Persistencia

Persistencia

- Persistencia significa trascender en el tiempo y/o en el espacio

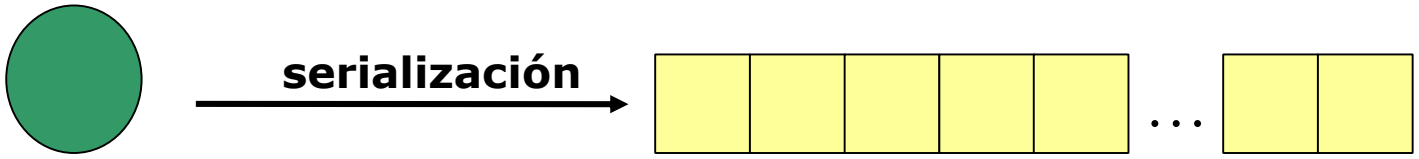
- Un ambiente orientado a objetos debe permitir que los objetos se persistan, para mantener su existencia más allá de la vida de la aplicación.

Persistencia

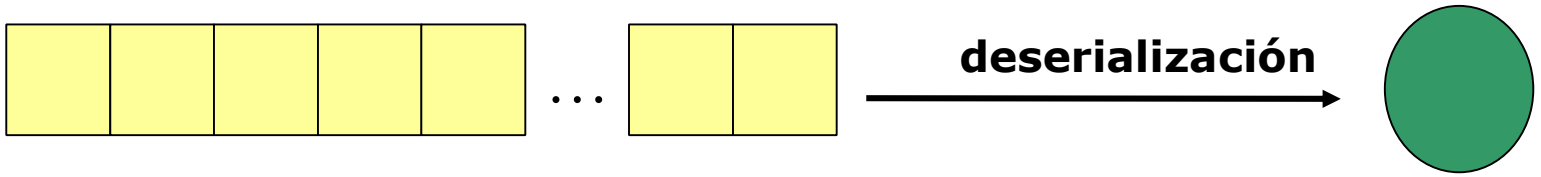
- Tipos de Persistencia
  - Nativa
    - Provista por la plataforma
      - EJ: Java, Smalltalk, .NET
  - No Nativa
    - A través de una biblioteca externa
    - Programada a mano

Serialización

Proceso que consiste en convertir la representación de un objeto en un *stream* (*flujo o secuencia*) de bytes.



Reconstituir un objeto a partir de un *stream* de bytes se denomina **deserialización**.



Persistencia y Serialización

- Para persistir debo primero serializar
- Serializar no implica necesariamente persistir
- Otras acciones luego de serializar:
  - Enviar por red
  - Mantener en memoria
  - Enviar a una impresora, etc

Serialización en Java

- Para definir una clase serializada, debe implementarse la interfaz ***Serializable***.
  - No tiene métodos
  - Sirve para avisarle a la máquina virtual que la clase puede serializarse (*marker interface*).
  - Todas las subclases de una clase serializable son serializables también
  - Por defecto, se serializan todos los atributos de un objeto que no posean el modificador `transient` (*transitorio*).
  - Si un objeto serializable tiene referencias a otro objeto serializable también lo serializa
  - Si algún objeto del “arbol de objetos” a serializar no es serializable se lanza la excepción NonSerializableException
  - Los objetos serializables deben tener un constructor sin parámetros para poder ser deserializados correctamente

Serialización en Java

```
Import java.io.Serializable;

class Vector implements Serializable {

 private static long serialVersionUID = 1L;

  private double x;
  private double y;
  private double z;

  private static final double NORMA_INVALIDA = -1.0;

  transient private double norma = NORMA_INVALIDA;
  ...
```

Marca Serializable

Versión de la clase

Atributo de clase. No se serializa

Atributo transitorio No se serializa

Persistencia en Java

- Paquete java.io
  - De objeto a archivo

objeto->stream (ObjectOutputStream)
stream->archivo (FileOutputStream)

```
UnaClase objeto = new UnaClase();
OutputStream fos = new FileOutputStream("objeto.dat");
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(objeto);
```

