

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:

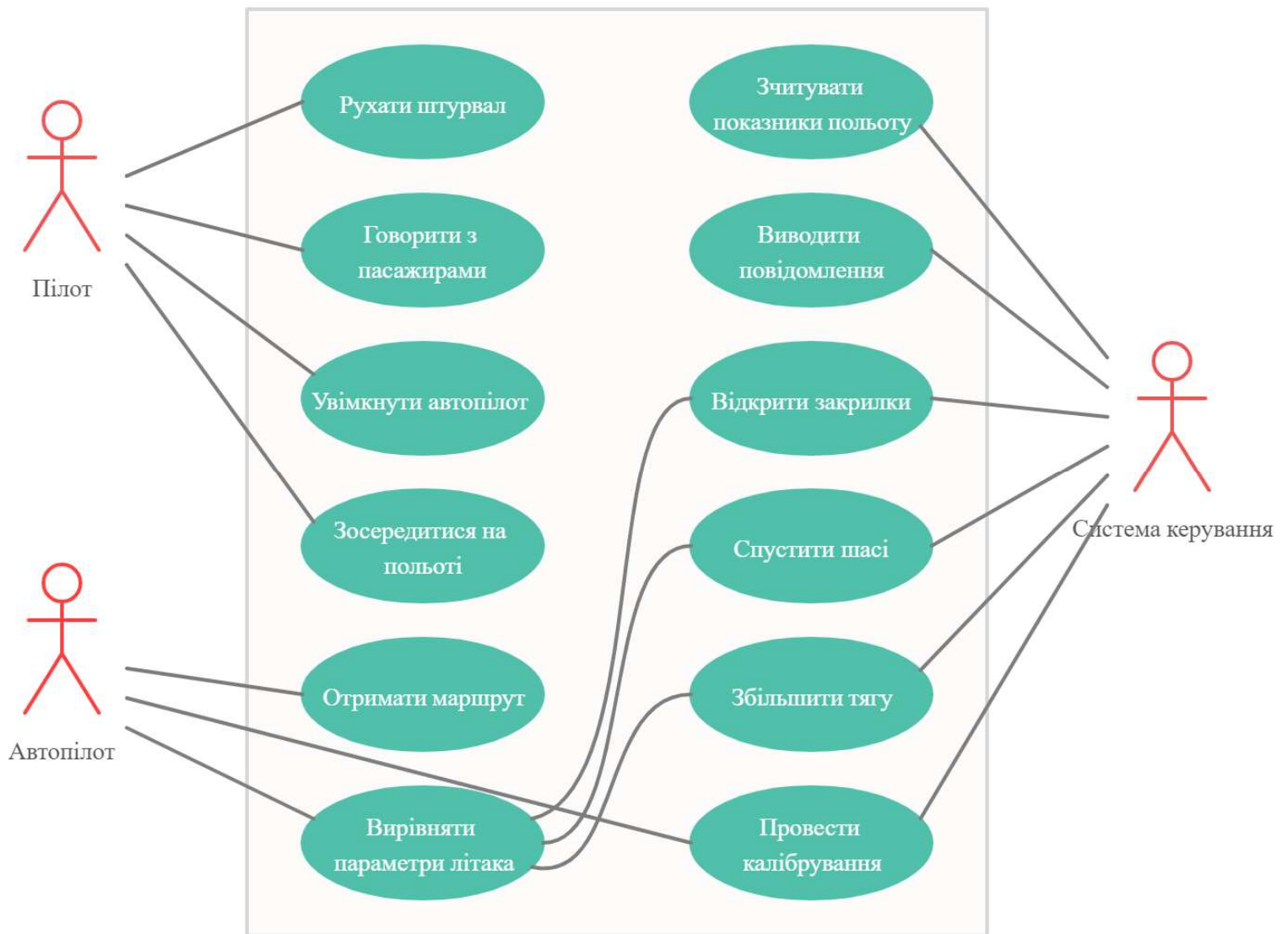
14. Розробити проект автопілота літака Boeing 737-900ER.

