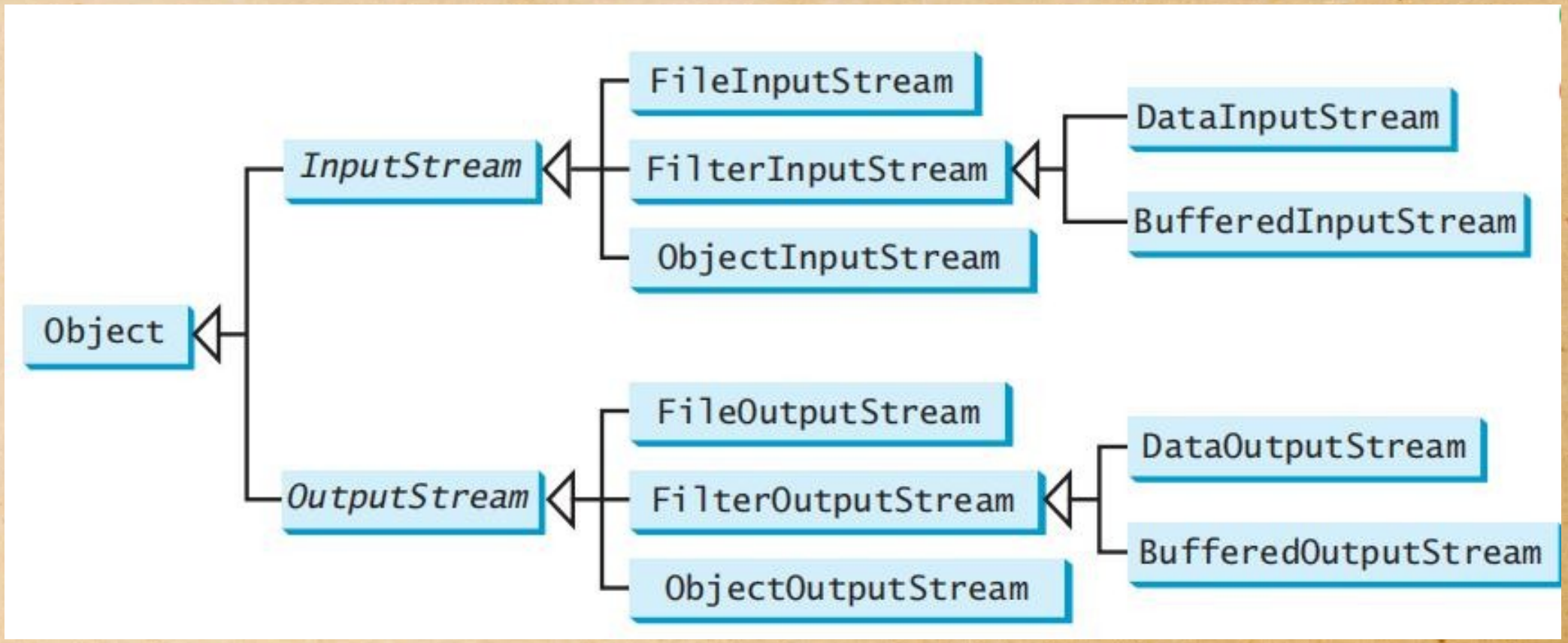
Programació

UT12.3. Entrada/Eixida de fitxers binaris

Introducció

- Ja hem vist com treballar amb fitxers de text (els que es poden llegir, crear i modificar, per exemple, amb un editor de text).
- La resta d'arxius, són fitxers binaris
- Este tipus de fitxers són llegits per programes (per exemple, els arxius .class, .exe, .dat ...)

