Лабораторна робота №5
Робота з файлами. Застосування колекцій.

Завдання.

Розробити програму, що матиме зручний інтерфейс користувача і дозволятиме виконати дії відповідно до варіанту.

Примітка. *У завданнях усіх варіантів передбачити можливість програмного створення файлу даних, перегляду всіх даних у файлі, додавання елементу даних у файл та виконання запиту вказаного в умові задачі. Для тимчасового збереження інформації у оперативній пам’яті використовувати колекції.*

Дано файл, що містить відомості про іграшки, указується назва іграшки (лялька, кубики, м'яч, конструктор), її вартість у копійках і вікові межі дітей, для яких іграшка призначена (наприклад, для дітей від двох до п'яти років). Крім того, для ляльки зазначено її розмір у сантіметрах, для кубиків — їхня кількість у наборі, для м'яча — його вага у грамах, для конструктора — кількість конструкцій, які з нього можна збудувати згідно інструкції. Одержати наступні відомості:

Варіант 1.

Перелік іграшок, ціна яких не перевищує вказану і які підходять дітям 5 років у порядку зростання ціни

Варіант 2.

Перелік конструкторів у порядку зростання ціни. (Ціну виводити за зразком …грн…коп).

Варіант 3.

Перелік найбільш коштовних іграшок (ціна яких відрізняється від ціни найкоштовнішої іграшки не більш ніж на 10 грн.) у порядку спадання ціни.

Варіант 4.

Перелік іграшок, що підходять дітям від 4 років до 10 років. Виводити в порядку алфавіту

Варіант 5.

Перелік усіх кубиків, у порядку зростання цін. Ціну виводити за зразком …грн…коп.

Варіант 6.

Вивести перелік іграшок, будь яких, крім м'яча, що підходять дитині 3 років, щоб вартість іграшки не перевищувала задану суму. Перелік виводити у порядку спадання ціни.

Варіант 7.

Вивести перелік м'ячів із ціною, що вказана, або менша за неї, призначених дітям від 3 до 8 років. Перелік виводити у порядку зростання ваги м'ячів

Варіант 8.

Вивести перелік ляльок, що підходять дитині 3 років, щоб розмір іграшки не перевищував заданий. Перелік виводити у порядку збільшення розміру ляльки

Варіант 9.

Вивести перелік кубиків, що підходять дитині 2 років, щоб їх вартість перевищувала задану суму, але була не більше 200 грн. Перелік виводити у порядку зростання цін.

Варіант 10.

Перелік найбільш дешевих іграшок (ціна яких відрізняється від ціни найдешевшої іграшки не більш ніж на 10% її вартості) у порядку зростання ціни.

Короткі теоретичні відомості.

В Java широко використовуються колекції (Collections) – "розумні" масиви з довжиною, що визначається динамічно, які підтримують ряд важливих додаткових операцій у порівнянні з масивами. Базовим для ієрархії колекцій є клас java.util.AbstractCollection. (У загальному випадку клас колекції не повинен бути наслідником AbstractCollection – він може бути будь-яким класом, що реалізує інтерфейс Collection).

Основні класи колекцій:

- Set, SortedSet, HashSet,TreeSet – множини (набори елементів, що не повторюються)
- List, ArrayList,LinkedList,Vector – списки (впорядковані набори елементів, які можуть повторюватися в різних місцях списку)
- Map, Sorted Map – таблиці (списки пар "ім'я" - "значення")

Доступ до елементів колекції в загальному випадку не може здійснюватися за індексом, через те що не всі колекції підтримують індексацію елементів. Цю функцію здійснюють за допомогою спеціального об'єкта – ітератора (Iterator). У кожної колекції collection є свій ітератор, який вміє з нею працювати, тому ітератор вводять таким чином:

Iterator iter = collection.iterator()

У ітераторів є такі три методи:

`boolean hasNext()`- надає інформацію, чи є колекції наступний об'єкт.

`Object next()` – повертає посилання на наступний об'єкт колекції.

