Serialització i persistència

Autor: Sergi Tur Badenas

Serialització i persistència

- **Serialització**
  - Conversió d'un objecte en un seqüència de bytes
  - Permet enviar dades i rebre objectes a través de fluxes d'E/S
- **Persistència**
  - És el procés de guardar la informació en memòria volàtil dels objectes a memòria persistent
  - El procés es realitza serialitzant l'objecte cap a la memòria persistent (fitxer, base de dades, etc...)

Autor: Sergi Tur Badenas

Reflection

- **Avantatges**
  - **Facilita la extensibilitat**
    - Aplicacions externes poden fer ús de classes amb reflection
  - **Navegadors de classes i IDE**
    - Exposar les característiques de les classes permet navegar per elles i implementar les ajudes de codi dels IDE
  - **Depuradors i eines de Test**
    - Els depuradors poden accedir als membres privats de les classes. Les eines de test poden fer un ús intensiu de reflection per implementar els test.

Autor: Sergi Tur Badenas

Reflection

- **Inconvenients**
  - **Costos de rendiment**
    - S'han de resoldre tipus de dades de forma dinàmica. Al no saber a priori el tipus no es poden aplicar certes optimitzacions
    - Cal evitar l'ús abusiu de reflection sobretot en aplicacions sensibles al rendiment.
  - **Restriccions de seguretat**
    - Requereix de certs permisos en temps d'execució que poden no estar disponibles sota un Security Manager (p.ex. Applets)
  - **Exposició de mètodes privats**
    - Pot tenir repercussions i efectes secundaris inesperats. Pot destruir la portabilitat de l'aplicació

Autor: Sergi Tur Badenas

Reflection

- **java.lang.Class**
  - Les implementacions de Java ME i Java SE són diferents. Java ME suporta reflection de forma limitada
    - Class a Java ME
    - Class a Java SE
  - No existeix el paquet **java.lang.reflect** a Java ME
  - L'operador **instanceof** forma part de reflection
  - **JUnit** no es pot aplicar a Java ME per falta de suport de reflection
    http://java.sun.com/docs/books/tutorial/reflect/class/inde

Autor: Sergi Tur Badenas

Serialització manual

- **CLDC no suporta serialització d'objectes**
  - Suport limitat per a *reflection*
  - Necessitem serialitzar objectes per enviar-los a través de la xarxa
- **Serialització manual**
  - Definirem mètodes **serialize()** i **deserialize()** en els objectes que vulguem fer persistents
  - Utilitzarem els fluxos **DataOutputStream** i **DataInputStream** per a codificar i decodificar les dades

Autor: Sergi Tur Badenas

Serialització i persistència

- **Java SE**
  - Al paquet ***java.io*** existeixen
    - Interfície Serializable
    - Els objectes ObjectOutputStream/ObjectInputStream i ObjectOutput i ObjectInput.
    - Hi han mètodes per escriure el objecte en un stream persistent: **ObjectOutputStream.writeObject()** **ObjectInputstream.readObject()**

En Java ME no disposem d'aquests objectes i per tant la persistència s'ha de fer manualment.

Autor: Sergi Tur Badenas

Serialització manual

- **Creem la interfície Persistent**

```
/**
 * Un interfície simple per a la creació d'objectes persistents
 */
public interface Persistent {
    /**
     * Implementació de la persistència de l'objecte
     */
    byte[] serialize() throws IOException;
    /**
     * Recuperació d'un objecte persistent
     */
    void deserialize( byte[] data ) throws IOException;
}
```


Autor: Sergi Tur Badenas

Serialització manual

- **Serialitzar i deserialitzar un objecte**

```
Professor profe = new Professor();
    try {
        byte[] persistedProfe = profe.serialize();
    }
    catch( java.io.IOException e ){
        // Control d'errors
    }
```

```
    byte[] persistedProfe = ....; // Codi per obtenir l'objecte
    Professor profe = new Professor();

    try {
        profe.deserialize( persistedProfe );
    }
    catch( java.io.IOException e ){
        // Control d'errors
    }
```


Autor: Sergi Tur Badenas

Persistència d'objectes complexos

- **Exemple Vector. Objecte dinàmic**
  - Cal guardar totes les dades necessàries, en aquest cas el nombre d'elements del vector

```
public static byte[] serialize(Vector v) throws IOException {
        ByteArrayOutputStream bout = new ByteArrayOutputStream();
        DataOutputStream dout = new DataOutputStream(bout);
        int n = v.size(); dout.writeInt(n);
        for (int i = 0; i < n; ++i) {
            Object o = v.elementAt(i);
            if (o instanceof String) {
                dout.writeUTF((String) o);
            }else {
                throw new IOException(
                        "No es pot persistir l'objecte de tipus” + o.getClass().getName());
            }
        }
        dout.flush();
        return bout.toByteArray();
}
```


Autor: Sergi Tur Badenas

Persistència d'objectes complexos

- **Objectes amb referències a altres objectes**
  - La cosa és complica...
  - Cal implementar tota la cadena de persistència (implementar la persistència de l'objecte i dels objectes als quals fa referencia)
  - Cal vigilar les referències creuades i fer un exhaustiu control d'errors

Es recomana que els objectes a persistir siguin autocontinguts.