Estructura de classes en Java



Classes per a l'eixida de bytes

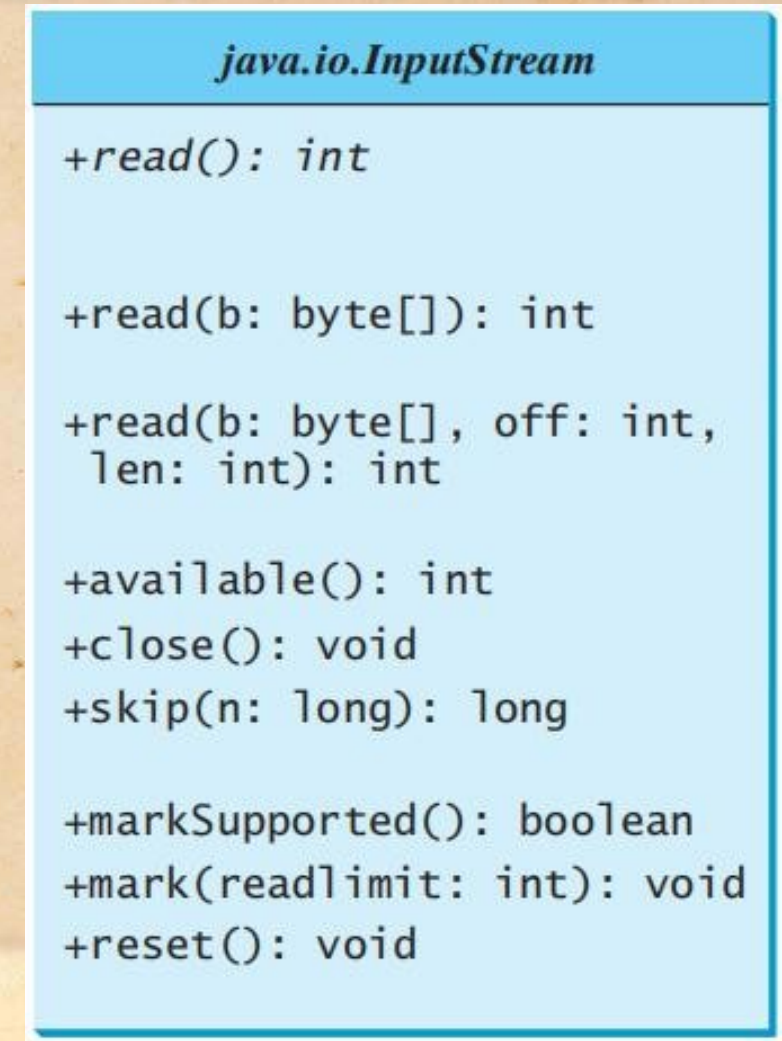
- **OutputStream**: classe abstracta, pare de la majoria dels fluxos de bytes.
- **FileOutputStream**: flux que permet escriure en un fitxer, byte a byte.
- **BufferedOutputStream**: flux que permet escriure grups (buffers) de bytes.
- **ByteArrayOutputStream**: flux que permet escriure en memòria, obtenint allò escrit en un array de bytes.

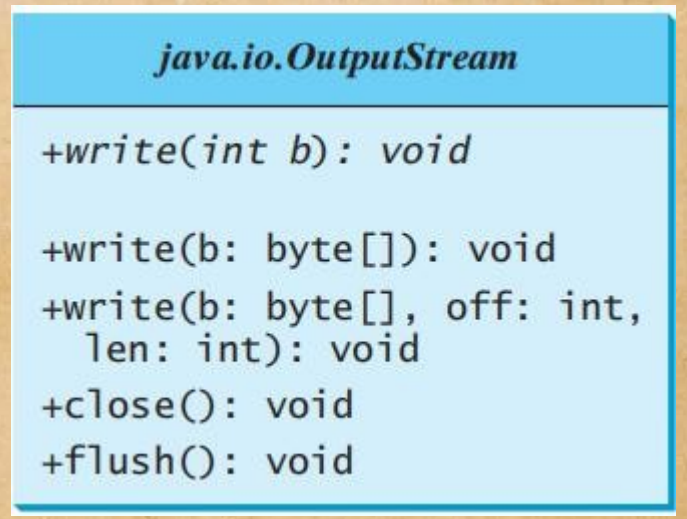
Classes per a l'entrada de bytes

- **InputStream**: classe abstracta, pare de la majoria dels fluxos de bytes.
- **FileInputStream**: flux que permet llegir d'un fitxer, byte a byte.
- **BufferedInputStream**: flux que permet llegir grups (buffers) de bytes.
- **ByteArrayInputStream**: flux que permet llegir de memòria (d'un array de bytes).

