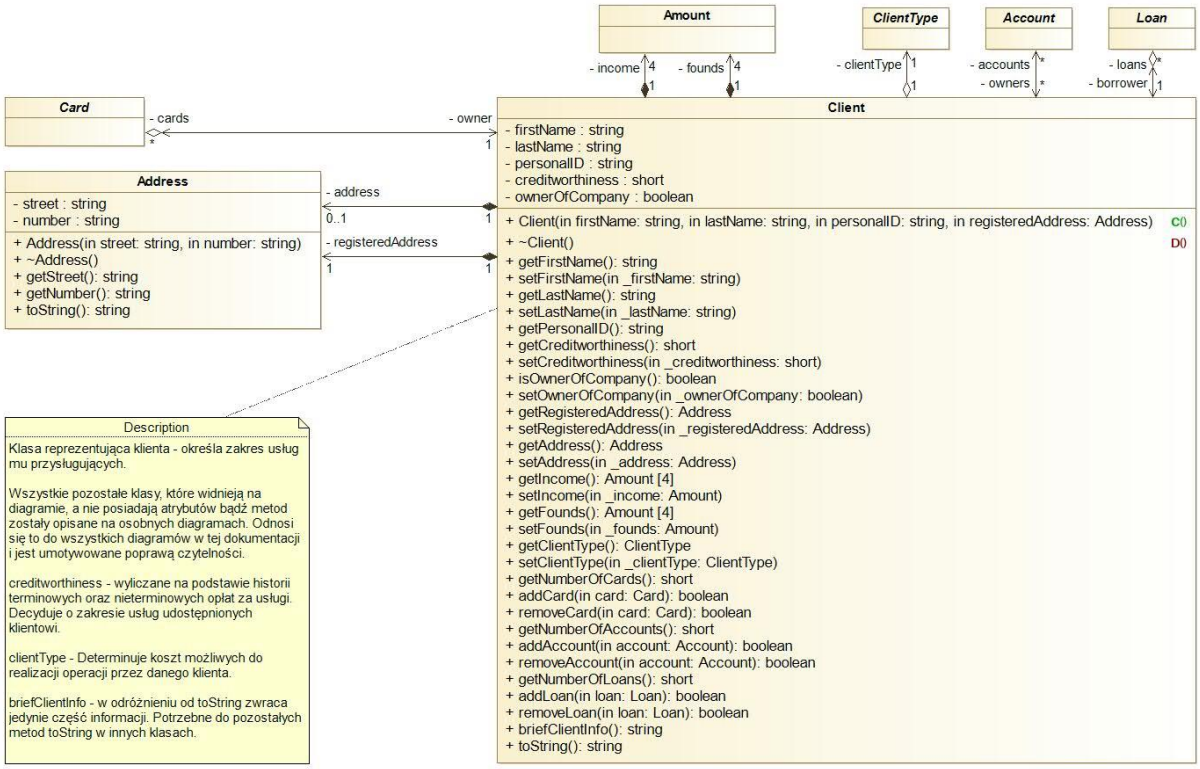
Client Type 7

Card Class 7

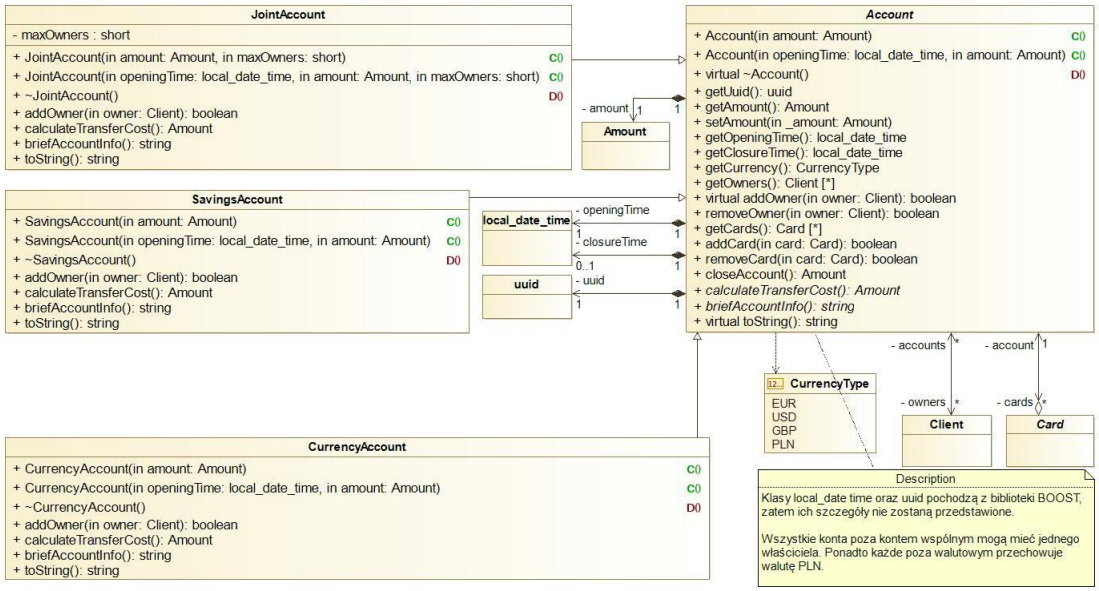
Exceptions 8

Amount 8

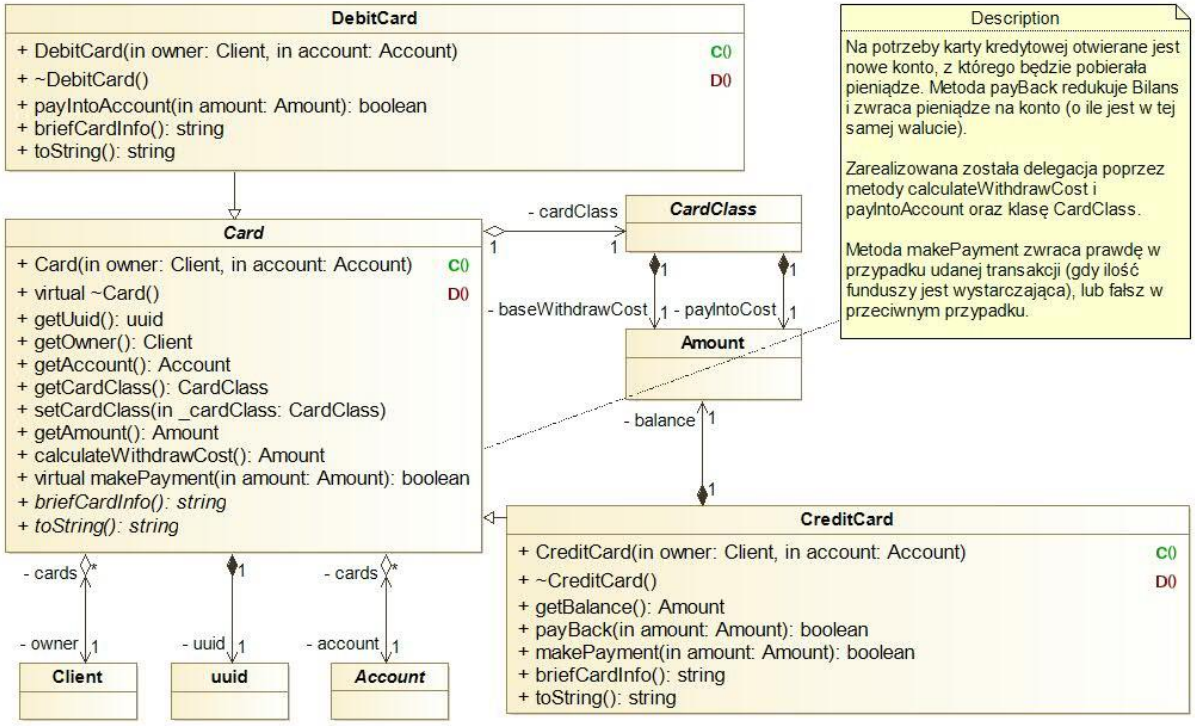
Client



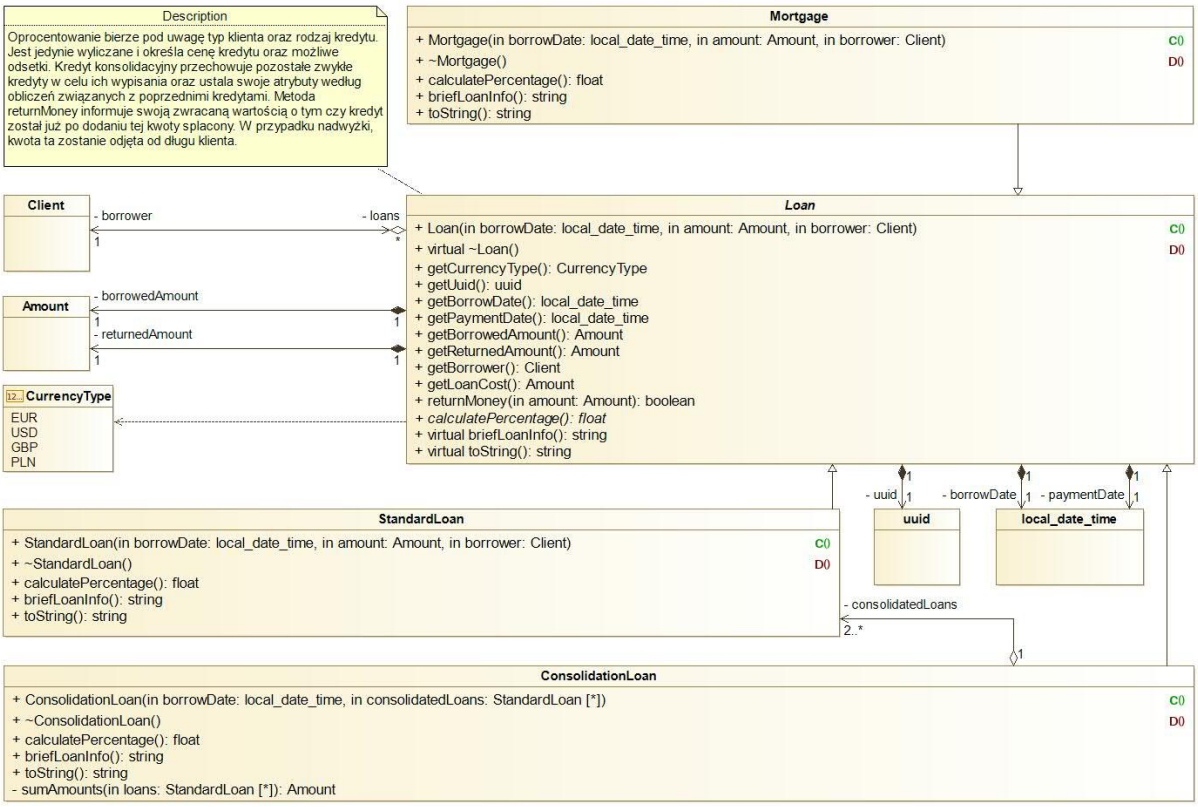
Account



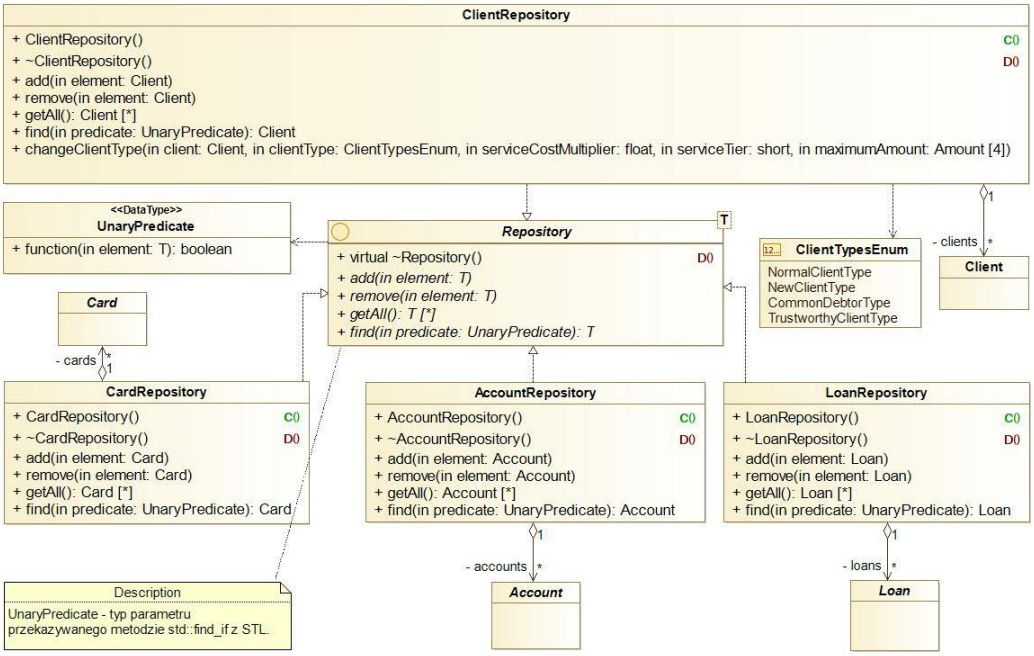
Card


DebitCard
+ DebitCard(in owner: Client, in account: Account)
+ ~DebitCard()
+ payIntoAccount(in amount: Amount): boolean
+ briefCardInfo(): string
+ toString(): string
C0
D0
Description
