**Курсов прокет**

Курсов проект No2022043

ТЕМА: „Система за управление на банкови сметки“

Автор:

Иван Николаев Михайлов

Специалност: Интелигентни системи и изкуствен интелект

Група: 21А

Факултетен номер: 361224033

Ръководител:

Милен Младенов

ТУ-София

Съдържание

[**1.** **Задание** 3](#_Toc180109303)

[**1.1.** **Основни задачи** 3](#_Toc180109304)

[**2.** **Използвани модули** 5](#_Toc180109305)

[**2.1.** **Класове** 5](#_Toc180109306)

[**2.1.1.** **Account** 5](#_Toc180109307)

[**2.1.1.1.** **Полета** 5](#_Toc180109308)

[**2.1.1.2.** **Методи** 5](#_Toc180109309)

[**2.1.2.** **CheckingAccount** 6](#_Toc180109310)

[**2.1.2.1.** **Полета** 7](#_Toc180109311)

[**2.1.2.2.** **Методи** 7](#_Toc180109312)

[**2.1.3.** **TestBank** 8](#_Toc180109313)

[**3.** **Резултатит от изпълнение на програмта** 9](#_Toc180109314)

[**3.1.** **При правилни данни** 9](#_Toc180109315)

[**3.2.** **При грешни данни** 9](#_Toc180109316)

# **Задание**

Проектирайте и реализирайте приложение, което служи като система за управление на банкови сметки – дава възможност за прилагане на различни банкови операции и отпечатва резултат след въвеждане на информацията.

# **Основни задачи**

1. Дефинирайте клас Account (Сметка) с полета за: номер на сметка – от целочислен тип int; име на притежателя на сметка – от тип String; баланс (вложени пари) – от реален тип double, декларирани като private. Напишете:

а) конструктор с три параметъра за инициализиране на полетата;

б) метод deposit() (операция внасяне на пари) с параметър сума за внасяне от реален тип double, който добавя сумата за внасяне към депозита.

в) метод withdraw() (операция теглене на пари) с параметър сума за теглене от реален тип double, който изважда сумата за теглене от депозита, ако сумата за теглене е по-малка или равна на депозита.

2. Дефинирайте клас CheckingAccount (Разплащателна сметка) като наследник на класа Account с поле за брой банкови операции за месец – от целочислен тип int, декларирано като private. Напишете:

а) конструктор с три параметъра за инициализиране на номера, името на притежателя и баланса в разплащателна сметка; установете броячът на банкови операции в нула;

б) предефиниран метод deposit() (операция внасяне на пари), който обновява брояча на банковите операции, като използвате следния алгоритъм:

- увеличете с единица брояча на банкови операции

- добавете сумата към депозита

в) предефиниран метод withdraw() (операция теглене на пари), който обновява брояча на банковите операции, като използвате следния алгоритъм:

- увеличете с единица брояча на банкови операции

- извадете сумата за теглене от депозита

3. Декларирайте тестов клас Банка (TestBank) с информация за сметки.

а) В метода main() създайте по 2 обекта от класовете Account и CheckingAccount за 2ма притежатели, като въвеждате данните от клавиатурата и ги добавяте в масива/в списъка. Проверявайте въведените данни, обработвайте възникналите изключения.

б) Отпечатайте на екрана информация за наличните суми по сметките.

в) от разплащателните сметки изтеглете сума от по 350 лв.;

г) внесете изтеглените суми към сметките и отпечатайте информацията.

