ManpowerGroup™

Eccezioni

Programmazione Java - Modulo 9

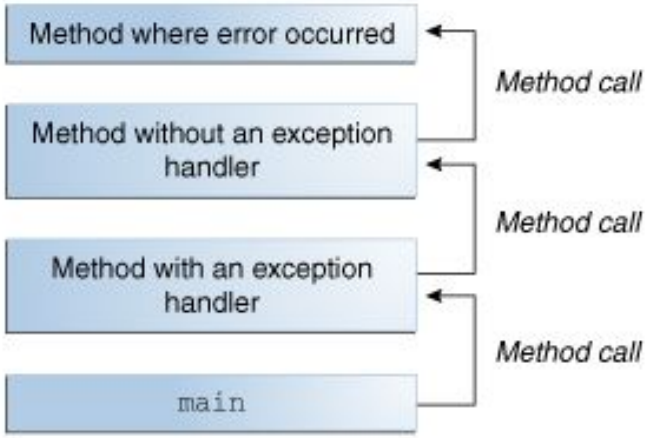
Nicola Atzei

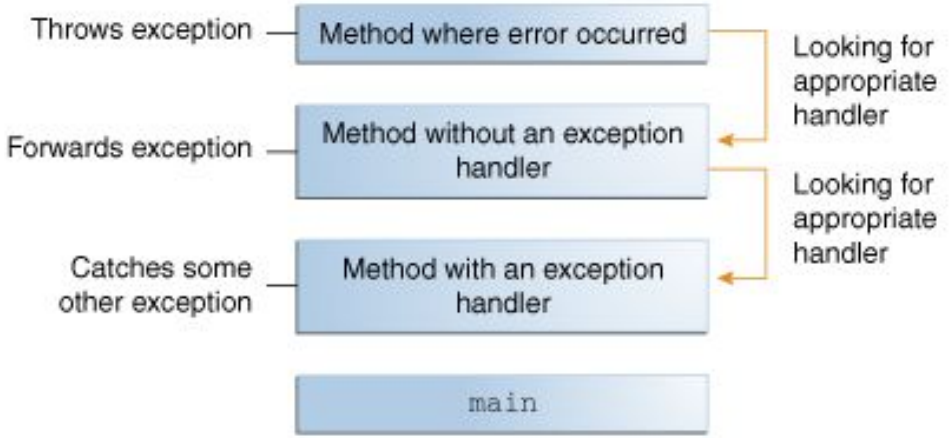
Eccezioni

Un'eccezione è un evento che occorre durante l'esecuzione di un programma.

Le eccezioni possono essere lanciate (**throw**) dai metodi e si propagano nei metodi chiamanti.

È possibile catturare (**catch**) un'eccezione e fare qualcosa in risposta a questo evento (**exception handling**).





