

Міністерство освіти і науки України
НТУУ «Київський політехнічний інститут»
Фізико-технічний інститут

Програмування

Комп'ютерний практикум №7

Використання об'єктно-орієнтованого підходу для розробки програмного забезпечення

Варіант №14

Виконав:

Студент 1 курсу ФТІ

Групи ФІ-92

Поночевний Назар Юрійович

Перевірив:

Прийняв:

Використання об'єктно-орієнтованого підходу для розробки програмного забезпечення

Мета роботи: засвоїти принципи об'єктно-орієнтованого проектування, включаючи аналіз предметної області та побудову фізичних і логічних моделей.

Завдання:

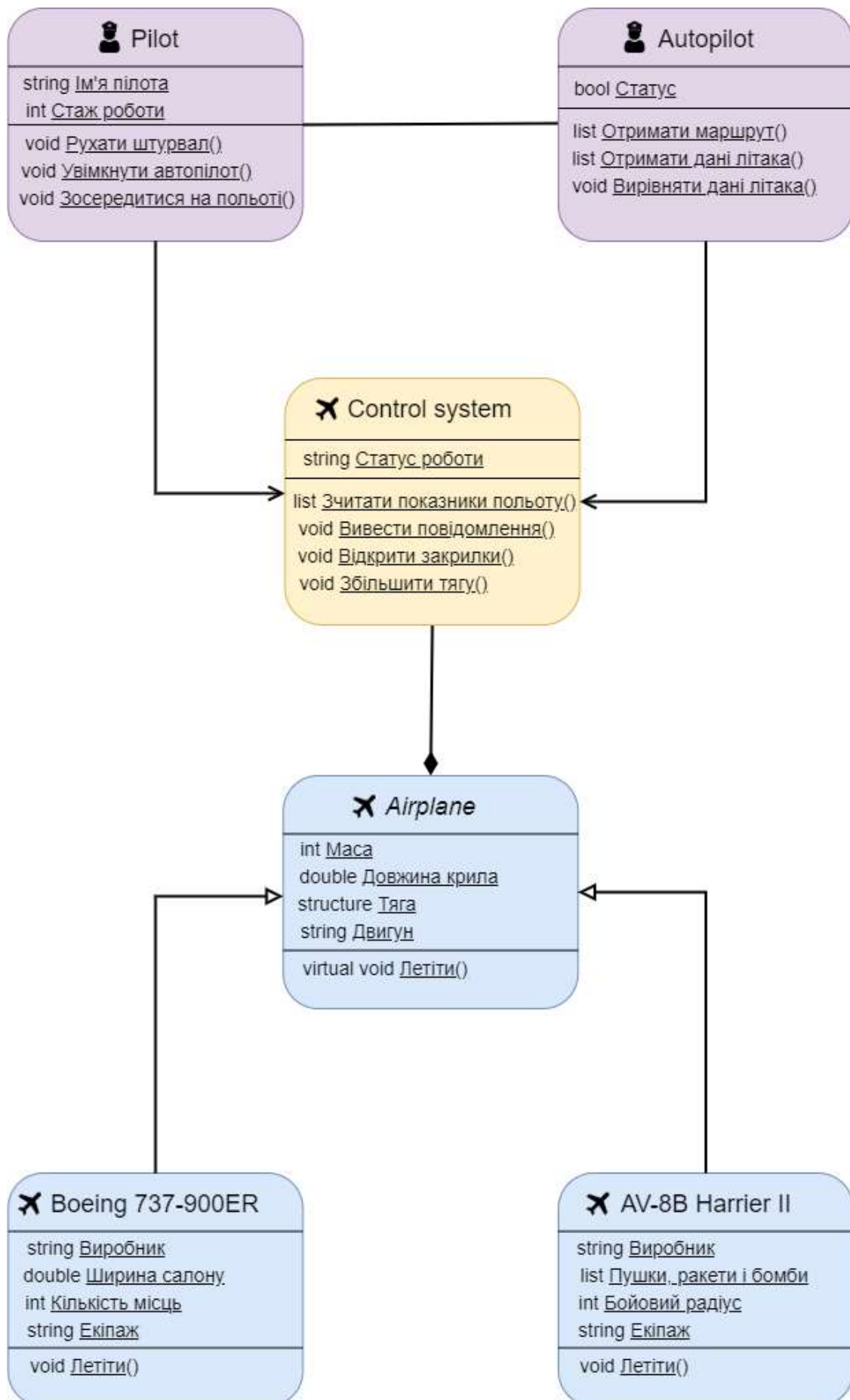
14. Розробити проект автопілота літака.

