Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №2

Технологія розробки програмного забезпечення

«ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ. СЦЕНАРІЇ ВАРІАНТІВ ВИКОРИСТАННЯ. ДІАГРАМИ UML. ДІАГРАМИ КЛАСІВ. КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ»

Варіант 28

Виконала:
студентка групи ІА–13
Хижняк Валерія Валеріївна

Перевірив:
Мягкий М.Ю.

Київ 2023

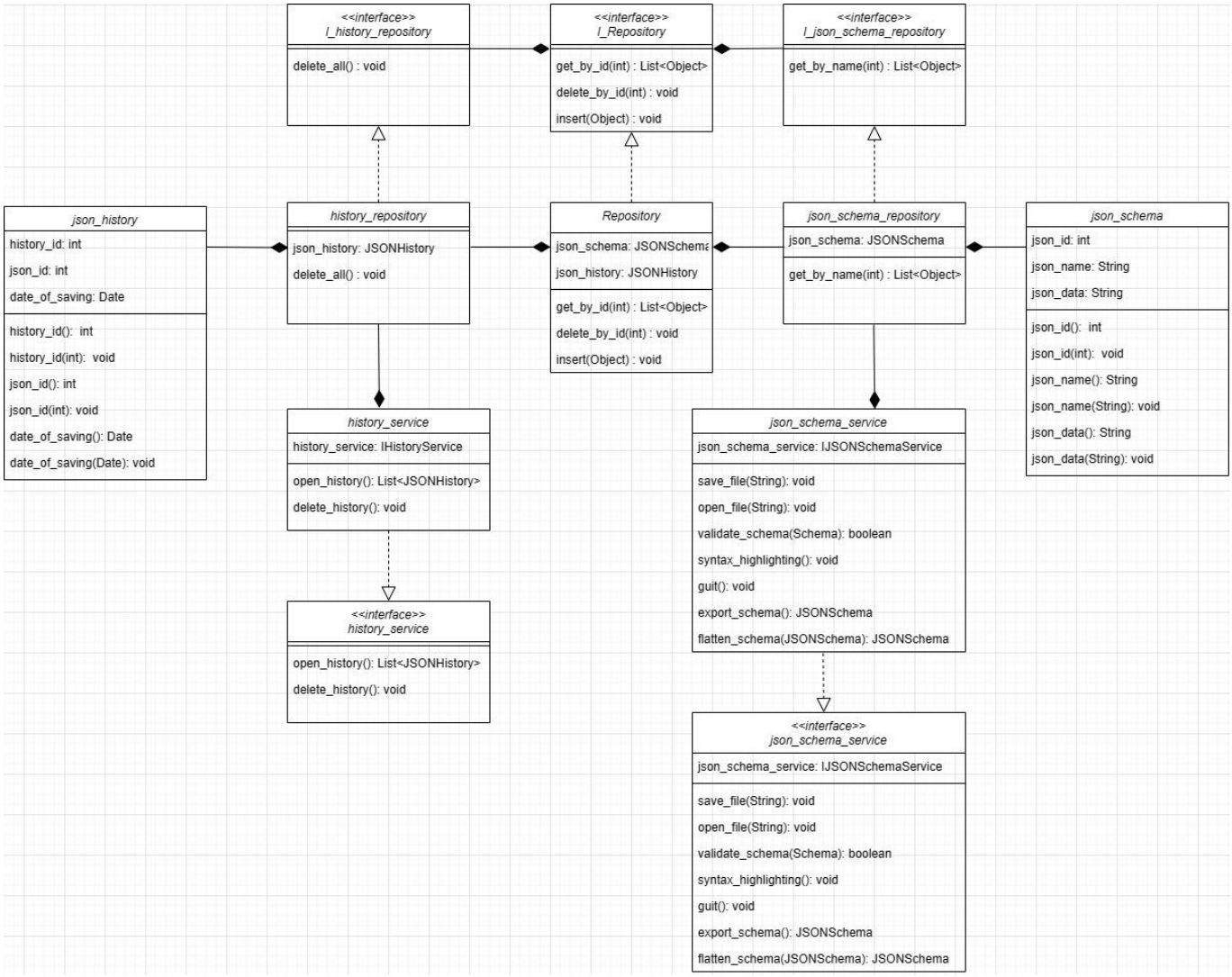
Тема: Діаграма варінтів використання. Сценарії варінтів використання. Діаграми UML. Діаграми класів. Концептуальна модель системи.

