

Programmazione ad Oggetti

I/O

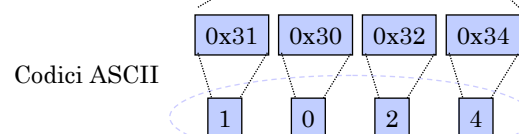
Attenzione a DataInputStream

- DataInputStream legge i dati in formato binario, quindi **NON** funziona con file di testo.

```
FileInputStream fis=new FileInputStream("file");  
DataInputStream file = new DataInputStream(fis);  
int numero = file.readInt();
```

- Il risultato è:

- numero = 825242164 = 0x31303234



InputStream

▪E' la classe dell'oggetto `System.in`

▪I metodi principali sono:

```
int available()
```

```
void close()
```

```
int read() throws IOException
```

```
int read(byte[] b) throws IOException
```

▪Problema:

- ◆ come fare a leggere una linea?

readLine from InputStream

```
final static int EOF = -1;
StringBuffer buffer=new StringBuffer();
int input;
try{
    while((input=System.in.read())!='\n'){
        if(input == EOF) break;
        if(input != '\r'){
            char ch = (char)input;
            buffer.append(ch);
        }
    }
}catch(Exception e){
    System.err.println("Error");
}
```

Termina la lettura quando incontra la fine dello stream

A capo:
in Unix: “\n”
in Win: “\n” + “\r”

Considera solo il primo byte di input

BufferedReader

- Per leggere una linea per volta da uno stream si utilizza la classe `BufferedReader` che fornisce il metodo

```
String readLine();
```

- `BufferedReader` NON può essere costruito direttamente a partire da un `InputStream` (come ad es. `System.in`), bisogna passare attraverso `InputStreamReader`:

```
BufferedReader in =  
    new BufferedReader(  
        new InputStreamReader(System.in));
```

Lettura da file di testo

- Il metodo per leggere un file di testo riga per riga è il seguente:

```
FileReader fr = new FileReader("es.txt");  
BufferedReader file = new BufferedReader(fr);  
while(true){  
    String line = file.readLine();  
    if(line==null) break;  
    System.out.println(line);  
}
```

Linee strutturate

- Leggere i prodotti della cassa da un file di testo con le linee strutturate nel modo seguente

```
<codice> ; <Nome> ; <Prezzo>
```

- Ogni linea letta dal file deve essere suddivisa in corrispondenza dei “;” (separatori)
-

StringTokenizer

- Permette di suddividere una stringa in base a dei separatori

- Costruttore:

```
StringTokenizer(String str, String delim)
```

- Metodi di iterazione:

```
boolean hasMoreTokens()
```

```
String nextToken()
```

Writers

- I writer non sono equivalenti
 - ◆ Non tutti rappresentano il new-line in maniera corretta

```
FileWriter fw = new FileWriter("provaFW.txt");  
fw.write("Hello!\n");  
PrintWriter pw = new PrintWriter(fw);  
pw.println("Hello!");
```

- Se non vengono chiusi il buffer si perde!
 - ◆ fw.close()
-

URLs

- Gli stream possono essere collegati, oltre che ai file, a degli URL

```
URL page = new URL(url);  
InputStream in = page.openStream();
```

- Occorre fare attenzione al tipo di file che si scarica
-

I/O Test

- Redirect output

```
buffer = new ByteArrayOutputStream();  
out = System.out;  
System.setOut(new PrintStream(buffer));
```

- Get captured output and restore

```
System.setOut(out);  
return buffer.toString();
```

```
private PrintStream out;  
private ByteArrayOutputStream buffer;
```

I/O Test

- Redirect input

```
in = System.in;  
ByteArrayInputStream buffer  
    =new ByteArrayInputStream(  
        input.getBytes());  
System.setIn(buffer);
```

- Restore input

```
System.setIn(in);
```

```
private InputStream in;
```
