**2. Завдання**

Написати програму алгоритмічною мовою Python згідно з завданням, отриманим від викладача за табл. 1. Додайте відповідні конструктори, деструктори, функції доступу та виведення інформації на екран. У методі main продемонструйте роботу класів.

Таблиця 1

| **№ п/п** | **Завдання** |
| --- | --- |
| 1 | Напишіть програму, яка дозволяє користувачеві вводити прізвища п’яти кандидатів на місцевих виборах та кількість голосів, отриманих кожним кандидатом (class **Candidate**). Потім програма повинна вивести ім’я кожного кандидата, кількість отриманих голосів та відсоток від загальної кількості голосів. Додайте class **Elections**, який містить набір кандидатів і може вирахувати переможця виборів. |
| 2 | Створити клас **TimeConverter**, який дозволяє переводити час у форматі hh:mm:ss у кількість секунд і навпаки. Використати перевантаження методу (наприклад outputConvertedTime). |
| 3 | Створити клас **Student**, який міститиме його ідентифікаційні дані, масив оцінок і дозволятиме порахувати середній бал студента. Додайте class **Group**, що містить набір студентів і може виводити їх рейтинг. Реалізуйте методи додавання, видалення студента з групи. |
| 4 | Створити клас **BankAccount**, який дозволяє виконувати основні банківські операції: створити акаунт (повинен містити id, owner\_name, balance), поповнити баланс і зняти готівку. Реалізуйте метод showAccount і додайте можливість поповнювати баланс на ціле і дробове число. Додайте клас **Bank**, що містить набір рахунків і дозволяє додавати/видаляти їх по id. Додати можливість сортування аккаунтів за балансом. Реалізуйте методи додавання, видалення аккаунт з банку. |
| 5 | Створити клас **Book** (price, numberOfPages, author, quantity, numberOfSales, ...) та клас **BookShop** - набір книжок, які можна купувати. Реалізуйте необхідні методи для виведення топу книжок (за ціною і за кількістю продажів). Реалізуйте методи додавання, видалення з магазину. |
| 6 | Створити кімнату своєї мрії! Додати класи для позначення об’єктів - наприклад **Bed** (чи **Sofa**), **Table**, **Carpet**, etc. Кожен об’єкт в кімнаті повинен мати масив координат (не обов'язково перевіряти їх валідність). Створити клас **Room**, у якому додати метод Rearrengement - який змінюватиме координати об’єктів. Реалізуйте методи додавання, видалення нової речі до кімнати. |
| 7 | Створити клас **Musician**, який міститиме поля для опису виконавця чи групи - наприклад назва, гонорар, вік і тд. Додати клас **MusicFestival**, який матиме максимальний бюджет і набір виконавців. Реалізувати конструктор, який прийматиме масив музикантів і повинен перевіряти, чи сумарний гонорар не перевищує бюджету і додавати лише тих музикантів, яких може собі дозволити. Реалізуйте методи додавання, видалення виконавців. |
| 8 | Створити клас **Medicine** (price, quantity, name, isPrescriptionNeeded, expirationDate, etc) та клас **Pharmacy**, який містить асортимент ліків. Реалізуйте методи переобліку (видалення прострочених ліків), знижки (зменшення цін на всі ліки на 10%), виведення найдешевших ліків та додавання, видалення лікарства з аптеки. |
| 9 | Створити клас **Flower** (height, size, color, price, quantity, deliveryRate) та клас **FlowerShop**, що містить в асортименті набір квітів. Створіть клас **Bouquet**, у який можна додавати будь-яку кількість квітів та дозвольте підрахувати його вартість. Реалізуйте методи виведення топу найдорожчих квітів в магазині та додавання, видалення з магазину. |
| 10 | Створити клас **File**(name, extension, size) та клас **Folder,** що може містити в собі і файли, і папки відповідно. Створіть файлову систему з вкладених об’єктів. Реалізуйте методи виведення дерева файлів, найдовшого шляху до об’єкту (path). |