`void remove()` – вилучає з колекції поточний об'єкт, тобто той, посилання на який було отримано останнім викликом **`next()`**.

Приклад перетворення масиву в колекцію та цикл з доступом до елементів цієї колекції, що здійснюється за допомогою ітератора:

```
java.util.List components= java.util.Arrays.asList(this.getComponents());
for (Iterator iter = components.iterator();iter.hasNext();) {
      Object elem = (Object) iter.next();
      javax.swing.JOptionPane.showMessageDialog(null,"Компонент: "+
                  elem.toString());
}
```

Основні методи колекцій:

Ім'я методу	Дія
`boolean add(Object obj)`	Додавання об'єкта в колекцію (в кінець списку). Повертає **true** у випадку успішного додавання – змінення колекції. Колекція може не дозволити додавання елементів несумісного типу чи таких що не підходять за будь-якою іншою ознакою.
`boolean addAll(Collection c)`	Додавання в колекцію всіх об'єктів з іншої колекції. Повертає **true** у випадку успішного додавання, тобто якщо додано хоча б один елемент.
`void clear()`	Очистка колекції – видалення з неї посилань на всі елементи, що входять в колекцію. При цьому ті об'єкти, на які є посилання у інших елементів програми, не видаляються з пам'яті.
`boolean contains(Object obj)`	Повертає **true** у випадку, якщо колекція містить об'єкт obj. Перевірка здійснюється за допомогою послідовного виклику метода obj.equals(e) для елементів e, що входять в колекцію.
`boolean containsAll(Collection c)`	Повертає **true** у випадку, якщо колекція містить всі елементи колекції c.
`boolean isEmpty()`	Повертає **true** у випадку, якщо колекція є порожньою, тобто не містить жодного елементу.

Ім’я методу	Дія
`Iterator iterator()`	Повертає посилання на ітератор – об’єкт, що дозволяє отримувати послідовний доступ до елементів колекції. Для однієї колекції дозволено мати довільну кількість об’єктів-ітераторів, в тому числі – різних типів. В процесі роботи вони можуть вказувати на різні елементи колекції. Після створення ітератор завжди вказує на початок колекції – виклик його метода **next()** дає посилання на початковий елемент колекції.
`boolean remove(Object obj)`	Вилучає з колекції перше, що зустрілося, входження об’єкта obj. Пошук та видалення здійснюється за допомогою ітератора. Повертає **true** у випадку, якщо видалення вдалося, тобто якщо колекція змінилась.
`boolean remove All(Collection c)`	Вилучає з колекції всі елементи колекції c. Повертає true у випадку, якщо вилучення відбулося, тобто якщо колекція змінилась.
`Boolean retainAll(Collection c)`	Залишає в колекції тільки ті з елементів, що входять до неї, які входять в колекцію c.
`int size()`	Повертає число елементів в колекції.
`Object[] toArray()`	Повертає масив посилань на об’єкти, що містяться в колекції. Тобто перетворює колекцію в масив.
`T[] toArray(T[n] a)`	Повертає масив елементів типу T, які будуть отримані в результаті перетворення елементів, що містяться в колекції. Тобто перетворює колекцію в масив. Якщо кількість елементів колекції не перевищує розмір n масиву a, розміщення даних виконується в існуючих комірках пам’яті, відведених для масиву. Якщо перевищує n – в пам’яті динамічно створюється та заповнюється новий набір комірок, та їхня кількість буде рівною числу елементів колекції. Після чого змінна a починає посилатися на новий набір комірок.
`String toString()`	Метод перевизначен – він повертає рядок зі списком елементів колекції. В списку виводяться рядкові представлення елементів, які взяті у квадратні дужки, які розділяються комбінацією “, ” – кома з пропуском після неї.

Найпоширенішими варіантами колекції є списки (Lists). Вони багато у чому схожі на масиви, але відрізняються від масивів тим, що в списках основними операціями є додавання та вилучення елементів, а не доступ до елементів за індексом, як в масивах.