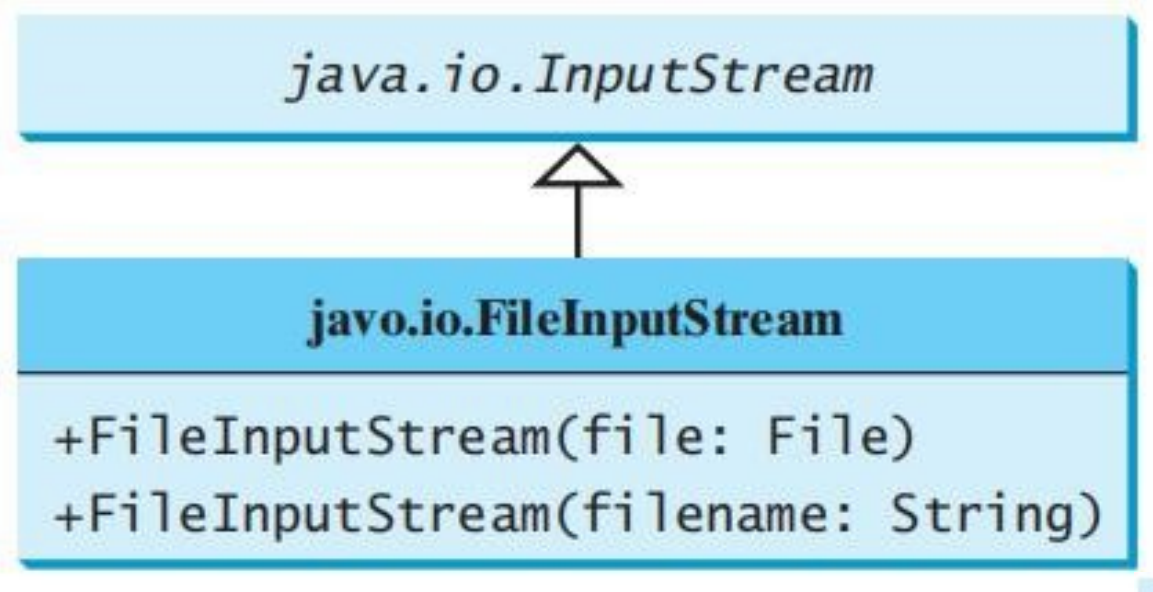
InputStream i OutputStream

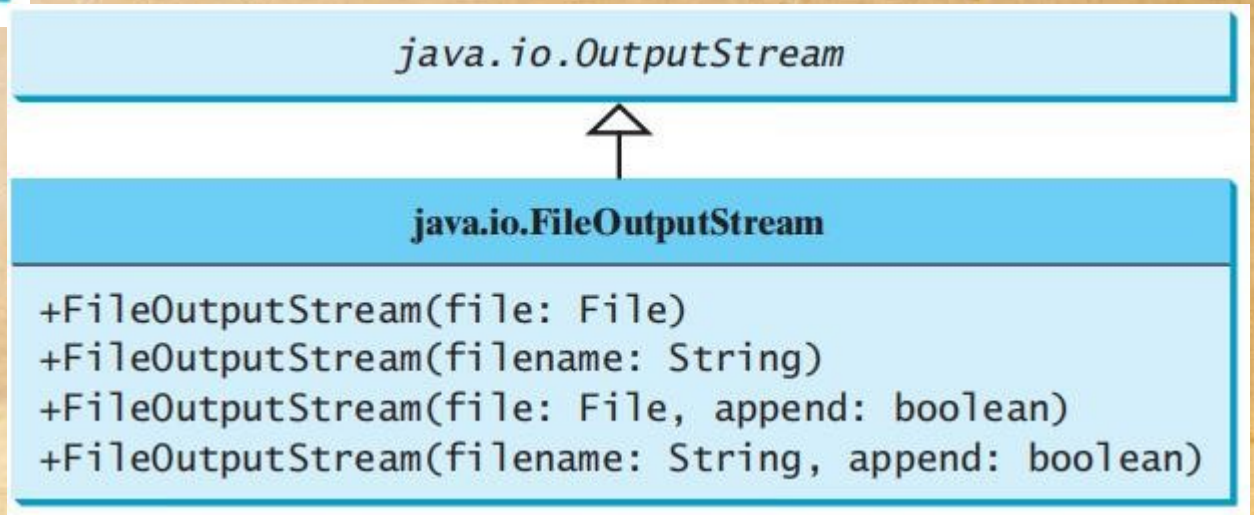
Llegeix i escriu per bytes d'informació





FileInputStream i FileOutputStream





Exemple

```java
import java.io.*;

public class TestFileStream {
  public static void main(String[] args) throws IOException {
    try (
      // Create an output stream to the file
      FileOutputStream output = new FileOutputStream("temp.dat");
    ) {
      // Output values to the file
      for (int i = 1; i <= 10; i++)
        output.write(i);
    }

    try (
      // Create an input stream for the file
      FileInputStream input = new FileInputStream("temp.dat");
    ) {
      // Read values from the file
      int value;
      while ((value = input.read()) != -1)
        System.out.print(value + " ");
    }
  }
}
```