Na potrzeby karty kredytowej otwierane jest nowe konto, z którego będzie pobierała pieniądze. Metoda payBack redukuje Bilans i zwraca pieniądze na konto (o ile jest w tej samej walucie).
Zarealizowana została delegacja poprzez metody calculateWithdrawCost i payIntoAccount oraz klasę CardClass.
Metoda makePayment zwraca prawdę w przypadku udanej transakcji (gdy ilość funduszy jest wystarczająca), lub fałsz w przeciwnym przypadku.
Card
+ Card(in owner: Client, in account: Account)
+ virtual ~Card()
+ getUuid(): uuid
+ getOwner(): Client
+ getAccount(): Account
+ getCardClass(): CardClass
+ setCardClass(in _cardClass: CardClass)
+ getAmount(): Amount
+ calculateWithdrawCost(): Amount
+ virtual makePayment(in amount: Amount): boolean
+ briefCardInfo(): string
+ toString(): string
C0
D0
- cardClass
CardClass
- baseWithdrawCost
- payIntoCost
Amount
- balance
CreditCard
+ CreditCard(in owner: Client, in account: Account)
+ ~CreditCard()
+ getBalance(): Amount
+ payBack(in amount: Amount): boolean
+ makePayment(in amount: Amount): boolean
+ briefCardInfo(): string
+ toString(): string
C0
D0
- cards
- owner
- uuid
- account
Client
uuid
Account

