# Упражнения: Статични полета и методи

## Дефиниране на клас Person

Създайте клас Person. Класът трябва да има:

* name: String - поле
* age: int - поле
* Name: String - свойство
* Age: int – свойство

Създайте статичен брояч (като поле), който съхранява колко обекта от класа са създадени до момента. Създайте и статично свойство, което да има само get, който да се използва от Main

## Дефиниране на клас Geometry

Създайте статичен клас Geometry. Класът трябва да има следните методи:

* double SquarePerimeter(double side);
* double SquareArea(double side);
* double RectanglePerimeter(double a, double );
* double RectangleArea(double a, double b);
* double CircleArea(double r);

Използвайте всеки един от тях в Main()

## Заявка за корен

Напишете клас, който съдържа метод, който връща корен квадратен при подадена заявка. Възможно е да получите голям брой заявки, така че трябва да отговаряте бързо на всяка една от тези заявки.

Реализирайте Main() метод, който да приема едно число – брой на последващите редове. От всеки следващ ред се задава едно цяло число в интервала [1; 1000].

|  |  |
| --- | --- |
| Вход | Изход |
| 5  25  8  3  100  4 | 5  2.82842712474619  1.73205080756888  10  2 |

## Един магазин

Създайте клас Product с полета за име на продукта и баркод – и двете са текстови низове, цена – double и количество - double. Създайте статичен клас, който да поддържа информация за продуктите в магазина и следните функционалности:

* Продажба на продукт – приема за параметри баркода и продаваното количество. Не допускайте продажба на продукта, ако той има по-малка наличност от желаното количество. Изведете подходящо съобщение на екрана (Sell).
  + Командата ще има вида: Sell <код> <количество>
  + Ако продуктът не съществува или няма достатъчно количество, изведете „Not enough quantity”.
* Добавяне на нов продукт – добавя се информация за продукта; баркод, име, цена и количество (Add)
  + Командата ще има вида: Add <код> <име> <количество>
* Зареждане на продукт – добавя се количество от даден продукт; параметрите са баркода и самото количество; не допускайте зареждане на продукт, ако той изобщо не съществува към момента (Update)
  + Командата ще има вида: Update <код> <количество>
  + Ако такъв продукт не съществува, изведете „Please add your product first!”
* Отпечатване на налични продукти по азбучен ред (PrintA)
* Отпечатване на информация за неналични продукти по азбучен ред (PrintU)
* Отпечатване на всички продукти по намаляща наличност - тези от които има най-много са в началото (PrintD)
* Изчисляване на стойността на всички налични продукти (Calculate)

За всичко това трябва да се създаде и програма, която приема команди и изпълнява съответните действия. Името на всяка команда е записано в скоби по-горе. Командата, която приключва въвеждането е „Close”. Когато се въведе тя програмата приключва. Всички реални числа се извеждат закръглени и с точно 2 знака след запетаята.

### **Примери**

|  |  |  |  |
| --- | --- | --- | --- |
| Вход | Изход | Коментар |  |
| Add 359293 ProductA 3.50 8.0  PrintA  Sell 359293 8.0  PrintA  Update 359293 5.0  PrintA  Add 555 ProductB 5.50 3.0  PrintA  Sell 359293 4.5  PrintD  Calculate  Close | ProductA (359293)  ProductA (359293)  ProductA (359293)  ProductB (555)  ProductB (555)  ProductA (359293)  18.25 | * Първата PrintA идва след като само сме добавили продукта, затова на нея съответства само първия ред от изхода. * После продуктът бива продаден и викаме отново PrintA. Този път няма налични продукти и не трябва да отпечатва нищо. * След това зареждаме ново количество и следващото PrintA го отпечатва отново. * Добавяме нов продукт и викаме PrintD: при него първо излиза ProductB, понеже е с по-голямо количество от ProductA, а условието за командата PrintD е продуктите да се извеждат в намаляващ ред, спрямо количеството. * При извикването на Calculate показваме сумата от всички налични продукти: (0.5 \* 3.50) + (3\*5.50) = 18.25 |  |

## Банкер

Създайте класа BankAccount

Този клас трябва да има полета за:

* id: int
* balance: double

Класът трябва да има свойства за:

* ID: int
* Balance: double

Създайте методите:

* Deposit(Double amount): void – който да вкарва пари в сметката
* Withdraw(Double amount): void – който да изтегля пари от сметката

Заменете метода ToString(), като в този метод изпечатвайте информация за банковата сметка

Създайте статичния клас Bank.

В този клас трябва да създадете следните функционалности:

* Теглене на средства – на този метод трябва да подадете ID-то, списъка с всички създадени сметки и желаната сума за теглене. Ако сумата я няма в наличност или сметката не съществува, изведете подходящо съобщение – за тази функционалност може да се наложи да реализирате няколко метода.
* Внасяне на средства – на този метод трябва да подадете ID-то, списъка с всички създадени сметки и желаната сума за внасяне. Ако сметката не съществува, изведете подходящо съобщение – за тази функционалност може да се наложи да реализирате няколко метода или да използвате вече реализирани такива от предходната точка

Изберете адекватна структура и логика за реализирането на желаните функционалности