Навчально-науковий інститут інформаційних технологій
Харківський національний економічний університет
імені Семена Кузнеця

Звіт
З Виконання лабораторної роботи №5
за дисципліною: “ Безпека банківських систем ”
на тему: “РОЗРОБКА СИСТЕМИ «БАНКОМАТИК»”

Виконав: студент кафедри
Кібербезпеки та інформаційних
технологій
4 курсу, спец. Кібербезпека,
групи 6.04.125.010.21.2
Бойко Вадим Віталійович
Перевірив:
Лимаренко Вячеслав Володимирович

ХНЕУ ім. С. Кузнеця
2024

Мета: ознайомитися з основами програмного забезпечення та логіки роботи банкоматів, розробити програмне забезпечення, що моделює роботу пристрою.

Завдання:

1. Реалізувати програму, опис якої наведено в Розділі 2.
2. В якості мови програмування використати будь-яку мову на ваш вибір.
3. Дані в програмі повинні бути зашифровані.
4. Навести вихідний код програми.
5. Разом зі звітом надати скомпільовану програму у архіві.

Контрольні питання:

1. Який мінімальний набір компонентів банкомату?
2. Що являє собою стандарт CEN/XFS?
3. Що таке стейти?
4. Як влаштований екран банкомату?

Хід роботи:

Напишу прграму, в результаті отримаю наступний код

```
from base64 import b64encode



class Card:
    def __init__(self, card_number: str, password: str, balance: int):
        self._card_number = card_number
        self._password = self._encode_value(password)
        self._balance = balance
        self._is_blocked = False

    def get_card_number(self) -> str:
        return self._card_number

    def get_balance(self) -> int:
        if not self._is_action_permitted():
            print('Looks like your card is blocked')
        return self._balance

    def validate_password(self) -> bool:
        if not self._is_action_permitted():
            return False
        c = 3
        while c != 0:
            password = input("Enter your password: ")
            if not self._is_valid_password(password):
                c -= 1
            else:
                break
        if not c:
            self._is_blocked = True
            print('Your card is blocked')
            return False
        return True

    def _is_valid_password(self, password):
        return self._password == self._encode_value(password)

    def add_money(self, amount: int):
        self._balance += amount
```

```
    def can_get_money(self, amount):
        return self._balance >= amount and not self._is_blocked

    def get_money(self, amount: int):
        if self._is_action_permitted:
            self._balance -= amount

    def _is_action_permitted(self) -> bool:
        if self._is_blocked:
            print('Your card is blocked')
            return False
        return True

    @staticmethod
    def _encode_value(value: str):
        return b64encode(value.encode('utf-8'))


class Bank:
    def __init__(self):
        self._cards: dict[str, Card] = {}

    def add_card(self, card: Card):
        self._cards.update({
            card.get_card_number(): card
        })

    def is_card_exists(self, card_number: str) -> bool:
        if card_number not in self._cards.keys():
            return False
        return True

    def validate_user(self, card_number: str) -> bool:
        return self._cards[card_number].validate_password()

    def add_money_to_card(self, card_number: str, amount: int):
        self._cards[card_number].add_money(amount)

    def can_get_money(self, card_number: str, amount: int) -> bool:
        card = self._cards[card_number]
        if not card.can_get_money(amount):
            return False
        return True
```

