МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра Інформаційної Безпеки

Системна інженерія

Індивідуальний проект

Перевірив:

_____

Виконав:

студент II курсу

групи ФБ-01

Сахній Н.Р.

Київ 2022

Завдання на індивідуальний проект

Метою індивідуального проекту з дисципліни «Системний інжиніринг» є побудова об’єктно-орієнтованої моделі системи на основі специфікації на мові SysML.

Основою проекту є діаграми, що описують систему, які були зроблені в ході виконання лабораторних робіт №1-3. Використовуючи даний матеріал необхідно:

Максимально детально описати подану систему на *будь-якій* мові програмування, що підтримує об’єктно-орієнтований підхід.

Критерії оцінювання проекту:

- коректність введених класів (правильне застосування ООП)
- детальність та вичерпність опису

Хід виконання роботи

Дана система: Комп’ютер.

1. Опис системи.

Для початку визначимо базові властивості (атрибути) та функції (методи) поданої системи. Це буде ***ескіз*** нашої системи.

Приклад мінімального опису.

- технічні характеристики: тип комп’ютера (моноблок, стаціонарний комп’ютер, лептоп та ін.), операційна система (Linux, Windows, MacOS та ін.), розмір дисплею (вказувати в дюймах).
- функціональні можливості: запустити комп’ютер, пройти автентифікацію, змінити користувача.

За допомогою мови програмування Python опишемо клас Комп’ютер:

```
class Computer:
    """Class that describes Computer."""
    def __init__(self, computer_type, os_type, display_size):
        self.computer_type = computer_type
        self.os_type = os_type
        self.display_size = display_size

    def start(self):
        pass

    def log_in(self):
        pass

    def change_user(self):
        pass
```

Приклад більш детального опису

Комп'ютер може: ввімкнутися або вимкнутися, пройти автентифікацію, змінити користувача, вводити або зберігати або обробляти або виводити інформаційні дані, використовувати прикладні програми, створювати або відкривати або зберігати або переглядати файли.

Комп'ютер не працює сам по собі: є користувачі (прості користувачі з обмеженими правами та адміністратори з усіма можливими правами у системі), які також є атрибутами комп'ютера.

Зафіксуємо означене у вигляді нового класу:

```
class Computer:
    """Class that describes Computer."""
    def __init__(self, admin, users, computer_type, os_type, display_size):
        self.admin = driver
        self.users = users
        self.computer_type = computer_type
        self.os_type = os_type
        self.display_size = display_size

    def start(self):
        """Turn on computer."""
        pass

    def stop(self):
        """Turn off computer."""
        shutdown = input("Do you wish to shutdown your computer ? (yes / no):")

        if shutdown == 'no':
              exit()
        else:
              os.system("shutdown /s /t 1")
        pass

    def log_in(self):
        """Log in system."""
        pass

    def change_user(self):
        """Change current user."""
        pass

    def data_interaction(self, action):
        """The computer can input, storage, processing, output data."""
        assert action in ('input', 'storage', 'processing', 'output')

    def use_apps(self):
        """Use any programm."""
        pass


    def file_interaction(self, action):
        """The computer can create, open, save, view files."""
        assert action in ('create', 'open', 'save', 'view')
```

2. Опис акторів.

Як вже було зазначено, Комп'ютер взаємодіє із користувачами, а головним серед них є адміністратор.

Опис сутності Користувачі:

```
class Users:
    """Class that describes Users."""
    def __init__(self, username, password, admin_permission = False):
        self.username = username
        self.password = password
        if admin_permission = True:
              print("Congratulation!!! Everything in your hands")
```

Основна ідея такого підходу: адміністартор та простий користувач комп'ютера це об'єкти одного класу Users, при цьому відмінною характеристикою адміністратора з-поміж інших користувачів є наявність адміністративних прав на користування комп'ютером, а простим користувачем може бути будь-хто, тобто ніяких додаткових обмежень не потрібно.