Autor: Sergi Tur Badenas

Persistència d'objectes complexos

- **Problemes de rendiment**
  - Com hem vist, els mètodes de reflection i l'operador **instanceof** són lents en Java ME
  - Millor guardar el tipus de classe al serialitzar
    - Nom complet de la classe més paquet! En comptes de:
      Professor --> edu.upc.ice.gestioAules.Professor

```
String cname = din.readUTF();
Object profe;
    if( cname.equals( "edu.upc.ice.gestioAules.Professor" ) ){
        profe = new Professor();
    } else {
        throw new UnknownClassException();
    }
    ((Persistent) profe).deserialize( tmp );
```


Autor: Sergi Tur Badenas

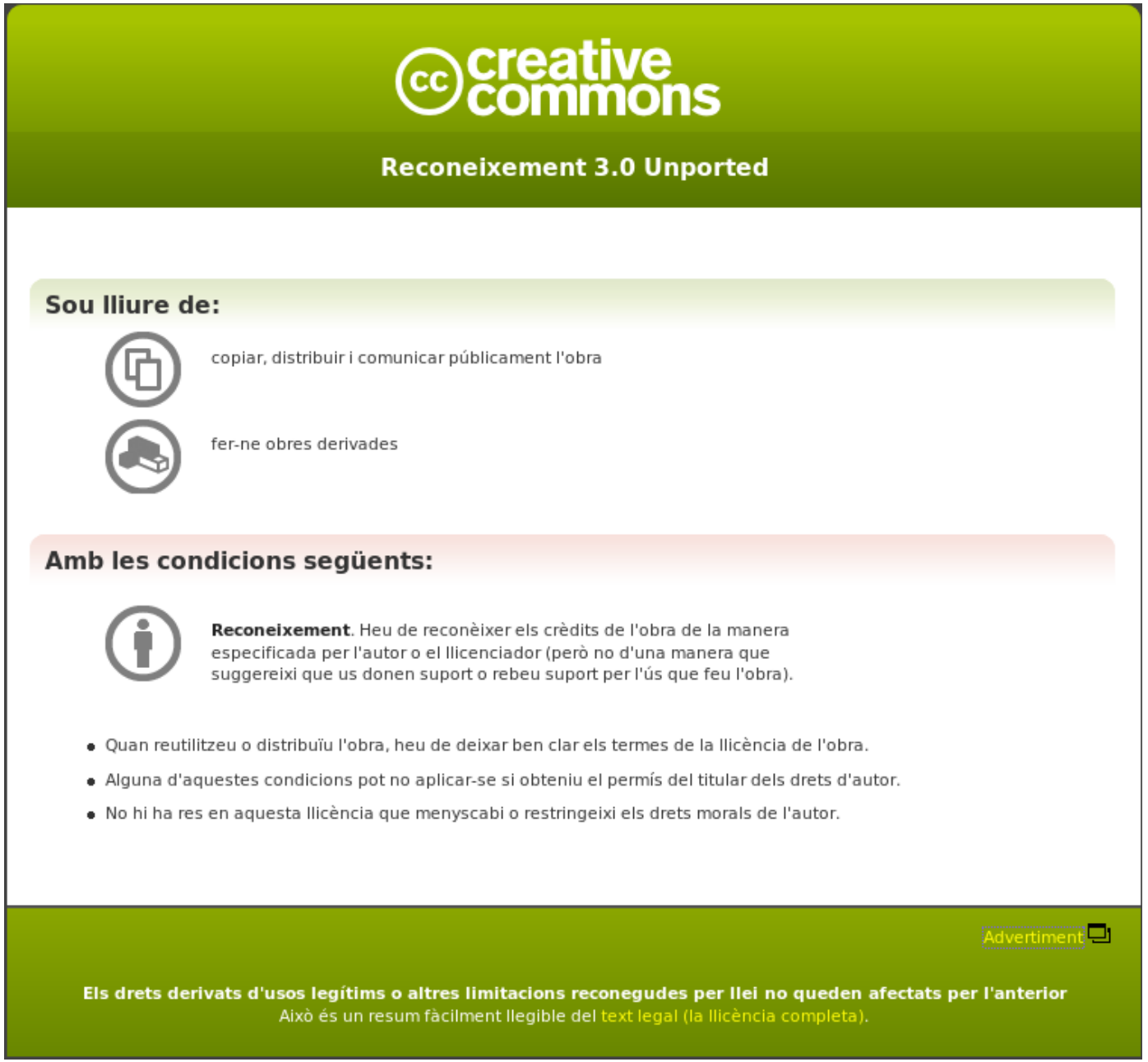
Persistència d'objectes complexos

- **Persistència d'objectes en cadena**

```
public static byte[] serialize( Vector v ) throws IOException {
   ...
   if( o instanceof Persistent ){
            dout.writeUTF(o.getClass().getName() );
            byte[] data = ((Persistent) o).serialize();
            dout.writeInt( data.length );
            if( data.length > 0 ){
               dout.write( data );
            }
         }
   ...
}
```


Autor: Sergi Tur Badenas

Llicència Creative Commons



http://creativecommons.org/licenses/by/3.0/deed.ca

Autor: Sergi Tur Badenas