Завдання № 2. У кожному класі повинен бути реалізований метод ініціалізації __init__ (крім випадків, коли клас не має атрибутів, а має лише методи). У більшості випадків встановлення атрибутів має відбуватись під час створення екземпляра класу через передачу необхідних аргументів у метод ініціалізації, якщо в завданні не вказаний інший спосіб ініціалізації. Кожен клас повинен містити лише ті атрибути і методи, які вказані у завданні. Створювати додаткові власні атрибути і методи заборонено.

Звертайте особливу увагу на текст завдання: якщо в завданні сказано, що у метод __init__ передаються деякі аргументи для ініціалізації певних атрибутів, то так треба й зробити. Але, якщо вказано, що деякі атрибути мають отримувати лише початкові значення у методі __init__, то це означає, що в метод __init__ для їх ініціалізації не передаються ніякі аргументи.

В основній програмі створіть екземпляри вказаних у завданні батьківських класів та всіх підкласів. Напишіть код таким чином, щоб перевірити працездатність усіх методів як батьківського класу, так і підкласів. Більшість завдань також містять опис додаткових дій, які обов’язково потрібно виконати в основній програмі.

№ вар	Умови завдання
1	Створити батьківський клас Особа (Person). У метод ініціалізації передається тільки аргумент для ініціалізації атрибуту **name,** при цьому у самому методі атрибуту **skills** має бути привласнене значення пустого словника, а атрибуту **person_type** має бути привласнене значення «Без спеціалізації». Атрибути: 1. Тип спеціалізації особи **person_type**. 2. Ім’я особи **name**. 3. Характеристики **skills** – словник, що містить інформацію про характеристики особи у форматі {"Характеристика": значення}. Методи: 1. Метод **describe_person()**, який генерує рядок з переліком спеціалізації особи, її імені, назви та значень унікальних характеристик. 2. Метод зміни значень існуючих характеристик або додавання нових **update_skills()**. На основі батьківського класу Особа (Person) створити три підкласи – Технар (Techie), Гуманітарій (Humanitarian) та Спортсмен (Athlete). У методі ініціалізації кожного підкласу атрибут **person_type** має прийняти

№ вар	Умови завдання
	відповідне значення «Технар», «Гуманітарій» та «Спортсмен». Значення характеристик **skills** повинні генеруватись випадково в процесі створення відповідного екземпляра класу (у діапазоні від 20 до 100). Самі ж характеристики мають бути наступними: – для Технаря: «Логіка», «Категоріальне мислення», «Результативність»; – для Гуманітарія: «Фантазія», «Інтуїція», «Образне мислення»; – для Спортсмена: «Сила», «Наполегливість», «Харизма». В основній програмі створити поліморфічну функцію **greetings()**, яка в якості аргументу приймає екземпляр класу та повертає рядок з привітанням, яке буде різним для кожного класу («Вітаємо {ім'я}» для класу Person, «Вітаємо видатного науковця {ім'я}» для класу Techie тощо). Створити список, який складається з декількох екземплярів різних класів. Далі в циклі обійти всі елементи списку та для кожного із них викликати метод **describe_person()**, а також передати кожен елемент до функції **greetings()**. Окремо у програмі без використання циклу для деяких екземплярів класів викликати метод **update_skills()** та змінити і/або додати деякі характеристики та відповідні їм значення.
2	Створити батьківський клас Службовець (Employe). У метод ініціалізації передаються тільки аргументи для ініціалізації атрибутів **name** та **proven_record**, а атрибуту **employe_salary** має бути привласнене значення 10000. Атрибути: 1. Ім'я службовця **name**. 2. Стаж роботи (років) **proven_record**. 3. Базовий оклад **employe_salary**. Методи: 1. Метод виведення інформації про службовця **info()**, який повертає назву та значення кожного атрибута. 2. Метод розрахунку зарплати **calculate_salary()**, який повертає значення зарплати, отримане за формулою Базовий оклад + Стаж * 0,1. На основі батьківського класу Службовець (Employe) створити три підкласи – Менеджер (Manager), Інженер (Engineer) і Охоронець (SecurityGuard). У кожному з підкласів має бути унікальний атрибут. У Менеджера це кількість проєктів (**number_projects**), у інженера – кількість вирішених завдань (**number_solved_tasks**), у охоронця – досвід (**experience**) у роках. У підкласах перевизначити метод для розрахунку зарплати **calculate_salary()**: – зарплата менеджера розраховується як: (результат роботи методу **calculate_salary()** батьківського класу) + Кількість проектів * 0,15; – зарплата Інженера: (результат роботи методу **calculate_salary()**

№ вар	Умови завдання
	батьківського класу) + Кількість завдань * 0,2; – Зарплата Охоронця: (результат роботи методу **calculate_salary()** батьківського класу) + Досвід * 0,05. В основній програмі додатково створити поліморфічну функцію **position()**, яка буде приймати екземпляр класу в якості аргументу та виводити повідомлення з інформацією про займану ним посаду (наприклад, «Працівник {ім'я} займає посаду {посада}).
3	Створити батьківський клас Транспорт (Transport). У метод ініціалізації передаються тільки аргументи, що відповідають атрибутам **production_year** та **fuel_consumption**, тоді як у тілі методу атрибуту **race** має бути привласнене значення 0. Атрибути: 1. Пробіг **race** (км). 2. Рік випуску **production_year**. 3. Витрата пального **fuel_consumption** (л/км). Методи: 1. Метод виведення інформації про транспортний засіб **about()**, який повертає назву та значення кожного атрибута. 2. Метод розрахунку витрати палива **calculate_fuel()**, який приймає відстань та повертає кількість витраченого палива. 3. Метод **consumption_growth()**, який розраховує та виводить інформацію про те, на скільки може зрости витрата палива у порівнянні із початковим значенням. Метод працює наступним чином: якщо до методу передано два аргументи (деяке значення пробігу та поточний рік), то приріст витрати пального становить 5% за кожні 100 000 км пробігу та на 1% за кожен рік віку авто; якщо до методу передано один аргумент (деяке значення пробігу), то приріст витрати пального становить 10% за кожні 150 000 км пробігу. Тут під терміном «деяке значення пробігу» мається на увазі число, що визначає пробіг, але не є атрибутом **race**. Метод не змінює ні атрибут **race**, ні атрибут **fuel_consumption**, а лише виводить інформацію. На основі батьківського класу Транспорт (Transport) створити підкласи Легковик (Car), Вантажівка (Truck) і Трактор (Tractor). У кожному з підкласів має бути додатковий атрибут **model**, який визначає марку відповідного транспортного засобу. Перевизначити у кожному підкласі метод **about()**, щоб він враховував цей додатковий атрибут. Також у підкласах перевизначити метод для розрахунку витрат палива: – для Легковика метод має повертати: (результат роботи методу **calculate_fuel()** батьківського класу) * 0,8. – для Вантажівки: (результат роботи методу **calculate_fuel()** батьківського класу) * 1,2. – для Трактора: (результат роботи методу **calculate_fuel()** батьківського класу) * 1,4.

