MUHAMMAD AL-XORAZMIY NOMIDAGI TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI

Dasturiy ta’minotni testlash
fanidan
3-amaliy mashg’ulot

Guruh: SWT 001-2
Bajardi: Murodullayev Og’abek
Tekshirdi: Qurbonov Bekzod

TOSHKENT – 2025

Topshiriq kodlari:

```
import unittest

class Car:
    def __init__(self, name, model, year, price, color, current_speed,
max_speed):
        self.name = name
        self.model = model
        self.year = year
        self.price = price
        self.color = color
        self.current_speed = current_speed
        self.max_speed = max_speed

    def get_name(self):
        return self.name

    def get_model(self):
        return self.model

    def get_year(self):
        return self.year

    def get_price(self):
        return self.price

    def get_color(self):
        return self.color

    def get_current_speed(self):
        return self.current_speed

    def get_max_speed(self):
        return self.max_speed

    def break_car(self, speed):
        if self.current_speed - speed >= 0:
            self.current_speed -= speed
            return "Tezlik pasaytirildi"
        return "Tezlik manfiy bo'lishi mumkin emas"

    def acceleration_up(self, speed):
        if self.current_speed + speed > self.max_speed:
            return "Telik max speeddan oshishi mumkin emas!"
        self.current_speed += speed
        return "Joriy tezlik yangilandi"


class TestCar(unittest.TestCase):
    def setUp(self):
        self.car = Car("BMW X6m", "BMW", 2020, 50000, "white", 80, 240)

    def test_name(self):
        self.assertEqual(self.car.name, "BMW X6m")

    def test_getters(self):
        self.assertEqual(self.car.get_name(), "BMW X6m")
        self.assertEqual(self.car.get_model(), "BMW")
```

```
        self.assertEqual(self.car.get_year(), 2020)
        self.assertEqual(self.car.get_price(), 50000)
        self.assertEqual(self.car.get_color(), "white")
        self.assertEqual(self.car.get_current_speed(), 80)
        self.assertEqual(self.car.get_max_speed(), 240)

    def test_break_car_normal(self):
        result = self.car.break_car(30)
        self.assertEqual(result, "Tezlik pasaytirildi")
        self.assertEqual(self.car.current_speed, 50)

    def test_break_car_negative(self):
        result = self.car.break_car(200)
        self.assertEqual(result, "Tezlik manfiy bo'lishi mumkin emas")

    def test_acceleration_up_normal(self):
        result = self.car.acceleration_up(100)
        self.assertEqual(result, "Joriy tezlik yangilandi")
        self.assertEqual(self.car.current_speed, 180)

    def test_acceleration_up_exceed(self):
        result = self.car.acceleration_up(300)
        self.assertEqual(result, "Telik max speeddan oshishi mumkin emas!")


if __name__ == '__main__':
    unittest.main()


class Transport:
    def __init__(self, model, manufacturer, year, mileage, fuel_type):
        self.model = model
        self.manufacturer = manufacturer
        self.year = year
        self.mileage = mileage
        self.fuel_type = fuel_type

    def get_vehicle_info(self):
        return (
            f"Model: {self.model}\n"
            f"Ishlab chiqaruvchi: {self.manufacturer}\n"
            f"Ishlab chiqarilgan yil: {self.year}\n"
            f"Yurgan masofa: {self.mileage} km\n"
            f"Yoqilg‘i turi: {self.fuel_type}"
        )

    def add_mileage(self, distance):
        if distance > 0:
            self.mileage += distance
        else:
            raise ValueError("Masofa musbat bo‘lishi kerak!")

    def get_vehicle_age(self, current_year):
        return current_year - self.year

    def __str__(self):
        return f"{self.manufacturer} {self.model} ({self.year}) - {self.mileage}
km, {self.fuel_type}"
```

```python
class TestTransport(unittest.TestCase):

    def setUp(self):
        self.car = Transport("Malibu", "Chevrolet", 2020, 45000, "Benzin")

    def test_str_method(self):
        expected = "Chevrolet Malibu (2020) - 45000 km, Benzin"
        self.assertEqual(str(self.car), expected)

    def test_get_vehicle_info(self):
        info = self.car.get_vehicle_info()
        self.assertIn("Model: Malibu", info)
        self.assertIn("Ishlab chiqaruvchi: Chevrolet", info)
        self.assertIn("Yoqilg‘i turi: Benzin", info)

    def test_add_mileage_positive(self):