Примітка: *пояснення для тих, хто не знайомий з мовою програмування Python:*

__init__-- метод, що викликається при ініціалізації об'єкту певного класу (один з «магічних» методів мови програмування Python)

***kwargs -- будь-які іменовані параметри (параметри, які можна передавати в функцію тільки разом з їх ім'ям)*

3. Опис підсистем.

Наступним етапом виконання проекту є опис підсистем даної системи та їх взаємодії.

Опишемо підсистему **апаратного забезпечення** комп'ютера ↓

```
class HardwareSystem:
    def __init__(self):
        self.current_action = 'waiting any action with Hardware System'

    def input(self):
        self.current_action = 'input these data in system'

    def storage(self):
        self.current_action = 'storage our data in HDD or SDD'

    def processing(self):
        self.current_action = 'processing all data'

    def output(self):
        self.current_action = 'output data'
```

Опишемо підсистему **програмного забезпечення** комп'ютера ↓

```
class SoftwareSystem:
    def __init__(self):
        self.current_action = waiting any action with Software System'

    def log_in(self):
        self.current_action = 'log in system with correct username and password'

    def change_user(self):
        self.current_action = 'change current user on another one'

    def use_app(self):
        self.current_action = 'use applications for your specific work'
```

Опишемо підсистему **інформаційних ресурсів** комп'ютера ↓

```
class InfoResourceSystem:
    def __init__(self):
        self.current_action = 'waiting any action with Info-Resource System'

    def create(self):
        self.current_action = 'create any type of file'

    def open(self):
        self.current_action = 'open this file'

    def save(self):
        self.current_action = 'save information in file'

    def view(self):
        self.current_action = 'view contents of the file'
```

В клас Комп'ютер допишемо, як підсистема **апаратного забезпечення**, **програмного забезпечення** та **інформаційних ресурсів** впливає на роботу усього механізму (взаємодія з системою Комп'ютер):

```
class Computer:
    """Class that describes Computer."""

    def __init__(self, admin, users, computer_type, os_type, display_size):
        self.admin = admin
        self.users = users
        self.computer_type = computer_type
        self.os_type = os_type
        self.display_size = display_size

        self.hardware_system = HardwareSystem()
        self.software_system = SoftwareSystem()
        self.info_resource = InfoResourceSystem()

    def start(self):
        """Turn on computer."""
        global hardware_system
        global software_system
        global info_resource
        hardware_system = 'waiting any action with Hardware System'
        software_system = 'waiting any action with Software System'
        info_resource = 'waiting any action with Info-Resource System'
        pass

    def stop(self):
        """Turn off computer."""
        global hardware_system
        print(hardware_system)
        if hardware_system == 'waiting any action with Hardware System':
            shutdown = input("Do you wish to shutdown your computer? (yes/no): ")

            if shutdown == 'no':
                exit()
            else:
                os.system("shutdown /s /t 1")
        else:
            print("You cannot stop the computer because it hasn't been started yet")

        pass

    def log_in(self):
        """Log in system."""
        global software_system
        if software_system == 'waiting any action with Software System':
            self.software_system.log_in()
        else:
            print("You cannot log in the computer because it hasn't been started yet")

        pass
```

```
def change_user(self):
    """Change current user."""
    global software_system
    if software_system == 'waiting any action with Software System':
        self.software_system.change_user()
    else:
        print("You cannot change current user because computer hasn't been started yet")

    pass

def data_interaction(self, action):
    """The computer can input, storage, processing, output data."""
    assert action in ('input', 'storage', 'processing', 'output'), 'Wrong action!'

    global hardware_system
    if hardware_system == 'waiting any action with Hardware System':
        if action == 'input':
            self.hardware_system.input()
        elif action == 'storage':
            self.hardware_system.storage()
        elif action == 'processing':
            self.hardware_system.processing()
        elif action == 'output':
            self.hardware_system.output()
    else:
        print("You cannot interact with data because computer hasn't been started yet")

    pass

def use_apps(self):
    """Use any program."""
    global software_system
    if software_system == 'waiting any action with Software System':
        self.software_system.use_app()
    else:
        print("You cannot open any app for using because computer hasn't been started yet")

    pass

def file_interaction(self, action):
    """The computer can create, open, save, view files."""
    assert action in ('create', 'open', 'save', 'view'), 'Wrong action!'

    global info_resource
    if info_resource == 'waiting any action with Info-Resource System':
        if action == 'create':
            self.info_resource.create()
        elif action == 'open':
            self.info_resource.open()
        elif action == 'save':
            self.info_resource.save()
        elif action == 'view':
            self.info_resource.view()
    else:
       print("You cannot interact with files because computer hasn't been started yet")

    pass
```