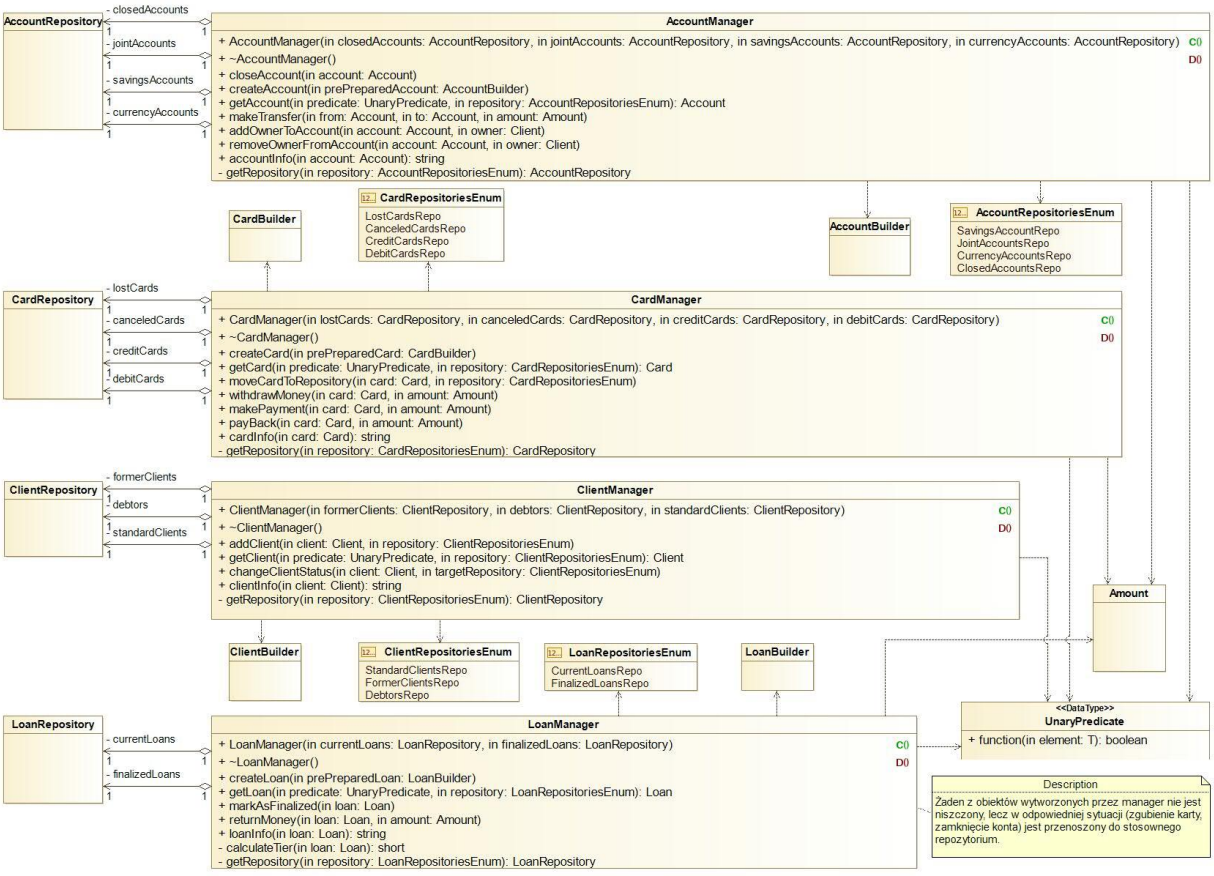
Loan


Description
Oprocentowanie bierze pod uwagę typ klienta oraz rodzaj kredytu. Jest jedynie wyliczane i określa cenę kredytu oraz możliwe odsetki. Kredyt konsolidacyjny przechowuje pozostałe zwykłe kredyty w celu ich wypisania oraz ustala swoje atrybuty według obliczeń związanych z poprzednimi kredytami. Metoda returnMoney informuje swoją zwracaną wartością o tym czy kredyt został już po dodaniu tej kwoty spłacony. W przypadku nadwyżki, kwota ta zostanie odjęta od długu klienta.
Mortgage
+ Mortgage(in borrowDate: local_date_time, in amount: Amount, in borrower: Client)
+ ~Mortgage()
+ calculatePercentage(): float
+ briefLoanInfo(): string
+ toString(): string
C0
D0
Client
- borrower
- loans
Loan
+ Loan(in borrowDate: local_date_time, in amount: Amount, in borrower: Client)
+ virtual ~Loan()
+ getCurrencyType(): CurrencyType
+ getUuid(): uuid
+ getBorrowDate(): local_date_time
+ getPaymentDate(): local_date_time
+ getBorrowedAmount(): Amount
+ getReturnedAmount(): Amount
+ getBorrower(): Client
+ getLoanCost(): Amount
+ returnMoney(in amount: Amount): boolean
+ calculatePercentage(): float
+ virtual briefLoanInfo(): string
+ virtual toString(): string
C0
D0
Amount
- borrowedAmount
- returnedAmount
CurrencyType
EUR
USD
GBP
PLN
- uuid
- borrowDate
- paymentDate
uuid
local_date_time
StandardLoan
+ StandardLoan(in borrowDate: local_date_time, in amount: Amount, in borrower: Client)
+ ~StandardLoan()
+ calculatePercentage(): float
+ briefLoanInfo(): string
+ toString(): string
C0
D0
- consolidatedLoans
2..*
ConsolidationLoan
+ ConsolidationLoan(in borrowDate: local_date_time, in consolidatedLoans: StandardLoan [*])
+ ~ConsolidationLoan()
+ calculatePercentage(): float
+ briefLoanInfo(): string
+ toString(): string
- sumAmounts(in loans: StandardLoan [*]): Amount
C0
D0