№ вар	Умови завдання
	У основній програмі створити список із декількох екземплярів кожного з класів. За допомогою циклу обійти елементи списку та викликати усі методи, щоб продемонструвати їх працездатність.
4	Створити батьківський клас Квітка (Flower). Встановлення усіх атрибутів має відбуватись у методі ініціалізації екземпляра класу. Атрибути: 1. Колір **color**. 2. Тип **flower_type** (наприклад, домашня, садова, дика або інша). Методи: 1. Метод виведення інформації про квітку **info()**. 2. Метод поливу квітки **watering()**, який виводить у консоль повідомлення про те, що квітка була полита. На основі батьківського класу Квітка (Flower) створити підкласи Троянда (Rose), Ірис (Iris) і Лілія (Lily). У кожному з підкласів мають бути нові атрибути: назва квітки **name** та вартість **price**. У підкласах перевизначити метод **watering()**, який повинен додатково виводити назву квітки, що була полита. Створити клас Садівник (Gardener), який не має атрибутів, але має два методи: 1. Метод, що дозволяє розпочати полив квітки **start_watering()**, який в якості аргументу приймає екземпляр квітки (троянду/ірис/лілію) та виводить повідомлення «Садівник розпочав полив квітки {назва квітки}» і викликає для отриманої квітки метод **watering()**. 2. Метод збирання букету **gather_bouquet()**, який приймає кількість квіток та повертає список із вказаної кількості квіток, сформованих випадковим чином. Також метод виводить повідомлення із вартістю зібраного букета.
5	Створити батьківський клас Студент (Student). Встановлення усіх атрибутів має відбуватись у методі ініціалізації екземпляра класу. Атрибути: 1. Ім’я **name**. 2. Вік **age**. Методи: 1. Метод виведення інформації про студента **info()**. 2. Метод **remains()**, який визначає кількість років до випуску в залежності від віку студента (з розрахунку: 17 років – залишилось навчатись 5 років, 22 роки – залишилось навчатись 0 років). На основі батьківського класу Студент (Student) створити підкласи Бакалавр (Bachelor), Магістр (Master) і Аспірант (PhDStudent). У кожному з підкласів мають бути нові атрибути: рік вступу **year** та тема диплому **topic**. Перевизначити метод **info()** для кожного підкласу так, щоб він виводив додаткове повідомлення «Тема дипломного проекту (для підкласу Бакалавр)

№ вар	Умови завдання
	/ магістерської дисертації (для Магістра) / дисертації PhD (для Аспіранта): {тема диплому}». Перевизначити метод **remains()** так, щоб він визначав залишок навчання на основі року вступу і поточного року (всього бакалаври навчаються 4 роки, магістри 2 роки, аспіранти – 4 роки). В основній програмі додатково створити поліморфічну функцію **give_diploma()**, за допомогою якої можна видати різний тип диплому (в залежності від переданого їй екземпляра класу). Функція має виводити повідомлення «{Ім'я} отримує диплом {бакалавра/магістра/доктора філософії}».
6	Створити батьківський клас Проїзний (Ticket). У метод ініціалізації не передаються ніякі аргументи, проте у самому методі атрибуту **ticket_type** має бути привласнене значення «Базовий», атрибуту **replenishment_cost** – 100, атрибуту **period** – 0, атрибуту **balance** – 0. Атрибути: 1. Термін дії **period** в днях. 2. Вартість одного поповнення **replenishment_cost** на 30 днів в грн. 3. Тип проїзного **ticket_type**. 4. Гроші на рахунку **balance** в грн. Методи: 1. Метод виведення загальної інформації про проїзний **info()**. 2. Метод **day_passed()**, який не приймає аргументів, проте його виклик зменшує атрибут **period** на 1. У методі повинна перевірятись умова досягнення атрибутом **period** значення 0. Коли це відбувається, то атрибут **period** уже не змінюється, а тільки виводиться повідомлення, що для того, щоб розпочати наступний період поїздок, необхідно поповнити рахунок. 3. Метод поповнення рахунку **replenish_account()**, який приймає один аргумент – суму поповнення і змінює відповідно атрибут **balance**. 4. Метод подовження дії проїзного **extend()**, який приймає один аргумент – кількість днів подовження days. У методі має змінитись атрибут **period** на задану кількість днів. Також має зменшитись кількість грошей на рахунку за формулою (**balance** - **replenishment_cost** * days / 30). Якщо на рахунку недостатньо коштів, то має виводитись повідомлення про необхідність поповнити рахунок. На основі батьківського класу Проїзний (Ticket) створити підкласи Студентський (Stud) та Учнівський (Pupil). У кожному із підкласів як аргумент у метод ініціалізації додатково має передаватись атрибут **education_institution**, де зберігається рядок з назвою навчального закладу. Атрибут **ticket_type** у методах ініціалізації підкласів має приймати значення, що відповідають підкласу: для Stud – «Студентський», Pupil – «Учнівський».

№ вар	Умови завдання
	У обох підкласах доповнити метод **info()** так, щоб він додатково виводив інформацію про навчальний заклад. Також у підкласах перевизначити метод **extend()** наступним чином: – для Студентського: у формулі розрахунку залишку на рахунку врахувати, що вартість одного поповнення складає 50% від базової вартості; – для Учнівського: подовження безкоштовне, тому вартість поповнення має = 0 (тобто, змінюється тільки атрибут **period**, а формула для розрахунку залишку на рахунку не використовується). Тільки у підкласі Pupil перевизначити метод **replenish_account()** таким чином, щоб він не змінював атрибут **balance**, а тільки виводив інформацію, що учнівський проїзний не потрібно поповнювати. В основній програмі, окрім перевірки роботи усіх методів, додатково створити функцію **get_ticket_type()**, яка визначає, до якого типу належить проїзний в залежності від переданого їй екземпляра класу та виводить відповідне повідомлення. Також під час перевірки роботи метода **day_passed()** використайте цикл, в якому деяку кількість разів викликався б цей метод на заданому екземплярі класу.
7	Створити батьківський клас Комп'ютер (Computer). Встановлення усіх атрибутів має відбуватись у методі ініціалізації екземпляра класу. Атрибути: 1. Процесор **processor** – рядкова змінна, яка визначає тип процесора («Intel» / «AMD» / «Other»). За замовчуванням – «Other». 2. Тактова частота **frequency** (ГГц). 3. Обсяг оперативної пам'яті **ram** (ГБ). Методи: 1. Метод виведення загальної інформації про комп'ютер **params()**. 2. Метод **cost()**, який повертає вартість комп'ютера, розраховану як cost = 10000 * (**ram** / 16 + **frequency** / 2), якщо процесор «Intel» або «AMD»; та cost = 8000 * (**ram** / 16 + **frequency** / 2.5), якщо процесор інший. На основі батьківського класу Комп'ютер (Computer) створити підкласи Настільний ПК (PC), Ноутбук (Laptop) і Планшет (Tab). Кожен підклас має додатковий атрибут **ssd**, який визначає обсяг SSD накопичувача в гігабайтах. У підкласах перевизначити метод **params()** із урахуванням додаткової інформації про новий атрибут. У підкласах перевизначити метод **cost()** наступним чином: – для настільного ПК вартість = (результат роботи методу **cost()** батьківського класу) * 1,2 + **ssd** * 10; – для Ноутбуку вартість = (результат роботи методу **cost()** батьківського класу) + **ssd** * 15; – для Планшету вартість = (результат роботи методу **cost()**