Актори	Сюжетна лінія
Пілот	Пілот сідає у літак, виконує зліт літака і вмикає автопілот, який за допомогою системи керування підтримує необхідні показники літака під час польоту
Автопілот	
Система керування	
Літак	

Внутрішні / зовнішні методи

Актор	Метод	Зовнішній	Внутрішній
Пілот	Рухати штурвал	✓	
	Говорити з пасажирами	✓	
	Увімкнути автопілот	✓	
	Зосередитися на польоті		✓
Автопілот	Отримати маршрут	✓	
	Вирівняти параметри літака	✓	
	Провести калібрування		✓
Система керування	Зчитувати показники польоту	✓	
	Виводити повідомлення	✓	
	Відкрити закрилки	✓	
	Спустити шасі	✓	
	Збільшити тягу	✓	

Class diagram



Відношення між класами і поліморфізм

Клас Pilot:

- двостороння асоціація з класом Autopilot;
- використання класу Control system.

Клас Autopilot:

- двостороння асоціація з класом Pilot;
- використання класу Control system.

Клас Control system:

- композиція (строга агрегація) з класом Airplane (пояснення: літак стає некерованим без системи керування, тому система керування є необхідною частиною літака).

Клас Boeing 737-900ER:

- відношення успадкування від абстрактного класу Airplane і перевизначення методу fly.

Клас AV-8B Harrier II:

- відношення успадкування від абстрактного класу Airplane і перевизначення методу fly.

Поліморфізм:

У файлі *main.py* є функція *iteration*, яка:

1. має параметр *airplanes* - масив об'єктів класів, які були успадковані від абстрактного класу *Airplane*
2. виконує для кожного об'єкта метод *fly*, який перевизначений для кожного окремого класу

Маємо параметричний поліморфізм, який утворений за допомогою абстрактного класу *Airplane*, чисто віртуального методу *fly* та функції *iteration*, яка змінює свою поведінку (сама цього не помічаючи) в залежності від того, об'єкт якого класу їй дали на вхід.

Код програми

pilot.py

```
1 from time import sleep
2 from random import choice, randint
3
4
5 class Pilot:
6     focus_level = 0
7
```

```
8     def __init__(self, name="Vasya Pechkin", work_experience=0):
9         self.name = name
10        self.work_experience = work_experience
11
12    def __str__(self):
13        return self.name
14
15    @property
16    def work_experience(self):
17        return self.__work_experience
18
19    @work_experience.setter
20    def work_experience(self, work_experience):
21        if work_experience < 0:
22            self.__work_experience = 0
23        elif work_experience > 100:
24            self.__work_experience = 100
25        else:
26            self.__work_experience = work_experience
27
28    @staticmethod
29    def move_helm(direction, control_system):
30        control_system.open_flaps(direction)
31
32    @staticmethod
33    def pull_lever(number, control_system):
34        control_system.change_traction(number)
35
36    @staticmethod
37    def enable_autopilot(control_system, time, debug=False):
38        autopilot = Autopilot(control_system)
39        autopilot.run(time, debug)
40        return autopilot.status
41
42    def focus_on_flying(self):
43        self.focus_level += 10
44
45
46    class Autopilot:
47        status = 0
48
49        def __init__(self, control_system):
50            self.control_system = control_system
```

```

51
52     def run(self, time, debug=False):
53         self.status = 1
54         for _ in range(time):
55             route = self.get_route()
56             airplane_data = self.get_airplane_data()
57             if debug:
58                 print("\nAutopilot ask for the route from dispatcher...")
59                 print("Route from dispatcher:", route)
60                 print("Airplane traction:", airplane_data['traction'])
61                 print("Autopilot corrected airplane's traction")
62                 self.correct_airplane_data(route, airplane_data)
63                 sleep(1)
64             self.status = 0
65
66     @staticmethod
67     def get_route():
68         # actually, we need to get the route from dispatcher,
69         # but we don't want to build a complex system, so we will generate route
        randomly
70         direction = choice(["Up", "Down", "Left", "Right"])
71         power = randint(1000, 5000)
72         traction = [direction, power]
73         return traction
74
75     def get_airplane_data(self):
76         return self.control_system.read_flight_indicators()
77
78     def correct_airplane_data(self, route, airplane_data):
79         route_direction, route_power = route
80         direction, power = airplane_data['traction']
81         if direction != route_direction:
82             self.control_system.open_flaps(route_direction)
83         if power != route_power:
84             self.control_system.change_traction(route_power)

