Programmazione in Java
e
gestione della grafica

Lezione 24

Parliamo di ….

- Files and Streams

Files e streams

- I dati memorizzati in variabili e array sono dati temporanei
- File si usano per conservare grosse quantità di dati anche dopo che il programma che li ha creati è terminato
- Dati persistenti – esistono oltre la durata dell'esecuzione del programma
- File si conservano su dispositivi di memoria secondaria
- Stream – dati ordinati scritti o letti su o da file

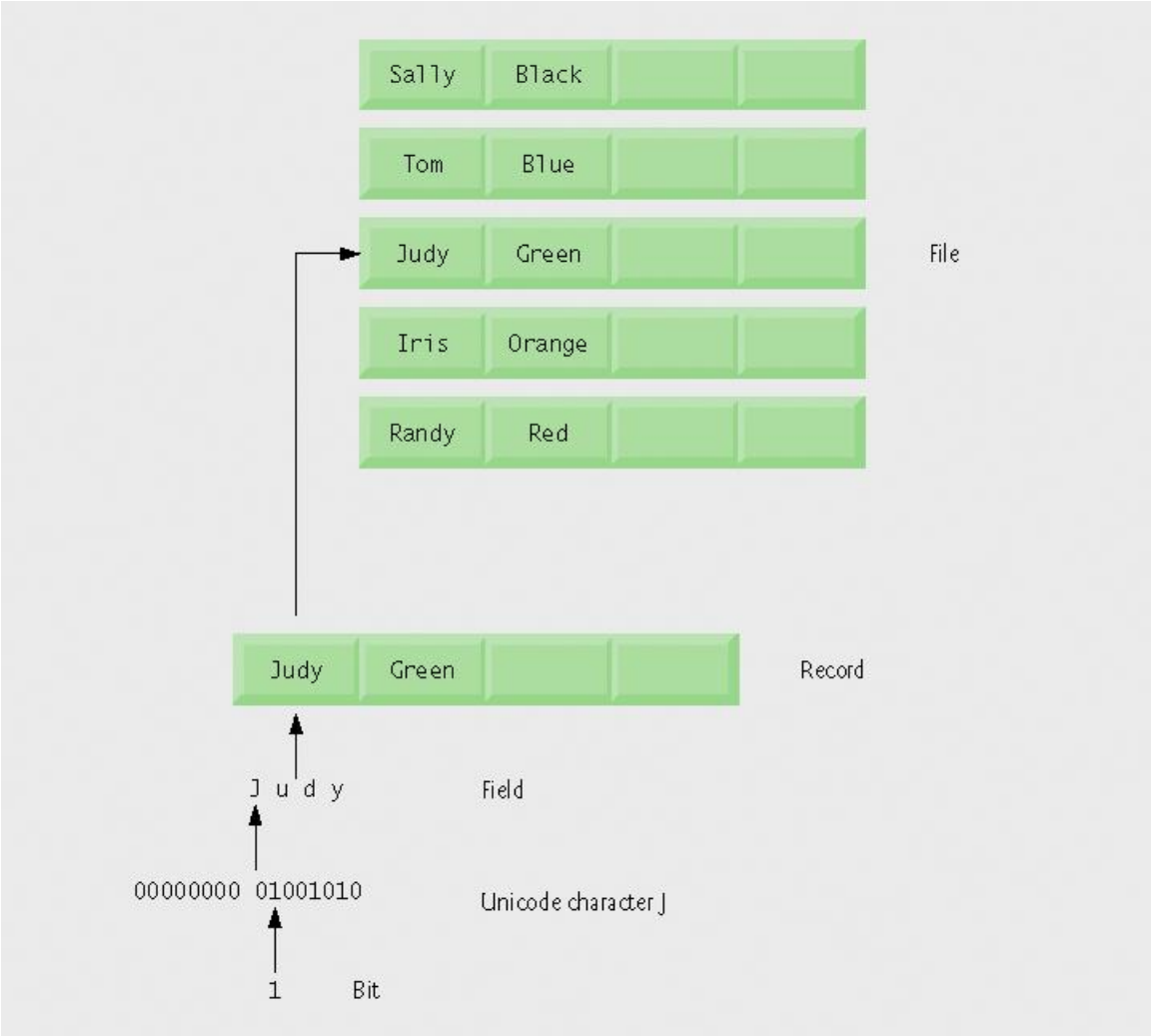
Gerarchia dei dati

- Computer elabora i dati come sequenze di zero e uno
- Bit (binary digit) – dato più piccolo in un computer, può avere valore 0 o 1
- Byte – 8 bits
- Character – dato più grande
  - Cifre decimali, lettere e simboli speciali
  - Character set – insieme di tutti I caratteri usati per scrivere i programmi e rappresentare I dati
    - Unicode – caratteri composti da due bytes
    - ASCII (American Stardard Code for Information Interchange) sono sottoinsieme di unicode

- Field (campo) – un gruppo di characters o bytes che hanno insieme un qualche significato
- Record – un gruppo di fields collegati
- File – un gruppo di record collegati

I dati elaborati da un computer formano cioè una gerarchia sempre più complessa dai bits ai files

- Record key (chiave del record) – usata per identificare il record – è unica per ogni record. E' usata per cercare e ordinare i records
- File Sequenziale – file in cui i records sono ordinati secondo il campo (field) record-key
- Database – un gruppo di file
- Database Management System – una collezione di programmi progettati per creare e gestire databases


Sally
Black
Tom
Blue
Judy
Green
Iris
Orange
Randy
Red
File
Judy
Green
Record
J u d y
Field
00000000 01001010
Unicode character J
1
Bit

Files e Streams

- Java vede ogni file come uno stream sequenziale di bytes
- I sistemi operativi hanno un proprio meccanismo per determinare la fine di un file
  - End-of-file marker
  - Conteggio del numero totale di bytes nel file
  - programma Java che processa uno stream riceve indicazioni dal sistema operativo sulla fine dello stream

Files e Streams

- File stream
  - Streams basati su Byte – memorizzano I dati in binario
    - File binari – creati da Streams basati su Byte possono essere letti da programmi specifici che convertono il formato
  - Streams basati su Character– memorizzano I dati come sequenze di characters
    - File di testo – creati da Streams basati su caratteri, possono essere letti dai , text editors