№ вар	Умови завдання
	батьківського класу) * 0,8 + **ssd** * 12. У основній програмі створити список із декількох екземплярів кожного з класів та написати код для визначення найдешевшого пристрою.
8	Створити батьківський клас Автомобіль (Car). Встановлення усіх атрибутів має відбуватись у методі ініціалізації екземпляра класу. Атрибути: 1. Марка **model**. 2. Максимальна швидкість **max_speed** (км/год). Методи: 1. Розрахунок вартості автомобіля **cost()**. Вартість має бути розрахована за формулою **max_speed** * 100. 2. Метод оновлення моделі **upgrade()**, який додає +10 км/год до максимальної швидкості. 3. Метод **info()** для виведення марки, макс. швидкості і вартості авто. На основі батьківського класу Автомобіль (Car) створити підкласи Елітний автомобіль (EliteCar) та Спортивний автомобіль (SportCar). У підкласах створити додатковий атрибут **rank** – ранг автомобілю (може бути 1, 2 або 3). Перевизначити у підкласах метод **info()**, щоб додатково виводити інформацію про ранг авто. У підкласі Елітний автомобіль (EliteCar) перевизначити метод **cost()**, який дозволяє розрахувати вартість за зміненою формулою **max_speed** * 150 * **rank**. У підкласі Спортивний автомобіль (SportCar) перевизначити метод **upgrade()**, який додає до максимальної швидкості величину (10 * **rank)** км/год. Створити клас Продавець авто (CarDealer), який не має атрибутів, але має два методи: 1. Метод для визначення ціни продажу авто **calculate_price()**, який приймає екземпляр автомобілю та повертає його ціну. 2. Метод продажу авто **sell_car()**, який приймає екземпляр автомобілю та виводить повідомлення «Автомобіль {марка} було продано за {ціна продажу} у.о.». В основній програмі передати Продавцю колекцію автомобілів для продажу.
9	Створити батьківський клас Камера (Camera). Встановлення усіх атрибутів має відбуватись у методі ініціалізації екземпляра класу. Атрибути: 1. Модель **model**. 2. Оптичне збільшення **zoom** (випадкове число від 1 до 20). 3. Матеріал корпусу **material** (метал або пластик).

№ вар	Умови завдання
	Методи: 1. Метод виведення загальної інформації про камеру **info()**. 2. Метод **cost()**, який повертає вартість камери, розраховану як (**zoom** + 2) * 15, якщо корпус пластиковий, та (**zoom** + 2) * 20, якщо корпус металевий. На основі батьківського класу Камера (Camera) створити підкласи Професійна камера (ProCam) та Екшн-камера (ActionCam). Обидва підкласи повинні мати додатковий атрибут **megapixels** – кількість мегапікселів. У підкласах перевизначити метод **info()**, щоб він виводив інформацію у тому числі і про кількість мегапікселів камери. Також у підкласах перевизначити метод **cost()**, який визначає вартість як (результат роботи методу **cost()** батьківського класу) * **megapixels**. В основній програмі додатково створити поліморфічну функцію **recommendation()**, яка в залежності від типу камери (звичайна, професійна або екшн) буде виводити повідомлення з назвою камери та рекомендаціями, кому ця камера найкраще підходить. Наприклад, повідомлення про екшн-камеру може бути таким: «Камера {модель} стане найкращим вибором для активних людей, які захоплюються екстремальними видами спорту та бажають зафіксувати найбільш вражаючі моменти своїх пригод». Інші рекомендації придумайте самостійно.
10	Створити батьківський клас Фільм (Film). Встановлення усіх атрибутів має відбуватись у методі ініціалізації екземпляра класу. Атрибути: 1. Назва **title**. 2. Тривалість **duration** у хвилинах. 3. Режисер **director**. 4. Прем’єра **premiere**, який приймає значення True або False (за замовчуванням має бути False). Методи: 1. Метод виведення загальної інформації про фільм **about()**. 2. Метод **calculate_price()**, який повертає вартість виробництва фільму, що визначається як тривалість * 20. Водночас, якщо режисером є К. Нолан або Дж. Кемерон, вартість фільму подвоюється. На основі батьківського класу Фільм (Film) створити підкласи Художній фільм (FeatureFilm), Мультфільм (Cartoon) та Документальний фільм (Documentary). Підкласи повинні мати додатковий атрибут **age_category** – вікова група, для якої рекомендовано стрічку (3+, 10+, 16+ тощо) та атрибут **film_type**, що визначає тип фільму і приймає відповідне значення у кожному із підкласів («Художній фільм», «Мультфільм» та «Документальний фільм»). Перевизначити у підкласах метод **about()**, доповнивши його виведенням інформації про вікову категорію.

№ вар	Умови завдання
	Створити клас Кінотеатр (Cinema), у якому в методі ініціалізації задається атрибут **film**. Цей атрибут отримуватиме своє значення через аргумент, що буде передаватись під час створення екземпляра класу Кінотеатр. Причому аргументом буде один із екземплярів фільмів (тобто, екземпляр класу Художній фільм, Мультфільм або Документальний фільм). Клас Кінотеатр (Cinema) також має два методи: 1. Метод **ticket_price()**, який повертає вартість квитка, що розраховується як (вартість виробництва фільму) / N (де N = 100 для художнього фільму, N = 120 – для мультфільму та N = 500 для документального). Якщо фільм – прем'єра, то отримана вартість квитка має бути помножена на 1,5. 2. Метод показу фільму **show_film()**, який виводить повідомлення «В кінотеатрі розпочато показ фільму {назва}. Фільм рекомендовано для вікової групи {вік}. Вартість квитка: {кількість} грн.» В основній програмі розпочати показ трьох довільних фільмів в кінотеатрі (назви, типи фільмів та приналежність до прем'єри того чи іншого фільму обираєте на власний розсуд).
11	Створити батьківський клас Літак (Plane). Встановлення усіх атрибутів має відбуватись у методі ініціалізації екземпляра класу. Атрибути: 1. Модель **model**. 2. Максимальна висота польоту **max_altitude** (м). 3. Максимальна швидкість **max_speed** (км/год). Методи: 1. Метод виведення загальної інформації про літак **info()**. 2. Метод **get_rank()**, який виводить інформацію про ранг літака. Ранг має визначатись як (**max_altitude** / 1000 + 2 * **max_speed** / 50). На основі батьківського класу Літак (Plane) створити підкласи Транспортний літак (TransportPlane), Пасажирський літак (PassengerPlane) та Безпілотник (Drone). Усі підкласи повинні мати додаткові атрибути: витрата палива **fuel_loss** (л/км) та максимальна дальність польоту **max_dist** (км). Перевизначити у підкласах метод **info()**, який додатково має виводити інформацію про нові атрибути. Також перевизначити метод **get_rank()**: – для транспортного літака метод має повертати ранг звичайного літака плюс 5 за кожні 2000 км максимальної дальності польоту; – для пасажирського літака – ранг звичайного літака плюс 10 за кожні 500 км максимальної дальності польоту; – для безпілотника – ранг звичайного літака плюс 15 за кожні 10 км максимальної дальності польоту. В основній програмі створити поліморфічну функцію, яка приймає

