Задачи за упражнение

1. Дефинирайте клас **Student**, който съдържа следната информация за студентите: трите имена, курс, специалност, университет, електронна поща и телефонен номер.

2. Декларирайте няколко конструктора за класа **Student**, които имат различни списъци с параметри (за цялостната информация за даден студент или част от нея). Данните, за които няма входна информация да се инициализират съответно с **null** или **0**.

3. Добавете статично поле в класа **Student**, в което се съхранява броя на създадените обекти от този клас.

4. Добавете метод в класа **Student**, който извежда пълна информация за студента.

5. Модифицирайте текущия код на класа **Student** така, че да капсулирате данните в класа чрез свойства.

6. Напишете клас **StudentTest**, който да тества функционалността на класа **Student**.

7. Добавете статичен метод в класа **StudentTest**, който създава няколко обекта от тип **Student** и ги съхранява в статични полета. Създайте статично свойство на класа, което да ги достъпва. Напишете тестова програма, която да извежда информацията за тях в конзолата.

8. Дефинирайте клас, който съдържа информация за мобилен телефон: модел, производител, цена, собственик, характеристики на батерията (модел, idle time и часове разговор /hours talk/) и характеристики на екрана (големина и цветове).

9. Декларирайте няколко конструктора за всеки от създадените класове от предходната задача, които имат различни списъци с параметри (за цялостната информация за даден студент или част от нея). Данните за полетата, които не са известни трябва да се инициализират съответно със стойности с **null** или **0**.

10. Към класа за мобилен телефон от предходните две задачи, добавете статично поле **nokiaN95**, което да съхранява информация за мобилен телефон модел Nokia 95. Добавете метод, в същия клас, който извежда информация за това статично поле.

11. Дефинирайте свойства, за да капсулирате данните в класовете **GSM**, **Battery** и **Display**.

12. Напишете клас **GSMTest**, който тества функционалностите на класа **GSM**. Създайте няколко обекта от дадения клас и ги запазете в масив. Изведете информация за създадените обекти. Изведете информация за статичното поле **nokiaN95**.

13. Създайте клас **Call**, който съдържа информация за разговор, осъществен през мобилен телефон. Той трябва да съдържа информация за датата, времето на започване и продължителността на разговора.

14. Добавете свойство архив с обажданията – **callHistory**, което да пази списък от осъществените разговори.

15. В класа **GSM** добавете методи за добавяне и изтриване на обаждания (**Call**) в архива с обаждания на мобилния телефон. Добавете метод, който изтрива всички обаждания от архива.

16. В класа **GSM** добавете метод, който пресмята общата сума на обажданията (**Call**) от архива с обаждания на телефона (**callHistory**) като нека цената за едно обаждане се подава като параметър на метода.

17. Създайте клас **GSMCallHistoryTest**, с който да се тества функционалността на класа **GSM**, от задача 12, като обект от тип **GSM**. След това, към него добавете няколко обаждания (**Call**). Изведете информация за всяко едно от обажданията. Ако допуснем, че цената за минута разговор е 0.37, пресметнете и отпечатайте общата цена на разговорите. Премахнете най-дългият разговор от архива с обаждания и пресметнете общата цена за всички разговори отново. Най-накрая изтрийте архива с обаждания.

18. Нека е дадена библиотека с книги. Дефинирайте класове съответно за библиотека и книга. Библиотеката трябва да съдържа име и списък от книги. Книгите трябва да съдържат информация за заглавие, автор, издателство, година на издаване и ISBN-номер. В класа, който описва библиотека, добавете методи за добавяне на книга към библиотеката, търсене на книга по предварително зададен автор, извеждане на информация за дадена книга и изтриване на книга от библиотеката.

19. Напишете тестов клас, който създава обект от тип библиотека, добавя няколко книги към него и извежда информация за всяка една от тях. Имплементирайте тестова функционалност, която намира всички книги, чийто автор е Стивън Кинг и ги изтрива. Накрая, отново изведете информация за всяка една от оставащите книги.

20. Дадено ни е училище. В училището имаме класове и ученици. Всеки клас има множество от преподаватели. Всеки преподавател има множество от дисциплини, по които преподава. Учениците имат име и уникален номер в класа. Класовете имат уникален текстов идентификатор. Дисциплините имат име, брой уроци и брой упражнения. Задачата е да се моделира училище с Java класове. Трябва да декларирате класове заедно с техните полета, свойства, методи и конструктори. Дефинирайте и тестов клас, който демонстрира, че останалите класове работят коректно.

Решения и упътвания

1. Използвайте **enum** за специалностите и университетите.

2. За да избегнете повторение на код извиквайте конструкторите един от друг с **this(<parameters>)**.

3. Използвайте конструктора на класа като място, където броя на обектите от класа **Student** се увеличава.

4. Отпечатайте на конзолата всички полета от класа **Student**, следвани от празен ред.

5. Направете **private** всички членове на класа **Student**, след което използвайки Eclipse (Source -> Generate -> Getters and Setters) дефинирайте автоматично публични методи за достъп до тези полета.

6. Създайте няколко студента и изведете цялата информация за всеки един от тях.

7. Можете да ползвате статичния конструктор, за да създадете инстанциите при първия достъп до класа.

8. Декларирайте три отделни класа: **GSM**, **Battery** и **Display**.

9. Дефинирайте описаните конструктори и за да проверите дали класовете работят правилно направете тестова програма.

10. Направете **private** полето и го инициализирайте в момента на декларацията му.

11. В класовете **GSM**, **Battery** и **Display** дефинирайте подходящи **private** полета и генерирайте getters / setters. Можете да ползвате автоматичното генериране в Eclipse.

12. Добавете метод **printInfo()** в класа **GSM**.

13. Прочетете за класа **ArrayList** в Интернет. Класът **GSM** трябва да пази разговорите си в списък от тип**ArrayList<Call>**.

14. Връщайте като резултат списъка с разговорите.

15. Използвайте вградените методи на класа **ArrayList**.

16. Понеже тарифата е фиксирана, лесно можете да изчислите сумарната цена на проведените разговори.

17. Следвайте директно инструкциите от условието на задачата.

18. Дефинирайте класове **Book** и **Library**. За списъка с книги ползвайте **ArrayList<Book>**.

19. Следвайте директно инструкциите от условието на задачата.

20. Създайте класове **School**, **SchoolClass**, **Student**, **Teacher**, **Disciplne** и в тях дефинирайте съответните им полета, както са описани в условието на задачата. Не ползвайте за име на клас думата "**Class**", защото в Java тя има специално значение. Добавете методи за отпечатване на всички полета от всеки от класовете.