Le basi del linguaggio Java

- La gestione dell'input/output da tastiera
- La gestione dell'input/output da file
- La gestione delle eccezioni

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

L'oggetto di tipo Scanner

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Un oggetto per la lettura dalla tastiera

Le API di Java hanno un oggetto **System.in che rappresenta la** tastiera del calcolatore, ma che non è semplice da usare direttamente.
Le stesse API offrono diversi modi per poter leggere da tastiera.

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Un oggetto per la lettura dalla tastiera

Un primo metodo che si vuole presentare è quello offerto dall'oggetto **in** **di tipo** **Scanner**, **del package** **java.util**

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Alcune operazioni consentite dall'oggetto di tipo scanner

- **int nextInt()**
  legge un numero intero, e restituisce il numero letto
- **double nextDouble()**
  legge un numero reale, e restituisce il numero letto
- **String nextLine()**
  legge una linea di testo, e la restituisce
- **String next()**
  legge un "token" (intuitivamente, una sequenza di caratteri contigui e senza separatori), e restituisce il token letto
- **boolean hashNextInt() – boolean hashNextDouble()**
  verifica se il prossimo token può essere interpretato come un numero intero/reale
- **boolean hashNextLine() – boolean hashNext()**
  verifica se in input è disponibile una ulteriore linea/token

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

utilizzo

Per usare un oggetto **in** di tipo **Scanner** bisogna:

- importare la classe **Scanner** dal package **java.util** usando la clausola **import java.util.*;**
- dichiarare una variabile **in** di tipo **Scanner**
- creare l'oggetto che rappresenta la tastiera mediante un'istruzione **in = new Scanner(System.in);**
- usare opportunamente le operazioni dell'oggetto **in**

int nextInt()
double nextDouble()
String nextLine()
String next()
boolean hashNextInt()
boolean hashNextDouble()
boolean hashNextLine()
boolean hashNext()

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

esempio

Si vuole scrivere un'applicazione Java che legge dalla tastiera due numeri interi, ne calcola la somma e la visualizza sullo schermo

Scrivi due numeri interi

10 15

La somma dei due numeri è 25

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

esempio

```
package progetto1;
import java.util.*;
/* Applicazione che legge dalla tastiera due numeri interi
* e ne calcola e visualizza la somma. */
class Somma
{
public static void main(String[] args)
        {
        Scanner in;  // per la lettura dalla tastiera definiamo l'oggetto in di tipo Scanner
        /* crea l'oggetto che rappresenta la tastiera */
        in = new Scanner( System.in );
        System.out.println("Scrivi due numeri interi");
        /* legge due numeri interi a e b */
        int a = in.nextInt();
        int b = in.nextInt();
        /* calcola la somma di a e b e la visualizza */
        int somma = a+b;
        System.out.print("La somma dei due numeri è: " +somma);
        }
}
```

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Input da file

Per leggere dati da un file presente sul disco, la classe Scanner si affida ad un'altra classe, File, che descrive file e cartelle presenti in un file system.

Per prima cosa si costruisce un oggetto di tipo File, fornendo il nome del file da leggere:

```
File inFile = new File("input.txt");
```

Successivamente si utilizza tale oggetto per costruire un oggetto di tipo Scanner:

```
Scanner in = new Scanner(inFile);
```

Output su file

Per scrivere dati in un file si costruisce un oggetto di tipo PrintWriter , fornendo il nome del file:

PrintWriter out = new PrintWriter("output.txt")

Se il file in cui si scrive esiste già, viene svuotato prima di scrivervi nuovi dati. Se il file non esiste viene creato un file nuovo

Con un oggetto di tipo PrintWriter potete usare gli usuali metodi print, println

Quando avete terminato di scrivere in un file accertatevi di chiudere l'oggetto PrintWriter:
out.close()

Output su file

Esempio:

```
import java.util.*;
import java.io.*;
Public  class LetturaFile {
    public static void main(String[]  args)
{
    try {
    File  inputF= new File("input1.txt");
    Scanner  in = new Scanner(inputF);
    PrintWriter out=  new  PrintWriter ("output1.txt");
    String riga = in.nextLine();
    out.println(riga);
    out.close();
    }
    Catch(FileNotFoundException exc)
    {
    System.out.println("Il file di input non esiste");
    }
}
}
```

Output su file

Esercizio

Scrivere un programma che legge tutte le righe presenti in un file e le scrive in un altro file inserendo per ciascuna riga il corrispondente numero di riga.