№ вар	Умови завдання
	екземпляр літака та дальність маршруту (в км) і повертає кількість літрів пального, що необхідна для подолання літаком цієї дальності. У випадку, якщо задана дальність перевищує максимальну дальність польоту літака, вивести повідомлення «Літак не зможе подолати настільки довгий маршрут». Якщо ж до функції окрім дальності маршруту передати ще один аргумент – середню швидкість польоту, функція має повернути два значення: затрати пального на виконання перельоту та час, який займе переліт. Якщо до функції передається екземпляр звичайного літака (класу Plane), то вивести повідомлення «Неможливо провести розрахунки».
12	Створити батьківський клас Студент (Student). У метод ініціалізації передаються тільки аргументи для ініціалізації атрибутів **name**, **surname** та **course**, а атрибуту **marks** має бути привласнене значення у вигляді списку з трьох цілих чисел в діапазоні від 25 до 100 (балів). Атрибути: 1. Ім'я **name** та прізвище **surname**. 2. Курс **course**. 3. Оцінки за екзамени на останній сесії **marks**. Методи: 1. Метод виведення загальної інформації про студента **about()**. 2. Метод переведення на наступний курс **next_year()**, який збільшує значення атрибуту **course** на 1, якщо усі оцінки за екзамени більше 60. Якщо ж хоча б одна із оцінок є нижчою за 60 балів, то має бути виведене повідомлення, що студент відрахований і може поновитись тільки на контракт. На основі батьківського класу Студент (Student) створити підкласи Студент-бюджетник (BudgetStud) та Студент-контрактник (ContractorStud). У підкласі Студент-бюджетник має бути додатковий атрибут булевого типу **stipend**, який приймає значення True, якщо студент отримує стипендію, інакше – False (встановлюється за замовчуванням). У підкласі Студент-контрактник має бути додатковий атрибут **contract_paid**, який приймає значення True, якщо студент заплатив за контракт, інакше – False (встановлюється за замовчуванням). Перевизначити у підкласах метод **about()**, щоб доповнити інформацію даними про додані атрибути. Також у підкласах перевизначити і доповнити метод **next_year()** так, щоб: – у підкласі Студент-бюджетник метод додатково розраховував і повертав розмір стипендії (1000 грн якщо середня оцінка за екзамени складає 70 балів, і далі плюс 100 грн за кожні 10 балів), при цьому встановлюючи атрибут **stipend** як True; якщо середня оцінка менше 70 балів, метод встановлює значення атрибуту **stipend** як False і повертає розмір стипендії 0 грн;

№ вар	Умови завдання
	– у підкласі Студент-контрактник метод додатково перевіряє, чи оплатив студент контракт (якщо не оплатив, то переведення не відбувається, навіть якщо усі оцінки за екзамени більші 60-ти балів). В основній програмі створіть поліморфічну функцію, яка приймає екземпляр класу та виводить інформацію про вартість контракту на рік, якщо студент – контрактник, або про розмір стипендії, якщо студент – бюджетник.
13	Створити батьківський клас Потяг (Train). Встановлення усіх атрибутів має відбуватись у методі ініціалізації екземпляра класу. Атрибути: 1. Кількість вагонів **carriages**. 2. Максимальна швидкість **max_speed**. Методи: 1. Метод виведення інформації про кількість вагонів та максимальну швидкість потяга **info()**. 2. Метод **characteristics()**, який розраховує довжину потяга в метрах, визначену як (кількість вагонів) * 25, та виводить повідомлення «Довжина потяга складає {число} метрів». На основі батьківського класу Потяг (Train) створити підкласи Пасажирський потяг (PassTrain) та Товарний потяг (FreightTrain). Для Пасажирського потягу створити додатковий атрибут **passengers** – кількість пасажирів в одному вагоні; для Вантажного потяга додатковий атрибут **max_weight** – максимальна вантажопідйомність одного вагона (у тонах). У підкласах перевизначити метод **info()**, щоб додатково до інформації цього методу із батьківського класу виводилось ще одне окреме повідомлення про значення свого унікального атрибута. У підкласах перевизначити метод **characteristics()** так, щоб для Пасажирського потяга він повертав кількість пасажирів, яку перевозить цей потяг, а для Товарного потяга – максимальну кількість товару (в тонах), яку можна перевезти цим потягом. Також у підкласах створити метод **calculate_path()**, який: – якщо до методу передано один аргумент (відстань до станції призначення), то розраховується час, за який потяг досягне кінцевої станції за умови руху зі своєю максимальною швидкістю; – якщо до методу додатково до відстані передано другий аргумент – розхід дизельного пального (кількість літрів на 100 км), то метод повертає кортеж, який містить час дороги до кінцевої станції та кількість пального, яку необхідно витратити на цей маршрут (для товарного потягу кількість пального подвоюється). В основній програмі створити список із декількох Пасажирських та Товарних потягів з різними значеннями атрибутів. В циклі для кожного із створених потягів по черзі викликати всі їх методи.

№ вар	Умови завдання
14	Створити батьківський клас Корабель (Boat). Встановлення усіх атрибутів має відбуватись у методі ініціалізації екземпляра класу. Атрибути: 1. Ім’я (назва) **name**. 2. Водотоннажність **displacement** (у тонах). Методи: 1. Виведення інформації про назву та водотоннажність судна **about()**. 2. Метод **calculate_carrying_capacity()** для розрахунку вантажопідйомності судна як 8/27 * **displacement**. 3. Визначення рангу судна **get_rank()**. Ранг збільшується на 1 за кожні 100 тон вантажопідйомності. На основі батьківського класу Корабель (Boat) створити підкласи Круїзний лайнер (CruiseShip) та Вантажне судно (FreightShip). У підкласі Круїзний лайнер має бути власний атрибут кількість палуб **decks**, у Вантажного судна – повна місткість судна **brutto** (в тонах). У підкласах перевизначити метод **about()** із урахуванням інформації про нові атрибути. У підкласах перевизначити метод **get_rank()**, який має повертати наступні значення: – для Круїзного лайнера: ранг звичайного судна * кількість палуб; – для Вантажного судна: ранг звичайного судна + (5 за кожні 100 тон повної місткості). В основній програмі створити поліморфічну функцію **ship_ready()**, яка приймає екземпляр класу корабля та виводить повідомлення «Лайнер {ім’я}, що має ранг {ранг}, готовий до подорожі», якщо це Круїзний лайнер, та «Судно {ім’я}, що має ранг {ранг}, готове до перевезення ваших товарів», якщо це Вантажне судно. Якщо ж переданий до функції об’єкт належить базовому класу Корабель, то має виводитись повідомлення «Корабель {ім’я}, що має ранг {ранг}, готовий до відправлення».
15	Створити батьківський клас Користувач (User). У метод ініціалізації передається тільки аргумент для ініціалізації атрибуту **login**, тоді як у самому методі атрибуту **group** має бути привласнене значення 'user', а атрибуту **rights** – словник із значеннями True для всіх прав. Атрибути: 1. Логін **login**. 2. Група **group**. 3. Права **rights** – словник, який містить перелік прав користувача на сайті у форматі {'Назва характеристики': значення (може бути True або False)}. Перелік прав: 'Читати матеріали', 'Публікувати матеріали', 'Писати