  - De archivo a objeto

archivo->stream (FileInputStream)
stream ->objeto (ObjectInputStream)

```
InputStream fis = new FileInputStream("objeto.dat");
ObjectInputStream ois = new ObjectInputStream(fis);
UnaClase objeto = (UnaClase)ois.readObject();
```

Persistencia en Java

- **Persistencia Nativa**
  - **Ventajas**
  - Es nativa del lenguaje (casi no hay que programar)
  - Resuelve referencias circulares

  - **Desventajas**
  - No es portable a otros lenguajes de manera sencilla
  - No es óptima en cuanto a tamaño (tiene *overhead* alto)
  - La información en el archivo es binaria
    - No es extensible ni reparable

Serialización Nativa en SmallTalk

- Objeto a Archivo

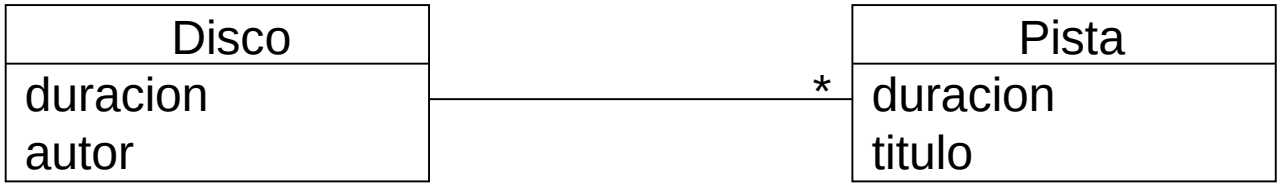
```
objeto := MiClase new.
rr := ReferenceStream fileNamed: 'objeto.bin'.
rr nextPut: objeto; close.
```

- Archivo a Objeto

```
rr := ReferenceStream fileNamed: 'objeto.bin'.
objeto := rr next.
rr close.
```

Ej: Serialización no nativa

- Persistencia en Texto (XML)



instancias

```
Disco disco = new Disco(50, “Queen”);
disco.addPista(new Pista(4,”the show must go on”);
disco.addPista(new Pista(3, “inuendo”);
```

Stream de texto

```
<disco duracion=“50” autor=“Queen”>
        <pista duracion=“4” titulo=“the show must go on” />
        <pista duracion=“3” titulo=“inuendo” />
</disco>
```

Persistencia no nativa

- Responsabilidad: cada clase conoce como serializarse
- Cada clase serializable tendrá
  - Un método serializar que devolverá un nodo XML
  - Un constructor sobrecargado que recibirá un nodo XML

Persistencia no nativa

```
public class Disco{
    private int duracion;
    private string autor;
    private List pistas = new ArrayList();

    public Disco(XMLNode nodo){
          duracion=Integer.ParseInt(nodo.getAttribute("duracion"));
          autor=nodo.getAttribute("autor");
          List nodosPistas = nodo.getChildrens("pista");
          for(XMLNode nodoPista : nodosPistas){
                  pistas.add(new Pista(nodoPista);
          }
    }

    public XMLNode serializar(){
          XMLNode nodo = CrearNodo();
          nodo.setAttribute("autor",this.autor);
          nodo.setAttribute("duracion",this.duracion);
          for(Pista pista : this.pistas){
                  nodo.addChildren(pista.serializar());
          }
    }

}
```

Deserialización

Serialización

La clase pista es similar

Persistencia no nativa

- Escritura a archivo

```
XmlDocument doc = CreateDocFromNode(disco.serializar());
XmlHelper.SaveXMLDocToFile(doc,"disco.xml");
```

- Lectura desde archivo

```
XmlDocument doc = XmlHelper.ReadFromFile(“disco.xml”);
Disco disco = new Disco(doc.getRootNode);
```

Persistencia No Nativa

- Queda en manos del programador
  - Cuestiones de diseño
    - Responsabilidad
      - Cada clase sabe como persistirse
      - Existe un gestor externo que sabe como persistir las clases
    - Formato
      - Binario, texto plano, texto jerárquico (XML)
    - Identidad de los objetos
      - Referencias circulares, duplicación de objetos, etc
  - Mayor versatilidad pero…
  - Mayor complejidad