Лабораторна робота № 10

Серіалізація об’єктів, опрацювання структурованих файлів.

Мета роботи: знайомство з потоками даних та серіалізацією об’єктів.

Завдання . Вдосконалити об’єктну модель попередньої лабораторної роботи, додавши до неї наступну функціональність методи для запису об’єктів з масиву з даними у файл і навпаки.

Варіанти завдань
Варіанти завдань відповідають лабораторній роботі № 9.

Приклад.

Варіант 16. Предметна область: бібліотека, клас: книга, орієнтовний перелік полів: назва, автор, рік видання, кількість сторінок, наявна у фонді/видана читачеві дата видачі.

В класі Library створюємо метод для запису масиву книг у файл. Для цього у класі Book слід реалізувати інтерфейс `Serializable`:

```
public class Book implements Serializable{
. . .
}
```

Метод запису в класі Library можна описати так:

```
public void saveToFile(String fileName){
      File file = new File(fileName);

      try {
                  FileOutputStream fileOut =  new FileOutputStream(file);
                  ObjectOutputStream outStream = new ObjectOutputStream(fileOut);
                  for(Book book: books){
                        outStream.writeObject(book);
                  }
                  outStream.close();
            } catch (FileNotFoundException e) {
                  // методи запису у файл вимагають обробки винятків
                  System.out.println("Файл для запису даних не створено - " +
                              e.getMessage() );

            } catch (IOException e) {

                  System.out.println("Помилка запису даних - " + e.getMessage() );
            }
}
```

При цьому слід додати відповідні імпорт-вирази для використаних класів у початок коду:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
```

При читанні даних з файлу можуть виникати різні помилки, тому операції з відкриття файлів та читання даних слід поміщати в блоки try{…}cath(){…}. Для цього рекомендується скористатися підказками IDE Eclipse.

Перевірку роботи методу можна виконати на масиві книжок, створеному у попередній лабораторній роботі за рахунок ініціалізації. При вдалому виконанні методу отримаємо готовий файл з даними, який можна буде використати для тестування наступного методу.

Метод зчитування даних може виглядати так:

```
public void loadFromFile(String fileName){
      File file = new File(fileName);
      int i = 0;
      Book book=null;
      ObjectInputStream inStream = null;
      try {
                  FileInputStream fileIn =  new FileInputStream(file);
                  inStream = new ObjectInputStream(fileIn);
                  if(inStream != null){
                  while(true){
                        book = (Book) inStream.readObject();
                        if(book==null)break;
                        books[i++] = book;
                        //читання даних з файлу в масив
                  }
                  inStream.close();
            }

      } catch (FileNotFoundException e) {
            System.out.println("Не знайдено файл з даними - " + e.getMessage() );
            }catch (ClassNotFoundException e) {
            System.out.println("Помилка в структурі даних - " + e.getMessage() );
            } catch (IOException e) {
            System.out.println("Помилка читання даних - " + e.getMessage() );
            }
}
```

Очевидно, що реалізація цього методу вимагатиме імпорту відповідних класів із стандартних пакетів, та опрацювання виняткових ситуацій, що можуть траплятися при читанні з файлу.

Для перевірки читання даних з файлу слід створити порожній масив книг books, достатнього для записаних даних розміру:

```
Book [] books = new Book[15];
Library myLibrary = new Library(books);
myLibrary.loadFromFile("C:\\Books.dat");
System.out.println("Список книг у бібліотеці:");
myLibrary.printList();
```

Очевидно, що частина масиву при цьому залишиться незаповненою, оскільки масив має фіксований розмір, програма наперед не може точно визначити скільки даних міститься у файлі. В таких випадках зручніше користуватися динамічними масивами – колекціями (див завдання б) до наступної лабораторної роботи).

Лабораторна робота № 11

Використання параметризованих колекцій для роботи з об'єктами.

Мета роботи: удосконалення навичок ООП та знайомство з колекціями Java.

Завдання . Вдосконалити об'єктну модель попередньої лабораторної роботи, додавши до неї наступну функціональність:

a) Реалізувати в класі з даними метод compareTo для порівняння двох примірників класу (), за допомогою якого виконати сортування масиву об'єктів в головному класі. Правила порівняння двох об'єктів розробити самостійно, відповідно до предметної області.

b) Замінити масив об'єктів, що використовується для тимчасового зберігання даних на динамічну структуру – список ArrayList. Протестувати роботу програми прочитавши дані з файла створеного в попередній лабораторній роботі, додавши 2-3 нових об'єкти і записавши файл заново.

Варіанти завдань

Варіанти завдань відповідають лабораторній роботі № 9.

Приклад.

Варіант 16. Предметна область: бібліотека, клас: книга, орієнтовний перелік полів: назва, автор, рік видання, кількість сторінок, наявна у фонді/видана читачеві дата видачі.

а) Реалізуємо порівняння двох книжок за правилом:

більшою вважається книжка, що вийшла з друку пізніше – метод compareTo повертає ціле число, тому різниця років видання книжок може бути його результатом;

якщо ж книжки видані в одному році, то порівняємо їх за алфавітним порядком прізвищ авторів – тут скористаємося власним методом compareTo класу String, що здійснює порівняння фрагментів тексту за алфавітним порядком;

нарешті, якщо книжки видані одним і тим же автором чи групою авторів протягом року, то порівнюватимемо книжки за назвою – таке потрійне порівняння дасть результат 0 лише у випадку, якщо йдеться про одну і ту ж книжку.

Орієнтовний код методу:

```
@Override
    public int compareTo(Book another){
      /*метод порівняння для впорядкування книжок
      *більшими вважаємо книжки, видані пізніше
      *якщо книжки видані в одному році - порівнюємо
      *за алфавітом прізвища авторів, а далі - назви книжок
      */
            int result = year - another.getYear();
      if(result==0) result = author.compareTo(another.author);
      if(result==0) result = title.compareTo(another.title);
      return result;

      }
```

Оскільки метод compareTo описаний в батьківському класі усіх об'єктів Object, то перед його описом додаємо анотацію @Override, яка вказує, що ми перевизначаємо батьківський метод, а в заголовку класу додаємо реалізацію інтерфейсу Comparable:

```
public class Book implements Serializable, Comparable<Book>{
```

Для реалізації сортування масиву книг можна скористатися методом сортування масиву з лабораторної роботи № 7, замінивши назву масиву та знак “<” на метод compareTo.

b) Замість масиву

```
private Book[] books;
```

Будемо використовувати список

```
private ArrayList<Book> books;
```

Відповідну структуру даних слід підключити імпорт-виразом

```
import java.util.ArrayList;
```

Наш клас має, конструктор з параметром типу `Book[]`, його слід замінити на `ArrayList<Book>`

Цикли з ітератором виду `for (Book book : books) {. . . }` можна залишити без зміни, а от звертання до елементів масиву за індексом `books[i]`) слід замінити відповідним методом get(i) класу `ArrayList`. Наприклад, метод сортування списку книг виглядатиме так:

```
public void sort(){

      for(int k=0; k<books.size() - 1; k++){

            Book min=books.get(k);
            int iMin = k;
            for(int i=k; i<books.size(); i++){
                  if(min.compareTo(books.get(i))>0){
                        min=books.get(i);
                        iMin = i;
                  }
            }
            books.remove(iMin);
            books.add(k,min);
      }
   }
```

Початкову ініціалізацію масиву, аналогічну до лабораторної 9 слід замінити на додавання елементів до списку:

```
ArrayList<Book> books = new ArrayList<Book>();
      books.add( new Book("Ткаченко  О.М.",
            "Комп'ютерне  програмування  на  мові  Java.", 2013, 147));
      books.add(new Book("Копитко М.Ф., Іванків К.С. ",
          "Основи програмування мовою Java: Тексти лекцій.", 2002, 83, "25.12.14"));
      books.add(new Book("Р. А. Мельник, М. М. Сенів",
                "Програмування на JAVA: метод. " +
       "Вказівки до лаборатор. робіт з дисципліни Об'єкт.-орієнт. програмув.",
                                    2007,42));
```

і т. д.