№ вар	Умови завдання
	коментарі'. Методи: 1. Виведення інформації про логін та групу користувача **info()**. 2. Метод **post()** для публікації матеріалу, якщо користувач має на це право. Цей метод має приймати рядок з текстом матеріалу і виводити цей текст та друкувати додаткове повідомлення «Запис успішно опубліковано!». Якщо користувач не має права на публікацію, то ніякий текст не виводиться, а тільки повідомлення «Ви не маєте права публікувати матеріали». На основі батьківського класу Користувач (User) створити підкласи Модератор (Moder) та Адміністратор (Admin). Підкласи повинні мати додатковий атрибут досвід **experience** (встановлюється автоматично під час ініціалізації, приймає значення випадкового числа від 1 до 100). У методі ініціалізації підкласів атрибуту **group** має бути автоматично привласнене значення 'moderator' або 'administrator' (в залежності від підкласу), а ініціалізація атрибуту **rights** не відрізняється від батьківського класу. У підкласах перевизначити метод **info()**, який тепер повинен виводити інформацію і про досвід користувача. Також у підкласах створити унікальний метод **change_rights()**, який приймає екземпляр класу (або Користувача, або Модератора, або Адміністратора) і назву права та змінює значення цього права на протилежне. При цьому Модератор може змінити право тільки Користувача і не може змінити права іншого Модератора та Адміністратора, тоді як Адміністратор може змінювати права будь-яких Користувачів і Модераторів. І Модератор, і Адміністратор можуть змінювати значення лише тих прав, які є в словнику **rights** (не можна додавати нові права).
16	Створити батьківський клас Водойма (Water). Встановлення усіх атрибутів має відбуватись у методі ініціалізації екземпляра класу. Атрибути: 1. Назва **name**. 2. Довжина **length**. 3. Ширина **width**. Методи: 1. Виведення інформації про водойму **about()**. 2. Метод **calculate_params()**, який дозволяє визначити площу поверхні водойми: обчислюється як **length** * **width**. На основі батьківського класу Водойма (Water) створити підкласи Озеро (Lake) та Море (Sea). Обидва підкласи мають додатковий атрибут **depth** – глибина водойми. Підклас Море має ще один власний атрибут **salinity** – солоність води у проміле (‰). У підкласах перевизначити метод **about()**, щоб він виводив інформацію з урахуванням нових атрибутів. У підкласах перевизначити метод **calculate_params()**, який замість

№ вар	Умови завдання
	площі поверхні має повертати водний об’єм, розрахований як (результат роботи методу батьківського класу) * **depth**. Створити клас Акваторія (Aquatorium), який містить атрибут **water_objects**, який створюється у методі ініціалізації та є списком із декількох озер та морів, які є екземплярами відповідних класів. У класі створити метод **fresh_salt()**, який підраховує та виводить кількість прісних і солоних водойм в акваторії. Також створити метод **water_resources()**, який підраховує загальний об’єм всіх водних ресурсів акваторії.
17	Створити батьківський клас Ракета (Rocket). Встановлення усіх атрибутів має відбуватись у методі ініціалізації екземпляра класу. Атрибути: 1. Модель **model**. 2. Об’єм паливного баку **fuel_volume** (у тонах). 3. Витрата палива **fuel_loss** (у тонах на кілометр). Методи: 1. Виведення інформації про параметри ракети **info()**. 2. Метод **distance()** – визначення максимальної дальності польоту, що обчислюється як об’єм паливного баку / витрата палива. На основі батьківського класу Ракета (Rocket) створити підкласи Космічна ракета (SpaceRocket) та Геофізична ракета (GeoRocket). Підклас Космічна ракета повинен мати додатковий атрибут **fuel_tanks** – кількість паливних баків, а Геофізична ракета – кількість вимірювальної апаратури на борту **equipment**. Перевизначити у підкласах метод **info()** для виведення додаткової інформації про нові атрибути. Також у підкласах перевизначити метод **distance()**: – для Космічної ракети метод має повертати (результат роботи методу **distance()** батьківського класу) * кількість паливних баків; – для Геофізичної ракети: (результат роботи методу **distance()** батьківського класу) / кількість вимірювальної апаратури. В основній програмі створити поліморфічну функцію **launch_rocket()**. Функція приймає екземпляр ракети та виводить повідомлення «Ракета {модель} розпочинає політ на Марс», якщо це Космічна ракета, або повідомлення «Ракету {модель} запущено у верхні шари атмосфери для геофізичних досліджень», якщо це Геофізична ракета. Запустити по черзі декілька різних ракет.
18	Створити батьківський клас Школа (School). У метод ініціалізації передається тільки аргумент для ініціалізації атрибуту **number**, тоді як у самому методі атрибуту **students** має бути автоматично привласнене випадкове ціле число в діапазоні від 200 до 1000. Атрибути:

№ вар	Умови завдання
	1. Номер школи **number**. 2. Кількість учнів **students**. Методи: 1. Виведення інформації про школу **info()**. 2. Метод визначення престижу школи **get_rank()**. Престиж збільшується на 1 за кожні 100 учнів у школі. На основі батьківського класу Школа (School) створити підкласи Гімназія (Gymnasium) та Ліцей (Lyceum). У підкласах має бути додатковий власний атрибут спеціалізація **specialization**, який передається у метод ініціалізації під час створення екземпляра класу. Цей атрибут має бути рядковою змінною (наприклад, 'математика', або 'іноземні мови', або 'економіка' тощо). Перевизначити у підкласах метод **info()**, щоб додатково до попередньої інформації виводити спеціалізацію навчального закладу. Також у підкласах перевизначити метод **get_rank()**, який повертає значення: – для Гімназії: (результат роботи методу **get_rank()** батьківського класу) + 5; – для Ліцею: (результат роботи методу **get_rank()** батьківського класу) + 10; Створити клас Учень (Student), який має атрибути ім’я **name**, клас **year_of_study** та школа **school**. Значенням атрибуту **school** має бути екземпляр однієї із шкіл/гімназій/ліцеїв. Створити метод **about()**, який виводить повідомлення «Учень {ім’я} навчається в {клас} класі {тип школи} номер {номер школи}, що має спеціалізацію {спеціалізація}». Наприклад, для одного із учнів повідомлення може виглядати так: «Учень Іван Іваненко навчається в 5 класі ліцею номер 30, що має спеціалізацію математика».
19	Створити батьківський клас Книга (Book). У метод ініціалізації передається тільки аргументи для ініціалізації атрибутів **title** та **author**, тоді як у самому методі атрибуту **pages** має бути автоматично привласнене випадкове ціле число в діапазоні від 100 до 700. Атрибути: 1. Назва **title**. 2. Автор **author**. 3. Кількість сторінок **pages**. Методи: 1. Виведення інформації про книгу **about()**. 2. Метод визначення ціни книги **price()**. Ціна розраховується як кількість сторінок * 3. Якщо автором є Тарас Шевченко або Леся Українка, то вартість книги збільшується вдвічі. На основі батьківського класу Книга (Book) створити підкласи Електронна книга (EBook) та Аудіокнига (Audiobook). Обидва підкласи