- Java apre un file creando un oggetto e associando ad esso uno stream

Files e Streams

- Standard streams – ogni stream può essere reindirizzato (redirected)
  - `System.in` – oggetto stream per standard input, redirected con `setIn`
  - `System.out` – oggetto stream per standard output, redirected con `setOut`
  - `System.err` – oggetto stream per standard error, redirected con `setErr`

Classi `java.io`

- `FileInputStream`
- `FileOutputStream` – I/O basati su byte

- `FileReader`
- `FileWriter` – I/O basati su character

- `ObjectInputStream`
  `ObjectOutputStream` – si usano per input e output di oggetti o variabili di tipi di dati primitivi

- Classe `File` – si usa per ottenere informazioni su files e directories

- Classi `Scanner` e `Formatter`
  - `Scanner` – si usa per leggere dati da un file
  - `Formatter` – si usa per scrivere dati su un file

Un file di *n* bytes visto da Java

La Classe `File`

- Classe `File` si usa per recuperare informazioni su I files e le directori del disco
- Oggetti della classe `File` non servono per aprire i file e non danno possibilità di processare i file .

Creare Oggetti `File`

- Costruttori per la classe `File` :
  1. Prende una `String` specifica nome e path (location del file sul disco)
  2. Prende due `Strings`, la prima specifica path e la seconda specifica il nome del file
  3. Prende oggetto `File` specificando il path e una `String` specificando il nome del file
  4. Prende oggetto `URI` (Uniform Reseurce Identifier) specificando nome e path
- Due diversi tipi di path
  - path assoluto– contiene tutte le directory a partire dalla root
  - path relativo– inizia dalla directory da cui è partita l'applicazione

Metodi di File

Method	Description
`boolean canRead()`	Returns `true` if a file is readable by the current application; `false` otherwise.
`boolean canWrite()`	Returns `true` if a file is writable by the current application; `false` otherwise.
`boolean exists()`	Returns `true` if the name specified as the argument to the `File` constructor is a file or directory in the specified path; `false` otherwise.
`boolean isFile()`	Returns `true` if the name specified as the argument to the `File` constructor is a file; `false` otherwise.
`boolean isDirectory()`	Returns `true` if the name specified as the argument to the `File` constructor is a directory; `false` otherwise.
`boolean isAbsolute()`	Returns `true` if the arguments specified to the `File` constructor indicate an absolute path to a file or directory; `false` otherwise.