Мета: навчитися створювати діаграму варіантів використання, сценарії варіантів використання, UML діаграми, діаграми класів, концептуальну модель системи

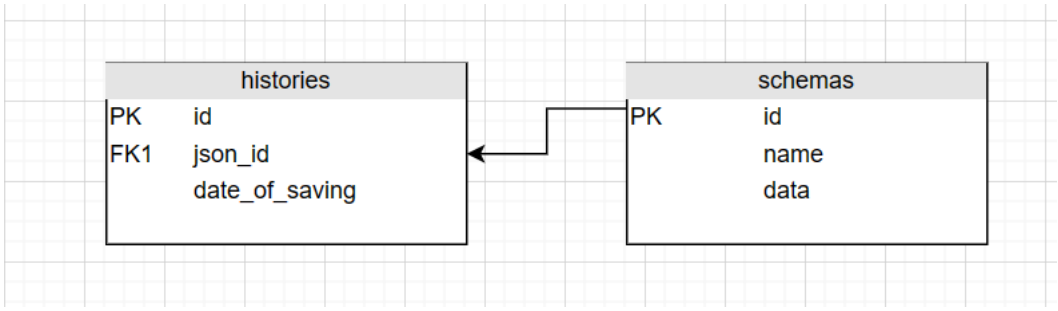
Хід роботи:

1. Діаграма прецедентів:

2. Діаграма класів

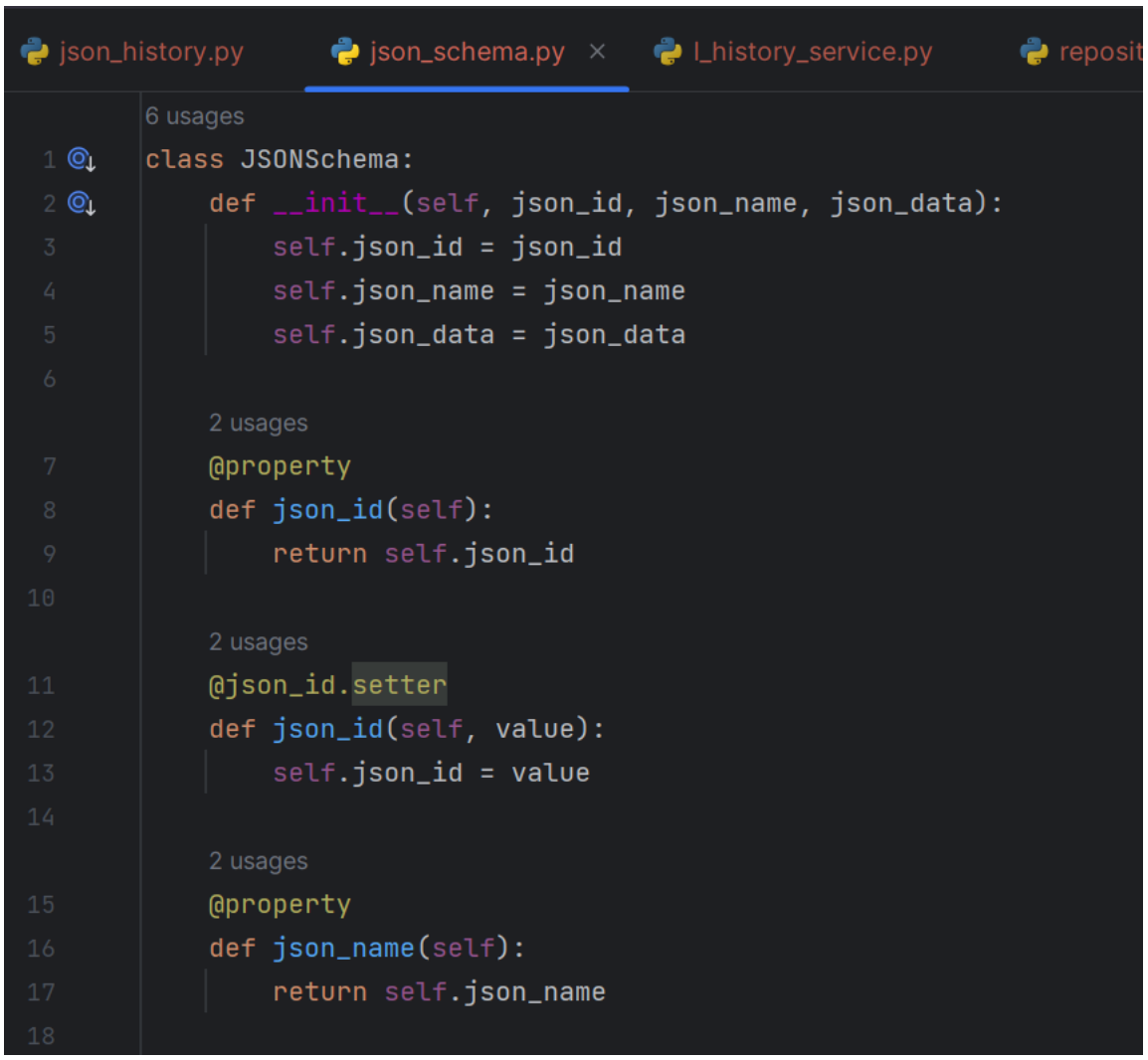


3. Схема БД:



4. Код програми:

```python
6 usages
class JSONHistory:
    def __init__(self, history_id, json_id, date_of_saving):
        self.history_id = history_id
        self.json_id = json_id
        self.date_of_saving = date_of_saving

    2 usages
    @property
    def history_id(self):
        return self.history_id

    2 usages
    @history_id.setter
    def history_id(self, value):
        self.history_id = value

    2 usages
    @property
    def json_id(self):
        return self.json_id

    2 usages
    @json_id.setter
    def json_id(self, value):
        self.json_id = value

    2 usages
    @property
    def date_of_saving(self):
        return self.date_of_saving

    2 usages
    @date_of_saving.setter
    def date_of_saving(self, value):
        self.date_of_saving = value.upper()
```


json_history.py
json_schema.py
I_history_service.py
reposit
6 usages
class JSONSchema:
def __init__(self, json_id, json_name, json_data):
self.json_id = json_id
self.json_name = json_name
self.json_data = json_data
2 usages
@property
def json_id(self):
return self.json_id
2 usages
@json_id.setter
def json_id(self, value):
self.json_id = value
2 usages
@property
def json_name(self):
return self.json_name


2 usages
@json_name.setter
def json_name(self, value):
self.json_name = value.upper()
2 usages
@property
def json_data(self):
return self.json_data
2 usages
@json_data.setter
def json_data(self, value):
self.json_data = value.upper()

json_history.py | json_schema.py | history_repository.py | I_history_service.

```python
from I_history_repository import IHistoryRepository
from repository import Repository
from app.models.json_history import JSONHistory


class HistoryRepository(IHistoryRepository, Repository, JSONHistory):
    def __init__(self, json_history: JSONHistory):
        super().__init__()
        self.json_history = json_history

    def delete_all(self):
        pass
```

I_history_repository.py | I_history_service.py | repository.py

```python
from abc import *
from I_repository import IRepository


4 usages
class IHistoryRepository(IRepository):

    @abstractmethod
    def delete_all(self):
        pass
```

I_history_repository.py | I_json_schema_repository.py | I_history_serv

```python
from abc import *
from I_repository import IRepository


4 usages
class IJSONSchemaRepository(IRepository):

    @abstractmethod
    def get_by_name(self):
        pass
```

I_history_repository.py I_json_schema_repository

```python
from abc import ABC, abstractmethod


6 usages
class IRepository(ABC):

    @abstractmethod
    def get_by_id(self):
        pass

    @abstractmethod
    def delete_by_id(self):
        pass

    @abstractmethod
    def insert(self):
        pass
```

I_repository.py json_schema_repository.py I_history_service.py repository.py

```python
from I_json_schema_repository import IJSONSchemaRepository
from repository import Repository
from app.models.json_schema import JSONSchema


class JSONSchemaRepository(IJSONSchemaRepository, Repository, JSONSchema):
    def __init__(self, json_schema: JSONSchema):
        super().__init__()
        self.json_schema = json_schema

    def get_by_name(self):
        pass
```

I_repository.py | json_schema_repository.py | I_history_service.py | repository.py

```python
from I_repository import IRepository
from app.models.json_history import JSONHistory
from app.models.json_schema import JSONSchema


4 usages
class Repository(IRepository, JSONSchema, JSONHistory):
    def __init__(self, json_schema: JSONSchema, json_history: JSONHistory):
        super().__init__()
        self.json_schema = json_schema
        self.json_history = json_history

    def get_by_id(self):
        pass

    def delete_by_id(self):
        pass

    def insert(self):
        pass
```

I_history_service.py | repository.py | history_service.py

```python
from I_history_service import IHistoryService
from app.repositories.I_history_repository import IHistoryRepository


class JSONSchemaService(IHistoryService):

    def __init__(self, history_repository: IHistoryRepository):
        super().__init__()
        self.history_repository = history_repository

    def open_history(self):
        pass

    def delete_history(self):
        pass
```

I_history_service.py × repository.py history_service.py

```python
from abc import ABC, abstractmethod


2 usages
class IHistoryService(ABC):

    @abstractmethod
    def open_history(self):
        pass

    @abstractmethod
    def delete_history(self):
        pass
```

