Corso di Programmazione
Anno Accademico 2013–2014
Prof. Giovanni Pighizzini

Note ed esercizi aggiuntivi

12. Eccezioni

Esempio. Conteggio del numero di lettere maiuscole presenti in un file di testo il cui nome è fornito sulla linea di comando (esempio già presentato relativamente ai file di testo).

```
import prog.io.FileInputManager;
import prog.io.ConsoleOutputManager;

class ContaMaiuscole {
  public static void main(String[] args) {
    FileInputManager in = new FileInputManager(args[0]);
    ConsoleOutputManager out = new ConsoleOutputManager();

    int nMaiuscole = 0;
    String s = in.readLine();
    while (s != null) {
      for (int i = 0; i < s.length(); i++)
        if (Character.isUpperCase(s.charAt(i)))
          nMaiuscole++;
      s = in.readLine();
    }
    out.println("Il testo contiene " + nMaiuscole + " lettere maiuscole");
  }
}
```

Note
Se sulla linea di comando non si specifica alcun argomento, o se l'argomento specificato non è il nome di un file (o il file non è accessibile), l'esecuzione viene interrotta in maniera eccezionale. Analizzate in dettaglio i messaggi d'errore forniti dalla Java Virtual Machine nei due casi, accertandovi di comprenderne il significato.

Esempio. L'esempio precedente modificato per *prevenire* le due tipologie di errori evidenziate.

```
import prog.io.FileInputManager;
import prog.io.ConsoleOutputManager;

class ContaMaiuscole {
  public static void main(String[] args) {
    ConsoleOutputManager out = new ConsoleOutputManager();
```

```
    if (args.length == 0)
      out.println("Non e' stato specificato il nome del file!");
    else if (!FileInputManager.exists(args[0]))
      out.println("Non esiste alcun file di nome " + args[0]);
    else {
      FileInputManager in = new FileInputManager(args[0]);

      int nMaiuscole = 0;
      String s = in.readLine();
      while (s != null) {
        for (int i = 0; i < s.length(); i++)
          if (Character.isUpperCase(s.charAt(i)))
            nMaiuscole++;
        s = in.readLine();
      }
      out.println("Il testo contiene " + nMaiuscole + " lettere maiuscole");
    }
  }
}
```

Note
Si verifica preliminarmente che sia stato specificato un argomento sulla linea di comando (prima selezione) e che esista un file con tale nome (secondo selezione). Si noti che scambiando l'ordine di questi due test il codice non risulterebbe corretto. All'interno del secondo `else` è contenuto il codice che deve essere eseguito quando è stato fornito correttamente il nome di un file esistente e accessibile. Questo è lo stesso codice della versione precedente (solo l'invocazione del costruttore di `ConsoleOutputManager` è stata spostata all'inizio, in quanto l'oggetto che rappresenta il monitor viene utilizzato anche nelle selezioni per segnalare gli errori).

Esercizio 12.1
Nell'esempio precedente il riferimento `args[0]` viene utilizzato tre volte. Modificate il codice in modo che il valore di `args[0]` venga assegnato a una variabile di tipo `String`, da utilizzare poi al posto di `args[0]`.

Esempio. L'esempio iniziale modificato per *curare* gli errori in esecuzione, mediante l'uso del costrutto `try-catch`.

```
import prog.io.FileInputManager;
import prog.io.ConsoleOutputManager;
import prog.io.FileNonPresenteException;

class ContaMaiuscole {
  public static void main(String[] args) {
    ConsoleOutputManager out = new ConsoleOutputManager();

    try {
      FileInputManager in = new FileInputManager(args[0]);

      int nMaiuscole = 0;
      String s = in.readLine();
      while (s != null) {
```

```
        for (int i = 0; i < s.length(); i++)
          if (Character.isUpperCase(s.charAt(i)))
            nMaiuscole++;
        s = in.readLine();
      }
      out.println("Il testo contiene " + nMaiuscole + " lettere maiuscole");
    } catch (ArrayIndexOutOfBoundsException e) {
      out.println("Non e' stato specificato il nome del file!");
    } catch (FileNonPresenteException e) {
      out.println("Non esiste alcun file di nome " + args[0]);
    }
  }
}
```

Note

- Si noti l'uso di due parti `catch` per trattare separatamente le due differenti tipologie di eccezioni.
- Si osservi l'uso della direttiva di importazione per la classe `FileNonPresenteException`. La direttiva non è necessaria per `ArrayIndexOutOfBoundsException` in quanto quest'ultima fa parte del package `java.lang`.
- Come specificato nella documentazione, la classe `FileNonPresenteException` è una sotto-classe di `RuntimeException`.
- L'esempio ha scopo didattico, per descrivere in un caso semplice l'uso del costrutto `try-catch`. Come mostrato in precedenza, per questo problema è abbastanza semplice prevenire gli errori utilizzando delle selezioni.

