Podstawy obiektowości

Zad. Zamówienie

1. Napisać program do obsługi zamówień. Program powinien składać się z dwóch klas: `Zamowienie` oraz `Pozycja`, przy czym każde zamówienie zawierać może jedną lub więcej pozycji.

   Klasa `Pozycja` powinna zawierać następujące pola:

   - nazwaTowaru (String)
   - ileSztuk (int) – liczba zamówionych sztuk
   - cena (double) – cena pojedynczej sztuki

   oraz metody:

   - konstruktor z parametrami umożliwiającymi ustalenie wartości pól klasy,
   - metodę `double obliczWartosc()` zwracającą wartość pozycji zamówienia,
   - metodę `String toString()` zwracającą łańcuch w formacie: nazwa towaru (20 znaków), cena (10 znaków), liczba sztuk (4 znaki), wartość zamówienia (10 znaków), przykład:

     ```
     Cukier                    4,00 zł    3 szt.      12,00 zł
     ```

   Klasa `Zamowienie` powinna zawierać następujące pola:

   - pozycje (tablica obiektów kl. Pozycja) – pozycje składowe zamówienia,
   - ileDodanych (int) – liczba pozycji w zamówieniu,
   - maksRozmiar (int) – maksymalna liczba pozycji w zamówieniu

   oraz metody:

   - konstruktor bezparametrowy – maksRozmiar ustalany na wartość 10,
   - konstruktor z parametrem określającym maksymalną liczbę pozycji w zamówieniu,
   - metodę `void dodajPozycje(Pozycja p)`, która dodaje podaną pozycję do zamówienia,
   - metodę `double obliczWartosc()` zwracającą wartość zamówienia,

- metodę `String toString()`, która zwraca łańcuch zawierający spis pozycji zamówienia oraz łączną wartość zamówienia.

Przykładowa metoda korzystająca z wspomnianych klas:

```
public static void main(String [] args) throws IOException {
    Pozycja p1 = new Pozycja("Chleb", 1, 3.5);
    System.out.println(p1);
    Pozycja p2 = new Pozycja("Cukier", 3, 4);
    System.out.print(p2);

    Zamowienie z = new Zamowienie(20);
    z.dodajPozycje(p1);
    z.dodajPozycje(p2);
    System.out.println(z);
}
```

Przykładowy wynik:

```
Chleb                    3,50 zł    1 szt.       3,50 zł
Cukier                   4,00 zł    3 szt.      12,00 zł

Zamówienie:
Chleb                    3,50 zł    1 szt.       3,50 zł
Cukier                   4,00 zł    3 szt.      12,00 zł

Razem: 15,50 zł
```

2. W klasie `Zamowienie`:
   - zaimplementować metodę `void usunPozycje(int indeks)`, która usuwa z zamówienia pozycję o podanym indeksie
   - zaimplementować metodę `void edytujPozycje(int indeks)`, która umożliwi edycję wybranej pozycji zamówienia, tj. nazwy towaru, ceny oraz liczby sztuk
   - zmodyfikować metodę `void dodajPozycje(Pozycja p)`, tak by w stytuacji, gdy dodawany jest ten sam towar nie dodawała kolejnej pozycji, lecz zwiększała liczbę sztuk w już istniejącej
3. W klasie `Pozycja`:
   - zaimplementować metodę `double obliczWartoscZRabatem`, która oblicza wartość pozycji zamówienia po uwzględnieniu rabatu zależnego od liczby sztuk:
     - 5–10 szt. rabat 5%,

   - 10–20 szt. rabat 10
   - powyżej 20 szt. rabat 15%.

4. Zmodyfikować metodę `obliczWartosc` w klasie Zamowienie, tak by również wyświetlała informacje o rabacie i łączny koszt zamówienia po jego uwzględnieniu.

5. zmodyfikować metodę `toString`, by wyświetlała również naliczony rabat i wartość z rabatem.

6. W obu klasach zaimplementować interfejs `Serializable` umożliwiający zapis i odczyt danych z pliku realizowany przez metody:

   - metodę `public static void zapiszZamowienie(Zamowienie z, String nazwaPliku)`, która zapisze podane w parametrze zamówienie do pliku o nazwie podanej drugim parametrem.
   - metodę `public static Zamowienie wczytajZamowienie(String nazwaPliku)`, która wczyta z pliku o podanej nazwie zamówienie i zwróci je jako wynik.

Zad. Lista

Napisać klasę `Lista`, której zadaniem będzie przechowywanie listy liczb całkowitych. Klasa ta ma mieć następujące pola prywatne:

- `int [] liczby;` – tablica, w której przechowywane będą liczby,
- `int pojemnosc;` – maksymalna liczba elementów, możliwych do przechowywania,
- `int rozmiar;` – aktualna liczba przechowywanych elementów.