        self.car.add_mileage(500)
        self.assertEqual(self.car.mileage, 45500)

    def test_add_mileage_negative(self):
        with self.assertRaises(ValueError):
            self.car.add_mileage(-100)

    def test_get_vehicle_age(self):
        age = self.car.get_vehicle_age(2025)
        self.assertEqual(age, 5)


class Person:
    def __init__(self, name, surname, age, born_year, passport):
        self.name = name
        self.surname = surname
        self.age = age
        self.born_year = born_year
        self.passport = passport

    def get_student_name(self):
        return self.name

    def get_student_surname(self):
        return self.surname

    def get_Passport_info(self):
        return self.passport

    def get_born_year(self):
        return self.born_year

    def get_age(self):
        return self.age


class Manzil:
    def __init__(self, country, city, province, district, village_name, street,
house_number, postcode):
        self.country = country
        self.city = city
        self.province = province
```

```
        self.district = district
        self.village_name = village_name
        self.street = street
        self.house_number = house_number
        self.postcode = postcode

    def get_location(self):
        return (
            f"{self.country}, {self.city}, {self.province}, {self.district}, "
            f"{self.village_name}, {self.street} {self.house_number},
{self.postcode}"
        )


class Student(Person, Manzil):
    def __init__(
        self,
        name, surname, age, born_year, passport,
        country, city, province, district, village_name, street, house_number,
postcode,
        university, faculty, group, gpa, subjects=None
    ):
        Person.__init__(self, name, surname, age, born_year, passport)
        Manzil.__init__(self, country, city, province, district, village_name,
street, house_number, postcode)
        self.university = university
        self.faculty = faculty
        self.group = group
        self.gpa = gpa
        self.subjects = subjects or {}

    def get_university(self):
        return self.university

    def get_faculty(self):
        return self.faculty

    def add_subjects(self, subject_name, grade):
        self.subjects[subject_name] = grade

    def calc_GPA_by_subjects(self):
        if not self.subjects:
            return 0
        total = sum(self.subjects.values())
        return round(total / len(self.subjects), 2)



class TestStudent(unittest.TestCase):
    def setUp(self):
        self.student = Student(
            name="Ogabek", surname="Murodullayev", age=20, born_year=2005,
passport="AB1234567",
            country="Uzbekistan", city="Tashkent", province="Tashkent",
district="Yunusabad",
            village_name="MFY Shodlik", street="Amir Temur", house_number="5A",
postcode="100093",
            university="TATU", faculty="Dasturiy injinering", group="311-23",
gpa=4.5,
```

```
            subjects={"Math": 5, "Physics": 4}
        )

    def test_get_student_name(self):
        self.assertEqual(self.student.get_student_name(), "Ogabek")

    def test_get_student_surname(self):
        self.assertEqual(self.student.get_student_surname(), "Murodullayev")

    def test_get_Passport_info(self):
        self.assertEqual(self.student.get_Passport_info(), "AB1234567")

    def test_get_born_year(self):
        self.assertEqual(self.student.get_born_year(), 2005)

    def test_get_age(self):
        self.assertEqual(self.student.get_age(), 20)

    def test_get_university(self):
        self.assertEqual(self.student.get_university(), "TATU")

    def test_get_faculty(self):
        self.assertEqual(self.student.get_faculty(), "Dasturiy injinering")

    def test_get_location(self):
        loc = self.student.get_location()
        self.assertIn("Tashkent", loc)
        self.assertIn("Amir Temur", loc)

    def test_add_subjects_and_calc_gpa(self):
        self.student.add_subjects("English", 5)
        self.assertIn("English", self.student.subjects)
        gpa = self.student.calc_GPA_by_subjects()
        self.assertAlmostEqual(gpa, 4.67, 2)

    def test_calc_gpa_empty(self):
        s2 = self.student
        s2.subjects = {}
        self.assertEqual(s2.calc_GPA_by_subjects(), 0)


if __name__ == "__main__":
    unittest.main()
```

Dastur natijasi: