Manejo de archivos serializados

Emmanuel Cruz Hernández
emmanuel_cruzh@ciencias.unam.mx

29 de Septiembre de 2020

Contenido

Introducción

La programación en Java se basa en gran medida en la creación de tipos de datos.

- **Tipos de datos**: Un tipo de datos es un conjunto de valores y un conjunto de operaciones sobre esos valores.
- **Tipos de datos abstractos**: Un tipo de datos abstracto es un tipo de datos cuya representación interna está oculta al cliente.

Introducción

- **Objetos**: Un objeto es una entidad que puede tomar un valor de tipo de datos.
- **Cliente**: Un cliente es un programa que usa un tipo de datos.
- **Implementación**: Una implementación es el código que implementa el tipo de datos especificado en una API.

Procesamiento de información

Los tipos de datos abstractos proporcionan un mecanismo natural para organizar y procesar la información. [1]

Ejemplos

Tipos de datos para representar el procesamiento de información:

- Date: representa una fecha compuesta por día, mes y año.
- Transaction: representa un cliente, una fecha y una cantidad.

Serialización de objetos

La **serialización** de un objeto consiste en generar una secuencia de bytes lista para su almacenamiento o transmisión.

La acción opuesta se conoce como **deserialización**, la cual consiste en regresar al objeto a su estado original.

Características de Serializable

Para que un objeto sea serializable se debe implementar la interfaz Serializable que se encuentra en *java.io.Serializable*.

Además, todas las variables de la clase deben ser serializables. Por defecto los tipos primitivos ya son serializables, así como los arreglos y otros tipos estándar.

Serialización

Para realizar este proceso se necesitan de las clases *FileOutputStream*, *ObjectOutputStream* e *IOException*. [2]

```
public void serializa(Object o) throws IOException{
    FileOutputStream file = new FileOutputStream("Nombre");
    ObjectOutputStream output = new ObjectOutputStream(file);
    output.writeObject(o);
    output.close();
}
```

Deserialización

Para realizar este proceso se necesitan de las clases *FileInputStream*, *ObjectInputStream*, *IOException* y *ClassNotFoundException*. [2]

```
public void deserializa() throws IOException, ClassNotFoundException{
    Clase c;
    FileInputStream file = new FileInputStream("Nombre");
    ObjectInputStream input = new ObjectInputStream(file);
    c = (Clase) output.readObject(o);
    output.close();
}
```

Bibliografía


Robert Sedgewick Kevin Wayne.
Algorithms.
Cuarta edition, 2011.


Fernando Berzal.
El sistema de E/S: Ficheros.