Insegnamento di Laboratorio di algoritmi e strutture dati

Le eccezioni in Java

Roberto Posenato

ver. 0.6, 31/01/2008

Si comincia...

Le eccezioni in Java

Esigenza...

- Solitamente, lo sviluppo di un algoritmo è focalizzato sui dati e sulle condizioni proprie del problema che si vuole risolvere.
- Condizioni particolari quali errori di I/O, accesso fuori range array, ecc. non sono considerate perché non sempre è possibile gestirle dall'interno del metodo.
- Dall'altra parte però queste condizioni devono essere gestite in qualche modo per garantire un comportamento certo del sistema.
- È "desiderabile" quindi un meccanismo che permetta di gestire queste condizioni speciali in modo semplice.
- A tal fine Java prevede il meccanismo delle eccezioni (exception).

Le eccezioni in Java

Definizione

Definizione 1 (Eccezione)

Un'eccezione è un oggetto che è creato dalla JVM quando si verifica una situazione anomala durante l'esecuzione di un metodo. L'oggetto eccezione contiene le informazioni circa il problema che ha generato la situazione anomala.

Le eccezioni in Java

Uso

Un'eccezione può essere "creata" e segnalata o può "capitare" e quindi deve essere gestita.

Creazione Se si verifica una condizione di errore, si può creare un oggetto eccezione che rappresenta l'errore e segnalare al sistema tale oggetto. In gergo, questa fase è detta lanciare o sollevare (`thrown`) un'eccezione.

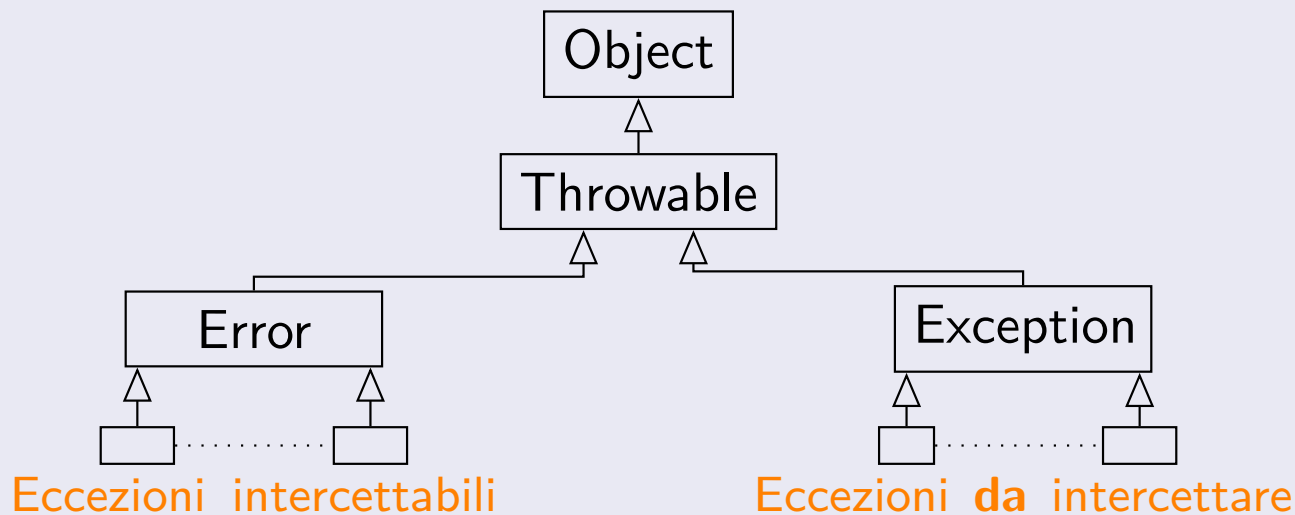
Gestione Se viene attivata un'eccezione, si può (o si deve, come si vedrà più avanti) gestirla in modo da correggere la situazione anomala.
In gergo, questa fase è detta cattura o intercettazione (`catch`) delle eccezioni.