4. Демонстрація роботи системи.

Використовуючи опис нашої системи, покажемо, як працює система Комп'ютер:

a) Ініціалізація об’єктів

```
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1929 64 bit (AMD64)] on win32
>>>
>>> from computer_system import Computer, Users
>>> admin = Users(username='Nazar24', password='dfv8e4#-L!', admin_permission=True)
Congratulation!!! Everything in your hands
>>> other_users = [Users(username='Fred_sample', password='QWERTY75!@'),
...                Users(username='Alex HV', password='TRo_fhi4a-28'),
...                Users(username='John Armstrong', password='cosmood_iiu')]
...
>>> my_computer = Computer(admin, other_users, "Laptop", "Windows 10", '15,6"')
>>>
```

b) Робота системи Комп'ютер

```
>>>
>>> my_computer.log_in()  # Спробуємо розпочати процес авнтентифікації, проте через те, що комп'ютер ще не увімкнений, отримаємо помилку
You cannot log in the computer because it hasn't been started yet
>>> my_computer.data_interaction('processing')  # Аналогічно щодо можливості процесу обробки даних
You cannot interact with data because computer hasn't been started yet
>>>
>>> my_computer.start()  # Отже, спочатку треба запустити комп'ютер, щоб можна було виконювати необхідні завдання
>>>
>>> my_computer.log_in()  # Пробуємо розпочати процес авторизуваці знову ...
>>> my_computer.software_system.current_action  # Так як виклик функції авторизації пройшов успішно, то можемо перевірити стан системи
'log in system with correct username and password'
>>>
>>> my_computer.data_interaction('processing')  # Знову перевіряємо можливость почати процес обробки даних
>>> my_computer.hardware_system.current_action  # Так як процес обробки даних розпочався, то можемо перевірити стан системи
'processing all data'
>>>
```

```
>>>
>>> my_computer.change_user()  # Викликаємо процес зміни поточного користувача
>>> my_computer.software_system.current_action
'change current user on another one'
>>>
>>> my_computer.use_apps()  # Починаємо використовувати прикладну програму для роботи з файлами
>>> my_computer.software_system.current_action
'use applications for your specific work'
>>>
```

```
>>>
>>> my_computer.file_interaction('nothing')  # Помилково вказуємо не правильну дію, через що отримаємо помилку
Traceback (most recent call last):
  File "<input>", line 1, in <module>
  File "D:\KPI\CI\Індивідуальний проект\Individual Project\computer_system.py", line 164, in file_interaction
    assert action in ('create', 'open', 'save', 'view'), 'Wrong action!'
AssertionError: Wrong action!
>>> my_computer.info_resource.current_action
'waiting any action with Info-Resource System'
>>>
>>> my_computer.file_interaction('create')  # А цього разу правильно викликаємо процес створення файлу
>>> my_computer.info_resource.current_action
'create any type of file'
>>>
```

