­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­

Контрольна робота №1

з курсу «Об’єктно-орієнтоване програмування»

для студентів базового напрямку 6.08.04 "Комп’ютерні науки"

(заочна форма навчання)

Варіант 14

Виконав студент гр. КНз-2

Чалий Михайло

­­

Львів 2014

# Серіалізація об’єктів у мові Java

**Серіалізація** — процес перетворення будь-якої [структури даних](http://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) в послідовність [бітів](http://uk.wikipedia.org/wiki/%D0%91%D1%96%D1%82). Зворотною до операції серіалізації є операція десеріалізації - відновлення початкового стану структури даних з бітової послідовності.

Серіалізація використовується для передачі [об'єктів](http://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82) по мережі і для збереження їх у [файли](http://uk.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB). Наприклад, потрібно створити розподілений [додаток](http://uk.wikipedia.org/wiki/%D0%94%D0%BE%D0%B4%D0%B0%D1%82%D0%BE%D0%BA), різні частини якого мають обмінюватися даними зі складною структурою. У такому випадку для типів даних, які передбачається передавати, пишеться код, який здійснює серіалізацію і десеріалізацію. Об'єкт заповнюється потрібними даними, потім викликається код серіалізації, в результаті виходить, наприклад, [XML](http://uk.wikipedia.org/wiki/XML)-документ. Результат серіалізації передається приймаючій стороні, наприклад, по [електронній пошті](http://uk.wikipedia.org/wiki/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0_%D0%BF%D0%BE%D1%88%D1%82%D0%B0) або [HTTP](http://uk.wikipedia.org/wiki/HTTP). Додаток-одержувач створює об'єкт того ж типу і викликає код десеріалізації, в результаті отримуючи об'єкт з тими ж даними, що були в об'єкті програми-відправника.

## Застосування

Серіалізація надає декілька корисних можливостей:

* метод реалізації зберігання об'єктів, який зручніший, ніж запис їх властивостей в текстовий файл на диск і повторна збірка об'єктів читанням файлів;
* метод здійснення віддалених викликів процедур, як, наприклад, у [SOAP](http://uk.wikipedia.org/wiki/SOAP);
* метод розповсюдження об'єктів, особливо в технологіях [компонентно-орієнтованого програмування](http://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), таких як [COM](http://uk.wikipedia.org/wiki/COM) і [CORBA](http://uk.wikipedia.org/wiki/CORBA);
* метод виявлення змін у даних, що змінюються з часом.

Для найефективнішого використання даних можливостей необхідно підтримувати незалежність від архітектури. Наприклад, необхідно мати можливість надійно відтворювати серіалізований [потік даних](http://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D1%96%D0%BA_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85), незалежно від [порядку байтів](http://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA_%D0%B1%D0%B0%D0%B9%D1%82%D1%96%D0%B2), що використовується в цій архітектурі. Це означає, що найбільш проста і швидка процедура прямого копіювання ділянки пам'яті, в якому розміщується структура даних, не може працювати надійно для всіх архітектур. Серіалізація структур даних в архітектурно-незалежний формат означає, що не повинно виникати проблем через різний порядок проходження байтів, механізмів розподілу пам'яті або відмінностей представлення структур даних в мовах програмування.

Будь-якій зі схем серіалізації властиво те, що кодування даних послідовно за визначенням, і вибірка будь-якої частини серіалізованої структури даних вимагає, щоб весь об'єкт був зчитаний від початку до кінця і був відновлений. У багатьох програмах така лінійність корисна, тому що дозволяє використовувати прості інтерфейси введення/виведення загального призначення для збереження і передачі стану об'єкта. У додатках, де важлива висока продуктивність, можливо буде доречніше використовувати складнішу, нелінійну організацію зберігання даних.

## Серіалізація в Java

Для запису/читання об’єктів використовуються потокові класи ObjectOutputStream або ObjectInputStream

Метод класу ObjectInputStream:  
  
           void writeObject(Object o); // записує (заносить) об’єкт в потік  
  
Метод класу ObjectInputStream:  
  
           Object readObject(); // вичитує (вибирає) об’єкт з потоку i повертає посилання на нього

Важливо пам’ятати що в потік можуть бути занесені тільки серіалізовані об’єкти. З іншого боку, об’єкт є серіалізованим, якщо його клас реалізує інтерфейс Serializable.

Відзначимо, що майже всі класи стандартних пакетів Javа реалізують цей інтерфейс. Зокрема, масиви (які є об’єктами спеціальних класів визначених на етапі компіляції) також є серіалізованими. Як приклад наведемо програму, в якій “закріпляються” об’єкти переважно стандартних класів Javа і масив. Напочатку, програма записує до потоку об’єкти – дати, масиву, який містить значення температур і масиву стрінгів, які є описами для значень температури. Далі об’єкти вибираються з потоку і можуть довільним чином бути зміненими.

Лістінг програми з прикладом серіалізації наведено нижче.

import java.io.\*;

import java.util.\*;

class Example

{

public static void main(String args[])

{

Date data = new Date();

int[] temperatura = { 25, 19 , 22};

String[] opis = { "день", "ніч", "вода" };

// Запис

try

{

ObjectOutputStream out = new ObjectOutputStream(

new FileOutputStream("c:\\test.dat")

);

out.writeObject(data);

out.writeObject(opis);

out.writeObject(temperatura);

out.close();

}

catch(IOException exc)

{

exc.printStackTrace();

System.exit(1);

}

// Відкривання (зазвичай в іншій програмі)

try

{

ObjectInputStream in = new ObjectInputStream(

new FileInputStream("c:\\test.ser")

);

Date odczytData = (Date) in.readObject();

String[] odczytOpis = (String[]) in.readObject();

int[] odczytTemp = (int[]) in.readObject();

in.close();

System.out.println(String.valueOf(odczytData));

for (int i=0; i<odczytOpis.length; i++)

System.out.println(odczytOpis[i] + " " +

odczytTemp[i]);

}

catch(IOException exc)

{

exc.printStackTrace();

System.exit(1);

}

catch(ClassNotFoundException exc)

{

System.out.println("Не можна віднайти клас об’єкту");

System.exit(1);

}

}

}

Серіалізація має декілька важливих особливостей:

* при серіалізації не зберігаються статичні поля та поля декларовані зі специфікатором transient; власне специфікатор transient використовується для елементів, які не треба піддавати “закріпленню”,
* повний контроль над способом серіалізації можемо отримати визначивши відповідні методи в класі серіалізованого об’єкту, методи ці повинні мати наступну сигнатуру:  
      private void readObject(java.io.ObjectInputStream stream)  
          throws IOException, ClassNotFoundException;  
    
      private void writeObject(java.io.ObjectOutputStream stream)  
          throws IOException
* цілковитий контроль над форматом і способом серіалізації отримується через реалізацією в класі інтерфейсу Externalizable i визначення методів writeExternal i readExternal

# Механізми розпаралелення мови Java.