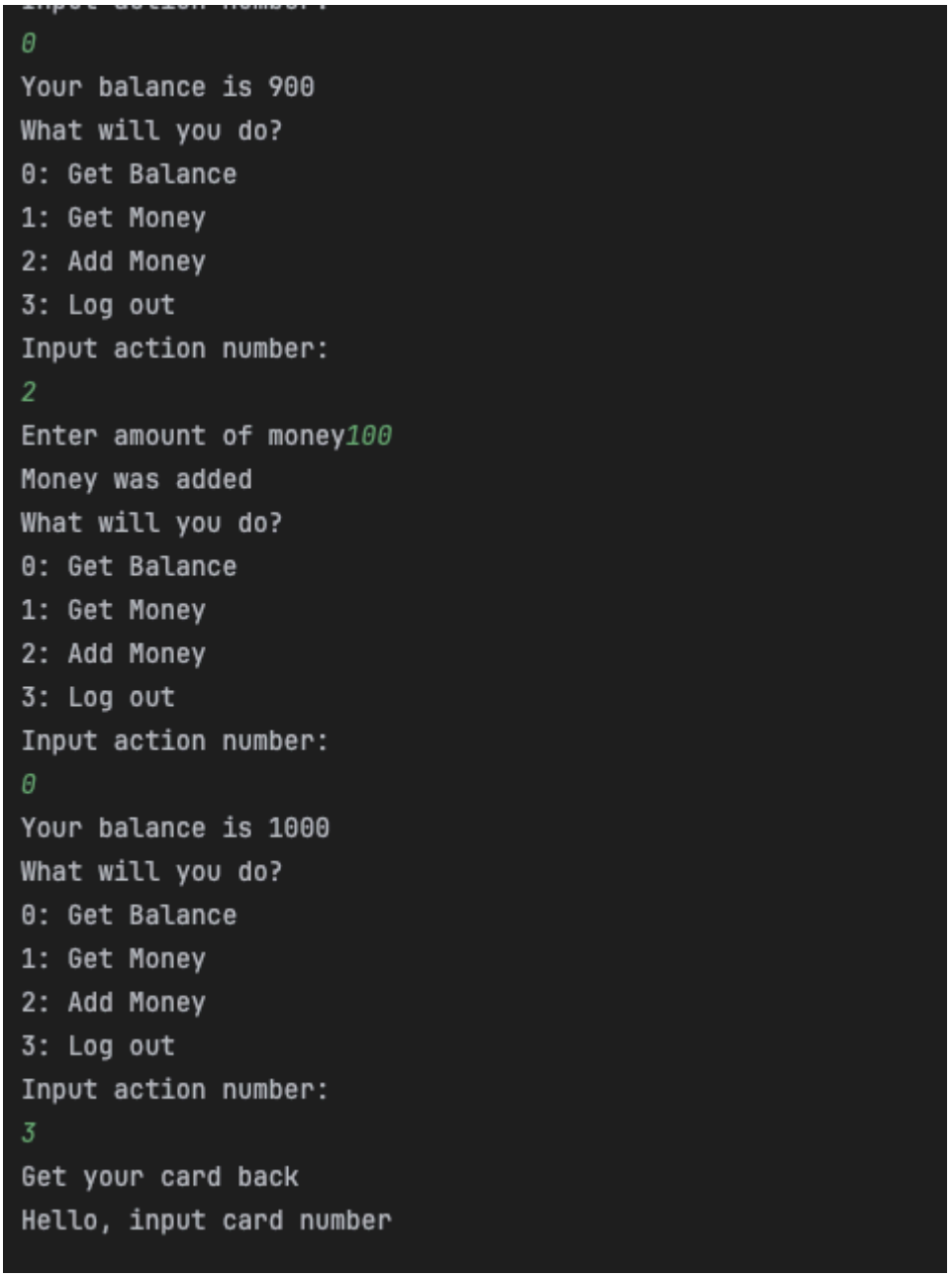
```
    def get_money(self, card_number, amount):
        card = self._cards[card_number]
        card.get_money(amount)

    def get_balance(self, card_number: str) -> int:
        return self._cards[card_number].get_balance()


class Terminal:
    _actions = {
        '0': 'Get Balance',
        '1': 'Get Money',
        '2': 'Add Money',
        '3': 'Log out',
    }

    def __init__(self, bank_instance: Bank = None):
        self.bank = bank_instance if bank_instance is not None else Bank()
        pass

    def start_terminal(self):
        while True:
            card_number = input('Hello, input card number\n')

            while not self._check_card_number(card_number):
                print('Card does not exist, please input valid card number')
                card_number = input('Input card number, please\n')

            if not self.bank.validate_user(card_number):
                print('See you next time')
                continue

            self.chose_action(card_number)

    def chose_action(self, card_number: str):
        print('What will you do?')
        for action_id, text in self._actions.items():
            print(f'{action_id}: {text}')
        action = input('Input action number:\n')
        if action not in self._actions.keys():
            self.chose_action(card_number)
```

```
        if action == '0':
            self.get_balance(card_number)
            self.chose_action(card_number)
        elif action == '1':
            self.get_money(card_number)
            self.log_out()
        elif action == '2':
            self.add_money(card_number)
            self.chose_action(card_number)
        else:
            self.log_out()

    def get_money(self, card_number: str):
        amount = int(input('Enter amount of money'))

        while amount < 0:
            amount = int(input('Enter correct amount of money'))

        if not self.bank.can_get_money(card_number, amount):
            print('Looks like you don\'t get enough money')
            self.get_money(card_number)

        self.bank.get_money(card_number, amount)

    def add_money(self, card_number: str):
        amount = int(input('Enter amount of money'))

        while amount < 0:
            amount = int(input('Enter amount of money'))

        self.bank.add_money_to_card(card_number, amount)
        print('Money was added')

    @staticmethod
    def log_out():
        print('Get your card back')

    def _check_card_number(self, card_number: str) -> bool:
        return self.bank.is_card_exists(card_number)

    def get_balance(self, card_number: str):
        balance = self.bank.get_balance(card_number)
        print(f'Your balance is {balance}')
```

```
bank = Bank()


def create_default_cards():
    bank.add_card(Card('0000 0000 0000 0000', '0000', 0))
    bank.add_card(Card('1111 1111 1111 1111', '1111', 1000))
    bank.add_card(Card('2222 2222 2222 2222', '2222', 2000))
    bank.add_card(Card('3333 3333 3333 3333', '3333', 3000))
    bank.add_card(Card('4444 4444 4444 4444', '4444', 4000))
    bank.add_card(Card('5555 5555 5555 5555', '5555', 5000))


create_default_cards()

Terminal(bank).start_terminal()
```