```

airplane.py

```

1 from pilot import Pilot
2
3
4 class ControlSystem:
5     status = "OK"

```

```

6
7     def __init__(self, airplane):
8         self.airplane = airplane
9
10    def read_flight_indicators(self):
11        return {
12            "traction": self.airplane.traction,
13            "position": self.airplane.position,
14            "weight": self.airplane.weight,
15            "engine": self.airplane.engine,
16        }
17
18    @staticmethod
19    def display_message(message):
20        print(message)
21
22    def open_flaps(self, direction):
23        self.airplane.traction[0] = direction
24
25    def change_traction(self, number):
26        self.airplane.traction[1] = number
27
28
29 class Airplane:
30     weight = 1000
31     wing_length = 3.2
32     traction = ["Down", 0]
33     engine = "Pratt & Whitney JT5D"
34     position = {'x': 0, 'y': 0}
35
36     def fly(self):
37         if self.traction[0] == "Up":
38             self.position['y'] += self.traction[1] / self.weight
39         if self.traction[0] == "Down":
40             self.position['y'] -= self.traction[1] / self.weight
41         if self.traction[0] == "Left":
42             self.position['x'] -= self.traction[1] / self.weight
43         if self.traction[0] == "Right":
44             self.position['x'] += self.traction[1] / self.weight
45
46
47 class Boeing737(Airplane):
48     manufacturer = "Boeing"

```

```

49     __interior_width = 5.25
50     seats_number = 220
51
52     weight = 5000
53     wing_length = 6.4
54     engine = "Pratt & Whitney JT8D"
55
56     def __init__(self, crew=None):
57         if crew is None:
58             crew = [Pilot("Michi Kovalsky", 10), Pilot("Kolya Stepanov", 7),
59                    "Natalya Kolesnikova"]
60
61         self.crew = crew
62
63     def fly(self):
64         if self.crew[0].work_experience > 5:
65             if self.traction[0] == "Up" and self.traction[1] > 1000:
66                 self.position['y'] += self.traction[1] / self.weight
67             if self.traction[0] == "Down" and self.traction[1] > 1000:
68                 self.position['y'] -= self.traction[1] / self.weight
69             if self.traction[0] == "Left" and self.traction[1] > 1000:
70                 self.position['x'] -= self.traction[1] / self.weight
71             if self.traction[0] == "Right" and self.traction[1] > 1000:
72                 self.position['x'] += self.traction[1] / self.weight
73             return "OK"
74         return "Work experience of the main pilot is lower than 5 years. Change
75         the main pilot."
76
77 class AV8B(Airplane):
78     manufacturer = "McDonnell Douglas"
79     __rockets_and_boms = ["4 × AIM-9L Sidewinder", "6 × AIM-120 Amraam", "4 × AGM-
80     65E Maverick", "10 × Mk.77"]
81     _combat_radius = 470
82
83     weight = 3000
84     wing_length = 3.6
85     engine = "Rolls-Royce Pegasus"
86
87     def __init__(self, pilot=None):
88         if pilot is None:
89             pilot = Pilot("Michi Kovalsky", 15)
90         self.pilot = pilot
91         self.crew = [pilot]

```



```

89
90     def fly(self):
91         if self.crew[0].work_experience > 10:
92             if self.traction[0] == "Up" and self.traction[1] > 500:
93                 self.position['y'] += self.traction[1] / self.weight
94             if self.traction[0] == "Down" and self.traction[1] > 500:
95                 self.position['y'] -= self.traction[1] / self.weight
96             if self.traction[0] == "Left" and self.traction[1] > 500:
97                 self.position['x'] -= self.traction[1] / self.weight
98             if self.traction[0] == "Right" and self.traction[1] > 500:
99                 self.position['x'] += self.traction[1] / self.weight
100             return "OK"
101         return "Work experience of the pilot is lower than 10 years. Change the
pilot."