Esempio. Calcolo della somma di numeri interi forniti sulla linea di comando. Ad esempio, mandando in esecuzione il programma mediante il comando

```
java Somma 23 45 18
```

verrà prodotot in output il numero `86`.

```
class Somma {
  public static void main(String[] a) {
    int somma = 0;
    for (String s: a) {
      Integer i = new Integer(s);
      somma = somma + i;
    }
    System.out.println(somma);
  }
}
```

Note
Osservate i messaggi d'errore che vengono prodotti nel caso sulla riga di comando vengano inserite delle stringhe che non sono costituite da cifre. Cercate di comprenderne il significato.

Esempio. L'esempio precedente, modificato per prevenire gli errori dovuti all'inserimento di stringhe non costituite da cifre.

```
class Somma {
  public static void main(String[] a) {
    //controlla che le stringhe inserite siano costituite esclusivamente da cifre
    boolean soloCifre = true;
    for (String s: a)
      for (int j = 0; j < s.length(); j++)
        if (!Character.isDigit(s.charAt(j)))
          soloCifre = false;

    if (soloCifre) {
      int somma = 0;
      for (String s: a) {
        Integer i = new Integer(s);
        somma = somma + i;
      }
      System.out.println(somma);
    } else
      System.out.println("Errore: sono state fornite stringhe che non rappresentano numeri!");
  }
}
```

Esercizio 12.2
Fate riferimento all'esempio precedente.

1. Nella parte di controllo, una volta che viene trovato un carattere che non è una cifra, è inutile esaminare i caratteri e le stringhe successive. Riscrivete questa parte in modo da evitare questi controlli, in due modi differenti:
   - ricorrendo all'istruzione `break` (esiste anche una versione "con etichetta", non presentata a lezione),
   - senza ricorrere all'istruzione `break`.
2. Modificate l'applicazione in modo che, in caso di errore, fornisca un messaggio che indichi la prima stringa, tra quelle inserite, che non rappresenta un numero.

Esempio. Di nuovo l'esempio della somma di numeri interi forniti sulla linea di comando, intercettando le eccezioni per trattare il caso di inserimento di stringhe non costituite da cifre.

```
class Somma {
  public static void main(String[] a) {
    int somma = 0;
    try {
      for (String s: a) {
        Integer i = new Integer(s);
        somma = somma + i;
      }
      System.out.println(somma);
    } catch (NumberFormatException e) {
      System.out.println("Errore: sono state fornite stringhe che non rappresentano numeri!");
    }
  }
}
```

Note
Il comportamento che l'utente osserva eseguendo questo programma è esattamente lo stesso della versione precedente, in cui le stringhe vengono controllate preliminarmente. I due programmi sono cioè *indistinguibili* durante l'esecuzione Tuttavia in questa versione il codice è decisamente più semplice e leggibile. In questo caso l'uso del meccanismo delle eccezioni permette di semplificare notevolmente la struttura del codice separando il codice "normale" (cioè ciò che si vorrebbe eseguire) dal codice che si occupa di trattare le situazioni anomale.

Esempio. Ancora l'esempio della somma di numeri interi forniti sulla linea di comando, intercettando le eccezioni per trattare il caso di inserimento di stringhe non costituite da cifre. In questo caso, vengono sommate tutte le stringhe che rappresentano numeri. Per ciascuna delle altre viene segnalato un errore.

```
class Somma {
  public static void main(String[] a) {
    int somma = 0;
    for (String s: a)
      try {
        Integer i = new Integer(s);
        somma = somma + i;
      } catch (NumberFormatException e) {
        System.out.println("La stringa \"" + s + "\" non rappresenta un numero: ignorata");
      }
    System.out.println(somma);
  }
}
```

Note
È stato utilizzato un costrutto `try-catch` all'interno del ciclo `for`: nel caso la stringa non rappresenti un numero l'applicazione segnala il problema e prosegue l'elaborazione con le altre stringhe. Confrontate il codice con quello della versione precedente in cui, invece, è stato utilizzato un ciclo `for` all'interno del costrutto `try-catch`: nel caso la stringa non rappresenti un numero viene segnala l'errore e l'applicazione termina l'esecuzione.

Esercizio 12.3
Scrivete un'applicazione che abbia esattamente lo stesso comportamento di quella presentata nell'ultimo esempio, senza ricorrere al meccanismo delle eccezioni, ma prevenendo le situazioni anomale.

Esercizio 12.4
Modificate l'applicazione che conta il numero di lettere maiuscole in un file il cui nome è specificato sulla linea di comando in modo che nel caso il file non sia accessibile chieda all'utente di inserire un altro nome di file da tastiera. Naturalmente, anche il nome fornito da tastiera potrebbe non corrispondere a un file accessibile e, dunque, la richiesta dovrà essere ripetuta. Per risolvere il problema potete combinare il costrutto `try-catch` con un costrutto iterativo, che termina quando l'utente inserisce il nome di un file accessibile.

Dopo avere risolto il problema, modificate ulteriormente il codice in modo che l'utente abbia la possibilità di rinunciare all'operazione (ad esempio inserendo come nome di file la stringa vuota).