Misto Si può anche intercettare un'eccezione, gestirla parzialmente e sollevare una nuova eccezione diversa dall'originale.

Le eccezioni in Java

Tipi di eccezioni

- L'eccezione è un oggetto. Quindi ha un tipo.
- La superclasse che rappresenta le eccezioni è `Throwable`.
- `Throwable` ha 2 sottoclassi: `Error`, che rappresenta le eccezioni che non ci aspetta di dover gestire e `Exception` che rappresenta le eccezioni che si deve prevedere di gestire.
- Tutte le eccezioni devono essere sottoclassi di una di queste due classi.



Le eccezioni in Java

Eccezioni di tipo Error

- Le eccezioni di tipo o sottotipo di `Error` dovrebbero rappresentare condizioni non prevedibili e **difficilmente gestibili** all'interno del metodo.
- Ci sono 3 sottoclassi dirette: `ThreadDeath`, `LinkageError` e `VirtualMachineError`.
- A titolo di esempio, `VirtualMachineError` rappresenta il verificarsi di errori nella Java Virtual Machine che bloccano il suo funzionamento.
- Si intuisce che difficilmente si può pretendere che un metodo gestisca questi tipi di errore.
- Per queste eccezioni infatti non è richiesto nulla: si possono ignorare!
- Se si verificano, il metodo termina con un codice di errore rappresentato dall'eccezione.

Le eccezioni in Java

Eccezioni di tipo Exception

- Le eccezioni di tipo o sottotipo di `Exception` dovrebbero rappresentare condizioni di errori del metodo che **devono essere *gestite***.
- Se non si gestisce un'eccezione di tipo `Exception`, il metodo non è compilabile... a meno che non sia del sottotipo `RuntimeException`!
- Il sottotipo `RuntimeException` rappresenta errori di runtime di metodo (non di JVM) che possono essere evitate a priori e che appesantirebbero troppo il codice se si dovessero gestire come eccezioni.
  Esempio: errore del tipo `NullPointerException`, che rappresenta *oggetto nullo non previsto*.
- È ammesso quindi che un'eccezione `RuntimeException` possa essere non gestita. Se si verifica, il metodo termina.

Le eccezioni in Java

Eccezioni di tipo RuntimeException

- Sono un sottotipo speciale di `Exception` in quanto si possono anche non gestire.
- Rappresentano condizioni di errori dei programmi che possono richiedere interventi esterni al metodo dove si verificano.
- Casi tipici: divisione per 0, indice fuori dai limiti dell'array, riferimento nullo a oggetto, ...
- Questi casi tipici sono già rappresentati nel package come sottotipi di `RuntimeException`.

Esercizio 1

Imparare cosa rappresentano le eccezioni `ArithmeticException`, `IndexOutOfBoundsException`, `NullPointerException`, `IllegalArgumentException`, `IllegalStateException` e `UnsupportedOperationException` del package `java.lang`.

Le eccezioni in Java

Sollevare eccezioni

- In generale, se all'interno di un metodo si verifica uno **stato di errore** di qualche genere, può essere conveniente interrompere l'esecuzione del metodo e *sollevare* l'eccezione.
- Il sollevare un'eccezione deve avvenire **solo** quando non è possibile gestire un modo corretto e/o efficiente l'errore.
- La sequenza di istruzioni per sollevare un'eccezione è:
  1. Dichiarazione che il metodo può sollevare eccezioni mediante l'istruzione `throws` nella dichiarazione del metodo;
  2. In caso di condizione di errore:
     1. Istanziare un oggetto di tipo eccezione appropriato;
     2. Sollevare mediante l'istruzione `throw`.

Le eccezioni in Java

Sollevare eccezioni

Esempio 1 (Dichiarazione di metodo che può sollevare un'eccezione)

```
/**
 * Esempio metodo con lancio di eccezione.
 */
void myMethod(List l) throws NullPointerException {
    if ( l == null ) {
        NullPointerException e =
          new NullPointerException(
            "Il parametro l non può essere nullo."
          );
        throw e;
    }
    ...
}
```

`java.lang.NullPointerException` è un sottotipo di `RuntimeException`: segnala quando un oggetto è nullo mentre non dovrebbe esserlo.

