Завдання №9

- Розробити програму, яка веде облік читачів та книг у бібліотеці. Для цього слід створити набір класів, що описують бібліотеку (*Автор, Книга, Бібліотека, Читач, Прокат, ...*).
- Клас ***Книга*** може характеризуватись назвою (поле типу String), списком авторів (список-масив з елементів типу ***Автор***), роком видання та номером видання (цілочисленні поля);
- Клас ***Читач***, який представляє читача (крім імені та прізвища читач характеризується його «реєстраційним номером» та списком отриманих книг (список-масив об'єктів ***Книга***));
- Клас **Бібліотека**, який представляє бібліотеку; характеризується назвою бібліотеки, списком книг, що зберігаються, і списком зареєстрованих читачів (списки-масиви, що зберігають об'єкти відповідних типів);
- крім того, можна створити клас ***LibraryDriver*** зі статичними методами, що виконують серіалізацію та десеріалізацію бібліотеки та методом **main**(),

У кожному класі має бути конструктор за замовчуванням, гетери та сеттери, перевизначений метод *toString*(), а також методи, що реалізують базову функціональність.
За допомогою механізму серіалізації збережіть у файлі стан системи та відновіть систему з цього ж файлу.

Реалізуйте три версії програми, що реалізують різні стратегії серіалізації:

1. всі класи, що входять до системи, є серіалізованим (реалізують інтерфейс java.io.Serializable);
2. серіалізуються (реалізують інтерфейс java.io.Serializable) не всі класи системи: класи *Автор, Читач, Книга* є **НЕ серіалізуються**;
3. класи, які входять у систему, реалізують інтерфейс java.io.Externalizable.

Перша версія програми має бути виконана у вигляді консольної програми, у функції main якої показана робота системи: створена «бібліотека» з книгами та списком читачів; видано кілька книг; виведено інформацію про поточний стан бібліотеки; проведена серіалізація та десеріалізація; показаний стан десеріалізованої системи.

Перша версія програми:

Усі класи повинні реалізовувати інтерфейс java.io.Serializable. При цьому можна вказати поле serialVersionUID за допомогою якого керують сумісністю різних версій класу, що серіалізується. Значення цього поля можна вказати «за умовчанням» - надавши йому певне значення,

```
private static final long serialVersionUID = 1L;
```

Друга версія програми:

Для організації серіалізації в другому випадку слід по-перше, вказати, що класи, що серіалізуються, реалізують інтерфейс ***java.io.Serializable***, а їх поля, що не підтримують цей інтерфейс, не повинні серіалізуватися автоматично: вони повинні бути позначені як **transient**. При цьому, для того, щоб стан, що визначається такими полями, все-таки був збережений і згодом відновлений, класи, що серіалізуються, повинні включити закриті методи:

```
private void writeObject(ObjectOutputStream out) throws IOException { ..}
private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException  {...}
```

для «ручного керування» серіалізацією. У них потрібно вказати, що слід автоматично серіалізувати те, що можна, а те, що не можна серіалізувати (десеріалізувати) вручну — за допомогою методів об'єктних потоків.

Третя версія програми:

Для організації серіалізації в третьому випадку слід зазначити, що кожен клас реалізує інтерфейс java.io.Externalizable. При цьому у кожному такому класі мають бути перевизначені методи:

```
@Override
public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException { }

@Override
public void writeExternal(ObjectOutput out) throws IOException { }
```

за допомогою яких потрібно організувати серіалізацію/десеріалізацію об'єктів даного класу.