```

main.py

```

1 from time import sleep
2
3 from pilot import Pilot
4 from airplane import ControlSystem, Boeing737, AV8B
5
6
7 def iteration(airplanes):
8     for airplane in airplanes:
9         airplane.fly()
10
11
12 def show_pilot_info(pilot):
13     print("Pilot name:", pilot.name)
14     print("Work experience:", pilot.work_experience)
15     print("Focus level:", pilot.focus_level)
16
17
18 def show_airplane_info(airplane, show_main=False):
19     print("Manufacturer:", airplane.manufacturer)
20     if not show_main:
21         print("Wing length:", airplane.wing_length)
22         print("Engine:", airplane.engine)
23         print("Weight:", airplane.weight)
24         print("Crew:")
25         for p in airplane.crew:
26             print(" - ", end='')
27             print(p)

```

```
28     print("Traction:", airplane.traction)
29     print("Position:", airplane.position)
30
31
32 def main():
33     military_airplane = AV8B()
34     control_system = ControlSystem(military_airplane)
35
36     crew = [Pilot("Kolya Stepanov", 10), Pilot("Taras Lopasov", 7), "Natalya
Kolesnikova"]
37     boeing737_airplane = Boeing737(crew)
38
39     show_pilot_info(military_airplane.pilot)
40
41     print("\nMilitary airplane pilot focusing on fly...\n")
42     military_airplane.pilot.focus_on_flying()
43     show_pilot_info(military_airplane.pilot)
44
45     print("\nAirplane status:")
46     show_airplane_info(military_airplane)
47     print("\nAirplane status:")
48     show_airplane_info(boeing737_airplane)
49
50     print("\nMilitary airplane pilot moving helm 'Up' and pull lever to '6000'...")
51     military_airplane.pilot.move_helm("Up", control_system)
52     military_airplane.pilot.pull_lever(6000, control_system)
53
54     print("\nAirplane start flying...")
55     for _ in range(10):
56         print("\nAirplane status (main):")
57         show_airplane_info(military_airplane, show_main=True)
58
59         iteration([military_airplane, boeing737_airplane])
60         sleep(1)
61
62     print("\nMilitary airplane pilot enabling autopilot...")
63     autopilot_status = military_airplane.pilot.enable_autopilot(control_system, 10,
debug=True)
64     print("\nAutopilot finished work with status", autopilot_status)
65
66
67 if __name__ == '__main__':
68     main()
```

Output:

Pilot name: Michi Kovalsky
Work experience: 15
Focus level: 0

Military airplane pilot focusing on fly...

Pilot name: Michi Kovalsky
Work experience: 15
Focus level: 10

Airplane status:
Manufacturer: McDonnell Douglas
Wing length: 3.6
Engine: Rolls-Royce Pegasus
Weight: 3000
Crew:
- Michi Kovalsky
Traction: ['Down', 0]
Position: {'x': 0, 'y': 0}

Airplane status:
Manufacturer: Boeing
Wing length: 6.4
Engine: Pratt & Whitney JT8D
Weight: 5000
Crew:
- Kolya Stepanov
- Taras Lopasov
- Natalya Kolesnikova
Traction: ['Down', 0]
Position: {'x': 0, 'y': 0}

Military airplane pilot moving helm 'Up' and pull lever to '6000'...

Airplane start flying...

Airplane status (main):
Manufacturer: McDonnell Douglas
Traction: ['Up', 6000]
Position: {'x': 0, 'y': 0}

Airplane status (main):
Manufacturer: McDonnell Douglas
Traction: ['Up', 6000]
Position: {'x': 0, 'y': 3.2}

Airplane status (main):
Manufacturer: McDonnell Douglas
Traction: ['Up', 6000]
Position: {'x': 0, 'y': 6.4}

...

Military airplane pilot enabling autopilot...

Autopilot ask for the route from dispatcher...

Route from dispatcher: ['Right', 2997]

Airplane traction: ['Up', 6000]

Autopilot corrected airplane's traction

Autopilot ask for the route from dispatcher...

Route from dispatcher: ['Left', 1095]

Airplane traction: ['Right', 2997]

Autopilot corrected airplane's traction

Autopilot ask for the route from dispatcher...