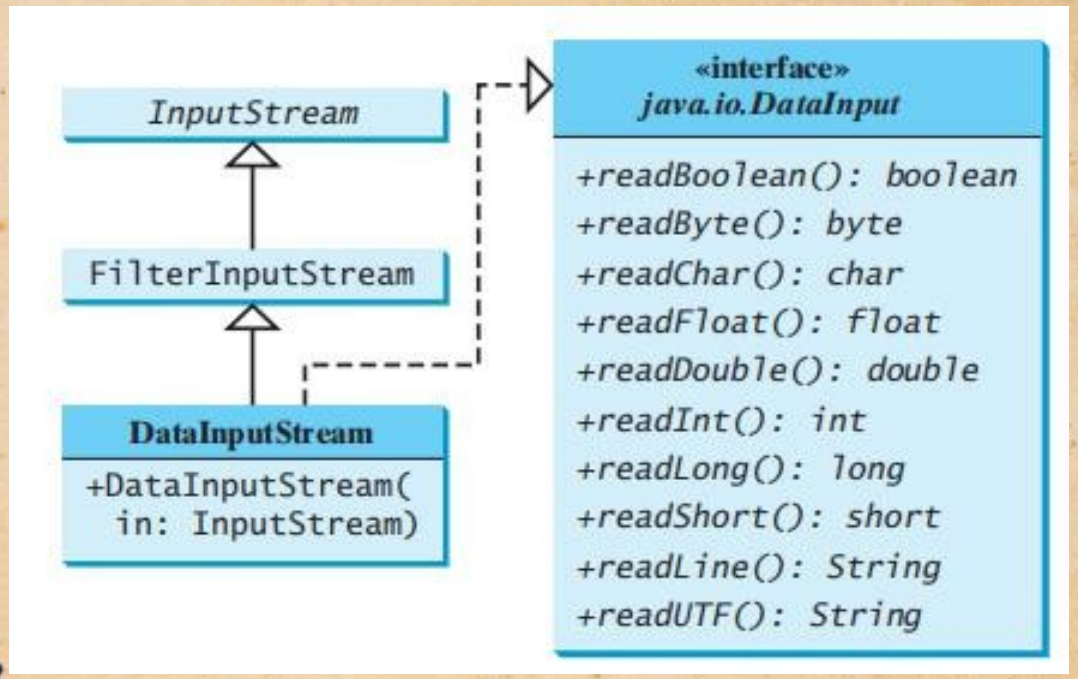
```
1  SOHSTXETXEOTENOACKBELBS
2
```

Contingut de temp.dat

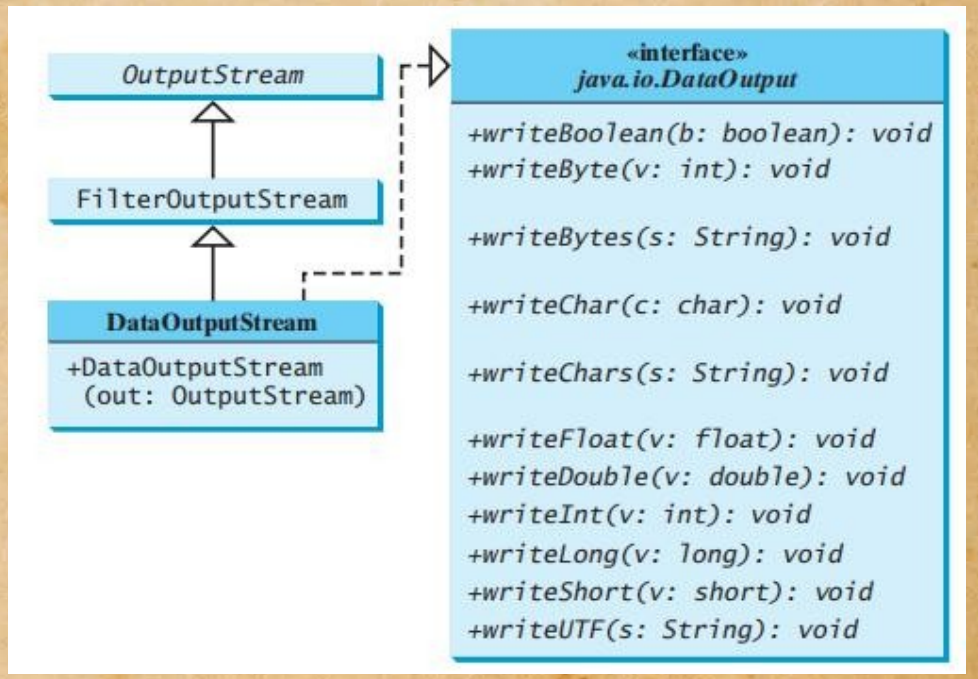
```
1 2 3 4 5 6 7 8 9 10
```

Clases per a entrada/eixida de tipus bàsics i String.

En cas de necessitar emmagatzemar o llegir informació, no per bytes, sinó per tipus de dada, es disposa de les classes `Data`



Entrada



Eixida

Exemple de ús de les classes Data

```java
public class TestDataStream {
  public static void main(String[] args) throws IOException {
    try ( // Create an output stream for file temp.dat
      DataOutputStream output =
        new DataOutputStream(new FileOutputStream("temp.dat"));
    ) {
      // Write student test scores to the file
      output.writeUTF("John");
      output.writeDouble(85.5);
      output.writeUTF("Jim");
      output.writeDouble(185.5);
      output.writeUTF("George");
      output.writeDouble(105.25);
    }

    try ( // Create an input stream for file temp.dat
      DataInputStream input =
        new DataInputStream(new FileInputStream("temp.dat"));
    ) {
      // Read student test scores from the file
      System.out.println(input.readUTF() + " " + input.readDouble());
      System.out.println(input.readUTF() + " " + input.readDouble());
      System.out.println(input.readUTF() + " " + input.readDouble());
    }
  }
}
```

```
John 85.5
Susan 185.5
Kim 105.25
```