№ вар	Умови завдання
	мають додатковий атрибут **size** – розмір файлу книги в мегабайтах. Підклас Аудіокнига також має власний атрибут **length** – тривалість запису (у хвилинах). Перевизначити у підкласах метод **about()** для виведення додаткової інформації про власні атрибути. Також у підкласах перевизначити метод **price()**, який повертає: – для електронної книги: (результат роботи методу **price()** батьківського класу) / розмір файлу книги; – для аудіокниги: (результат роботи методу **price()** батьківського класу) / розмір файлу книги + тривалість запису * 2. В основній програмі створіть функцію **sell_book()** для продажу книги. Функція приймає екземпляр книги та виводить повідомлення «{тип книги} книга {назва книги} була продана за {ціна} грн», де {тип книги} – Друкована / Електронна / Аудіо (в залежності від класу). Створити список з декількох екземплярів книг різних класів та по черзі продати їх усі.
20	Створити батьківський клас Тварина (Animal). Встановлення усіх атрибутів має відбуватись у методі ініціалізації екземпляра класу. Атрибути: 1. Прізвисько **alias**. 2. Улюблена їжа **food**. Методи: 1. Метод **eat()**, який виводить повідомлення «Тварина їсть». 2. Метод **sound()**, який виводить повідомлення «Тварина видає звуки». На основі батьківського класу Тварина (Animal) створити підкласи Кіт (Cat), Собака (Dog) та Кінь (Horse). Усі підкласи повинні мати додатковий власний атрибут порода **breed**. У підкласах перевизначити метод **eat()**, змінивши повідомлення наступним чином: «{тип тварини} на {прізвисько} їсть», де тип тварини – кіт, собака або кінь (в залежності від підкласу). Також перевизначити у підкласах метод **sound()**, замінивши повідомлення на «Няв!» (для кота), «Гав!» (для собаки) або «Іго-го!» (для коня). Створити клас Ветеринар (Vet), який не має атрибутів, але має методи: – лікувати тварину **treat_animal()** – приймає екземпляр тварини та виводить повідомлення «Ветеринар лікує {тип тварини} на прізвисько {прізвисько} та дає поїсти {улюблена їжа}». – визначити породу **determine_breed()** – виводить повідомлення «Ветеринар визначив породу тварини – це {порода}». Причому має існувати шанс 20%, що ветеринар не зможе визначити породу. Тоді має бути виведене повідомлення «Не вдалося визначити породу». В основній програмі створіть список з тварин усіх описаних підкласів та в циклі відправте їх до ветеринара для проведення лікування та

№ вар	Умови завдання
	визначення породи.
21	Створити батьківський клас Електротранспорт (EVehicle). У метод ініціалізації передається тільки аргумент для ініціалізації атрибуту **capacity**, а атрибуту **max_distance** має бути автоматично привласнене випадкове ціле число в діапазоні від 20 до 100. Атрибути: 1. Ємність акумулятора **capacity** (в ампергодинах). 2. Максимальна дальність їзди на одному заряді **max_distance** (у кілометрах). Методи: 1. Виведення інформації про параметри транспорту **params()**. 2. Метод **calculate_loss()**, що дозволяє визначити втрату заряду акумулятора на 1 км. Втрата розраховується як ємність акумулятора / максимальна дальність їзди. 3. Метод **charge_loss()** – визначення витрат заряду акумулятора для поїздки на задану відстань. Метод приймає відстань поїздки (в км) та повертає відсоток заряду акумулятора, що буде витрачений на здійснення такої поїздки. На основі батьківського класу Електротранспорт (EVehicle) створити підкласи Електровелосипед (EBike) та Електросамокат (EScooter). У обох підкласах має бути додатковий атрибут бренд (виробник) **brand** та **freeway**, який визначає частину часу поїздки без використання акумулятора, тобто коли рух здійснюється силами людини (має бути числом у діапазоні від 0 до 1). Перевизначити у підкласах метод **params()**, який тепер повинен виводити інформацію і про бренд виробника транспортного засобу. У підкласах перевизначити метод **charge_loss()**, який тепер повинен повертати (результат роботи батьківського методу **charge_loss()**) * (1 - N). В основній програмі створити функцію **ride()**, яка приймає екземпляр транспортного засобу та виводить повідомлення «Розпочинаю поїздку на {тип транспортного засобу}», де тип транспортного засобу – електровелосипед або електросамокат (в залежності від класу).
22	Створити батьківський клас Напій (Drink). Встановлення усіх атрибутів має відбуватись у методі ініціалізації екземпляра класу. Атрибути: 1. Назва **title**. 2. Густина **density** (кг/м^3). Методи: 1. Інформація про напій **info()** – виводить повідомлення про значення всіх атрибутів. 2. Метод розрахунку вартості **cost()**, який розраховує вартість

№ вар	Умови завдання
	одного літру напою як густина / 0.15 та виводить повідомлення «Напій {назва} коштує {вартість} грн за 1 л». На основі батьківського класу Напій (Drink) створити підкласи Алкогольний напій (Alcohol) та Сік (Juice). Підклас Алкогольний напій має додатковий атрибут міцність **strength** (% об.), а підклас Сік – смак **taste**. У підкласах перевизначити метод **info()** для виведення цієї додаткової інформації. Перевизначити у підкласі Алкогольний напій метод **cost()**, який тепер повинен розраховувати вартість як (результат роботи методу **cost()** батьківського класу) * (1 + міцність/100). В основній програмі створити функцію для замовлення напою **order_drink()**. Якщо до функції передається екземпляр напою та необхідний об'єм замовлення (в літрах), функція має повертати повідомлення «{Об'єм замовлення} літрів {тип напою} буде коштувати Вам {вартість напою * об'єм замовлення} грн», де тип напою – алкоголь або сік (в залежності від підкласу). Наприклад, якщо до функції передати екземпляр Алкогольного напою та об'єм 0.5 л, повідомлення має бути таким: «0.5 літрів алкогольного напою буде коштувати Вам 100 грн». Якщо до функції передати тільки екземпляр напою і не вказувати об'єм замовлення, функція має вивести аналогічне повідомлення з вартістю 1 л даного напою.
23	Створити батьківський клас Будівля (Building). У метод ініціалізації передається тільки аргументи для ініціалізації атрибутів **length** та **width**, а атрибуту **floors** має бути автоматично привласнене випадкове ціле число в діапазоні від 1 до 20. Атрибути: 1. Довжина **length** (м). 2. Ширина **width** (м). 3. Кількість поверхів **floors**. Методи: 1. Розрахунок площі фундаменту будівлі **square()**. 2. Розрахунок площі всіх приміщень будівлі **total_square()**, що обчислюється як площа фундаменту * кількість поверхів. На основі батьківського класу Будівля (Building) створити підкласи Житловий будинок (House), Офісний центр (OfficeCenter) та Торговий центр (TradeCenter). Підкласи повинні мати додатковий атрибут: кількість квартир **flats** / кількість офісів **offices** / кількість магазинів **shops** (в залежності від конкретного підкласу). Також усі підкласи повинні мати атрибут **price**, в якому зберігається значення вартості 1 $м^2$ приміщення у цьому будинку. У підкласах перевизначити метод **total_square()** так, щоб окрім загальної площі він повертав площу однієї квартири / офісу / магазину (вважати, що всі приміщення однакові). В основній програмі створити функцію **rent()**. Функція приймає