| 11 | Створити клас **Smartphone**, який містить поля для опису телефону, як мінімум - ціна, модель, масив (або пара) мобільних номерів, обсяг пам’яті, ємність батареї і тд. Створити клас **PhoneStore**, який має асортимент(масив чи колекцію) різних телефонів. Реалізуйте метод вибору найкращого телефону із заданим максимальним бюджетом та виведення списку телефонів, сортованого за ціною. |
| 12 | Створити клас **Bug**(проблема у проекті чи продукті), який міститиме опис, severity, deadline, статус, assignee і тд. Додати клас **Backlog**, який містить список баг. Реалізувати можливість виведення баг певного розробника, які мають статус RESOLVED та сортування беклогу за severity. |
| 13 | Створити клас **Fish**(name, age, species, size, prefferedFood, isAggressive, neededSpace (cubic meters)). Створити клас **Aquarium**, який мiстить різних рибок, та має поля totalVolume (cubic meters) і freeSpace (cubic meters). Додати методи для виведення 3-ьох найбільших риб у акваріумі. У методі main створіть декілька акваріумів, враховуючи вимоги рибок (агресивних можна додавати лише до агресивних, вільне місце в акваріумі повинне завжди залишатися >= 0) та продемонструйте роботу класів. |
| 14 | Створити клас **Fruit** (name, size, color, taste(SWEET, SOUR, NEUTRAL), etc). Додати enum **Topping**, який також має поле taste (SWEET, SOUR, NEUTRAL). Створити клас **FruitSalad**, який складається з набору фруктів і певного топінгу (можна використати enum). Реалізувати метод для вибору топінгу, який залежить від смаку інгредієнтів (на свій смак)). Реалізувати метод для перемішування інгредієнтів. |
| 15 | Створити клас **Sneakers** (brand, size, color, price, quantity, material, numberOfSales, etc). Додати клас **SportShoesStore**, який містить асортимент кросівок. Реалізувати методи для сортування взуття за ціною і кількістю та можливість вивести топ найпопулярніших кросівок. |
| 16 | Створіть клас **Fighter** (name, health, damagePerAttack). Додайте клас **Fight**, який міститиме пару бійців і метод, який повертає ім’я переможця в поєдинку двох бійців. Кожен боєць по черзі нападає на іншого, і той, хто першим вбиває іншого, перемагає. Смерть визначається як стан здоров'я <= 0. |
| 17 | Створити клас **Point (x, y).** Створити клас **Polynom**, у котрого точки (будь-яка кількість) - це об’єкти Point. Додати enum **colour**: RED, GREEN, BLUE, ... Додайте функцію визначення периметру, найдовшої діагоналі та виведіть їх довжини. Визначте функції сортування точок за абсцисою(х) та ординатою (у). |
| 18 | Створити клас **Planet (name, mass in kg, orbital velocity, mean temperature, length of day, distance from sun ).** Додати enum **type**: TERRESTRIAL, JOVIAN. Створити клас **Planetary**, котрий складається з певної кількості планет. Додайте функцію сортування за довжиною світлового дня. Визначте функції **findDistanceBetween**(planetA, planetB), **findAverageMass**( Planet\* ) - знайти середню масу планет серед заданих. |
| 19 | Створити клас **Pet (name, breed, age, greeting, mass).** Додати enum **kind**: DOG, CAT, BIRD, …. . Додати клас **Home**, котрий містить певний перелік домашніх улюбленців. Додати функцію **areFriends( Pet\*),** котра приймає на вхід перелік тварин та шукає друзів між ними (дві тваринки вважаються друзями, якщо різниця віку у них менше 2 років). До класу **Pet** дати метод **isPolite(self)**: якщо **greeting** містить “Hello”, то поверніть True, у іншому випадку - False. Додайте функцію сортування за віком. |
| 20 | Створити клас **Atom (name, atomic mass unit, neutrons number, protons number, electrons number ).** Додати enum **type**: ISOTYPE, RADIOACTIVE, ION, ANTIMATTER, STABLE. Створити клас **Molecule (name)**, котра складається з певної кількості атомів. Додайте функцію сортування атомів за масою. Визначте функцію **findAverageMass**( Atom\* ) - знайти середню масу атомів серед заданих. До класу **Atom** дати метод **isNeutral(self)**: якщо сума кількості нейтронів та електронів рівна**,**  то поверніть True, у іншому випадку - False. |