Method	Description
`String getAbsolutePath()`	**Returns a string with the absolute path of the file or directory.**
`String getName()`	**Returns a string with the name of the file or directory.**
`String getPath()`	**Returns a string with the path of the file or directory.**
`String getParent()`	**Returns a string with the parent directory of the file or directory (i.e., the directory in which the file or directory can be found).**
`long length()`	**Returns the length of the file, in bytes. If the `File` object represents a directory, `0` is returned.**
`long lastModified()`	**Returns a platform-dependent representation of the time at which the file or directory was last modified. The value returned is useful only for comparison with other values returned by this method.**
`String[] list()`	**Returns an array of strings representing the contents of a directory. Returns `null` if the `File` object does not represent a directory.**

- Può essere utile usare il metodo `isFile` di `File` per determinare se un oggetto `File` rappresenta effettivamente un file (no una directory) prima di cercare di aprire il file.

- Metodi di `File` più comuni:
  - `exists` – return `true` se il file esiste dove specificato
  - `isFile` – return `true` se `File` è un file, no una directory
  - `isDirectory` – return `true` se `File` è una directory
  - `getPath` – return file path come string
  - `list` – restituisce contenuto della directory
- Carattere Separatore – usato nei path delle directories
  - Windows usa \
  - UNIX usa /
  - Java processa entrambi,
  - `File.pathSeparator` può essere utilizzato per ottenere quello usato nel computer locale

Attenzione…

- Usare \ come separatore di directory invece di \\ in una stringa è un errore logico.

File ad accesso sequenziale

- Records sono salvati in ordine rispetto al campo chiave
- Può essere creato come file di testo o come file binario

Creare un file di testo ad accesso sequenziale

- Java non impone una struttura ai file, I record non esistono come parte del linguaggio Java (bisogna scrivere programmi a questo scopo!)
- Classe `Formatter` può essere usata per aprire un file di testo per scrivere l'output
  - Passa nome file al costruttore
  - Se il file non esiste, viene creato
  - Se il file esiste il contenuto viene If file already exists, contents are truncated (discarded)
  - Uso metodo `format` per scrivere sul file il testo formattato
  - Uso metodo `close` per chiudere l'oggetto `Formatter` (in ogni caso il sistema operativo chiude il file quando il programma termina)

Eccezioni relative ai files

- `SecurityException` – quando, aprendo il file usando un oggetto `Formatter`, l'utente non ha il permesso di scrittura
- `FileNotFoundException` – quando, aprendo il file usando un oggetto `Formatter`, il file non si trova e non si può creare nuovo file
- `NoSuchElementException` – quando si legge input non valido con un oggetto `Scanner`
- `FormatterClosedException` – quando si cerca di scrivere su un file usando un oggetto `Formatter` già chiuso

Combinazione tasti per end-of-file

Operating system	Key combination
UNIX/Linux/Mac OS X	***<return> <ctrl> d***
Windows	***<ctrl> z***

Aggiornare file ad accesso sequenziale

- In genere i dati nei file ad accesso sequenziale non possono essere modificati senza rischiare di distruggere altri dati nel file
- Vecchi dati non possono essere riscritti se i nuovi non sono esattamente della stessa lunghezza
- In genere si riscrive tutto il file.

Serializzazione degli oggetti

- Meccanismo per leggere o scrivere un intero oggetto da un file
- Serializzazione – rappresentare un oggetto come una sequanza di bytes (che contiene tutte le info sull'oggetto)
- Deserializzazione – ricreare oggetto in memoria da I dati nel file
- Si usano le classi `ObjectInputStream` e`ObjectOutputStream`, e i metodi `readObject` e`writeObject`

CreateTextFileTest.java

ReadTextFileTest.java