Le eccezioni in Java

Sollevare eccezioni

- Il `throws` nella dichiarazione può accettare un elenco di tipo di eccezioni, separati da ',':
  `throws IOException, FileNotFoundException, ...`
- Se si esegue l'istruzione `throw e`, il metodo viene interrotto e il controllo torna al metodo che ha invocato `myMethod()` con la segnalazione di eccezione.

Le eccezioni in Java

Intercettare le eccezioni

- Se un metodo che viene invocato può sollevare un'eccezione diversa da `Error` o `RuntimeException`, questa deve essere gestita (intercettata).
- Ci sono 2 possibilità:
  1. O si dichiara che il metodo dove l'eccezione può verificarsi può *sollevare* a sua volta l'eccezione;
  2. O si inserisce il codice necessario per intercettare e risolvere l'eccezione nel caso si verifichi.

Nota!

Il punto 1 equivale a dichiarare che è noto il fatto che si può determinare un'eccezione ma non si è in grado o non si vuole gestirla. Si *delega* quindi a chi invoca il metodo la gestione della stessa.

Le eccezioni in Java

Intercettare le eccezioni: delega

- Si supponga di avere un metodo `myMethod()` dentro il quale si invoca un metodo che può sollevare l'eccezione `IOException`: Esempio: `myMethod()` contiene

  ```
  File = new File("nuovo").createNewFile();
  ```

- L'eccezione `IOException` indica che si è verificato un errore di I/O. Non può essere ignorata.
- Se non si vuole gestire un errore di tale tipo dentro `myMethod()`, si può **delegare** la gestione a chi invocherà il metodo `myMethod()`.
- La delega avviene completando la segnatura del metodo `myMethod()` con l'istruzione **`throws`**.

Le eccezioni in Java

Intercettare le eccezioni: delega

Esempio 2 (Dichiarazione di metodo che delega un'eccezione)

```
/**
 * Esempio metodo con delega di eccezioni.
 */
void myMethod() throws IOException {
    ...
    File = new File("nuovo").createNewFile();
    ...
}
```

Le eccezioni in Java

Intercettare le eccezioni: delega

- Se durante l'esecuzione della chiamata `createNewFile();` all'interno del metodo `myMethod()` si verifica l'eccezione `IOException`,
  1. Il metodo `createNewFile()` istanzia un oggetto di tipo `IOException` con il dettaglio;
  2. "Lancia" l'oggetto all'interprete;
  3. L'interprete ritorna al metodo `myMethod()` segnalando l'eccezione;
  4. Dato che questo delega la gestione, l'inteprete interrompe l'esecuzione di `myMethod()` e ritorna al metodo che ha invocato `myMethod()` segnalando l'eccezione;
  5. Questa ritorno continua fino a quando o l'interprete trova un metodo in grado di gestire l'eccezione o esaurisce lo stack delle chiamate.
  6. Nel secondo caso, l'interprete termina l'esecuzione del programma segnalando l'errore sullo `stderr`.

Le eccezioni in Java

Intercettare le eccezioni: intercettazione

- Si supponga di volere invece gestire (intercettare) un'eventuale eccezione che può accadere.
- Si deve allora inserire nel metodo almeno 2 (3 per completezza) istruzioni nel seguente ordine:
  1. Un'istruzione `try {...}` che racchiude le istruzioni che possono sollevare l'eccezione;
  2. Un'istruzione `catch (...) {...}`, da mettere subito dopo il try, che intercetta l'oggetto eccezione specificato tra () e che esegue, in caso di intercettazione, le istruzioni racchiuse tra {};
  3. Un'eventuale istruzione `finally {...}` che racchiude tra {} le istruzioni che si devono eseguire sempre: sia se si è verificata un'eccezione sia che non si sia verificata.