Exemple amb detecció de fi de fitxer

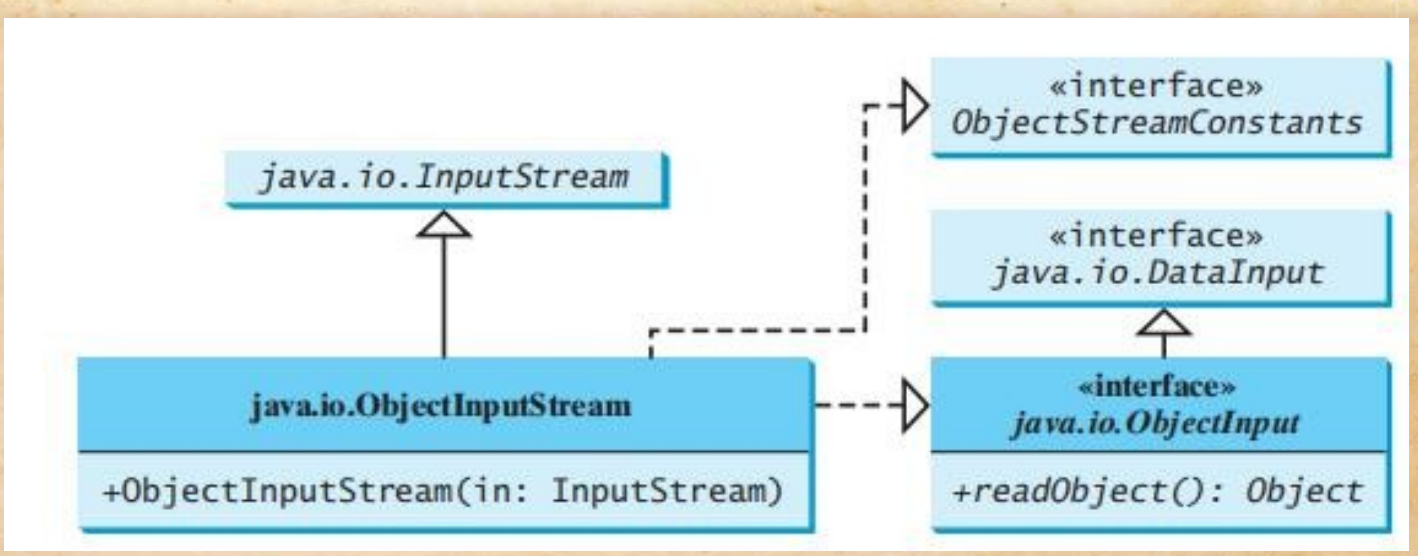
```
import java.io.*;

public class DetectEndOfFile {
  public static void main(String[] args) {
    try {
      try (DataOutputStream output =
        new DataOutputStream(new FileOutputStream("test.dat"))) {
        output.writeDouble(4.5);
        output.writeDouble(43.25);
        output.writeDouble(3.2);
      }

      try (DataInputStream input =
        new DataInputStream(new FileInputStream("test.dat"))) {
        while (true)
          System.out.println(input.readDouble());
      }
    }
    catch (EOFException ex) {
      System.out.println("All data were read");
    }
    catch (IOException ex) {
      ex.printStackTrace();
    }
  }
}
```

```
4.5
43.25
3.2
All data were read
```

