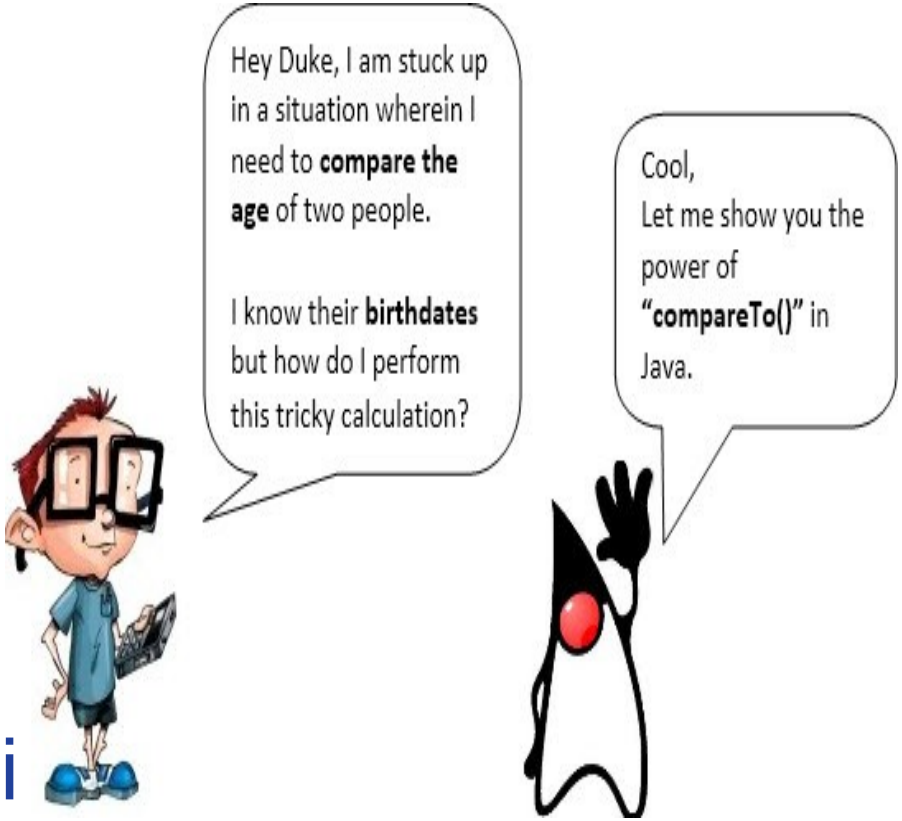
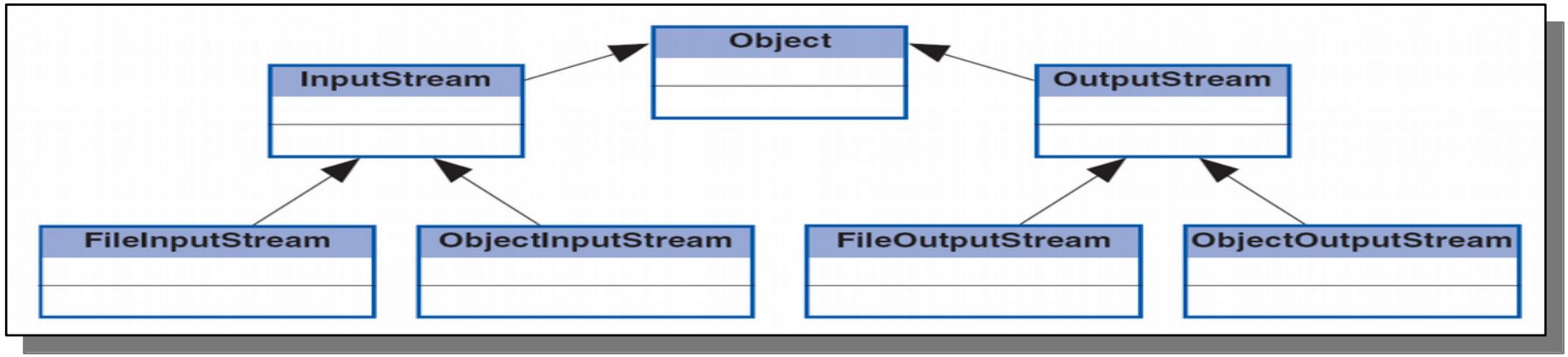
I File Strutturati



Prof. Francesco Viglietti
Www.in4matika.altervista.org

Classi basate sui byte



Vedremo le classi ObjectInputStream ed ObjectOutputStream, che consentono di leggere e scrivere file binari, mettendo a disposizione metodi che agiscono un byte per volta. Naturalmente il file binario non è modificabile con un editor di testi.

