**Bank system**

Вие работите за банка. Както всяка банка в 21 век и тази има електронни системи. Една от първите такива е потребителската система. Тъй като е много остаряла и написана на Pascal (yikes) Вашите шефове Ви възлагат задачата да я пренапишете. Трябва да започнете като създадете клас BankAccount или иначе казано, банковият акаунт на всеки потребител. Напишете класа в отделен файл BankAccount.py. След това направете нов файл BankSystemem.py и там ще направите отделен клас – системата на банката.

**Bank Account**

Какви данни трябва да съдържа класа:

• Лично име на потребителя

• Фамилно име на потребителя

• Наличност по сметката – не може да е < 0

• История на транзакциите – списък с всички движения по сметката (теглене или депозиране на пари). //hint – каква структура от данни ще изпозлвате?

Методи на класа:

• Конструктор (\_\_Init\_\_ метод) в конструктора подавайте само имената на потребителя и първоначалната наличност по сметката. Създаването на потребителски акаунт не се брои за транзакция.

• Dunder method който да връща името на потребителя във форматиран вид. Кой ще е този dunder method? //Hint: ползвали сме го.

• withdraw(amount) – Изтегляне на сума от сметката (ако е възможно)

• deposit(amount) – Депозиране на сума в сметката

• \_\_save\_transaction(is\_deposit, amount) – private method който да записва транзакцията по следния начин: „transaction X, type – Y, amount - Z, new balance W“, където Х е пореден номер на транзакцията, Y може да бъде deposit или withdraw, Z – сумата на транзакцията, W – наличността по сметката след като транзакцията е приключила. Параметърът is\_deposit има стойност true ако транзакцията е от тип депозит и false в противен случай. Параметърът amount – стойността на транзакцията.

Пример – за акаунт с наличност 1000 лева по сметката, от който току що е са изтеглени 500 лева, и това е била първата транзакция след създаване на сметката, методът save\_transaction трябва да запише: “transaction 1, type – withdraw, amount 500 new balance 500”

• print\_all\_transactions() – метод който изпринтва всички запазени транзакции до този момент.

• Return\_loan\_payment(amount) // Повече за това надолу. Този метод както и член данните свъзани с него може да ги промените по най-удобния за Вас начин.

**Bank system**

• Какви данни трябва да съдържа системата

Списък със **уникални** потребители. //Какво ще ползвате за user\_id?

Потребители с едно и също име се допускат. Пример – Димитър Димитров Сума на наличните пари в банката.

• Какви методи трябва да съдържа системата

Register\_new\_user( new\_Bank\_Account : BankAcconut )

Delete\_user( user\_id )

Give\_loan\_to\_user( user\_id, amount, interest, return\_period\_in\_months ) //Ако е възможно

Show\_all\_transactions( user\_id = “” ) //Default-на стойност за user\_id – ако е празна функцията да изпише всички транзакции на всички потребители, ако не е да изпише всички транзакции на съответния потребител.

В main.py създайте банкова система, и демонстрирайте нейните възможности – добавете потребители, направете транзакции, теглете кредити, добавете нови потребители, изтрийте стари и т.н. След като приключите с имплементацията запазете файловете. Ако повечето от вас се справят със задачата и имаме достатъчно време, по нататък през срока ще направим графичен интерфейс който да борави със системата Ви.

**БОНУС**

Тук няма да давам много подсказки, тази част е за по-любознателните от Вас, които нямат страх да се поровят в интернет . Добавете още едно поле – парола на потребителя, **НО НЕ ЗАПАЗВАЙТЕ ПАРОЛАТА КАТО ЧИСТ ТЕКСТ,** все пак това е информация която не искаме

потенциални не ауторизирани потребители да получат. Не имплементираме система на НАП все пак. Намерете начин да шифровате паролата преди да я запазите. Освен това напишете функции за проверка и смяна на паролата.