# **Използвани модули**

# **Класове**

# **Account**

Базов клас, който пази основните полета и методи, с които можем да извършваме операции със сметките на потребители.

|  |  |  |
| --- | --- | --- |
|  | Полета |  |
| Тип | Име на променлива | Описание |
| Int(целочислен тип) | accountNumber | Пази номера на банков акаунт |
| String(Низ) | owner | Име на потребител |
| Double(реален тип) | balance | Баланс на потребител |

# **Полета**

# **Методи**

|  |  |  |
| --- | --- | --- |
|  | Методи |  |
| Параметри | Име | Описание |
| accountNumber, owner, balance | Account | Конструктор за инициализиране на стойностите на съответните полета от класа |
| amount | deposit : virtual void | Метод, който позволява на потребителите да правят депозит по сметка. |
| amount | withdraw : virtual void | Позволява на потребителите да теглят пари от текущия баланс. |
|  | displayAccountInfo | Отпечатва информацията за сметка на потребител. |
|  | getBalance : double | Гетър – връща текущия баланс по сметката. |
| newBalance | setBalance : void | Сетър – задава нов баланс на сметката, като приема стойността newBalance. |
|  | getAccountNumber : int | Гетър – връща номер на сметка. |

# **CheckingAccount**

Клас, който наследява всички методи и променливи от базовия клас Account. В CheckingAccount методите за достъп до баланса и номер на сметката са пренаписани, за да се добави възможност за броене на банковите операции.

# **Полета**

|  |  |  |
| --- | --- | --- |
|  | Полета |  |
| Тип | Име на променлива | Описание |
| Int(целочислен тип) | bankingOperationsCount | Съхранява броя на извършените операции (депозит и теглене). |

# **Методи**

|  |  |  |
| --- | --- | --- |
|  | Методи |  |
| Параметри | Име | Описание |
| accountNumber, owner, balance | CheckingAccount | Конструктор, който приема accountNumber, owner и balance, които се предават към базовия клас Account чрез неговия конструктор. Също инициализира bankingOperationsCount със стойност 0. |
| amount | deposit : override | Пренаписва метода deposit от класа Account, като добавя възможността за увеличаване на броя на банковите операции при извървшане на депозит. |
| amount | withdraw : override | Пренаписва метода withdraw от класа Account, като инкрементира стойността на bankingOperationsCount при всяко теглене на пари от баланса. |
|  | displayAccountInfo | Отпечатва информацията за сметка на потребител. |
|  | getBalance : double | Гетър – връща текущия баланс по сметката. |
| newBalance | setBalance : void | Сетър – задава нов баланс на сметката, като приема стойността newBalance. |
|  | getAccountNumber : int | Гетър – връща номер на сметка. |

# **TestBank**

* std::vector<Account\*> accounts – Вектор от указатели към класа Account. Използваме указател към Account за да можем да пълним вектора с обекти от тип Account и CheckingAccount, тъй като CheckingAccount наследява Account.
* Метод run() – Основният метод на класа, в който създаваме по две инстанции на класовете Account и CheckingAccount и извършваме тестови операции с тях. Ръчно се въвеждат данните сметките и разплащателните сметки(номер на сметка, име на собственик и баланс). При въвеждане на данните от потребителя се извършав проверки, дали данните са въведени правилно. Цялата функционалност е обгърната от try-catch блок, като при възникване на грешка или невалидни данни се хвърлят изключения(exceptions) от тип std::invalid\_argument, които се преработват като грешки и биват изписани на стандратния изход. Всяка нова сметка/разплащателна сметка се добавя към вектора accounts.
* Извеждане на информация за сметки на стандартния изход – След добавянето на сметките във вектора accounts, методът го обхожда и извежда информацията за всяка сметка чрез метода displayAccountInfo()
* Теглене от разплащателна сметка: Методът обхожда разплащателните сметки и използва dynamic\_cast за да провери дали даден обект е от правилния тип. Ако е от правилния тип, от разплащателната сметка биват изтеглени 350 лв.
* Депозиране в стандартна сметка: Отново сметките биват обходени и чрез dynamic\_cast проверяваме дали дадената ни сметка е от ти Account. Ако е, биват депозирани 350 лв. в стандартната сметка.
* Освобождаване на паметта – След извършване на всички операции, свързани с вектора accounts, освобождаваме паметта с помощта на delete, за да не се получи memory leak.

# **Резултатит от изпълнение на програмта**

# **При правилни данни**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# **При грешни данни**

A black background with white text

Description automatically generated

A screen shot of a computer

Description automatically generated