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 Boeing 737-900ER	

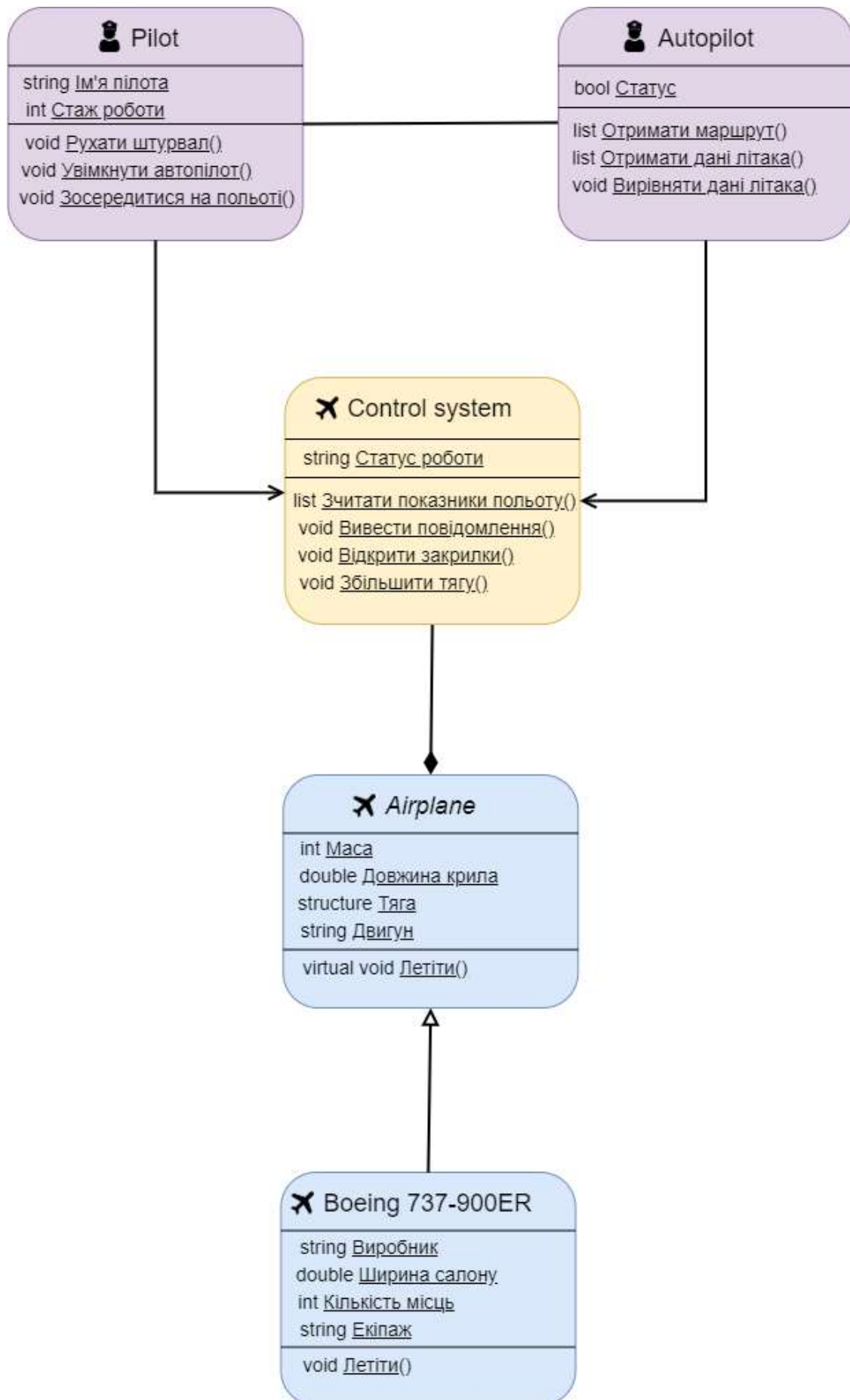
Внутрішні / зовнішні методи

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	✓	

Use case diagram



Class diagram



Відношення між класами

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.

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
70 randomly
71         direction = choice(["Up", "Down", "Left", "Right"])
72         power = randint(1000, 5000)
73         traction = [direction, power]
74         return traction
75
76     def get_airplane_data(self):
77         return self.control_system.read_flight_indicators()

```

```
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             self.crew = crew
61
62     def fly(self):
63         if self.crew[0].work_experience > 5:
64             if self.traction[0] == "Up" and self.traction[1] > 1000:
65                 self.position['y'] += self.traction[1] / self.weight
66             if self.traction[0] == "Down" and self.traction[1] > 1000:
67                 self.position['y'] -= self.traction[1] / self.weight
68             if self.traction[0] == "Left" and self.traction[1] > 1000:
69                 self.position['x'] -= self.traction[1] / self.weight
70             if self.traction[0] == "Right" and self.traction[1] > 1000:
71                 self.position['x'] += self.traction[1] / self.weight
72             return "OK"
73         return "Work experience of the main pilot is lower than 5 years. Change the
74         main pilot."

```

main.py