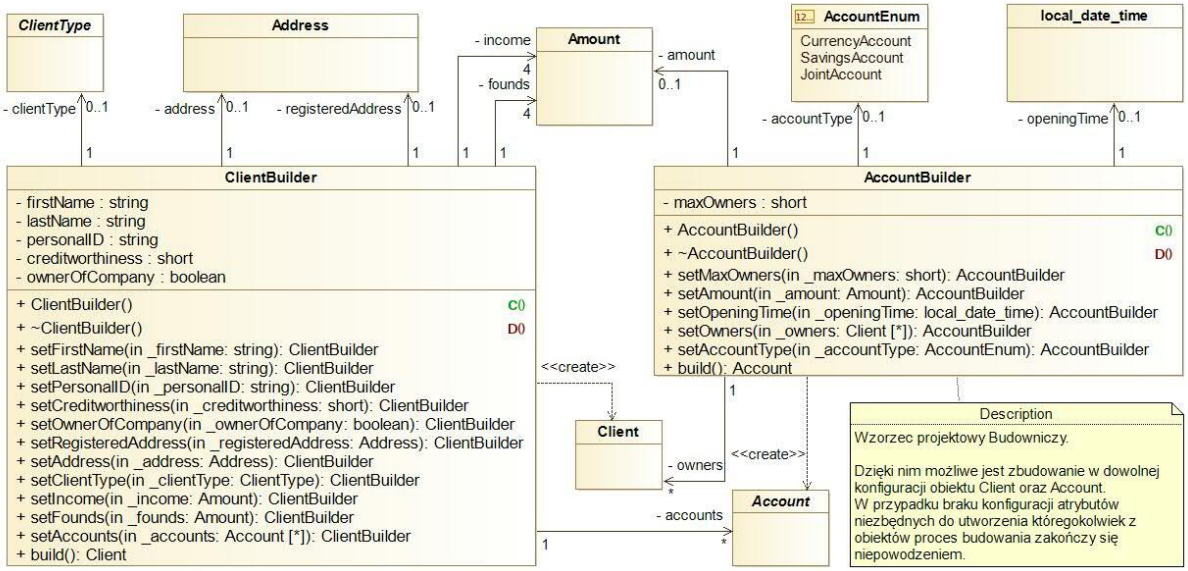
Repository



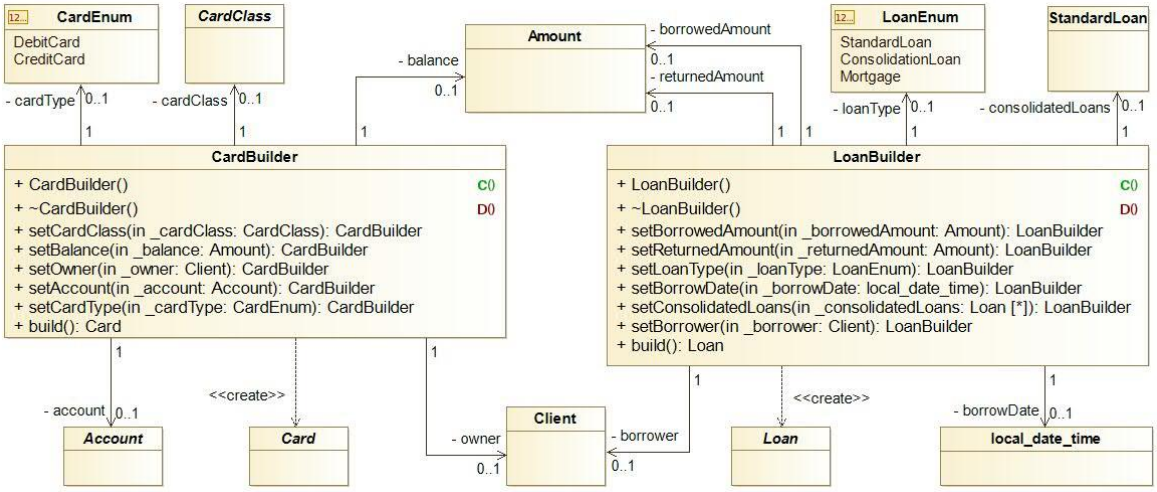
