Задачи за домашно към упражнение 10

Задача 1

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

Да се състави програма за стоки, предлагани в хранителен магазин. За целта са необходими:

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

- Енумерация за тип стока (ItemType) със стойности основни храни (FOOD_ESSENTIALS), храни (FOOD) и напитки (DRINKS);

- Енумерация за тип вода (WaterType) със стойности минерална (MINERAL), трапезна (TABLE), изворна (SPRING) и газирана (SODA);

- Абстрактен клас стока (Item) с атрибути за тип на стоката (itemType), единична цена (itemPrice) и наличност (availableQuantity), който имплементира интерфейси доставка и Comparable. Дефинирайте конструктор по всички полета, методи за достъп и метод за равенство (по всички полета). Методът за сравнение да е по наличност;

- Клас бисквити (Biscuits), наследник на клас стока, с атрибут име (name). Дефинирайте конструтор по всички полета, методи за достъп, равенство (по всички полета) и за текстово описание. Доставка е необходима ако наличността е под 5 бройки;

- Клас хляб (Bread), наследник на клас стока, с атрибут име (name). Дефинирайте конструтор по всички полета, методи за достъп, равенство (по всички полета) и за текстово описание. Доставка е необходима ако наличността е под 15 бройки;

- Клас масло (Butter), наследник на клас стока, с атрибути грамаж на опаковката (packing) и масленост (butterContent). Дефинирайте конструтор по всички полета, методи за достъп, равенство (по всички полета) и за текстово описание. Доставка е необходима ако има за опаковка под 250 грама има по-малко от 20 броя, или ако наличността е под 30 броя;

- Клас шоколад (Chocolate), наследник на клас стока, с атрибути име (name) и процент съдържание на какао (cocoaContent). Дефинирайте конструтор по всички полета, методи за достъп, равенство (по всички полета) и за текстово описание. Доставка е необходима ако наличността е под 10 броя;

- Клас яйца (Eggs), наследник на клас стока, с атрибут брой в опаковка (numberInPack). Дефинирайте конструтор по всички полета, методи за достъп, равенство (по всички полета) и за текстово описание. Доставка е необходима ако наличността е под 15 броя;

- Клас мляко (Milk), наследник на клас стока, с атрибути име (name) и годност в дни (daysToExpire). Дефинирайте конструтор по всички полета, методи за достъп, равенство (по всички полета) и за текстово описание. Доставка е необходима ако годността изтича след по-малко от седем дни и наличността е по-малка от 15 броя;

- Клас вода (Water), наследник на клас стока, с атрибути име (name) и тип (waterType). Дефинирайте конструтор по всички полета, методи за достъп, равенство (по всички полета) и за

текстово описание. Доставка е необходима ако газираната вода е по-малко от 30 броя или ако наличността е по-малка от 15 броя;

- Клас хранителен магазин (Shop), който има като атрибут списък с уникални стоки (items). Методи:

  - за добавяне на стока (addItem);
  - за намиране и връщане броя стоки, за които е необходима доставка (countItemsForDelivery);
  - за изчисляване и връщане на цената на стоките в магазина (calculateItemsPrice);
  - за намиране и връщане стоката с най-голяма наличност (getItemWithMostAvailableQuantity);
  - за намиране и връщане на типа стока, от която има най-малка наличност (getItemTypeWithLeastQuantity);
  - за текстово описание.

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 състави програма за банка.
За целта са необходими:

- Интерфейс лихвен калкулатор (InterestCalculator) с метод, който изчислява стойността на лихвата по дадена сметка
  (calculateInterestValue);
- Енумерация за валута (Currency) със стойности лев, долар, евро и паунд;
- Енумерация за тип на сметката (AccountType) със стойности лична (PERSONAL) и корпоративна (CORPORATE);
- Изключение за невалидни данни (InvalidDataException);
- Абстрактен клас сметка (Account) с атрибути за номер (id), тип (type), валута (currency) и наличност (balance), който
  имплементира интерфейс лихвен калкулатор. Дефинирайте конструктор по всички полета, методи за
  достъп и метод за равенство (по номер);
- Клас депозитна сметка (DepositAccount), който разширява сметката. Дефинирайте конструктор по всички полета и метод за
  текстово описание. Лихвата се изчислява като:
  - 9% от наличността при лична сметка;
  - 7% от наличността при сметка в лева;
  - 5% от наличността при сметка в долари;
  - 2% от наличността при сметка в евро;
  - 1% от наличността в останалите случаи;
- Клас сметка за заплата (SalaryAccount), който разширява сметката. Дефинирайте конструктор по всички полета и метод за
  текстово описание. Лихвата се изчислява като:
  - 0 за корпоративна сметка;
  - 5% от наличността за сметка в лева;
  - 3% от наличността за сметка в долари;
  - 2% от наличността за сметка в евро;
  - 1% от наличността в останалите случаи;
- Клас спестовна сметка (SavingsAccount), който разширява сметката. Дефинирайте конструктор по всички полета и метод за
  текстово описание. Лихвата се изчислява като:
  - 15% от наличността за лична сметка в лева;
  - 5% от наличността за сметка в лева;
  - 10% от наличността за лична сметка в долари;
  - 1% от наличността в останалите случаи;
- Клас банка (Bank), който има като атрибут асоциация <Сметка, Лихва> (bankAccounts). Методи:
  - за добавяне на сметка (addAccount), който добавя сметката към колекцията ако тя не съществува;
  - за сортиране на сметките по наличност (sortAccountsByBalance);
  - за сортиране на сметките по лихва (sortAccountsByInterest);
  - за намиране и връщане валутата с най-много сметки (findCurrencyWithMostAccounts);
  - за текстово описание.

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.

Създайте приложение за нуждите на териториално бюро за регистрация на автомобили. Регистрацията изисква съхранение на данните на автомобила и на неговия собственик. Всеки автомобил има само един собственик, докато един собственик може да притежава повече от един автомобил.

Предвидете интерфейс Competency с boolean метод hasCompetence(). Методът не приема параметри.

Създайте клас Person, имплементиращ интерфейса Competency, който да описва данните на дадено лице:

- ЕГН (egn) – последователност от 10 символа;
- име (firstName) и фамилия (lastName) – не могат да са празни стойности или да са null;
- възраст (age) – положително число.

Конструктор по всички полета, методи за достъп до полетата – съобразно избраните модификатори за достъп и нуждите. Обектите от класа да се сравняват по стойностите на полето ЕГН. Изключение за въведени невалидни данни - PersonalDataException (създайте класа).

Добавете клас Owner, който да разширява Person. Да допълва данните на собственика с поле с номер на шофьорската книжка (drivingLicense), като при липса на подобна полето да приема стойност “none”. Конструктор и методи за достъп до полетата – съобразно нуждите. Сравнение на обектите – съобразно предвиденото в родителския клас. Текстово представяне на обекта. Класът имплементира метода hasCompetence(), като правоспособност за управление на автомобил е предвидено да има лице на възраст между 18 и 75 години, което има шофьорска книжка.

Създайте клас автомобил (Car), който съдържа описание на неговите:

- регистрационен номер (registrationNumber);
- цвят (color) – избира се от списък с цветове (Color);
- марка на автомобила (brand);
- модел на автомобила (model);
- гориво (fuel) – избира се от списък (Fuel) с възможности petrol, diesel и PHEV

Валидирайте всички полета (CarDataException) за наличие на празни стойности или null. Конструктор по всички полета, методи за достъп – съобразно нуждите и модификаторите за достъп. Сравнението на обектите да се извършва по техния регистрационен номер. Текстово представяне на обектите, съдържащо всички данни на автомобила.

Добавете клас Register, съдържащ информация за населеното място, в което е териториалното бюро (unit) и колекция за регистрации на автомобили, обвързани във връзка <автомобил, собственик> (registeredCars). Добавете конструктор и полета за достъп съобразно нуждите. Класът да включва методи за:

- добавяне на регистрация (addRegistration). Приема като параметри обекти от класовете Car и Owner, не връща резултат.
- извеждане на всички регистрирани автомобили (без данни за собственика им) (printRegisteredCars). Не приема параметри и не връща резултат.
- търсене на собственик по подаден регистрационен номер на автомобила (findOwnerByRegistrationNumber). Приема като параметър номера на автомобила, връща като резултат текст, съдържащ първото и фамилното име на собственика или “Not found”, ако автомобил с подобен номер не присъства в колекцията.
- Извеждане данните на собствениците, които имат правоспособност да управляват автомобил (printLicensedDrivers). Методът не приема параметри и не връща резултат. Изведените резултати не трябва да съдържат повторения.

Задача 4

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

Създайте програма за целите на предстоящи президентски избори. За целта предвидете следните класове:

- Личност Person, съдържащ полета, описващи първото и фамилното име на съответното лице. Добавете конструктор по всички полета. Модификатори за достъп и методи за достъп – съобразно нуждите. Сравнение на обектите по всички полета. Текстово представяне по всички полета.
- Списък с партии (Party) - предварително предвидени партии със абревиатури AAA, BBB, CCC, DDD и NONE.
- Кандидатура (Candidacy). Описва се от:
    - Бюлетина № (candidacyId) – цяло число;
    - Личност – кандидат за президент (candidatePresident);
    - Личност – кандидат за вицепрезидент (candidateVicePresident);
    - Подкрепяща кандидатурата партия (party).

Добавете конструктор по всички полета. Модификатори за достъп и методи за достъп – съобразно нуждите. Сравнение на обектите по номер на бюлетина. Предвидете текстово представяне на обектите.

- Избори (Election), описани от:
    - дата на провеждане на изборите dateofElection (задайте като String);
    - колекция results, съдържаща резултатите на всеки от кандидатиращите се екипи на принципа “кандидатура:брой получени гласове”.

Добавете конструктор и методи за достъп съобразно нуждите. Предвидете методи за:

- Добавяне на резултат от изборите (addResult) – приема като параметри кандидатура от изборите и брой получени гласове, не връща резултат. Добавя резултата към колекцията.
- Извеждане на кандидат-президентските двойки, подредени по номер на бюлетина (printOrderedByCandidacyId) – извежда в конзолата кандидатпрезиденските двойки и резултатите им, подредени във възходящ ред по номера на тяхната бюлетина. Не приема параметри и не връща резултат.
- Извеждане на кандидат-президентските двойки, подредени по получен брой гласове (printOrderedByVotes) – извежда в конзолата кандидатпрезиденските двойки и резултатите им, подредени в низходящ ред по броя на получените гласове. Не приема параметри и не връща резултат.
- Търсене на резултат по кандидатиращо се лице (getVotesByPerson). Методът приема като параметър обект личност и връща като резултат броя на получените гласове или 0, ако не бъде открит сред кандидатиращите се двойки.
- Извеждане на неподкрепените от никоя партия кандидатури (getUnsupportedCandidacies). Не приема параметри и не връща резултат.

- Извеждане на резултатите от изобрите (calculateElectionResults) - не приема параметри и не връща резултат. Ако резултатът на двойката с най-много получени гласове надвишава 50% от общия брой гласове, да се изведе съобщение, че <дадената кандидатура> са изборът на нацията. В противен случай да се изведе, че се преминава към втори тур на изборите.

Задача 5

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

Съставете програма за управление на учебно звено.

За целта ще се нуждаете от:

Клас Person, който описва човек с име (name), задължително името трябва да се състой от минимум 2 думи разделени с един празен символ (space) и всяка дума да започва с главна буква. Когато се опита да се създаде обект от клас Person с невалидна стойност на името да се продуцира грешка (PersonExeption с конструктор по подразбиране който да инициализира грешката със съобщение „Invalid value for Person name“).

Клас Teacher, който разширява класа Person с (academicPositions), поле от тип енумерация (AcademicPositions със стойности: Assistant(ac.), ChiefAssistant(гл. ас.), Lecturer(пр.), SeniorLecturer (ст. пр.), AssociateProfessor (доц.), Professor (проф.)).

Клас Student, който разширява класа Person с (couse), поле от тип енумерация (Course със стойности: FIRST(1 Курс), SECOND (2 Курс), THIRD (3 Курс), FORTH (4 Курс), FIFTH (5 Курс).

Клас Discipline с полета уникално име (name), списък от уникални студенти с точки (0-40) от семестриален контрол, подреден по факултетен номер на студентите (students), лектор (lector), списък уникални преподаватели (teachers), който включва и лектора. Когато се опитате да добавите студент с точки извън интервала от 0-40 да се продуцира грешка (SemestrialControlExeption с конструктор по подразбиране който да инициализира грешката със съобщение „Invalid semestrial control“).

Клас Exam съхранява поле discipline, дисциплина по която се провежда изпит. Метод run, за провеждане на изпит, създава случайна стойност от 0 до 60 за всеки студент (точките от проведения изпит). Добавя генерираните точки към семестриалния контрол и връща списъка със студенти и общия сбор на точки от семестриален контрол и изпит.