Задачи за домашно към упражнение 7

Задача 1

Подберете подходящи модификатори за достъп в задачата и поставете сорс файловете в пакет bg.tu_varna.sit.task1

Да се създаде програма за книжарница.

За целта са необходими:

- Клас Author с атрибути име (firstName), фамилия (lastName) и държава (country).

- Интерфейс Margin с метод calculateMargin(), който изчислява надценката на дадена книга;

- Клас за описание на грешка InvalidDataException с подходящо съобщение за грешка, при инициализиране на обекти с невалидни стойности;

- Енумерация за тип на корицата (CoverType), със стойности твърди корици (hardcover) и меки корици (paperback);

- Абстрактен клас Book, който имплементира интерфейс Margin. Класът има атрибути заглавие (title), автор (author), година на издаване (publishingYear), цена (price) и тип на корицата (coverType).

Дефинирайте необходимите конструктор и метод за текстово описание, както и метод getFinalPrice(), който изчислява и връща крайната цена на книгата. Необходими са проверки за заглавието, автора и цената (да не е под 5);

- Клас CrimeNovel, наследяващ Book. Надценката се изчислява като 2% от цената ако книгата е с твърди корици и като 1% от цената в случай на меки корици;

- Клас PoetryBook, наследяващ Book. Надценката се изчислява като 1% от цената ако книгата е с твърди корици и е издадена след 2000 година, и като 5% в останалите случаи;

- Клас SciFiNovel, наследяващ Book. Надценката се изчислява като 9% от цената ако книгата е с меки корици и 12% от цената в останалите случаи.

Създайте клас Bookstore, който има поле масив от книги (books). Дефинирайте необходимият конструктор и методи:

- calculateTotalBookPrice(), който изчислява и връща цената на всички книги;

- calculateAverageBookPrice(), който изчислява и връща средната цена на книгите в книжарницата.

Дефинирайте клас Application с главна функция и тествайте описаните функционалности.

Задача 2

Подберете подходящи модификатори за достъп в задачата и поставете сорс файловете в пакет bg.tu_varna.sit.task2

Да се създаде програма, описваща агенция за недвижими имоти.

За целта са необходими:

- Интерфейс Commission с метод calculateCommission(), който изчислява и връща комисионната за даден имот;

- Клас за описание на грешка InvalidDataException с подходящо съобщение за грешка, при инициализиране на обекти с невалидни стойности;

- Енумерация за тип на имота (PropertyType) със стойности за отдаване под наем (rent), за продажба (sale) и неопределен (undefined);

- Енумерация за паркинг () с атрибути, указващи броя места за паркиране (numberOfParkingPlaces) и тип на имота (за отдаване под наем или за продажба). Стойностите са: няма паркинг (noParkingLot), едно място за отдаване под наем (onePlaceForRent), едно място за продажба (onePlaceForSale), две места за отдаване под наем (twoPlacesForRent), две места за продажба (twoPlacesForSale), места за отдаване под наем (placesForRent) и места за продажба (placesForSale).

Последните две стойности имат като дефолтен брой места 100;

- Абстрактен клас Property, който имплементира интерфейс Commission. Класът има атрибути площ(area), цена(price) и тип на имота (propertyType). Дефинирайте необходимите конструктор и методи за достъп;

- Клас Apartment, наследник на Property, който има като атрибути брой стаи (numberOfRooms), етаж (numberOfRooms) и паркинг (parkingLot). Комисионната се изчислява като 15% от цената ако апартамента е за отдаване под наем и има едно или две паркоместа за отдаване под наем; 10% ако площта му е по-малка от 60 кв.м. и 7% от цената в останалите случаи;

- Клас House, наследник на Property, който има като атрибути брой етажи (numberOfFloors) и атрибут, указващ наличието на градина (hasGarden). Комисионната се изчислява като 8% ако къщата се отдава под наем и има градина; 5% ако къщата е за продажба и площта е по-малка от 100 кв.м. и 3% от цената в останалите случаи;

- Клас Office, наследник на Property, който има атрибути паркинг (parkingLot) и брой стаи (numberOfRooms). При създаване на обекта е необходима проверка за съвпадение на типа имот. Комисионната се изчислява като 18% от цената ако офисът е за отдаване под наем, 15% ако има паркинг и повече от две стаи, и 11% във всички останали случаи;

- Клас Shop, наследник на Property. Комисионната се изчислява като 2% ако магазинът е за отдаване под наем и площта му е по-малка от 50 кв.м., 1% ако площта му е над 100 кв.м. и 6% във всички останали случаи.

Създайте клас RealEstateAgency, който има полета за име (name) и масив от недвижими имоти (properties). Дефинирайте необходимите конструктор, методи за достъп и методи:

- calculateTotalExpectedCommission(), който изчислява и връща очакваната комисионна от всички предлагани имоти;

- getPropertiesForSale(), който намира и връща броя недвижими имоти, предлагани от агенцията за продажба.

Дефинирайте клас Application с главна функция и тествайте описаните функционалности.

Задача 3

Подберете подходящи модификатори за достъп в задачата и поставете сорс файловете в пакет bg.tu_varna.sit.task3

Да се създаде програма за магазин.

За целта са необходими:

- Интерфейс Delivery с метод needsDelivery(), който указва дали дадената стока се нуждае от доставка;

- Енумерация тип стока (ItemType) със стойности храна (food) и напитка (drink);

- Абстрактен клас Item, имплементиращ Delivery. Класът има атрибути име (name), срок на годност в дни (daysToExpire),

налично количество (availableQuantity) и тип на стоката (itemType). Дефинирайте необходимия конструктор;

- Клас Bread, наследяващ Item и с атрибут съдържание на бяло брашно (whiteFlourPercentage). Дефинирайте необходимия конструктор. Има необходимост за доставка ако срокът за годност е по-малък от 5 дни и хлябът е със съдържание на бяло брашно над 75 или са останали по-малко от 10 бройки;

- Клас Cheese, наследяващ Item и с атрибут грамаж на опаковката (gramsPerPackage). Дефинирайте необходимия конструктор. Има необходимост от доставка ако са останали по-малко от 10 бройки сирене в опаковки до 400 грама;

- Клас Milk, наследяващ Item и с атрибут, указващ процента масленост (fatPercentage). Дефинирайте необходимия конструктор. Има необходимост от доставка ако млякото е трайност до 15 дни, останали са по-малко от 10 броя и маслеността е по-голяма от 2%;

- Клас Water, наследяващ Item и с атрибут обща минерализация (mineralContent). Дефинирайте необходимия конструктор. Има необходимост от доставка ако са останали по-малко от 30 броя или ако срокът на годност изтича след по-малко от 20 дни.

Създайте клас Shop, който има като атрибут масив от стоки (items) и дефинирайте необходимия конструктор. Класът има методи:

- getItemCountDelivery(), който изчислява и връща броя стоки, които се нуждаят от доставка;

- getAverageDaysToExpire(), който изчислява и връща средния срок на годност за стоките в магазина;

- getAvailableDrinksCount(), който намира и връща колко са наличните напитки в магазина.

Дефинирайте клас Application с главна функция и тествайте описаните функционалности.

Задача 4

Подберете подходящи модификатори за достъп в задачата и поставете сорс файловете в пакет bg.tu_varna.sit.task4

Съставете програма, която има за цел да изчислява добива на риба от плавателните водни басейни на дадена територия.

Предвидете клас FishList, съдържащ списък на всички риби, обитаващи територията. За казуса използвайте рибите ester, catfish, perch, tench, piranha.

Създайте клас риба (Fish) с полета наименование (name), приемащо стойности от представения по-горе списък, и количество от нея по експертна оценка, тона (quantity). Класът се разширява от:

- ядлива риба (EdibleFish), която допълва информацията с процент от популацията, която е годна за улов (възрастни индивиди) (percentOfYield);

- негодна за консумация риба (NonEdibleFish), където допълнително като текст се описва причината тя да е неизползваема или да е заплаха за здравето (threat).

Създайте интерфейс Usage, който да съдържа boolean метод isProductabe().

