Azzolini Riccardo 2018-11-23

Eccezioni

1 Eccezioni

Durante l'esecuzione di un programma, possono verificarsi varie anomalie (ad esempio: divisione per zero, accesso a una posizione non esistente di un array, ecc.).

Spesso è possibile prevenire tali anomalie facendo prima dei controlli, ma questa non è sempre la soluzione migliore:

- il punto in cui si verifica l'anomalia può essere lontano dal punti in cui intervenire per porvi rimedio
- l'anomalia può essere dovuta a eventi esterni e, quindi, impossibile da prevenire

Per questo Java mette a disposizione il meccanismo delle eccezioni. Un'**eccezione** è un oggetto (istanza di un'apposita classe) usato per segnalare il verificarsi di un'anomalia: quando questa si verifica, durante l'esecuzione di un metodo o di un costruttore, viene **sollevata** un'eccezione, creando un'istanza di una classe che descrive l'evento anomalo. Se possibile, il metodo/costruttore la **intercetta** e **gestisce**, altrimenti la sua esecuzione si interrompe e l'eccezione viene **rinviata** al metodo chiamante, e così via fino a raggiungere, eventualmente, la JVM, che ferma il programma e stampa un messaggio di errore.

2 Gerarchia delle eccezioni

Le classi delle eccezioni sono organizzate in modo gerarchico:

```
Object
  Throwable
    Error
      NoClassDefFoundError
      OutOfMemoryError
      ecc.
    Exception
      FileNotFoundException
      IOException
      ecc.
      RuntimeException
        ArithmeticException
        IndexOutOfBoundsException
          ArrayIndexOutOfBoundsException
          StringIndexOutOfBoundsException
        ecc.
```

3 Intercettazione delle eccezioni

Per intercettare le eccezioni si utilizza il costrutto `try-catch`:

```
try {
    blocco_try;
} catch (tipo_eccezione_1 id1) {
    blocco_gestore_1;
} catch (tipo_eccezione_2 id2) {
    blocco_gestore_2;
} ...
```

La JVM esegue innanzitutto il blocco `try`:

- se non vengono sollevate eccezioni, l'esecuzione passa direttamente all'istruzione successiva al costrutto
- altrimenti, appena si verifica un'anomalia, interrompe l'esecuzione del blocco `try` e scorre la lista degli argomenti dei blocchi `catch` nell'ordine in cui sono scritti, cercando un tipo uguale o supertipo di quello dell'eccezione sollevata
  - se lo trova, esegue il blocco `catch` corrispondente, quindi l'esecuzione prosegue dall'istruzione successiva al costrutto
  - altrimenti, l'eccezione viene rinviata al chiamante (come se il costrutto `try-catch` non ci fosse)

All'interno di un blocco `catch`, è possibile accedere all'oggetto eccezione tramite l'identificatore appositamente dichiarato nell'intestazione del blocco.

3.1 Esempio

```
String s = in.readLine("Inserisci una stringa ");
int indice = in.readInt("Inserisci una posizione ");
try {
    char x = s.charAt(indice);
    out.println("Il carattere in posizione " + indice +
                " della stringa " + s + " è " + x);
} catch (StringIndexOutOfBoundsException e) {
    out.println(e.toString());
}
```

4 Clausola `finally`

Il costrutto `try-catch` prevede anche una clausola facoltativa, `finally`:

```
try {
    blocco_try;
} catch (...) {
    ...
} finally {
    blocco_finally;
}
```

Le istruzioni contenute nel `blocco_finally` vengono eseguite indipendentemente dal sollevamento di eccezioni (e dal fatto che vengano intercettate o rinviate al chiamante), e anche nel caso in cui l'esecuzione del blocco try sia interrotta da istruzioni come `return` o `break`.

5 Classificazione delle eccezioni

Le eccezioni di Java si suddividono in due categorie:

non controllate: istanze di `RuntimeException` (e di `Error`)

controllate: istanze di `Exception`, ma non di `RuntimeException`

Se nel corpo di un metodo/costruttore possono essere sollevate eccezioni controllate, il compilatore verifica che esse siano **trattate esplicitamente**, cioè:

- intercettate nel metodo/costruttore stesso
- **delegate esplicitamente** al chiamante, specificando nell'intestazione del costruttore/metodo i tipi di eccezioni delegate mediante la parola chiave `throws`

Al contrario, per le eccezioni non controllate non vengono effettuati controlli: quando esse non sono intercettate, vengono delegate automaticamente al chiamante.

5.1 Scopo

- Le eccezioni controllate sono usate per modellare anomalie legate a *eventi esterni* al programma, che il programmatore deve quindi essere obbligato a trattare.
- Le eccezioni non controllate modellano invece anomalie dovute a *eventi interni*, che si potrebbero evitare scrivendo il codice in modo diverso.
- Le istanze di `Error` (anch'esse non controllate), infine, rappresentano in genere situazioni irreparabili, come ad esempio l'esaurimento della memoria disponibile.

6 Sollevamento di eccezioni

Per **sollevare esplicitamente** un'eccezione si utilizza l'istruzione `throw`. Essa deve essere seguita da un riferimento a un'eccezione (solitamente si scrive direttamente un'espressione di creazione):

```
classe_eccezione riferimento = new classe_eccezione(argomenti);
throw riferimento;
// oppure
throw new classe_eccezione(argomenti);
```

Quando viene eseguita, quest'istruzione solleva l'eccezione, provocando quindi la terminazione anomala del metodo.