I_history_service.py I_json_schema_service.py × repos

```python
from abc import ABC, abstractmethod


2 usages
class IJSONSchemaService(ABC):

    @abstractmethod
    def save_file(self):
        pass

    @abstractmethod
    def auto_save(self):
        pass

    @abstractmethod
    def validate_schema(self):
        pass

    @abstractmethod
    def syntax_highlighting(self):
        pass
```

```python
    @abstractmethod
    def quit(self):
        pass

    @abstractmethod
    def export_schema(self):
        pass

    @abstractmethod
    def flatten_schema(self):
        pass
```

I_history_service.py | I_json_schema_service.py | json_schema_service.py | reposit

```python
import tkinter as tk
from tkinter.filedialog import *
from I_json_schema_service import IJSONSchemaService
from app.repositories.I_json_schema_repository import IJSONSchemaRepository


class JSONSchemaService(tk.Tk, IJSONSchemaService):
    def __init__(self, schema_repository: IJSONSchemaRepository):
        super().__init__()
        self.schema_repository = schema_repository

    def open_file(self):...

    def save_file(self):
        pass

    def auto_save(self):
        pass

    def validate_schema(self):
        pass

    def syntax_highlighting(self):
        pass
```

```python
    def quit(self):
        pass

    def export_schema(self):
        pass

    def flatten_schema(self):
        pass
```

I_json_schema_view.py × repository.py history_service.py

```python
from abc import ABC, abstractmethod


1 usage
class IJSONSchemaView(ABC):

    @abstractmethod
    def create_menu(self):
        pass

    @abstractmethod
    def create_schema_frame(self):
        pass

    @abstractmethod
    def create_toolbar(self):
        pass
```

I_json_schema_view.py | json_schema_view.py × | repository.py | history_service.py

```python
import tkinter as tk
from tkinter import ttk, scrolledtext
import ctypes


# Створюємо клас для нашого додатку
2 usages
class JSONSchemaView(tk.Tk):
    # Ініціалізуємо атрибути класу
    def __init__(self):
        super().__init__()
        # Встановлюємо назву вікна
        self.title("JSON Schema Editor")
        ctypes.windll.shcore.SetProcessDpiAwareness(True)
        # Встановлюємо розмір вікна
        self.geometry("1200x600")
        # Створюємо меню для вікна
        self.create_menu()
        # Створюємо панель іннструментів для редагування JSON - схеми
        self.create_toolbar()
        # Створюємо рамку для відображення JSON-схеми
        self.create_schema_frame()
```

```python
    # Створюємо меню для вікна
    1 usage
    def create_menu(self):
        # Створюємо об'єкт меню
        self.menu = tk.Menu(self)
        # Додаємо пункт меню "Файл"
        self.file_menu = tk.Menu(self.menu, tearoff=0)
        # self.file_menu.add_command(label="Відкрити", command=self.open_file)
        # self.file_menu.add_command(label="Зберегти", command=self.save_file)
        # self.file_menu.add_command(label="Вийти", command=self.quit)
        self.menu.add_cascade(label="Файл", menu=self.file_menu)
        # Встановлюємо меню для вікна
        self.config(menu=self.menu)

```

```python
    # Створюємо панель інструментів для редагування JSON-схеми
    1 usage
    def create_schema_frame(self):
        # Створюємо об'єкт рамки
        self.schema_frame = tk.LabelFrame(self, text="JSON-схема", padx=25, pady=20, font='25')
        # Створюємо об'єкт текстового поля з прокруткою для редагування JSON-схеми
        self.schema_text = scrolledtext.ScrolledText(self.schema_frame, width=40, height=20, wrap=tk.NONE
        # Розташовуємо текстове поле у рамці
        self.schema_text.pack(fill=tk.BOTH, expand=True)
        # Розташовуємо рамку у лівій частині вікна
        self.schema_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

    1 usage
    def create_toolbar(self):
        # Створюємо об'єкт панелі інструментів
        self.toolbar = tk.Frame(self, bd=1, relief=tk.RAISED)
        self.toolbar.pack(side=tk.TOP, fill=tk.X)
```

main.py repository.py history_service.py

```python
# Імпортуємо клас JSONSchemaView з файлу json_schema_view.py
from app.views.json_schema_view import JSONSchemaView

# Створюємо об'єкт додатку
app = JSONSchemaView()


# Запускаємо додаток
app.mainloop()
```

Висновок: на даній лабораторній роботі я побудувала діаграму класів для реалізованої частини системи, вибрала прецеденти і написала на їх основі прецеденти, розробила основні класи і структуру системи баз даних.