| 21 | Створити клас **Weather (day, city, country, temp, humidity, wind speed).** Додати enum **type**: SUNNY, CLOUDY, RAINY, FOGGY, …. Додати клас **WeatherCalendar**, котрий містить об’єкти **Weather.** Визначити функцію **findMaxTemprature**( Weather\*, day) - знайти максимальну температуру з переліку записів погоди за певним днем, якщо такого дня не існує, то вивести: “Not enough data”. Додати метод передбачення, чи запис погоди належить до Львова **isLvivWeather(humidity, type)**: якщо humidity > 80% і type = “RAINY”, то вивести: “The typical day in Lviv”, у іншому випадку: “You’re lucky, man”. Просортуйте записи погоди за днем. |
| 22 | Створити клас **Clothing (name, description, location, colour, size).** Додати enum **type**: SHIRT, JEANS, JACKET, …. Додати клас **Wardrobe**, котрий містить одяг**.** Визначити функцію are **goOut**( **Clothing**\*) - вивести кількість типів одягу та чи готова людина вийти на вулицю (якщо кількість типів більша, ніж три - так, менша - ні). Посортуйте одяг за розміром. Додайте ще один метод до класу одягу. |
| 23 | Створити клас **Guest( id, name, age, city, phone number)**. Додати enum **gender**: FEMALE, MALE, NON\_BINARY, …. Додати клас **Party** (day, reason), на яку прийде певна кількість гостей **guests.** Додати функцію **findAverageAge**( Guest\*, gender) - знайти середній вік гостей за статтю. Додати метод визначення, чи номер телефону є щасливий **isLuckyPhoneNumber** (якщо більше, ніж три сімки - True). Посортуйте гостей за номером id. |
| 24 | Додати клас **Candy (name, mass in grams, amount, price).** Додати enum **type**: BAR, BUTTON, POPCORN, GUM, … Додати клас **Dinner (day, time),** який складається з певної кількості цукерок. Додати функцію пошуку топ 3 найдорожчих цукерок **findTheMostExpensiveCandies** ( Candy\*, name). До класу **Candy** додати метод **ate,** котрий типово повертає: “What delicious candies!”, але коли **mass \* amount** > **2 кг**, повинен вивести “You’re on a diet!”. Додайте сортування за певним параметром. |
| 25 | Додати клас **Movie (id, title, ranking, release date, character number, ticket price, comment).** Додати enum **type**: ACTION, COMEDY, DRAMA, FANTASY, … Додати клас **Cinema (name, location),** у якому відбуваються покази певного переліку фільмів. Додати функцію підрахунку прибутку за певним днем **calculateProfit** ( Movie\*, day). Реалізувати метод для вибору фільму, який залежить від певних параметрів (на свій смак). Додати можливість сортування фільмів за датою випуску. |
| 26 | Додати клас **Country (name, capital, code, population, area, GDP).** Додати enum **governmentType**: DEMOCRACY, REPUBLIC, AUTOCRACY, … Додати клас **Land (name, ...),** до якого можуть належати певні країни. Додати функцію підрахунку густини населення серед певних країн **сalculatePopulationDensity** (Countries\*). Реалізувати метод сортування за GDP та виводу топу країн з найкращою економікою. Реалізувати метод для вибору країни, котрий залежить від певних параметрів (на свій смак). Додати можливість сортування фільмів за датою випуску. |
| 27 | Створити клас **RaceHorse (speed, age, name, placeInRace,…)** Додати клас **Race**, що міститиме колекцію учасників перегонів. Клас повинен містити функції для додавання та видалення нових учасників з перегонів, вибору переможця за формулою найбільша швидкість + найближчий до середнього вік серед усіх учасників. Створити умови для додавання коней (наприклад - за віком від 3 до 7). Додати можливість сортування за швидкістю. Першим трьом учасникам присвоїти відповідні **placeInRace**. |
| 28 | Створити клас **Автомобіль (brand, age, maxSpeed, horsePower, і тд)**. Зробити поле марки enum. Створити клас **Паркінг** з колекцією автомобілів і максимальною кількістю місць. У разі відсутності вільних місць виводити повідомлення про переповнення стоянки. Присвоїти ціну однієї години стоянки. Додати методи **parkCar**, **leaveParking**. Посортувати машини за тривалістю стоянки та порахувати ціну паркінгу для кожної. |