Corso di Programmazione
Anno Accademico 2015–2016
Prof. Giovanni Pighizzini

Note ed esercizi aggiuntivi

Gli esercizi proposti sono utili per rivedere gli esempi riportati, che sono stati sviluppati e discussi in dettaglio a lezione.

12. Eccezioni

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 prodotto 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 12.1
Fate riferimento all'esempio precedente.

1. Nella parte di controllo, quando venga trovato un carattere che non è una cifra, risulta 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` (osservate che l'istruzione termina l'esecuzione del ciclo più interno; esiste anche una versione "con etichetta" per forzare la terminazione di un ciclo esterno),
   - 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 facendo eseguire 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 segnalato l'errore e l'applicazione termina l'esecuzione.

Esercizio 12.2
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.

Esempio. Esempio di codice che può sollevare svariati problemi in esecuzione (individuarli tutti!)

```
class Prova {
  public static void main(String[] a) {
    int somma = 0;
    for (String s: a)
      somma = somma + s.length();
    System.out.println(somma);
    System.out.println(somma / a.length);
    Integer n = new Integer(a[1]);
    System.out.println(10 / n);
  }
}
```

Lo stesso codice con un blocco `try-catch` per intercettare le singole eccezioni.

```
class Prova {
  public static void main(String[] a) {
    int somma = 0;
    for (String s: a)
      somma = somma + s.length();
    System.out.println(somma);
    try {
      System.out.println(somma / a.length);
      Integer n = new Integer(a[1]);
      System.out.println(10 / n);
    } catch (ArithmeticException e) {
      if (a.length > 0)
        System.out.println("Il secondo argomento deve essere un numero diverso da 0");
      else
        System.out.println("Sono richiesti almeno due argomenti");
    } catch (NumberFormatException e) {
      System.out.println("Il secondo argomento deve essere un numero");
    } catch (ArrayIndexOutOfBoundsException e) {
      System.out.println("Sono richiesti almeno due argomenti");
    }
  }
}
```

Esempio. Visualizza sul monitor il contenuto di un file di caratteri, il cui nome viene specificato sulla riga di comando.

```
import java.io.FileReader;
import java.io.IOException;

class VisualizzaFile {

  public static void main(String[] args) throws IOException {
    //costruzione dello stream di caratteri
    FileReader frd = new FileReader(args[0]);

    int i;
    //lettura e visualizzazione
    while ((i = frd.read()) != -1)
      System.out.print((char)i);

    //chiusura dello stream
    frd.close();
  }
}
```

Note
Il costruttore di `FileReader` può sollevare una `FileNotFoundException` (sottoclasse di `IOException`). Il metodo `read` e il metodo `close` possono sollevare una `IOException`. Poiché si tratta di *eccezioni controllate*, `main` deve intercettarle con un costrutto `try-catch` oppure delegarle esplicitamente al chiamante. Qui si è scelta la seconda alternativa: la delega esplicita al chiamante è espressa da `throws IOException` alla fine dell'intestazione di `main`.

Esercizio 12.3
Modificate l'applicazione `VisualizzaFile` in modo che nel caso il file non esista o vi siano problemi durante la lettura, visualizzi un messaggio di errore e termini. A tale scopo intercettate opportunamente le eccezioni mediante il costrutto `try-catch`