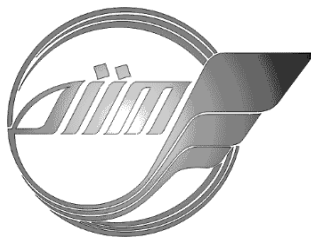


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп'ютерні інформаційні технології»

Лабораторна робота №7

з дисципліни «Архітектура та проектування програмних засобів»

на тему: «Генерація коду»

Виконав:
студент гр.ПЗ1911
Сафонов Д. Є.
Прийняла:
Куруп'ятник О. С.

Дніпро, 2020

Тема. Генерація коду.

Мета. Ознайомитися із взаємозв'язком діаграм та етапами генерації коду. Отримати практичні навички генерації коду для UML-моделі.

Постановка задачі згідно завдання.

Для проекту, який розроблявся протягом лабораторних робіт № 1 – 6, виконати генерацію коду. Визначити відповідності між кодом та елементами моделі: на прикладах коду продемонструвати всі застосовані специфікатори до елементів класу, відношення асоціації (композиції, агрегації), спадкування. Доповнити модель компонентами для створення, видалення об'єктів, доступу до елементів.

Згенерований код з поясненнями.

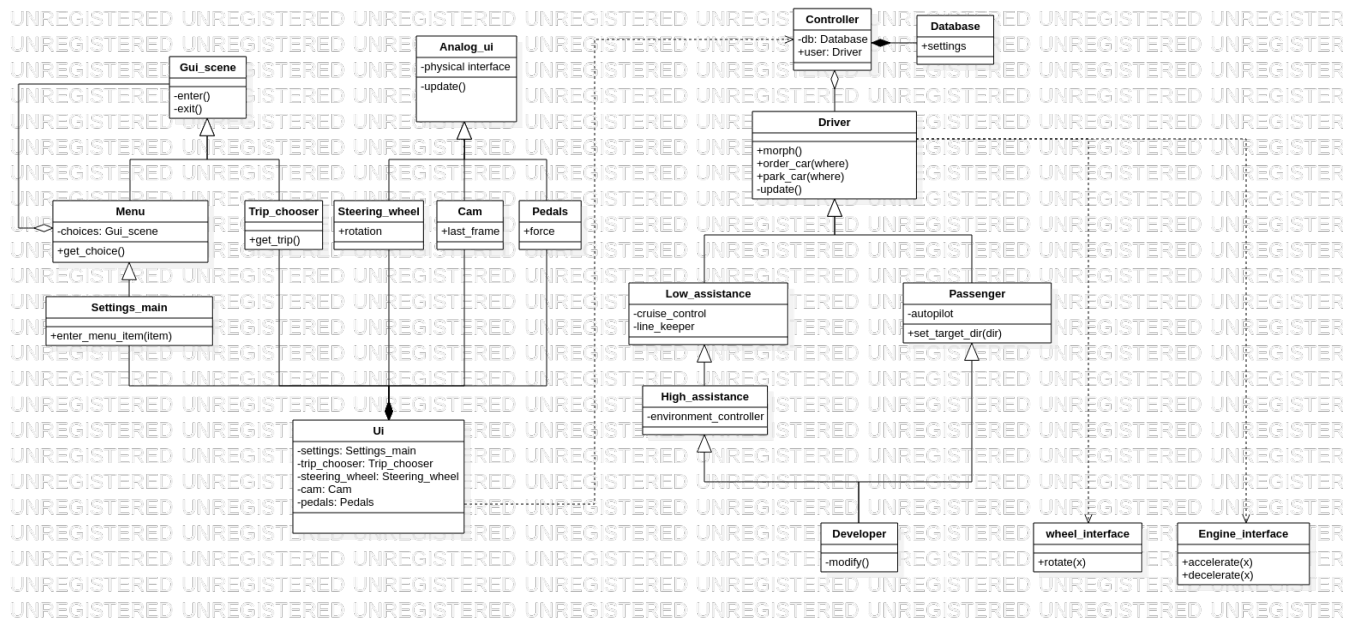


Рисунок 1 (Діаграма класів з ЛР2)

“Menu.py”

```
#!/usr/bin/python
#-*- coding: utf-8 -*-

from Gui_scene import Gui_scene

class Menu(Gui_scene):#Menu extends Gui_scene
    def __init__(self):#initialiazer
        self.choices = None#pseudo private attribute, may use prefixes to immitate(none — public, _
        - protected, __ - private(e.g. public, _protected, __private))
        #also this variable has type Gui_scene, which is not specified because python is dynamically
        typed language

    def get_choice(self, ):#public method
        pass
```

“Analog_ui.py”

```
#!/usr/bin/python
#-*- coding: utf-8 -*-

class Analog_ui:
    def __init__(self):
        self.physical interface = None#pseudo private attribute
```

```
def update(self, ):#pseudo private method
    pass
```

“Controller.py”

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
```

```
class Controller:
    def __init__(self):
        self.db = None#pseudo private, type class Database
        self.user = None#public, because class Ui depends on it
```

“Ui.py”

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
```

```
class Ui:#depends on Controller.user
    def __init__(self):#pseudo declaration of attributes
        self.settings = None
        self.trip_chooser = None
        self.steering_wheel = None
        self.cam = None
        self.pedals = None
    def morph_user(mode);
        pass#calls Controller.user.morph(mode), also sets according value in Controller.db
```

“Пояснення”

Через те, що в обраній мові не існує вказівників також відсутня й різниця між композицією та агрегацією.

Наслідування демонструється наступним синтаксисом:

```
class Ім'я_класу(Ім'я_базового_класу).
```

Залежність ніяк не вказана через те, що в обраній мові відсутні модифікатори доступу, тож завжди є можливість отримати доступ до потрібних методів та атрибутів будь яких класів та об'єктів.

Аналіз результатів та висновки щодо відповідності елементів моделі коду.

Код відповідає моделі, загалом такий спосіб проєктування дуже зручний тому що набагато зручніше редагувати зв'язки між класами в одному файлі, ніж в багатьох, а результат той самий. Тож цей метод проєктування зберігає дуже багато часу, який можна витратити наприклад на реалізацію методів.