Наступним кроком протесутуй його

```
Hello, input card number
1
Card does not exist, please input valid card number
Input card number, please
0000 0000 0000 0000
Enter your password: 1
Enter your password: 2
Enter your password: 3
Your card is blocked
See you next time
Hello, input card number
0000 0000 0000 0000
Your card is blocked
See you next time
Hello, input card number
1111 1111 1111 1111
Enter your password: 1111
What will you do?
0: Get Balance
1: Get Money
2: Add Money
3: Log out
Input action number:
0
Your balance is 1000
What will you do?
0: Get Balance
1: Get Money
2: Add Money
3: Log out
Input action number:
1
Enter amount of money100
Get your card back
Hello, input card number
1111 1111 1111 1111
Enter your password: 1111
What will you do?
0: Get Balance
1: Get Money
2: Add Money
3: Log out
Input action number:
0
Your balance is 900
What will you do?
0: Get Balance
1: Get Money
```



В результаті код працює

Відповіді на контрольні запитання:

1. Який мінімальний набір компонентів банкомату?
    a. Операції з поповнення рахунку
        i. Поповнення власного рахунку: Ви можете внести готівку на свій рахунок через банкомат, який обладнаний для прийому банкнот.
        ii. Поповнення рахунку іншої особи: Можливість поповнити рахунок іншої людини, якщо така функція підтримується вашим банком.
    b. Операції з переказу коштів
        i. Переказ коштів між власними рахунками: Ви можете перевести гроші з одного свого рахунку на інший.
        ii. Переказ коштів на рахунок іншої особи: Можливість переказати гроші на рахунок іншої людини за її реквізитами.
    c. Операції з отримання інформації
        i. Перевірка балансу: Ви можете перевірити залишок коштів на своєму рахунку.
        ii. Отримання виписки по рахунку: Деякі банкомати дозволяють роздрукувати міні-виписку, яка відображає останні операції по рахунку.
        iii. Отримання інформації про тарифи: Ви можете ознайомитися з актуальними тарифами на банківські послуги.
    d. Інші операції
        i. Зміна PIN-коду: Ви можете змінити свій PIN-код для додаткової безпеки.
        ii. Блокування картки: У разі втрати або крадіжки картки ви можете заблокувати її через банкомат.

iii. Оплата послуг: Деякі банкомати дозволяють оплачувати комунальні послуги, штрафи та інші платежі.

iv. Отримання готівки під розписку: Ця послуга дозволяє отримати невелику суму готівки без використання картки, за умови пред'явлення документа, що посвідчує особу.

2. Що являє собою стандарт CEN/XFS?

   CEN/XFS (Extended Financial Services) – це міжнародний стандарт, який визначає інтерфейси для взаємодії між різними компонентами банкомату. Цей стандарт дозволяє різним виробникам створювати сумісні компоненти, що спрощує розробку, обслуговування та модернізацію банкоматів.

   Основні функції CEN/XFS:

   a. Стандартизація функціональності компонентів банкомату.
   b. Спрощення інтеграції нового обладнання.
   c. Забезпечення безпеки та надійності роботи банкомату.
   d. Підвищення сумісності різних систем.

3. Що таке стейти?

   Термін "стейти" у контексті банкоматів може мати кілька значень. Найчастіше він використовується для опису:

   a. Станів транзакції: Кожна транзакція проходить через кілька станів: ініціалізація, введення PIN-коду, вибір операції, завершення.
   b. Станів банкомату: Банкомат може перебувати в різних станах: робочий, очікування, несправний тощо.
   c. Станів компонентів: Кожен компонент банкомату може мати свої стани (наприклад, принтер може бути готовий до друку або потребувати заправки паперу).

4. Як влаштований екран банкомату?

   Екран банкомату – це зазвичай рідкокристалічний дисплей (LCD) або

більш сучасний OLED-дисплей. Він призначений для відображення інформації для користувача, такої як:

a. Інструкції щодо проведення операцій.
b. Повідомлення про стан операції.
c. Рекламні матеріали.
d. Інформація про баланс рахунку.