Manager



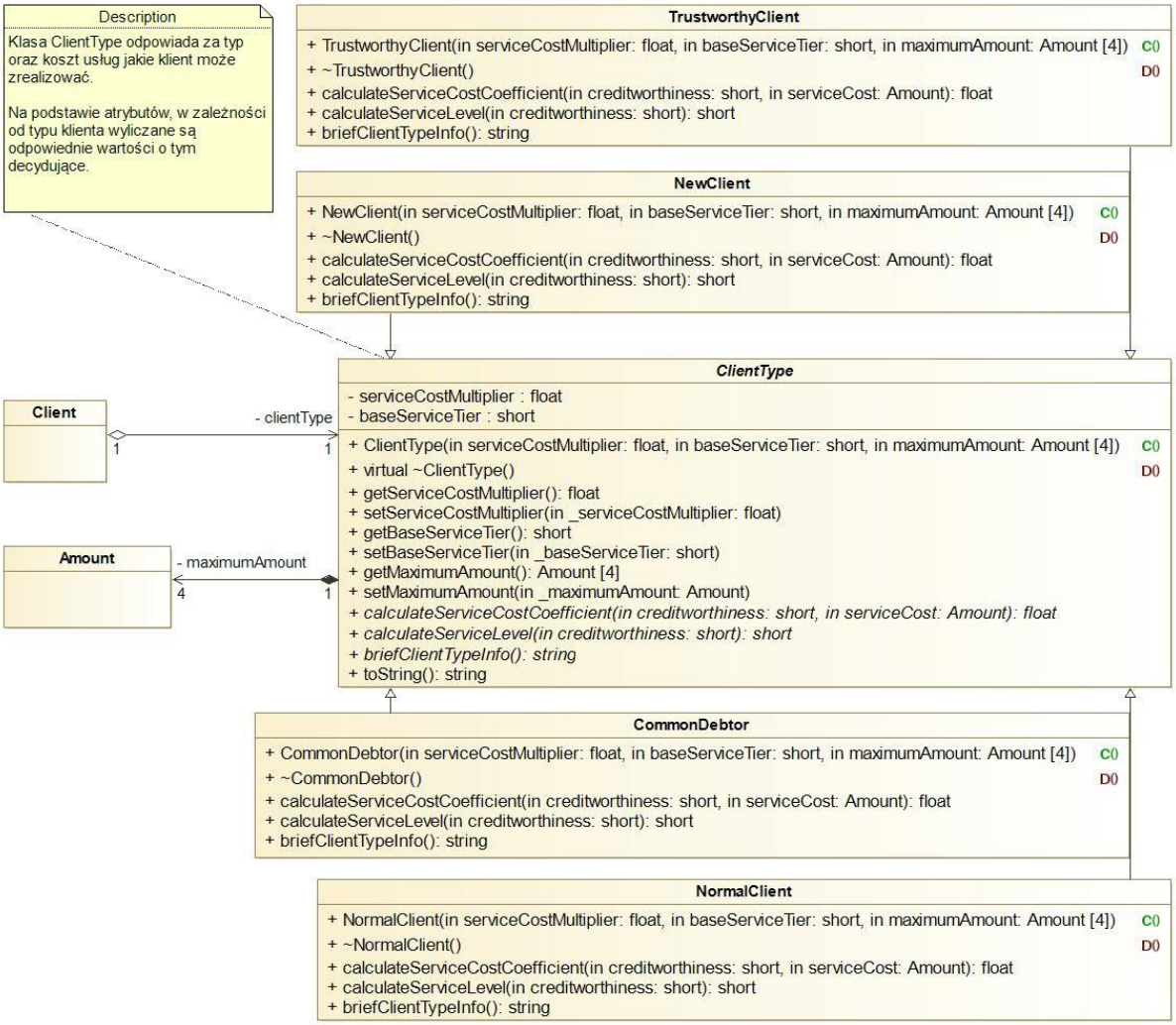
Client and Account Builders