Lectura/escriptura d'objectes

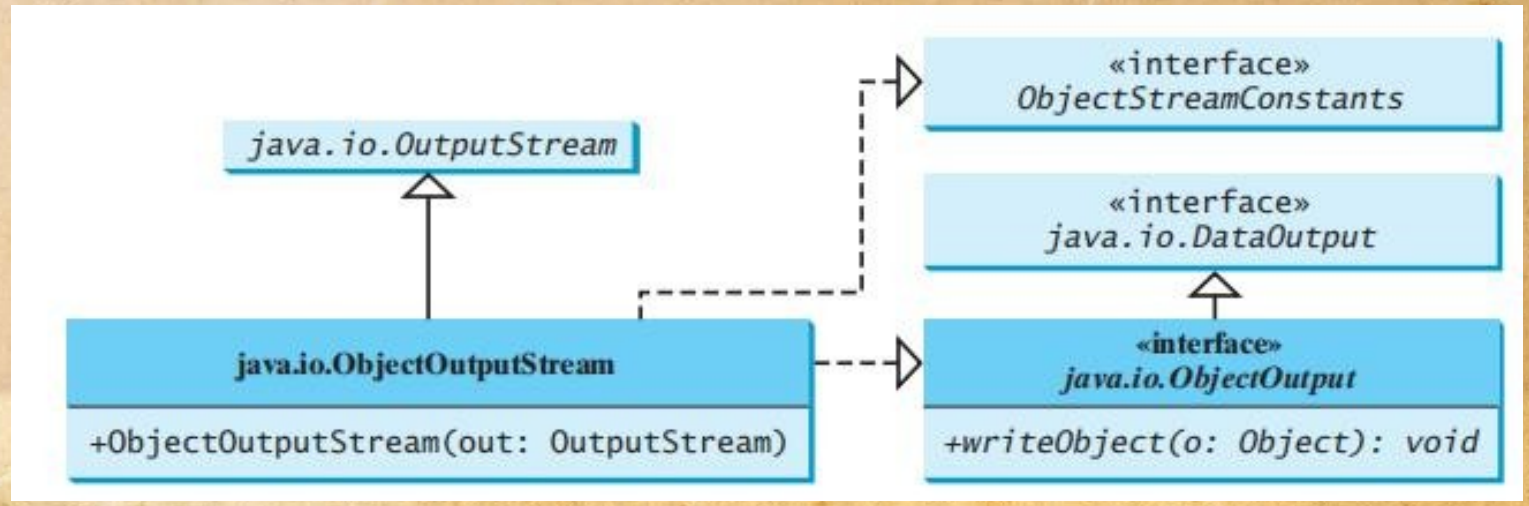
- És **molt habitual** que la informació s'emmagatzeme en bloc (és a dir, per objectes) i no per totes i cadascuna de les dades primitives que el componen.
- Això ens permet també, reconstruir la informació que compon un objecte .
- És important tenir clar que en emmagatzemar la informació d'un objecte en un fitxer es guarda només allò que pertany al propi objecte, i no a la classe (tot el que siga static NO s'emmagatzema).

Clases para la lectura /escritura deobjectes



Entrada

Eixida



Exemple d'escriptura d'un objecte Date

```
import java.io.*;

public class TestObjectOutputStream {

   public static void main(String[] args) throws IOException {
     try ( // Create an output stream for file object.dat
       ObjectOutputStream output =
         new ObjectOutputStream(new FileOutputStream("object.dat"));
     ) {
       // Write a string, double value, and object to the file
       output.writeUTF("John");
       output.writeDouble(85.5);
       output.writeObject(new java.util.Date());
     }
   }
}
```

Exemple de lectura d'un objecte Date

```
import java.io.*;

public class TestObjectInputStream {
  public static void main(String[] args)
    throws ClassNotFoundException, IOException {
    try ( // Create an input stream for file object.dat
      ObjectInputStream input =
        new ObjectInputStream(new FileInputStream("object.dat"));
    ) {
      // Read a string, double value, and object from the file
      String name = input.readUTF();
      double score = input.readDouble();
      java.util.Date date = (java.util.Date)(input.readObject());
      System.out.println(name + " " + score + " " + date);
    }
  }
}
```

Si en llegir la dada es detecta que no és un objecte

Interfície `Serializable`

- No tots els objectes poden ser escrits a un fitxer.
- Només els objectes serialitzables
- Moltes classes de l'API de Java implementen la interfície `Serialitzable: String, StringBuilder, StringBuffer,Date, ArrayList...`
- Si intentem guardarun objecte que no implementa esta interfície obtenim l'excepció `NotSerializableException`

Exemple de classe pròpia Serialitzable

- Per que una classe siga `Serializable`, únicament deu implementar esta interfície

```
public class Datos implements Serializable
{
   public int a;
   public String b;
   public char c;
}
```