Додаток. Програмний код системи Комп'ютер.

```
import os

global hardware_system
global software_system
global info_resource


class Users:
    """Class that describes Users."""

    def __init__(self, username, password, admin_permission=False):
        self.username = username
        self.password = password
        if admin_permission:
            print("Congratulation!!! Everything in your hands")

    pass


class HardwareSystem:
    def __init__(self):
        self.current_action = 'waiting any action with Hardware System'

    def input(self):
        self.current_action = 'input these data in system'

    def storage(self):
        self.current_action = 'storage our data in HDD or SDD'

    def processing(self):
        self.current_action = 'processing all data'

    def output(self):
        self.current_action = 'output data'

    pass


class SoftwareSystem:
    def __init__(self):
        self.current_action = 'waiting any action with Software System'

    def log_in(self):
        self.current_action = 'log in system with correct username and password'

    def change_user(self):
        self.current_action = 'change current user on another one'

    def use_app(self):
        self.current_action = 'use applications for your specific work'

    pass
```

```
class InfoResourceSystem:
    def __init__(self):
        self.current_action = 'waiting any action with Info-Resource System'

    def create(self):
        self.current_action = 'create any type of file'

    def open(self):
        self.current_action = 'open this file'

    def save(self):
        self.current_action = 'save information in file'

    def view(self):
        self.current_action = 'view contents of the file'

    pass


class Computer:
    """Class that describes Computer."""

    def __init__(self, admin, users, computer_type, os_type, display_size):
        self.admin = admin
        self.users = users
        self.computer_type = computer_type
        self.os_type = os_type
        self.display_size = display_size

        self.hardware_system = HardwareSystem()
        self.software_system = SoftwareSystem()
        self.info_resource = InfoResourceSystem()

       global hardware_system
       hardware_system = "nothing"
       global software_system
       software_system = "nothing"
       global info_resource
       info_resource = "nothing"


    def start(self):
        """Turn on computer."""
        global hardware_system
        global software_system
        global info_resource
        hardware_system = 'waiting any action with Hardware System'
        software_system = 'waiting any action with Software System'
        info_resource = 'waiting any action with Info-Resource System'
        pass

    def stop(self):
        """Turn off computer."""
        global hardware_system
        print(hardware_system)
        if hardware_system == 'waiting any action with Hardware System':
            shutdown = input("Do you wish to shutdown your computer? (yes/no): ")

            if shutdown == 'no':
                exit()
            else:
                os.system("shutdown /s /t 1")
        else:
            print("You cannot stop the computer because it hasn't been started yet")
        pass
```

```
def log_in(self):
    """Log in system."""
    global software_system
    if software_system == 'waiting any action with Software System':
        self.software_system.log_in()
    else:
        print("You cannot log in the computer because it hasn't been started yet")
    pass

def change_user(self):
    """Change current user."""
    global software_system
    if software_system == 'waiting any action with Software System':
        self.software_system.change_user()
    else:
        print("You cannot change current user because computer hasn't been started yet")
    pass

def data_interaction(self, action):
    """The computer can input, storage, processing, output data."""
    assert action in ('input', 'storage', 'processing', 'output'), 'Wrong action!'

    global hardware_system
    if hardware_system == 'waiting any action with Hardware System':
        if action == 'input':
            self.hardware_system.input()
        elif action == 'storage':
            self.hardware_system.storage()
        elif action == 'processing':
            self.hardware_system.processing()
        elif action == 'output':
            self.hardware_system.output()
    else:
        print("You cannot interact with data because computer hasn't been started yet")

    pass

def use_apps(self):
    """Use any program."""
    global software_system
    if software_system == 'waiting any action with Software System':
        self.software_system.use_app()
    else:
        print("You cannot open any app for using because computer hasn't been started yet")
    pass

def file_interaction(self, action):
    """The computer can create, open, save, view files."""
    assert action in ('create', 'open', 'save', 'view'), 'Wrong action!'

    global info_resource
    if info_resource == 'waiting any action with Info-Resource System':
        if action == 'create':
            self.info_resource.create()
        elif action == 'open':
            self.info_resource.open()
        elif action == 'save':
            self.info_resource.save()
        elif action == 'view':
            self.info_resource.view()
    else:
        print("You cannot interact with files because computer hasn't been started yet")

pass
```