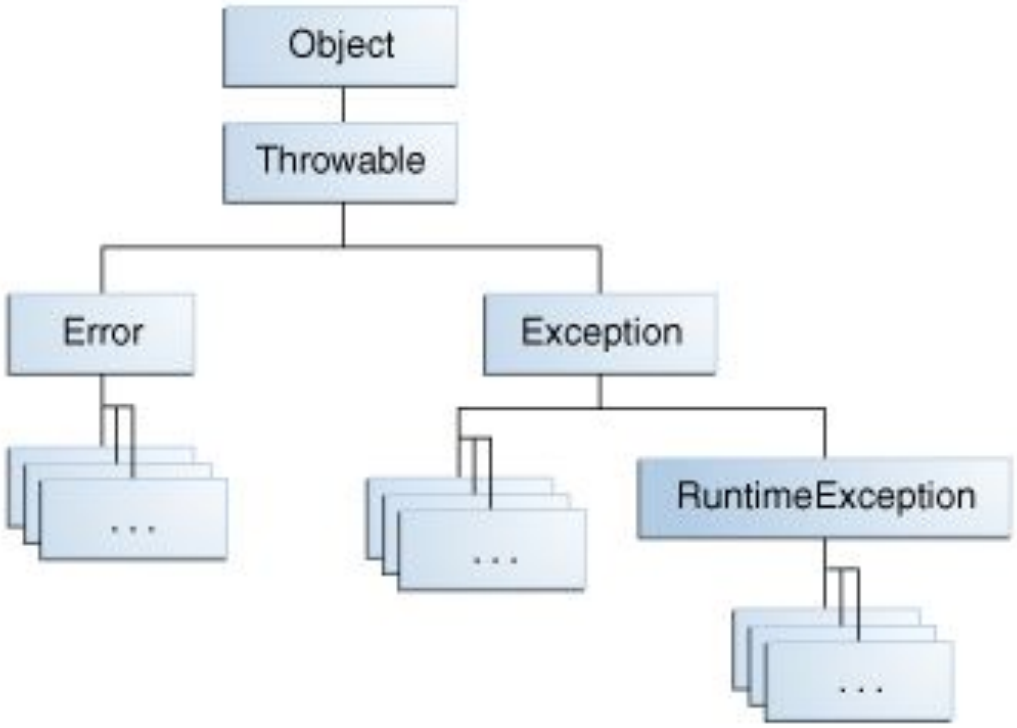
Young
Talent in Action
#crediamoneltalentodelfuturo

Tipi di eccezioni

- checked exceptions
  - eccezioni che andrebbero gestite dall'applicazione (exception handling)
  - sottoclasse di `java.lang.Exception`
- unchecked exceptions
  - errors
    - errori irreparabili **non imputabili all'applicazione** (esterni). In linea di massima **non sono prevedibili** né è possibile fare qualcosa per porre rimedio
    - sottoclasse di `java.lang.Error`
  - runtime exceptions
    - errori irreparabili **imputabili all'applicazione** (interni). Indicano generalmente un bug nell'applicazione e non dovrebbero essere gestite
    - sottoclasse di `java.lang.RuntimeException`

Young Talent in Action
#crediamoneltalentodelfuturo

Gerarchia delle classi



Exception

```
public class Exception extends Throwable {

    public Exception() {
        super();
    }

    public Exception(String message) {
        super(message);
    }

    public Exception(String message, Throwable cause) {
        super(message, cause);
    }

    public Exception(Throwable cause) {
        super(cause);
    }

}
```

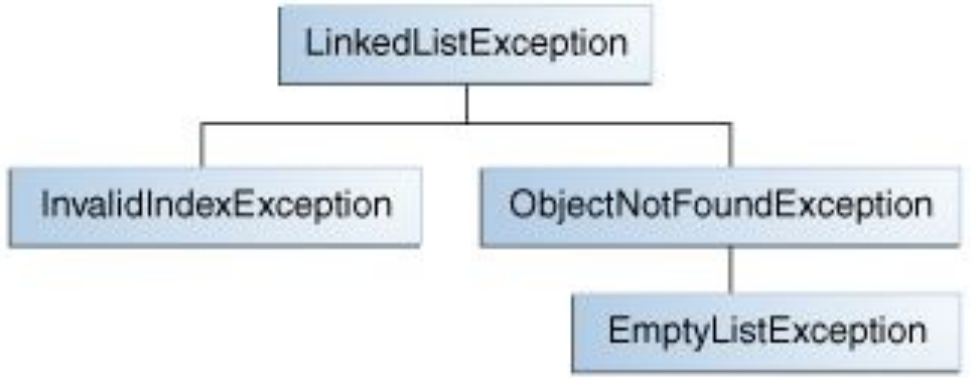
RuntimeException

```
public class RuntimeException extends Exception {

    public RuntimeException() {
        super();
    }

    public RuntimeException(String message) {
        super(message);
    }

    public RuntimeException(String message, Throwable cause) {
        super(message, cause);
    }

    public RuntimeException(Throwable cause) {
        super(cause);
    }

}
```