Creare file binario

Per creare un file binari usiamo la classe ObjectOutputStream, ***ObjectOutputStream outStream=new ObjectOutputStream (new FileOutputStream(“out.dat”));***
Si può notare la similitudine con i file di testo... ma variano le classi!

```
ObjectOutputStream outStream = null;
try{ //apre il file
    outStream = new ObjectOutputStream (new FileOutputStream(“out.dat”))
    outStream.writeXXX(argomento);//scrive nel file elementi base (int, char,...)
    outStream.close();//chiude il file
}catch (FileNotFoundException e){
    ....//istruzioni per generazione dell'eccezione
}catch (IOException e){
    ....//istruzioni per generazione dell'eccezione
}
```

scrivere in un file binario

Per scrivere in un file binario si usa il metodo **writeXXX(argomento)** dove al posto di XXX si inserisce la tipologia di dato (int, double, boolean,…).
Per le stringhe invece si usa il metodo **writeUTF(stringa)**.
Per gli oggetti si usa **writeObject(oggetto),** l'oggetto dev'essere serializzabile altrimenti si genera eccezione. Tratteremo l'argomento più avanti

Gestione di file strutturati

- Metodi per la **scrittura**. Ogni metodo assume la forma **write*XXX(...)***.

```
outStream.writeInt(25400);
outStream.writeDouble(12.36);
outStream.writeUTF("ciao a tutti");
```

- il metodo **flush** serve per scrivere su disco tutti i dati che sono attualmente contenuti nel buffer dello stream.

- il metodo **writeObject** salva su file un oggetto

leggere file binari

- L'**apertura** di un file strutturato per le operazioni di input, cioè per la lettura:

```
ObjectInputStream inStream = new ObjectInputStream(new
FileInputStream("in.dat"));
```

- La **chiusura** di uno stream, sia di input che di output:

```
inStream.close();
```

Gestione di file strutturati

I metodi della classe **ObjectInputStream** assumono la forma **read*XXX()*.**

- metodo **readObject()** recupera un oggetto precedentemente salvato.
- Il metodo **readUTF()** recupera una stringa precedentemente salvata

Eccezione **EOFException:** segnala che si è raggiunta la fine del file e non ci sono più dati da leggere.

Leggere da file binario

Per leggere da file binari usiamo la classe ObjectInputStream, ***ObjectInputStream inStream=new ObjectInputStream (new FileInputStream("in.dat"));***

Si può notare la similitudine con i file di testo... ma variano le classi!

```
try{ //apre il file
    ObjectInputStream inStream=new ObjectInputStream(new FileInputStream("in.dat"))
    inStream.readXXX();//legge dal file elementi base (int, char,...)
    inStream.close();//chiude il file
}catch (FileNotFoundException e){
    ....//istruzioni per generazione dell'eccezione
}catch (EOFException | IOException e){
    ....//istruzioni per generazione dell'eccezione
}
```

EOFException può essere sfruttata per la fine della lettura del file!

EOFException

Di seguito viene riportato il frammento di codice che gestisce la fine della lettura di un file, che sfrutta l'eccezione.

```
…
try{ //apre il file
    while(true){ //il ciclo termina quando viene generata l'eccezione
       int n=inStream.readInt(); //legge dal file numeri interi
       System.out.print(n+" ");
     }
}catch (EOFException e){
    System.out.println("\n Fine letture file ");
}
inStream.close();//chiude il file
catch (FileNotFoundException){....//istruzioni per generazione dell'eccezione }
catch (IOException e){ ....//istruzioni per generazione dell'eccezione }
```

File binari di oggetti

Java fornisce un modo semplice per la gestione dei file binari di oggetti (o anche array), ***chiamato serializzazione degli oggetti.***
Rendere serializzabile una classe è molto semplice basta che implementi l'inferfaccia serializable.
Per rendere l'interfaccia accessibile al main dev'essere importata la libreria ***java.io.Serializable***
La scrittura degli oggetti aviene mediante il metodo ***writeObject(parametro)***, mentre la lettura avviene mediante il metodo ***readObject()*** che restutuira un Object e avrà bisogno del casting!