L'oggetto di tipo System.out

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

System.out

System.out rappresenta un oggetto associato allo standard output. Esiste anche un oggetto associato allo standard error che è System.err. Entrambi fanno riferimento, in assenza di una diversa specifica (per default) allo schermo.

System.out

println è un metodo. Sull'oggetto System.out viene invocato il metodo println(), con un parametro che rappresenta il valore da visualizzare. Il parametro può essere una stringa, un numero intero o reale, un carattere o un booleano. Quando il metodo println() viene richiamato, stampa sul video il parametro e inserisce un ritorno a capo.

È possibile utilizzare anche l'oggetto print(): esso stampa il parametro passato, ma non il ritorno a capo.

I metodi print() e println() ricevono un unico parametro, ma utilizzando l'operatore + è possibile concatenere stringhe e numeri.

System.out

Esercizio

Scrivere un programma che consenta di stampare il valore di tre variabili: un numero intero, un numero reale e un booleano.

System.out

Soluzione:

```
class stampa
{
   public static void main (String args[])
   {
         int intero = 5;
         float reale = 25.68f;
         boolean trovato = false;
         System.out.println(“Numero intero = “ + intero);
         System.out.println(“Numero reale = “ +   reale);
         System.out.println(“Trovato = “ + false);
   }
}
```

System.in

Per le operazioni di input esiste un oggetto analogo System.in che gestisce il flusso di dati inseriti da tastiera. L'oggetto System.in, in pratica viene mascherato con altri oggetti più potenti che forniscono maggiori funzionalità.

Si utilizza la classe BufferReader nel seguente modo:

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

System.in

Con queste dichiarazioni viene definito un oggetto, di classe BufferedReader, usando l'operatore new che crea un nuovo oggetto.

La classe BufferedReader mette a disposizione il metodo readLine() che consente di leggere, una riga per volta.

Una riga viene considerata terminata quando si preme il tasto Invio.

Questo metodo acquisisce solo stringhe , quindi se si vogliono acquisire valori numeri si deve effettuare la conversione tra stringhe e numeri.

System.in

Per effettuare la lettura di una stringa dobbiamo dichiarare una variabile di tipo stringa e poi usare il metodo readLine() con l'oggetto tastiera definito in precedenza.

Esempio:

```
String nome;
nome = tastiera.readLine();
```

Le eccezioni

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Le eccezioni

L'operazione deve essere completata considerando le eccezioni che possono essere generate dal metodo readLine(). L'eccezione segnala una situazione anomala.

Quando si verifica un eccezione bisogna prevedere un blocco di istruzioni per gestire questa situazione.

Il blocco viene realizzato con il costrutto try {} catch {}

Le eccezioni

In Java una **classe Exception**, e le eccezioni sono degli oggetti appartenenti a questa classe, questo ci permette di trattarle come gli altri componenti del linguaggio.

Le eccezioni

Quindi la lettura di una stringa si effettua correttamente nel seguente modo:

```
String nome;
try
{
  nome = tastiera.readLine();
}
catch(exception e) {}
```

Le eccezioni

Il segmento di codice precedente stabilisce che, al verificarsi dell'eccezione non deve essere effettuata alcuna operazione.

Eccezioni predefinite:

AritmeticException: segnala errori aritmetici;

NullPointyerException: errore dovuto all'utilizzo di un riferimento che possiede il valore null;

IndexOutBoundException: errore nell'indice dell'array;

IOException: errore generico di input/output

Lettura di numeri

Per effettuare la lettura di numeri, bisogna convertire la stringa ottenuta dall'operazione di input nel seguente modo:

```
String leggiNumero;
int num;
 try
{
    leggiNumero = tastiera.readLine();
   num = Integer.valueOf(leggiNumero).intValue;
}
catch(exception e) {
   System.out.println("Numero non corretto");
return;
}
```

finally

Se nel try è presente una clausola **finally**, il suo codice viene eseguito dopo aver completato tutte le altre operazioni del try, indipendentemente dal fatto che questa abbiano lanciato una eccezione o no.

Esercizi

1. Leggere da tastiera l'età di tre persone e calcolare l'età media;
2. Eseguire una divisione con divisore uguale a zero e gestire l'eccezione generata.