Le eccezioni in Java

Intercettare le eccezioni: intercettazione 1

Esempio 3 (Metodo che intercetta un'eccezione (I versione))

```
/**
 * Esempio metodo che intercetta IOException.
 */
void myMethod() { //Tolto throws!
    ...
    try {  //Codice che può lanciare una o + eccezioni!
        File = new File("nuovo").createNewFile();
    }
    catch (IOException e) {
        //Codice che gestisce eccezione IOException
        System.out.println(
          "Non si può creare il file 'nuovo'."
        );
    }
    ...
}
```

Le eccezioni in Java

Intercettare le eccezioni: intercettazione 1

- L'interprete esegue il codice dentro il blocco `try {}` normalmente;
- Se si verifica un'eccezione:
  - Si interrompe l'esecuzione;
  - Se esiste un blocco `catch {}` corrispondente all'eccezione, si esegue il blocco e poi prosegue con il codice presente DOPO il blocco `catch {}`.
  - Se non esiste un blocco `catch {}` corrispondente, (l'eccezione DEVE essere di tipo Error o RuntimeException), si esegue un return dal metodo segnalando l'eccezione al metodo chiamante;
- Se NON si verifica un'eccezione: dopo aver eseguito il codice dentro il `try {}`, si prosegue con il codice eventualmente presente dopo il blocco `catch {}`.

Le eccezioni in Java

Intercettare le eccezioni: intercettazione 2

Esempio 4 (Metodo che intercetta più eccezioni)

```
void myMethod(int k) {
    try {  //Codice che può lanciare una o + eccezioni!
        File = new File("nuovo").createNewFile();
        int i = 100 / k;
    }
    catch (IOException e) {
        System.out.println(
          "Non si può creare il file 'nuovo'!"
        );
    }
    catch (ArithmeticException e1) {
        System.out.println(
          "Non si può dividere per 0!"
        );
    }
    ...
}
```

Le eccezioni in Java

Intercettare le eccezioni: finally

- Il blocco `finally {}` è utile per garantire che un certo insieme di istruzione venga eseguito a prescindere dalle eccezioni.
  Esempio: se si apre un file e si eseguono delle operazioni, nel finally si può inserire il codice di chiusura del file.

Le eccezioni in Java

Intercettare le eccezioni: intercettazione 3

Esempio 5 (Metodo che intercetta un'eccezione e esegue il finally)

```
void myMethod(int k) {
    try {  //Codice che può lanciare una o + eccezioni!
        File = new File("nuovo").createNewFile();
        int i = 100 / k;
    }
    catch (IOException e) {
        System.out.println(
          "Non si può creare il file 'nuovo'!"
        );
    }
    finally {
        System.out.println(
          "Siamo giunti alla fine!"
        );
    }
    ...
}
```

Le eccezioni in Java

Intercettare le eccezioni: misto

- Talvolta è opportuno intercettare un'eccezione, gestirla parzialmente e poi risollevare una nuova eccezione.
- Il risollevare dell'eccezione avviene come il sollevare standard. Ricordarsi di dichiarare che il metodo solleva la nuova eccezione!

Le eccezioni in Java

Intercettare le eccezioni: intercettazione 4

Esempio 6 (Metodo che intercetta un'eccezione e risolleva)

```
void myMethod(int k) throws IllegalArgumentException {
    try {
         int i = 100 / k;
    }
    catch (ArithmeticException e) {
        IllegalArgumentException e1 =
          new IllegalArgumentException(
           "Il parametro deve essere diverso da 0!"
          );
        throw e1;
    }
    ...
}
```

Le eccezioni in Java
Conclusioni

In questa lezione:

- Si è introdotto il concetto di eccezione.
- Si è introdotto come le eccezioni sono gestite nel linguaggio Java.
- A completamento delle nozioni inerenti alla gestione delle eccezioni, si rimanda al capitolo 10 del libro Dai fondamenti agli oggetti di Pighizzini & Ferrari.
- Si consiglia di studiare il capitolo e di svolgere gli esercizi in quanto saranno oggetto di esame.