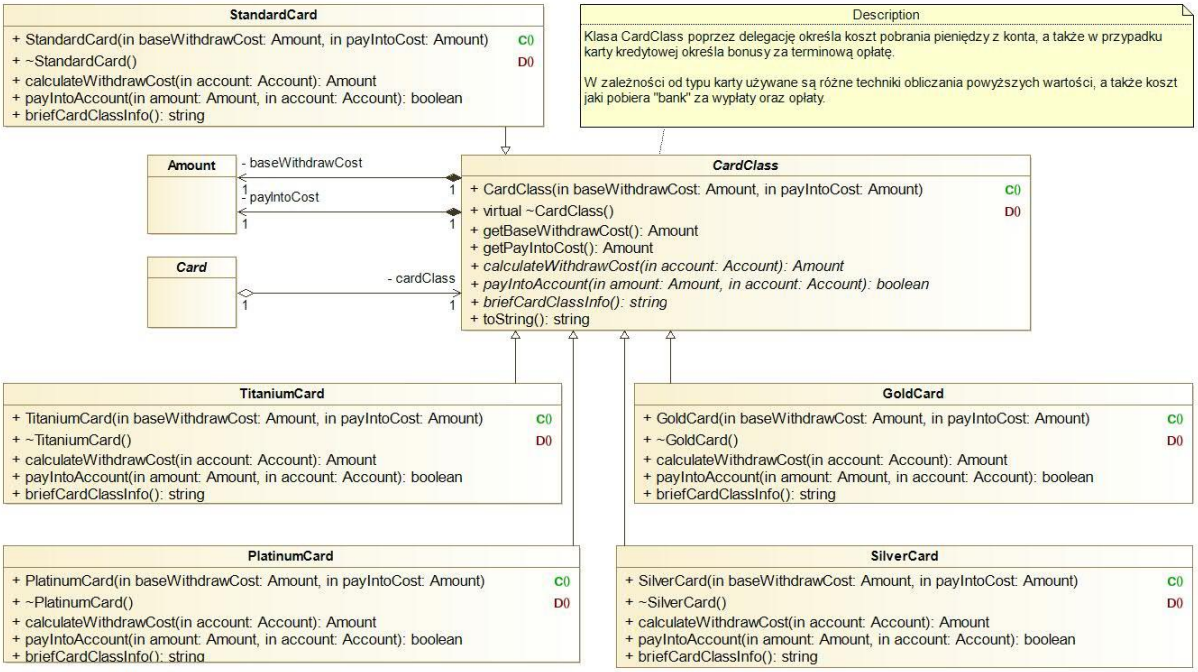
Card and Loan Builders



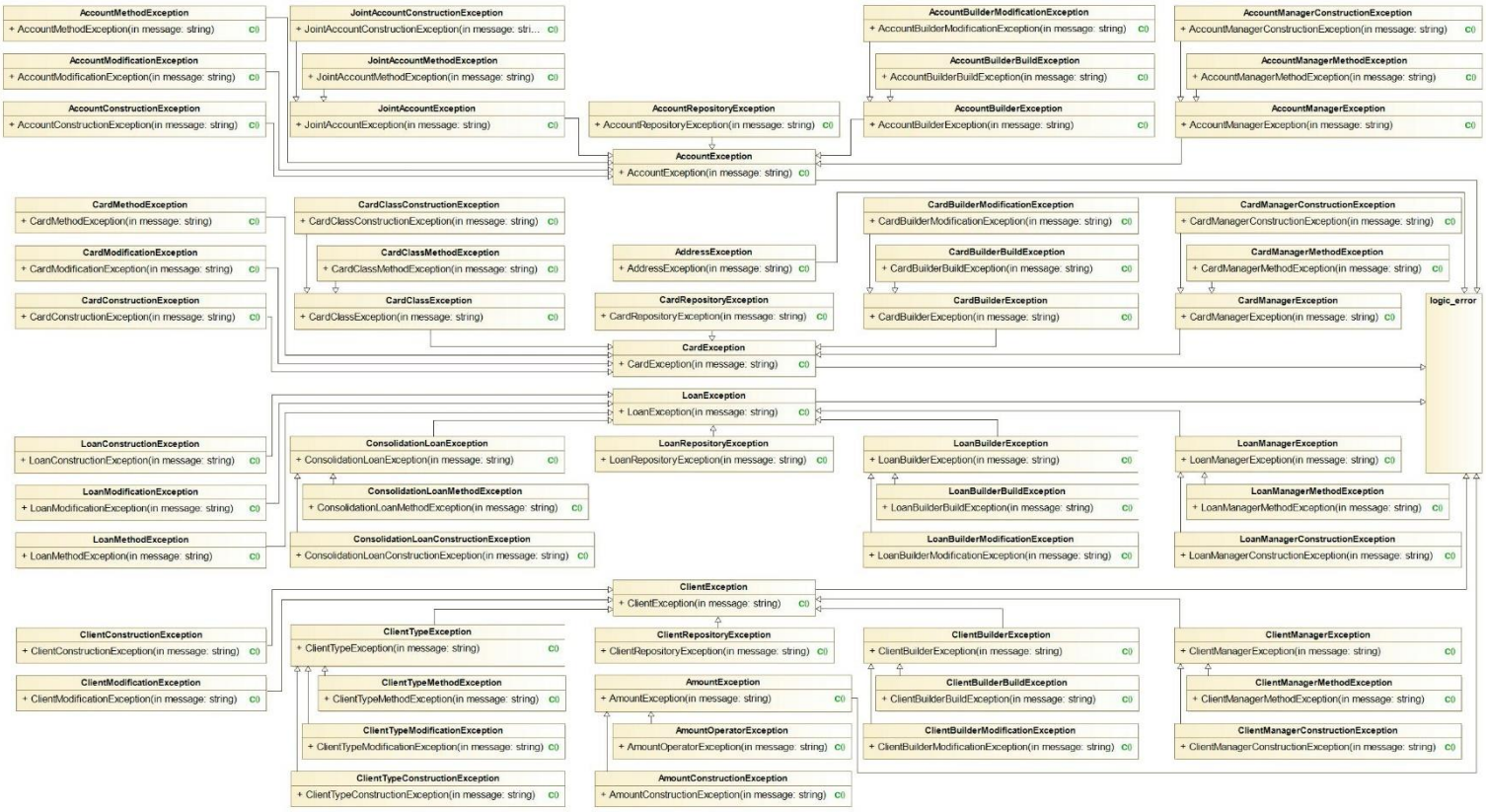