№ вар	Умови завдання
	екземпляр будинку та розраховує вартість оренди однієї квартири / офісу / магазину в цій будівлі. Функція має виводити повідомлення «Вартість оренди {тип приміщення} складає {число} грн», де тип приміщення – квартира, офіс або магазин (визначається автоматично в залежності від підкласу).
24	Створити батьківський клас Персональний комп'ютер (PC). Встановлення усіх атрибутів має відбуватись у методі ініціалізації екземпляра класу. Атрибути: 1. Частота процесора **frequency** (ГГц). 2. Об'єм оперативної пам'яті **ram** (ГБ). Методи: 1. Виведення загальної інформації про частоту процесора та об'єм оперативної пам'яті **about()**. 2. Метод **power()**, який розраховує обчислювальну потужність ПК як частота процесора * об'єм оперативної пам'яті. На основі батьківського класу Персональний комп'ютер (PC) створити підкласи Ігровий ПК (GamerPC) та Робоча станція (Workstation). Кожен підклас має додатковий атрибут відеокарта **graph_card** – рядкова змінна, значення якої автоматично обирається під час ініціалізації зі списку ['NVIDIA', 'Radeon', 'Integrated']. У підкласах перевизначити метод **about()** для додаткового виведення інформації про новий атрибут. Перевизначити у підкласах метод **power()** наступним чином: обчислювальна потужність становить (результат роботи методу **power()** батьківського класу) * k, де k – коефіцієнт, який залежить від типу комп'ютера та відеокарти: – для Ігрового ПК: k=1.5, якщо відеокарта NVIDIA; k=1.2, якщо відеокарта Radeon; k=0.8, якщо відеокарта інтегрована; – для Робочої станції: k=1.2, якщо відеокарта NVIDIA; k=1.3, якщо відеокарта Radeon; k=0.9, якщо відеокарта інтегрована. У основній програмі створити список із декількох екземплярів кожного з типів ПК. В циклі обійти всі створені ПК та для кожного із них вивести повідомлення «Це {тип ПК} з обчислювальною потужністю {значення}», де тип ПК – ігровий / робоча станція (в залежності від класу об'єкта).
25	Створити батьківський клас Лікар (Doctor). У метод ініціалізації передається тільки аргумент для ініціалізації атрибуту **name**, тоді як атрибуту **specialization** у методі ініціалізації має бути привласнене значення «Без спеціалізації», а атрибуту **cost** має бути привласнене значення 500. Атрибути: 1. Ім'я лікаря **name**.

№ вар	Умови завдання
	2. Спеціалізація лікаря **specialization**. 3. Базова вартість прийому **cost**. Методи: 1. Метод виведення інформації про лікаря **info()**, який повертає назву та значення кожного атрибута. 2. Метод розрахунку вартості прийому **reception_cost()**, який повертає базове значення вартості прийому. На основі батьківського класу Лікар (Doctor) створити три підкласи – Терапевт (Therapist), Стоматолог (Dentist) або Окуліст (Ophthalmologist). Атрибут **specialization** у підкласах має містити текст із назвою відповідної спеціалізації. У кожному з підкласів додатково має бути унікальний атрибут. У Терапевта це кількість зареєстрованих пацієнтів (**patients**), у Стоматолога – стаж роботи (**experience**), у Окуліста – день тижня, в який відбувається прийом пацієнтів (**schedule**) (вибір реалізувати рандомно під час ініціалізації екземпляру зі списку ['пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'нд']). У підкласах перевизначити метод **info()** для додаткового виведення інформації про нові атрибути. Також у підкласах перевизначити метод для розрахунку вартості прийому **reception_cost()**: – Вартість прийому терапевта розраховується як: (результат роботи методу **reception_cost()** батьківського класу) + 0,2% за кожні 500 пацієнтів понад норму в 1500; – Вартість прийому стоматолога: (результат роботи методу **reception_cost()** батьківського класу) + 10% за кожні 10 років досвіду; – Вартість прийому окуліста: (результат роботи методу **reception_cost()** батьківського класу) * D, де D – коефіцієнт дня тижня: якщо будній день, то D=0.05, якщо вихідний – D=2. В основній програмі додатково створити поліморфічну функцію **appointment()**, яка буде приймати екземпляр класу в якості аргументу та виводити повідомлення з інформацією про вартість прийому у лікаря (наприклад, «Прийом у лікаря {спеціалізація} {ім’я} обійдеться у {вартість прийому} грн»).
26	Створити батьківський клас Інформаційний ресурс (Resource). У метод ініціалізації передається тільки аргументи для ініціалізації атрибутів **title** та **owner**, а атрибуту **subscribers** має бути автоматично привласнене випадкове ціле число в діапазоні від 100 до 100000. Атрибути: 1. Назва **title**. 2. Власник **owner**. 3. Кількість підписників **subscribers**. Методи:

№ вар	Умови завдання
	1. Виведення інформації про ресурс **about()**. 2. Метод визначення достовірності інформації на ресурсі **reliability()**. Вважатимемо, що чим більше підписників, тим достовірніша інформація публікується: достовірність розраховується як кількість підписників * 0.001. Якщо автором є видання Нью Йорк Таймс (NYT) або BBC, то достовірність автоматично 99%. На основі батьківського класу Інформаційний ресурс (Resource) створити підкласи Новини (News), Спорт (Sport) та Наука (Science). Усі підкласи мають додатковий атрибут **publications** – кількість публікацій в день. Підклас Новини також має власний атрибут **comments** – середня кількість відгуків/коментарів читачів на 1 публікацію. Перевизначити у підкласах метод **about()** для виведення інформації про додаткові нові атрибути. Також у підкласах перевизначити метод **reliability()**, який повертає: – для ресурсу з новинами: (результат роботи методу **reliability()** батьківського класу) * кількість відгуків / кількість публікацій в день; – для ресурсу про спорт: (результат роботи методу **reliability()** батьківського класу) / кількість публікацій в день; – для ресурсу про науку: (результат роботи методу **reliability()** батьківського класу) / (0.5 * кількість публікацій в день). В основній програмі створіть функцію **read()** для формування переліку ресурсів, що читаються. Функція приймає екземпляр ресурсу та виводить повідомлення «{тип ресурсу} ресурс {назва} подає надійну інформацію на {достовірність}%», де {тип ресурсу} – Новини / Спорт / Наука (в залежності від класу). Створити список з декількох екземплярів рандомних ресурсів різних класів та вивести їх перелік (застосувати функцію **read()**).
27	Створити батьківський клас Черга (Queue). У метод ініціалізації не передаються ніякі аргументи, тоді як у самому методі атрибуту **title** має бути привласнене значення «Просто черга», атрибуту **people** – пустий словник. Атрибути: 1. Назва **title**, яка характеризує призначення черги. 2. Словник **people**, який буде в якості ключа містити номер клієнта у черзі, а як значення – деякий список параметрів клієнта. Методи: 1. Інформація **info()** – виводить повідомлення про чергу. 2. Метод додавання клієнтів у чергу **add()**. Метод приймає тільки один аргумент – ім’я та прізвище клієнта. При цьому у методі відбувається аналіз останнього ключа в словнику **people**, після чого створюється новий ключ із наступним послідовним номером клієнта, значенням якого буде введене прізвище та ім’я. Наприклад, у черзі 5 клієнтів, причому останній ключ дорівнює 8 (так може бути тому, що три клієнти були видалені із черги), тоді виклик методу **add()** повинен призвести до створення нового