Klasa `Lista` powinna mieć również następujące metody:

- konstruktor z parametrem określającym pojemność, który przydziela pamięć dla tablicy liczby oraz ustala wartości pozostałych pól klasy;
- metodę `dodajElement`, która przyjmuje dokładnie jeden element – liczbę całkowitą, która dodawana jest do listy; w przypadku, gdy lista jest pełna powinien zostać wyświetlony komunikat o błędzie;
- metodę `znajdz`, której jedynym parametrem powinna być szukana liczba, natomiast wynikiem pozycja podanej liczby w liście (licząc od 0) lub -1, gdy liczby nie ma na liście;
- bezparametrową metodę `pisz`, która wypisuje informacje o liście, w tym jej rozmiar, pojemność oraz listę przechowywanych elementów;
- metodę `usunPierwszy`, która usuwa pierwsze wystąpienie podanej jako parametr liczby, jeżeli znajduje się ona na liście, tzn. jeżeli podana liczba występuje więcej niż jeden raz, to usuwane jest jedynie pierwsze jej wystąpienie;

- metodę `usunPowtorzenia`, która usuwa wszystkie powtórzenia elementów na liście, tzn. po jej wykonaniu na liście nie powinno być żadnych powtórzonych liczb;
- metodę `odwroc`, która odwraca kolejność elementów przechowywanych na liście;
- metodę `zapiszDoPliku`, która zapisuje zawartość listy do pliku tekstowego, którego nazwa podana powinna być jako pierwszy parametr;

Przykładowo, po wykonaniu poniższego fragmentu:

```
final int N = 10;
Lista l = new Lista(N);
for (int i = 0; i < N/2; ++i) {
    l.dodajElement( (1 << i) );
}
l.dodajElement(2);
l.dodajElement(8);
l.pisz();
l.usunPierwszy(2);
l.pisz();
for (int i = 0; i < N/2; ++i) {
    l.dodajElement( (1 << i) );
}
l.pisz();
System.out.println("Po usunięciu powtórzeń:");
l.usunPowtorzenia();
l.pisz();
```

Na ekranie powinno zostać wyświetlone:

```
Lista:
    Pojemność: 10
    Rozmiar: 7
    Elementy: 1 2 4 8 16 2 8
Lista:
    Pojemność: 10
    Rozmiar: 6
    Elementy: 1 4 8 16 2 8
Nie można dodać więcej elementów, lista pełna!
Lista:
    Pojemność: 10
    Rozmiar: 10
    Elementy: 1 4 8 16 2 8 1 2 4 8
Po usunięciu powtórzeń:
Lista:
```

```
Pojemność: 10
Rozmiar: 5
Elementy: 16 1 2 4 8
```

Zad. Czas

Napisać klasę `Czas` służącą do zapamiętania okresu czasu tj. liczby godzin i minut. Klasa ta powinna mieć dwa pola prywatne:

- `int godz;`
- `int minuty;`

oraz metody publiczne:

- konstruktor z parametrami będącymi liczbą godzin i minut,
- konstruktor przyjmujący jako parametr łańcuch znaków na podstawie którego można ustalić wartość godzin i minut np. "12 h 58 min"
- `String toString()` której wynikiem jest łańcuch znaków opisujący dany okres czasu, np. "29 h 19 min"
- `Czas dodaj(Czas t)` której wynikiem jest nowy obiekt klasy Czas będący sumą bieżącego i podanego jako parametr obiektu
- `Czas odejmij(Czas t)` analogicznie jak dodaj, tyle że odejmowanie,
- `Czas pomnoz(int ile)` wynikiem ma być okres czasu pomnożony podaną liczbę razy,
- `static Czas sumuj(Czas [] tab, int n)` statyczna metoda klasy służąca do sumowania wszystkich okresów czasu podanych w tablicy będącej pierwszym parametrem

Przykładowy program:

```
Czas t1(10, 56);
Czas t2(0, 123);
System.out.println("t1 = " + t1);
System.out.println("t2 = " + t2);
System.out.println("t1 + t2 = " + t1.dodaj(t2));
System.out.println("t1 - t2 = " + t1.odejmij(t2);

Czas [] tab = { t1, t2, t2 };
System.out.println("Czas.sumuj dla t1 + t2 + t2 = " +
                   Czas.sumuj(tab, 3));
```

```
    System.out.println("t1 * 2 = " + t1.pomnoz(2));

    Czas t3("3 h 17 min");
    System.out.println("Konstruktor z łańcuchem: " + t3);
```

Wydruk dla przykładowego programu:

```
t1 = 10 h 56 min
t2 = 2 h 3 min
t1 + t2 = 12 h 59 min
t1 - t2 = 8 h 53 min
Czas.sumuj dla t1 + t2 + t2 = 15 h 2 min
t1 * 2 = 21 h 52 min
Konstruktor z łańcuchem: 3 h 17 min
```