Интерфейсът да се имплементира от клас воден басейн (WaterBody). Водният басейн се описва с наименование (name), дълбочина (depth) и масив от риби, които го обитават.

Имплементира метода isProductabe(), където за потенциално продуктивен се счита басейн, който е обитаван от поне една годна за консумация риба с количества, които разрешават нейния улов (над 10 тона).

Да се добави абстрактно поведение isFloaty(), чието предназначение е да определи дали водният басейн е плаваем или не.

Да се допълни с метод calculateProduction(), който да изчислява възможно най-големите количества риба, които законно могат да се уловят от плавателния басейн.

WaterBody се наследява от:

- Езеро (Lake) с широчина (width) и дължина (length). Плаваемо е, ако дълбочината му е поне 5 м., а широчината и дължината – поне 1000 м.

- Река (River) със скорост на течението (km/h) (speed). Плаваема е, ако дълбочината е поне 3 м. и скоростта на течението не надвишава 30 km/h.

Създайте клас за изключения, които възникват в класа WaterBody (WaterBodyException). Класът за изключенията да приема съобщение за грешка като параметър на конструктора си.

Приложете класа за изключения WaterBodyException в параметризирания конструктор на класа WaterBody. Да обработва подадени отрицателни стойности на полето depth, като извежда съобщение "Дълбочината не може да бъде отрицателна величина". Предайте изключението към конструкторите на класовете Lake и River.

Задача 5

Подберете подходящи модификатори за достъп в задачата и поставете сорс файловете в пакет bg.tu_varna.sit.task5

Съставете програма, с която да управлявате наличностите от напитки в бар.

Създайте интерфейс Delivery, който да съдържа два метода:

- needOfDelivery(), с който да се установява има ли (true) или няма (false) нужда от доставка от дадена напитка;

- deliver(), с помощта на който да се доставят нови количества от съответната напитка. Приема като параметър доставените количества.

Добавете втори интерфейс Serving, който съдържа метод serve() за сервиране на единица от съответната напитка. Приема като параметър броя единици, поръчани от съответната напитка. Например при поръчка на 5 чаши натурален сок, броят на единиците е 5.

Създайте клас Drink, който имплементира двата интерфейса. Всяка напитка има наименование (name); налично количество, литра (quantity) и обем на една единица за сервиране, литра (serveQuantity). Включва имплементацията на методите:

- deliver() – увеличава наличните количества с подадените като параметър доставени количества;

- serve() - при подадени като параметър единици за сервиране извършва проверка дали има достатъчна наличност от напитката. Ако да, те се сервират директно на клиентите. Ако не, се извършва проверка колко единици са налични и само те се сервират. (сервирането става като се извадят сервираните количества от общото количество)

В текстовото представяне на напитките включете тяхното наименование, налични количества и необходимост от доставка на нови количества.

Drink се разширява от класове:

- безалкохолна напитка (SoftDrink), която допълва информацията за напитката с това дали е без захар (isSugarFree) и какъв е нейният вкус (drinkFragrance). Вкусът ѝ се избира от предварително изготвен списък с константи (Fragrances), който включва възможностите lemon, orange, kiwi, peach, apricot, pear, cola, none.

- алкохолна напитка (AlcoholicBeverage), която допълнително описва и обемния процент на съдържащия се в нея алкохол (vol). Да включва булев метод (isProper), с чиято помощ да се определя дали дадената напитка е препоръчителна за дадена възрастова категория. Методът да приема като параметър възрастта на съответното лице. Ако възрастта е в диапазона [18;21) години или над 70 г., препоръчителни са само напитките с обемен процент под 10. Между 21 и 70 г. могат да се консумират всички напитки. При въведена възраст в диапазона (0;18] да се хвърли изключение „Лица под 18 г. не могат да консумират алкохолни напитки“, а при стойност по-малка от 0 - „Въведена е невалидна възраст“. За нуждите на изключенията създайте клас AlcoholicBeverageException, наследяващ Exception.

Минималното количество от дадена безалкохолна напитка, които барът се стреми да поддържа, е 10 литра, а от алкохолна – 5 литра. Под тези наличности е необходима доставка.