- Això implica que si una classe té un atribut d'una altra classe diferent, eixe atribut deu ser també serialitzable perque el total de la informació de l'objecte puga ser emmagatzemada.

```
public class DatoGordo implements Serializable
{
   public int d;
   public Integer e;
   Datos f;
}
```

Si no implementa `Serialitzable`, obtindríem una excepció en emmagatzemar-lo

Exemple de classe pròpia Serialitzable

Programa que escriu un ArrayList en el fitxer “persones.dat”

```
ArrayList<Persona> persones = new ArrayList<>();
persones.add(new Persona("78954184V", "Pere", LocalDate.of(1998, 2, 20)));
persones.add(new Persona("89954745X", "Maria", LocalDate.of(2002, 1, 12)));
persones.add(new Persona("15548746Q", "David", LocalDate.of(1992, 5, 30)));
persones.add(new Persona("22987451s", "Anna", LocalDate.of(1990, 8, 5)));

try(ObjectOutputStream eixida = new ObjectOutputStream(new FileOutputStream("persones.dat"))){
    eixida.writeObject(persones);
} catch (IOException ex) {
    System.out.println(ex.getMessage());
}
```

Programa que llig el fitxer “persones.dat” i guarda les dades en un ArrayList

```
ArrayList<Persona> recollides;
try (ObjectInputStream entrada = new ObjectInputStream(new FileInputStream("persones.dat"))) {
    recollides = (ArrayList<Persona>) entrada.readObject();
    System.out.println(recollides);
} catch (IOException | ClassNotFoundException ex) {
    System.out.println(ex.getMessage());
}
```

persones.dat

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	AC	ED	00	05	73	72	00	13	6A	61	76	61	2E	75	74	69
00000010	6C	2E	41	72	72	61	79	4C	69	73	74	78	81	D2	1D	99
00000020	C7	61	9D	03	00	01	49	00	04	73	69	7A	65	78	70	00
00000030	00	00	04	77	04	00	00	00	04	73	72	00	16	70	72	75
00000040	65	62	61	65	72	72	6F	72	65	73	31	2E	50	65	72	73
00000050	6F	6E	61	12	59	B5	91	E6	73	AA	7A	02	00	03	4C	00
00000060	0D	64	61	74	61	4E	61	69	78	65	6D	65	6E	74	74	00
00000070	15	4C	6A	61	76	61	2F	74	69	6D	65	2F	4C	6F	63	61
00000080	6C	44	61	74	65	3B	4C	00	03	64	6E	69	74	00	12	4C
00000090	6A	61	76	61	2F	6C	61	6E	67	2F	53	74	72	69	6E	67
000000A0	3B	4C	00	03	6E	6F	6D	71	00	7E	00	04	78	70	73	72
000000B0	00	0D	6A	61	76	61	2E	74	69	6D	65	2E	53	65	72	95
000000C0	5D	84	BA	1B	22	48	B2	0C	00	00	78	70	77	07	03	00
000000D0	00	07	CE	02	14	78	74	00	09	37	38	39	35	34	31	38
000000E0	34	56	74	00	04	50	65	72	65	73	71	00	7E	00	02	73
000000F0	71	00	7E	00	06	77	07	03	00	00	07	D2	01	0C	78	74
00000100	00	09	38	39	39	35	34	37	34	35	58	74	00	05	4D	61
00000110	72	69	61	73	71	00	7E	00	02	73	71	00	7E	00	06	77
00000120	07	03	00	00	07	C8	05	1E	78	74	00	09	31	35	35	34
00000130	38	37	34	36	51	74	00	05	44	61	76	69	64	73	71	00
00000140	7E	00	02	73	71	00	7E	00	06	77	07	03	00	00	07	C6
00000150	08	05	78	74	00	09	32	32	39	38	37	34	35	31	73	74
00000160	00	04	41	6E	6E	61	78									