Serializzazione in Java

Serializzazione degli oggetti

Al termine della esecuzione di un programma, i dati utilizzati vengono distrutti.

Per poterli preservare fra due esecuzioni consecutive è possibile ricorrere all'uso dell'I/O su file.

Nel caso di semplici strutture o di valori di un tipo primitivo, questo approccio è facilmente implementabile.

I problemi si presentano quando si desidera memorizzare strutture complesse (e.g., collezioni di oggetti): in questo caso occorrerebbe memorizzare tutte le parti di un oggetto separatamente, secondo una ben precisa rappresentazione, per poi ricostruire l'informazione dell'oggetto all'occorrenza. Questo processo può risultare impegnativo e noioso.

Serializzazione degli oggetti

La ***persistenza*** di un oggetto indica la capacità di un oggetto di poter "vivere" separatamente dal programma che lo ha generato.

Java contiene un meccanismo per creare oggetti persistenti, detto ***serializzazione degli oggetti***: un oggetto viene serializzato trasformandolo in una sequenza di byte che lo rappresentano. In seguito questa rappresentazione può essere usata per ricostruire l'oggetto originale. Una volta serializzato, l'oggetto può essere memorizzato in un file o inviato a un altro computer perché lo utilizzi.

Serializzazione degli oggetti

In Java la serializzazione viene realizzata tramite

- una interfaccia e
- due classi.

Ogni oggetto che si vuole serializzare deve implementare l'interfaccia ***Serializable***, la quale non contiene metodi e serve soltanto al compilatore per comprendere che un oggetto di quella determinata classe può essere serializzato.

Per serializzare un oggetto si invoca poi il metodo ***writeObject*** della classe ***ObjectOutputStream***; per deserializzarlo si usa il metodo ***readObject*** della classe ***ObjectInputStream***.

Serializzazione degli oggetti

ObjectInputStream e *ObjectOutputStream* sono stream di manipolazione e devono essere utilizzati congiuntamente a un *OutputStream* e un *InputStream*. Quindi gli stream di dati effettivamente usati dall'oggetto serializzato possono rappresentare file, comunicazioni su rete, stringhe, ecc.

Esempio:

```
FileOutputStream outFile = new
FileOutputStream("info.dat");
ObjectOutputStream outStream = new
ObjectOutputStream(outFile);
outStream.writeObject(myCar)
```

dove `myCar` è un oggetto di una classe `Car` definita dal programmatore e che implementa l'interfaccia `Serializable`.

Serializzazione degli oggetti

Per poter leggere l'oggetto serializzato e ricaricarlo in memoria centrale si procederà come segue:

```
FileInputStream inFile = new
  FileInputStream("info.dat");
ObjectInputStream inStream = new
  ObjectInputStream(inFile);
Car myCar = (Car) inStream.readObject();
```

La serializzazione di un oggetto si occupa di serializzare tutti gli eventuali riferimenti ad esso collegati. Dunque, se la classe `Car` contenesse dei riferimenti (variabili di classe o di istanza) a oggetti di classe `Engine`, questa verrebbe serializzata automaticamente e diverrebbe parte della serializzazione di `Car`. La classe `Engine` dovrà, pertanto, implementare anch'essa l'interfaccia serializable.

Serializzazione degli oggetti

Attenzione:

Gli attributi di classe, cioè definiti come **static**, NON vengono serializzati. Per poterli salvare occorre provvedere in modo personalizzato.

Esempio:

Attributo statico nroNavi in Nave.

```
class Nave implements Serializable{
        private static int nroNavi=1;
        private int nroNave;
        private String nomeNave;
        Nave(String nomeNave){
            nroNave=nroNavi++;
            this.nomeNave=nomeNave;
        }
        public String toString(){
            return nomeNave+":"+nroNave;
        }
        public void salva() throws FileNotFoundException, IOException {
                ObjectOutputStream out = new ObjectOutputStream(new
                                        FileOutputStream("info.dat"));
                out.writeObject(this);
                out.writeObject(nroNavi);
                out.close();
        }
        public static Nave carica() throws FileNotFoundException, IOException {
                ObjectInputStream in = new ObjectInputStream(new
                                        FileInputStream("info.dat"));
                Nave n=(Nave)in.readObject();
                Nave.nroNavi=(Integer)in.readObject();
                in.close();
                return n;
        }
}
```

Serializzazione degli oggetti

Molte classi della libreria standard Java implementano l'interfaccia Serializable in modo da essere serializzate quando necessario.

Esempi di tali classi sono: String, HashMap, …

Ovviamente, nel caso di HashMap anche gli oggetti memorizzati nella struttura dati devono implementare l'interfaccia Serializable.

Il modificatore *transient*

A volte, quando si serializza un oggetto, si può desiderare di escludere delle informazioni, ad esempio, una password.

Questo accade quando le informazioni vengono trasmesse via rete.

Il pericolo è che, pur dichiarandola con visibilità privata, la password possa essere letta e usata da soggetti non autorizzati quando viene serializzata.

Un'altra ragione potrebbe essere quella di voler escludere l'informazione dalla serializzazione semplicemente perché tale informazione può essere semplicemente riprodotta quando l'oggetto viene deserializzato. In questo modo lo stream di byte che contiene l'oggetto serializzato non ha informazioni inutili che ne aumenterebbero la dimensione.

Il modificatore *transient*

Per modificare la dichiarazione di una variabile può essere usata la parola chiave ***transient***: questa indica al compilatore di non rappresentarla come parte dello stream di byte della versione serializzata dell'oggetto.

Ad esempio, si supponga che un oggetto contenga la seguente dichiarazione:

```
private transient String password
```

Questa variabile, quando l'oggetto che la contiene viene serializzato, non viene inclusa nella rappresentazione.

Riferimenti bibliografici

La parte sulla serializzazione degli oggetti è presa da:

J. Lewis, W. Loftus.

Java: Fondamenti di progettazione software (prima edizione italiana).

Addison-Wesley, 2001