№ вар	Умови завдання
	ключа, що є числом 9, а його значенням буде введене прізвище та ім'я. 3. Метод видалення клієнтів із черги **remove()**. Метод приймає лише один аргумент – номер клієнта у черзі, після чого видаляє відповідний ключ і значення зі словника **people**. При цьому, якщо як аргумент було передано неіснуючий номер клієнта, то має бути виведене повідомлення, що клієнта із таким номером немає у черзі. 4. Метод розрахунку довжини черги **length()**, який повертає поточну кількість клієнтів у черзі. На основі батьківського класу Черга (Queue) створити підкласи Черга на реєстрацію місця проживання (Registration) та Черга на отримання допомоги (Help). Атрибут **title** має приймати відповідне текстове значення у кожному підкласі (для класу Registration це має бути текст «Черга на реєстрацію місця проживання», для класу Help – «Черга на отримання допомоги»). Перевизначити у підкласі Черга на реєстрацію місця проживання (Registration) метод **length()**, який тепер повинен розраховувати довжину не у кількості людей, а у кількості часу, яка знадобиться на обслуговування цієї черги: (результат роботи методу **length()** батьківського класу) * (10 хвилин), та друкувати додаткове повідомлення типу «До черги {назва} записано {кількість} клієнтів, що складає {години} годин та {хвилини} хв». У підкласі Черга на отримання допомоги (Help) перевизначити метод **add()** таким чином, щоб він приймав два аргументи: ім'я та прізвище (одна рядкова змінна) і вік. Принцип формування номера ключа словника такий самий, як у методі батьківського класу, тоді як значення формується у вигляді списку із двох елементів: перший – ім'я та прізвище (одна рядкова змінна), другий – вік. У підкласі Черга на отримання допомоги (Help) створити додатковий метод **calculate_help()**, за допомогою якого визначається орієнтовна необхідна допомога кожному клієнту з черги з урахуванням віку: (базова ставка в 5000 грн) * K, де K – коефіцієнт віку (до 10 років включно K=2, від 10 до 16 років включно K=1.5, від 16 до 60 років включно K=1, після 60 років K=1.8). Метод повинен повертати повідомлення типу «Орієнтовна допомога клієнту за номером {номер} складає {розмір допомоги} грн» (і таких повідомлень має генеруватись стільки, скільки є клієнтів наразі у черзі). В основній програмі додатково створити поліморфічну функцію **result()**, яка буде приймати екземпляр класу будь-якої із черг в якості аргументу та виводити інформацію про чергу шляхом застосування доступних методів.
28	Створити батьківський клас Інженер (Engineer). У метод ініціалізації передаються тільки аргументи для ініціалізації атрибутів **name** та **proven_record**, а атрибуту **employe_salary** має бути привласнене значення 15000.

№ вар	Умови завдання
	Атрибути: 1. Ім’я службовця **name**. 2. Стаж роботи (років) **proven_record**. 3. Базовий оклад **employe_salary** (у грн). Методи: 1. Метод виведення інформації про службовця **info()**, який повертає назву та значення кожного атрибута. 2. Метод розрахунку зарплати **salary()**, який повертає значення зарплати, отримане за формулою Базовий оклад + Стаж * 0,15. На основі батьківського класу Інженер (Engineer) створити три підкласи – Інженер-конструктор (DesignEng), Інженер програмного забезпечення (SoftEng) і Веб-інженер (WebEng). У кожному з підкласів має бути унікальний атрибут. У Інженера-конструктора – це досвід (**experience**) в роках, у Інженера ПЗ – кількість вирішених задач (**solved_tasks**), у Веб-інженера – середня оцінка від замовників (**estimation**) в межах від 1 до 5. Перевизначити у підкласах метод **info()** для виведення інформації про додаткові нові атрибути. Також у підкласах перевизначити метод для розрахунку зарплати **salary()**: – зарплата Інженера-конструктора розраховується як: (результат роботи методу **salary()** батьківського класу) + Досвід * 0,05; – зарплата Інженера ПЗ: (результат роботи методу **salary()** батьківського класу) + Кількість вирішених задач * 0,15; – зарплата Веб-інженера: (результат роботи методу **salary()** батьківського класу) + Оцінка від замовників * 0,5 (за умови що Оцінка рівна або вище 4, у іншому випадку оцінка від замовників не враховується). В основній програмі створити поліморфічну функцію **greetings()**, яка в якості аргументу приймає екземпляр класу та повертає рядок з привітанням, яке буде різним для кожного класу («Вітаємо інженера {ім’я}» для класу Engineer, «Вітаємо інженера-конструктора {ім’я}» для класу DesignEng тощо). Створити список, який складається з декількох екземплярів різних класів. Далі в циклі обійти всі елементи списку та для кожного із них викликати метод **salary()**, а також передати кожен елемент до функції **greetings()**.
29	Створити батьківський клас Паркінг (Parking). Встановлення усіх атрибутів має відбуватись у методі ініціалізації екземпляра класу. Атрибути: 1. Назва **name**. 2. Місткість **spaces**. 3. Вартість оренди одного місця за добу **cost**. Методи:

№ вар	Умови завдання
	1. Виведення інформації про паркінг **about()**. 2. Метод **income()** – визначення прибутку, який проносить паркінг за добу за умови повного завантаження: обчислюється як **cost * spaces**. На основі батьківського класу Паркінг (Parking) створити підкласи Велопаркінг (BicycleParking) та Автомобільний паркінг (CarParking). Обидва підкласи мають додатковий атрибут **long_term** – кількість місць, що орендовані на довгий термін. Підклас Автомобільний паркінг має ще один власний атрибут **charging_station** – кількість зарядних станцій на паркінгу. У підкласах перевизначити метод **about()**, щоб він виводив інформацію і про ці нові атрибути. Також у підкласах перевизначити метод **income()**, який має повертати дохід, розрахований як: (результат роботи методу батьківського класу) – * **long_term** * **cost * 0.8**. Створити клас Міські паркінги (CityParking), що містить атрибут **parking**, який створюється у методі ініціалізації та є списком із декількох екземплярів автомобільних та велопаркінгів. У класі створити метод **calculate_spaces()**, який розраховує та виводить загальну кількість паркомісць в місті. Також створити метод **general_income()**, який розраховує загальний дохід від паркінгів.