```

1 from time import sleep
2
3 from pilot import Pilot
4 from airplane import ControlSystem, Boeing737
5
6
7 def show_pilot_info(pilot):
8     print("Pilot name:", pilot.name)
9     print("Work experience:", pilot.work_experience)
10    print("Focus level:", pilot.focus_level)
11
12

```



```
13 def show_airplane_info(airplane, show_main=False):
14     print("Manufacturer:", airplane.manufacturer)
15     if not show_main:
16         print("Seats number:", airplane.seats_number)
17         print("Wing length:", airplane.wing_length)
18         print("Engine:", airplane.engine)
19         print("Weight:", airplane.weight)
20         print("Crew: {}, {}, {}".format(airplane.crew[0], airplane.crew[1],
airplane.crew[2]))
21     print("Traction:", airplane.traction)
22     print("Position:", airplane.position)
23
24
25 def main():
26     crew = [Pilot("Michi Kovalsky", 10), Pilot("Kolya Stepanov", 7), "Natalya
Kolesnikova"]
27     boeing737_airplane = Boeing737(crew)
28     control_system = ControlSystem(boeing737_airplane)
29
30     main_pilot = crew[0]
31     show_pilot_info(main_pilot)
32
33     print("\nMain pilot focusing on fly...\n")
34     main_pilot.focus_on_flying()
35     show_pilot_info(main_pilot)
36
37     print("\nAirplane status:")
38     show_airplane_info(boeing737_airplane)
39
40     print("\nMain pilot moving helm 'Up' and pull lever to '6000'...")
41     main_pilot.move_helm("Up", control_system)
42     main_pilot.pull_lever(6000, control_system)
43
44     print("\nAirplane start flying...")
45     for _ in range(10):
46         print("\nAirplane status (main):")
47         show_airplane_info(boeing737_airplane, show_main=True)
48
49         boeing737_airplane.fly()
50         sleep(1)
51
52     print("\nMain pilot enabling autopilot...")
53     autopilot_status = main_pilot.enable_autopilot(control_system, 10, debug=True)
54     print("\nAutopilot finished work with status", autopilot_status)
55
56
57 if __name__ == '__main__':
58     main()
```

Output:

Pilot name: Michi Kovalsky
Work experience: 10
Focus level: 0

Main pilot focusing on fly...

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

Airplane status:
Manufacturer: Boeing
Seats number: 220
Wing length: 6.4
Engine: Pratt & Whitney JT8D
Weight: 5000
Crew: Michi Kovalsky, Kolya Stepanov, Natalya Kolesnikova
Traction: ['Down', 0]
Position: {'x': 0, 'y': 0}

Main pilot moving helm 'Up' and pull lever to '6000'...

Airplane start flying...

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

Airplane status (main):
Manufacturer: Boeing
Traction: ['Up', 6000]
Position: {'x': 0, 'y': 1.2}

Airplane status (main):
Manufacturer: Boeing
Traction: ['Up', 6000]
Position: {'x': 0, 'y': 2.4}

...

Main pilot enabling autopilot...

Autopilot ask for the route from dispatcher...
Route from dispatcher: ['Right', 1755]
Airplane traction: ['Up', 6000]
Autopilot corrected airplane's traction

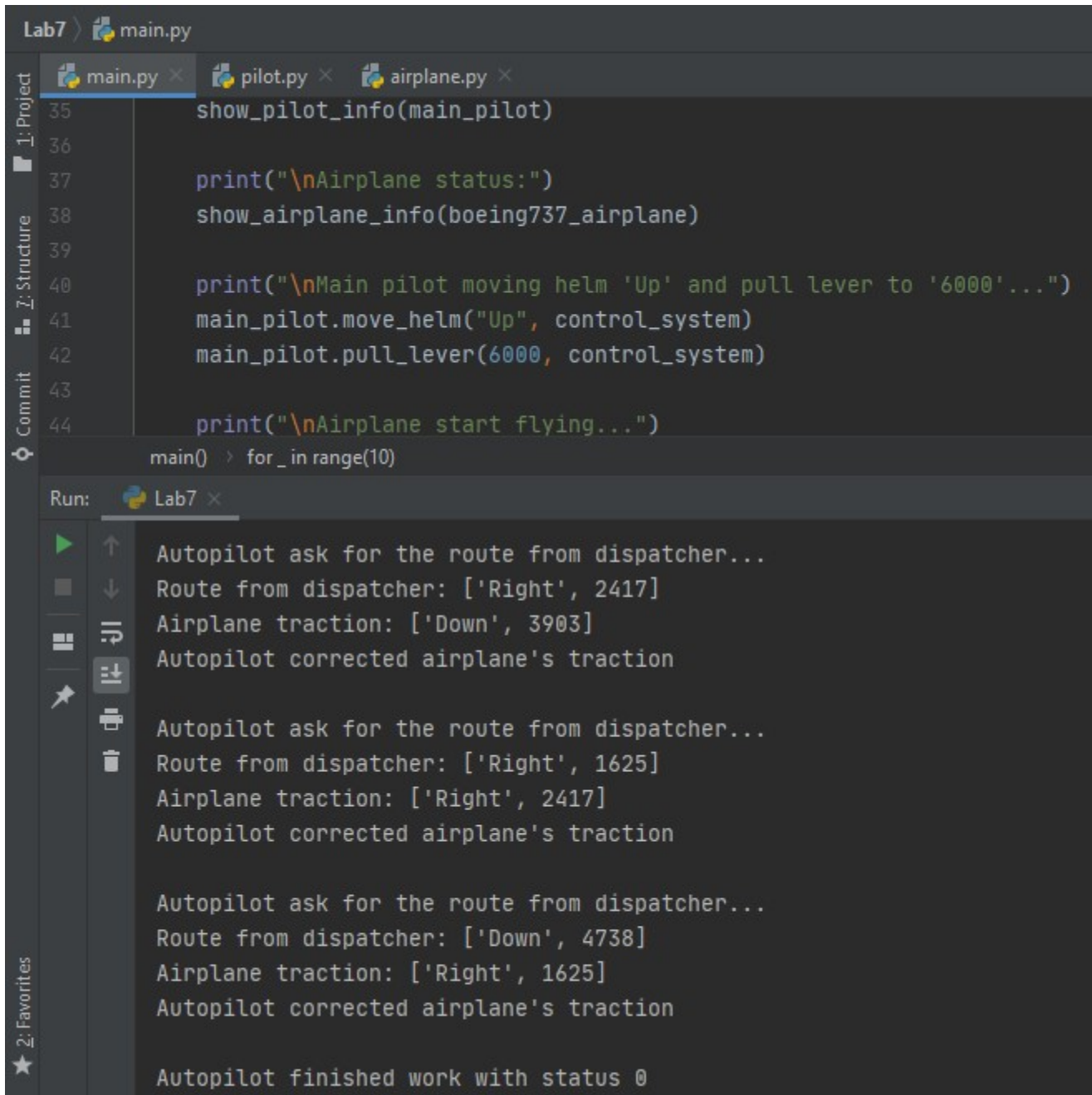
Autopilot ask for the route from dispatcher...
Route from dispatcher: ['Up', 1526]
Airplane traction: ['Right', 1755]
Autopilot corrected airplane's traction

Autopilot ask for the route from dispatcher...
Route from dispatcher: ['Down', 1120]
Airplane traction: ['Up', 1526]
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 is active, showing lines 35 to 44. The code in `main.py` includes calls to `show_pilot_info`, `print` statements for airplane status and main pilot actions, `show_airplane_info`, and `main_pilot.move_helm` and `main_pilot.pull_lever` calls. The output window shows the execution of the `main()` function, which runs a loop for 10 iterations. The output displays the autopilot asking for a route, the route being 'Down' with a value of 1120, the airplane's traction being 'Up' with a value of 1526, and the autopilot correcting the traction. The output also shows the autopilot asking for a route, the route being 'Down' with a value of 4738, the airplane's traction being 'Right' with a value of 1625, and the autopilot correcting the traction. Finally, the output shows the autopilot finished work with status 0.

```
Lab7 > main.py
main.py x pilot.py x airplane.py x
35 show_pilot_info(main_pilot)
36
37 print("\nAirplane status:")
38 show_airplane_info(boeing737_airplane)
39
40 print("\nMain pilot moving helm 'Up' and pull lever to '6000'...")
41 main_pilot.move_helm("Up", control_system)
42 main_pilot.pull_lever(6000, control_system)
43
44 print("\nAirplane start flying...")

main() > for _ in range(10)

Run: Lab7 x
Autopilot ask for the route from dispatcher...
Route from dispatcher: ['Right', 2417]
Airplane traction: ['Down', 3903]
Autopilot corrected airplane's traction

Autopilot ask for the route from dispatcher...
Route from dispatcher: ['Right', 1625]
Airplane traction: ['Right', 2417]
Autopilot corrected airplane's traction

Autopilot ask for the route from dispatcher...
Route from dispatcher: ['Down', 4738]
Airplane traction: ['Right', 1625]
Autopilot corrected airplane's traction

Autopilot finished work with status 0
```