Route from dispatcher: ['Left', 3008]

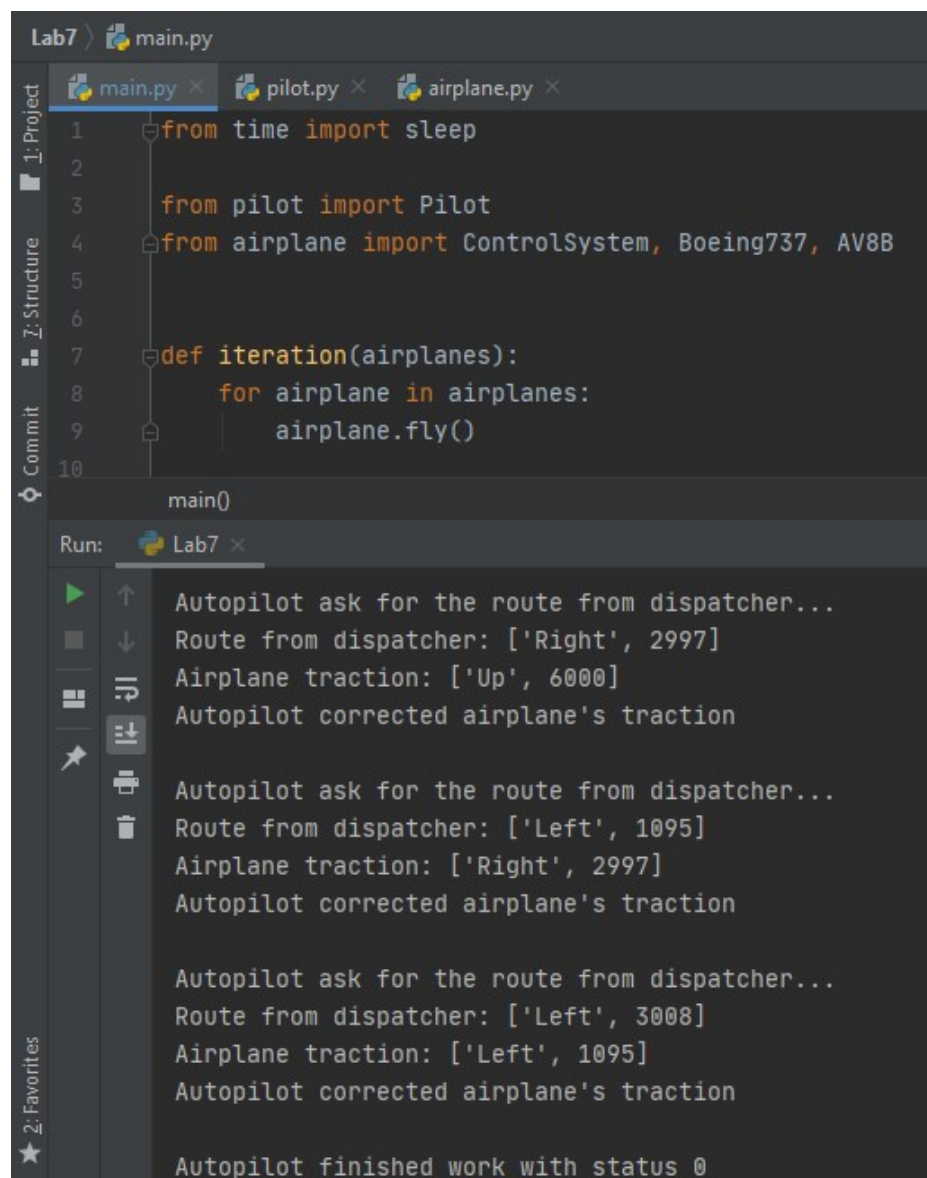
Airplane traction: ['Left', 1095]

Autopilot corrected airplane's traction

...

Autopilot finished work with status 0

Screenshot



The screenshot shows a Python IDE with three files open: `main.py`, `pilot.py`, and `airplane.py`. The `main.py` file contains the following code:

```
1 from time import sleep
2
3 from pilot import Pilot
4 from airplane import ControlSystem, Boeing737, AV8B
5
6
7 def iteration(airplanes):
8     for airplane in airplanes:
9         airplane.fly()
10
11
12 main()
```

The terminal output shows the execution of the program, displaying the following messages:

```
Autopilot ask for the route from dispatcher...
Route from dispatcher: ['Right', 2997]
Airplane traction: ['Up', 6000]
Autopilot corrected airplane's traction

Autopilot ask for the route from dispatcher...
Route from dispatcher: ['Left', 1095]
Airplane traction: ['Right', 2997]
Autopilot corrected airplane's traction

Autopilot ask for the route from dispatcher...
Route from dispatcher: ['Left', 3008]
Airplane traction: ['Left', 1095]
Autopilot corrected airplane's traction

Autopilot finished work with status 0
```