Young Talent in Action
#crediamoneltalentodelfuturo

try - catch - finally (1/2)

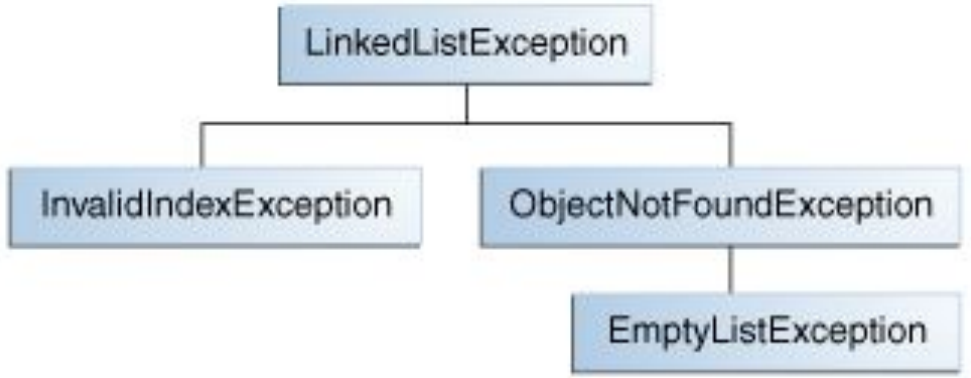
```
try {
   // some code
}
catch (InvalidIndexException | EmptyListException e) {
   // handle
}
catch (LinkedListException e) {
   // handle
}
finally {
   // always executed
}
```



Young Talent in Action
#crediamoneltalentodelfuturo

try - catch - finally (1/2)

```
try {
   // some code
}
catch (LinkedListException e) {
   // handle
}
catch (InvalidIndexException | EmptyListException e) {
   // handle
}
finally {
   // always executed
}
```



Young Talent in Action
#crediamoneltalentodelfuturo

Gestione delle eccezioni

```
public class IndexOutOfBoundException extends RuntimeException {}

public class WriteException extends Exception {}

public class List {

    public int get(int i) {
        ... throw new IndexOutOfBoundException();
    }

    public void writeToFile() throws WriteException { ... }
}
```

```
List l = new List();
int n = l.get(5);
l.writeToFile();
```

Compila? Perché?

```
List l = new List();
int n = l.get(5);
try {
    l.writeToFile();
} catch (WriteException e) {...}
```

Young Talent in Action
#crediamoneltalentodelfuturo

Finally

Il blocco finally viene **sempre** eseguito. SEMPRE!

Viene eseguito anche se il blocco try contiene **return**, **break**, **continue**.

```
try {
    ...
    return true;
}
finally {
    // chiusura di stream aperti
}
```

ATTENZIONE
il blocco finally non dovrebbe contenere istruzioni di ritorno né modificare l'oggetto eventualmente ritornato nel blocco try

Young Talent in Action
#crediamoneltalentodelfuturo

Finally - cose da NON fare

```
try {
   return 0;
}
finally {
   return 1;
}
```

```
Persona p = new Persona();
p.nome = "Gianni";
try {
   return p;
}
finally {
   p.nome = "Salvatore";
}
```

Young
Talent in Action
#crediamoneltalentodelfuturo

Un metodo deve dichiarare **sempre** le Exception che potrebbe lanciare

Mentre le RuntimeException si possono omettere

Young Talent in Action
#crediamoneltalentodelfuturo

Esempio

```
public class PersonNotFoundException extends Exception {}


/**
 * @throws PersonNotFoundException
 *     quando non trova la persona corrispondente
 */
public Persona getByName(String nome) throws PersonNotFoundException {
   if (! found)
      throw new PersonNotFoundException();

   return p;
}
```

Esempio

```
public class PersonNotFoundException extends RuntimeException {}


...
public Persona getByName(String nome) {
    if (! found)
        throw new PersonNotFoundException();

    return p;
}
```

Young Talent in Action
#crediamoneltalentodelfuturo

Esempio

```
public class PersonNotFoundException extends RuntimeException {}


/**
 * @throws PersonNotFoundException
 *     quando non trova la persona corrispondente
 */
public Persona getByName(String nome) throws PersonNotFoundException {
   if (! found)
      throw new PersonNotFoundException();

   return p;
}
```

Young
Talent in Action
#crediamoneltalentodelfuturo

Faccio una RuntimeException o una Exception?