В класі List є методи колекції, а також ряд додаткових методів:
list.get(i) – отримання посилання на елемент списку list за індексом i.
list.indexOf(obj) – отримання індексу елемента obj у списку list. Повертає -1 якщо об’єкт не знайдено.
list.listIterator(i) – отримання посилання на ітератор типу ListIterator, що має додаткові методи у порівнянні з ітераторами типу Iterator.
list.listIterator(i) – теж саме з позиціонуванням ітератора на елемент з індексом i.
list.remove(i) – вилучення зі списку елемента з індексом i.
list.set(i,obj) – заміна в списку елемента з індексом i на об’єкт obj.
list.subList(i1,i2) – повертає посилання на підсписок, що складається з елементів списку з індексами від i1 до i2.
Крім них в класі List є ще багато інших корисних методів.

Ряд корисних методів для роботи з колекціями міститься в класі Collections:
Collections.addAll(c,e1,e2,…,eN) - додавання в колекцію c довільної кількості елементів e1,e2,…,eN.
Collections.frequency(c,obj) – повертає кількість входжень елемента obj в колекцію c.
Collections.reverse(list) – перевертає порядок слідування елементів в списку list (перші стають останніми і навпаки).

Collections.sort(list) – сортує список в порядку зростання елементів. Порівняння іде викликом методу e1.compareTo(e2) для чергових елементів списку e1 та e2.
Крім них в класі Collections є ще багато інших корисних методів.

В класі Arrays є метод
Arrays.asList(a) – повертає посилання на список елементів типу T, що є ”обгорткою” над масивом T[] a. При цьому і масив, і список містять ті ж самі елементи, і зміна елемента списку призводить до зміни елемента масиву, и навпаки.

Робота з файлами

- Об’єкти типу File забезпечують роботу з іменами файлів та папок (перевірка існування файлу чи папки з вказаним іменем, знаходження абсолютного шляху за відносним і навпаки, перевірка та встановлення атрибутів файлів та папок).
- При роботі з файлами в більшості програм потрібен виклик файлового діалогу JFileChooser (пакет javax.swing).
- Потоки введення-виведення забезпечують роботу не тільки з файлами, але і з пам’яттю, а також різноманітними пристроями введення-виведення. Відповідні класи розміщені в пакеті java.io. Абстрактний клас InputStream (“вхідний потік”) інкапсулює модель вхідних потоків, що дозволяють зчитувати з них дані. Абстрактний клас OutputStream (“вихідний потік”) - модель вихідних потоків, що дозволяють записувати до них дані.
- Для читання рядків (у вигляді масиву символів) використовуються нащадки абстрактного класу Reader (“читач”). Наприклад, для читання з файлу – клас FileReader. Аналогічно, для запису рядків використовуються класи, які наслідуються від абстрактного класу Writer (“записувач”). Наприклад, для запису масиву символів в файл – клас FileWriter.
- Є ще один важливий клас для роботи з файлами - RandomAccessFile (“файл з довільним доступом”), який призначений для читання і запису даних у довільному місці файлу. Такий файл з точки зору класу RandomAccessFile представляє масив байт, які збережені на зовнішньому носії. Клас RandomAccessFile не є абстрактним, тому можна створювати його екземпляри.

Приклад використання класів для введення даних:
```
try {
    BufferedReader in = new BufferedReader(new FileReader("file.txt"));
    String line = null;
    while ((line = in.readLine())!=null){
        System.out.println(line);
    }
    in.close();
} catch (IOException ex) {
    ex.printStackTrace();
}
```
В цьому прикладі вміст файлу file.txt роздруковується на екрані.

Якщо рядок, що прочитаний з файлу містить декілька елементів, то для розділення їх можна скористатися класами розбору рядків Scanner або StringTokenizer, наприклад:
```
Scanner s = new Scanner(line);
int n = s.nextInt();
String str = s.next();
double x = s.nextDouble();
```
або
```
StringTokenizer st = new StringTokenizer(line);
int n = Integer.parseInt(st.nextToken());
String str = st.nextToken();
double x = Double.parseDouble(st.nextToken());
```