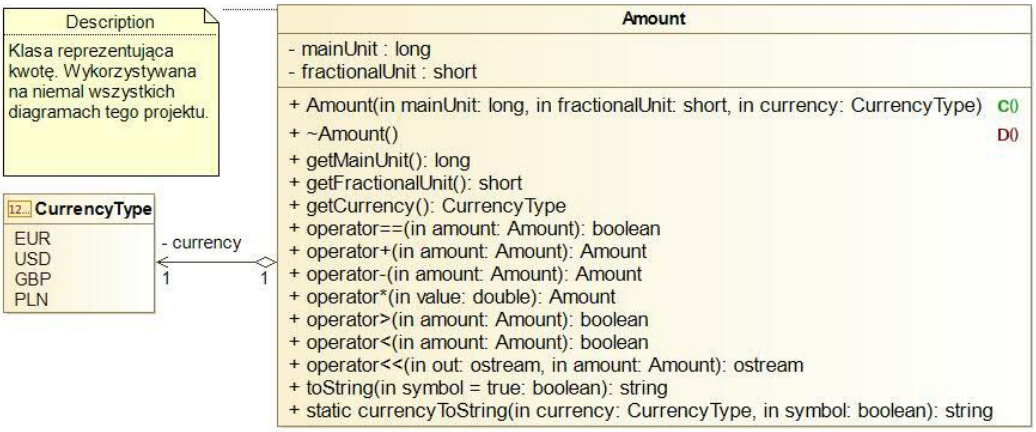
Client Type



Card Class



Exceptions



Amount