Exception vs RuntimeException

Il compilatore segnala eventuali *checked exception* (derivate da Exception) che il programmatore ha dimenticato di gestire

- Semplice, faccio solo RuntimeException!
- **NO**

Le eccezioni (checked) che un metodo può lanciare fanno parte della sua interfaccia d'uso, al pari del valore di ritorno. Il programmatore che utilizza il metodo de decide cosa fare.

- Perché non usare comunque RuntimeException se posso gestirle e documentarle nell'API?

Le eccezioni a runtime non sono state pensate per essere gestite. Sono sintomo di bug nel codice e vanno risolte diversamente. Inoltre, le eccezioni a runtime sono troppe per pensare di renderle parte dell'API

Young Talent in Action
#crediamoneltalentodelfuturo

Esercizio

Configurazione

Clonare il repository *https://github.com/civraxiu/GiovaniTalentiInAzione.git*

Importare in Eclipse il progetto *Anagrafica* del Modulo 9:

```
./Modulo 9 - Java nelle aziende/Progetto finale/Anagrafica
```

Eseguire il main e assicurarsi che funzioni correttamente:

- Run Configurations: il Main riceve in input un file per salvare e leggere l'anagrafica

Esercizio

Creare un nuovo package `it.ytia.anagrafica.logica.exceptions`

Creare una nuova classe `InitializationException`

- estende `RuntimeException`
- viene lanciata dal metodo `initFromFile()` in caso di errori nell'inizializzazione, al posto di `throw new RuntimeException(...)`
- oltre al costruttore di default, ha un costruttore che riceve in input un messaggio di errore che viene stampato quando l'eccezione viene lanciata
- se l'inizializzazione fallisce a causa di un'altra eccezione, quest'ultima dev'essere passata nel costruttore di `InitializationException` come **causa** dell'eccezione (un altro costruttore)

Young Talent in Action
#crediamoneltalentodelfuturo

Esercizio

Creare una nuova classe `SaveException` analoga a `InitializationException`

- estende `RuntimeException`
- viene lanciata dal metodo `salvaAnagrafica()` in caso di errori nel salvataggio, al posto di `throw new RuntimeException(...)`
- oltre al costruttore di default, ha un costruttore che riceve in input un messaggio di errore che viene stampato quando l'eccezione viene lanciata
- se il salvataggio fallisce a causa di un'altra eccezione, quest'ultima dev'essere passata nel costruttore di `SaveException` come **causa** dell'eccezione (un altro costruttore)

Young
Talent in Action
#crediamoneltalentodelfuturo

Esercizio

Validazione dei parametri nelle classe Persona. I metodi `setNome(...)` e `setCognome(...)` lanciano l'eccezione `java.lang.IllegalArgumentException` se:

- viene passato **null** come argomento
- viene passata una stringa vuota
- viene passata una stringa contenente solo spazi/tab (suggerimento: utilizzare il metodo `trim()` della classe `String`)

I controlli vanno fatti anche in fase di costruzione dell'oggetto: usare i setter all'interno del costruttore.

Young
Talent in Action
#crediamoneltalentodelfuturo

Esercizio

Estendere i controlli anche ai setter della classe `DataDiNascita`:

- un giorno dev'essere compreso tra 1 e 31
- un mese dev'essere compreso tra 1 e 12
- un anno dev'essere maggiore di 0

Young
Talent in Action
#crediamoneltalentodelfuturo

Reference

https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html