Serialization trong Java

Java cung cấp một kỹ thuật, được gọi là serialization đối tượng (object serialization), tại đây một đối tượng có thể được biểu diễn như là một dãy byte liên tục mà bao gồm dữ liệu của đối tượng cũng như thông tin về kiểu đối tượng và kiểu dữ liệu được lưu giữ trong đối tượng.

Sau khi một đối tượng được serialize đã được ghi vào trong một file, nó có thể được đọc từ file này và được deserialize từ đó, đó là thông tin kiểu và các byte mà biểu diễn đối tượng và dữ liệu của nó có thể được sử dụng để tái tạo đối tượng này trong bộ nhớ.

Ấn tượng nhất là toàn bộ tiến trình là JVM độc lập, nghĩa là một đối tượng có thể được xếp thứ tự (serialize) trên một platform và deserialize trên một platform hoàn toàn khác.

Các lớp **ObjectInputStream** và **ObjectOutputStream** trong Java là các luồng bậc cao mà chứa các phương thức để serialize và deserialize một đối tượng.

Lớp ObjectOutputStream chứa nhiều phương thức write để ghi các kiểu dữ liệu đa dạng, nhưng với một phương thức trong đầu ra chuẩn cụ thể:

```
public final void writeObject(Object x) throws IOException
```

Phương thức trên xếp thứ tự một Object theo thứ tự và gửi nó tới đầu ra chuẩn. Tương tự, lớp ObjectInputStream chứa phương thức sau để deserialize một đối tượng:

```
public final Object readObject() throws IOException,
                                ClassNotFoundException
```

Phương thức này thu nhận Object tiếp theo ra khỏi luồng và deserialize nó. Giá trị trả về là Object, vì thế bạn sẽ cần ép nó thành kiểu dữ liệu thích hợp.

Để minh họa cách serialization làm việc trong Java, tôi sử dụng lớp Employee mà đã bàn luận trong các chương trước. Giả sử rằng chúng ta có lớp Employee sau, mà triển khai Serializable Interface:

```
public class Employee implements java.io.Serializable
{
   public String name;
   public String address;
   public transient int SSN;
```

```
   public int number;
   public void mailCheck()
   {
      System.out.println("Mailing a check to " + name
                         + " " + address);
   }
}
```

Ghi chú rằng, với một lớp để được xếp thứ tự theo thứ tự một cách thành công, phải có hai điều kiện sau:

- Lớp phải triển khai java.io.Serializable interface
- Tất cả các trường trong lớp phải là có thể xếp thứ tự (Serializable). Nếu một trường là không thể xếp thứ tự, nó phải được đánh dấu.

Nếu bạn tò mò để biết: lớp Java chuẩn nào là có thể xếp thứ tự hoặc không, bạn kiểm tra văn kiện cho lớp đó. Việc kiểm tra khá đơn giản: Nếu lớp triển khai java.io.Serializable interface thì nó là Serializable, nếu không thì nó không thể xếp thứ tự.

Xếp thứ tự một Object trong Java

Lớp ObjectOutputStream được sử dụng để xếp thứ tự một Object. Chương trình SerializeDemo sau thuyết minh một đối tượng Employee và xếp thứ tự nó vào trong một file.

Khi chương trình thực thi, một file với tên employee.ser được tạo. Chương trình không tạo bất kỳ đầu ra nào, nhưng bạn xem xét code và cố gắng xác định xem chương trình đang làm cái gì.

Chú ý: Khi xếp thứ tự một đối tượng vào một file, qui ước chuẩn trong Java là cung cấp một file có đuôi là **.ser** .

```
import java.io.*;


public class SerializeDemo
{
   public static void main(String [] args)
   {
      Employee e = new Employee();
```

```
      e.name = "Reyan Ali";
      e.address = "Phokka Kuan, Ambehta Peer";
      e.SSN = 11122333;
      e.number = 101;
      try
      {
         FileOutputStream fileOut =
         new FileOutputStream("/tmp/employee.ser");
         ObjectOutputStream out = new ObjectOutputStream(fileOut);
         out.writeObject(e);
         out.close();
         fileOut.close();
         System.out.printf("Serialized data is saved in /tmp/employee.ser");
      }catch(IOException i)
      {
          i.printStackTrace();
      }
   }
}
```

Deserialize một Object trong Java

Chương trình DeserializeDemo sau sẽ deserialize đối tượng Employee được tạo trong chương trình SerializeDemo. Bạn xem xét chương trình này và xác định đầu ra của nó:

```
import java.io.*;
public class DeserializeDemo
{
   public static void main(String [] args)
   {
      Employee e = null;
      try
      {
         FileInputStream fileIn = new FileInputStream("/tmp/employee.ser");
         ObjectInputStream in = new ObjectInputStream(fileIn);
```

```
         e = (Employee) in.readObject();
         in.close();
         fileIn.close();
      }catch(IOException i)
      {
         i.printStackTrace();
         return;
      }catch(ClassNotFoundException c)
      {
         System.out.println("Employee class not found");
         c.printStackTrace();
         return;
      }
      System.out.println("Deserialized Employee...");
      System.out.println("Name: " + e.name);
      System.out.println("Address: " + e.address);
      System.out.println("SSN: " + e.SSN);
      System.out.println("Number: " + e.number);
    }
}
```

Nó sẽ cho kết quả sau:

```
Deserialized Employee...
Name: Reyan Ali
Address:Phokka Kuan, Ambehta Peer
SSN: 0
Number:101
```

Dưới đây là một số điểm quan trọng cần ghi nhớ:

- Khối try/catch cố gắng để bắt một ClassNotFoundException, được khai báo bởi phương thức readObject(). Với một JVM để có thể deserialize một đối tượng, nó phải có thể tìm thấy bytecode cho lớp đó. Nếu JVM không thể tìm một lớp trong khi deserialize một đối tượng, nó ném một ClassNotFoundException.

- Chú ý rằng giá trị trả về của readObject được ném tới một tham chiếu Employee.
- Giá trị của trường SSN là 11122333 khi đối tượng được xếp thứ tự, nhưng bởi vì trường này là tạm thời (transient), giá trị này không được gửi tới luồng đầu ra. Trường SSN của đối tượng Employee được deserialize là 0.