Le eccezioni in Java

Programmazione
Corso di laurea in Informatica

Eccezioni

- L'istruzione `try/catch`
- La propagazione dell'eccezioni
- Intercettare e gestire eccezioni

Eccezioni in breve

- Un'***eccezione*** è un oggetto che descrive una situazione anomala o di errore
- L'eccezioni vengono ***lanciate*** da una parte di un programma e possono essere ***raccolte*** e ***gestite*** da altre parti del programma
- Un programma può perciò essere suddiviso nel normale flusso d'esecuzione e in quello eccezionale
- Anche un ***errore*** è rappresentato come un oggetto Java, ma solitamente rappresenta una situazione non recuperabile e da non gestire

Gestire l'eccezioni

- Java ha un insieme predefinito di eccezioni ed errori che possono accadere durante l'esecuzione di un programma
- 3 modi di gestire l'eccezioni:
  - Ignorarle
  - Gestirle quando avvengono
  - Gestirle altrove nel programma
- La scelta del modo di gestire gli eventi anomali o eccezionali è un'importante caratteristica del disegno del programma

Ignorare l'eccezioni

- Se un'eccezione è ignorata da un programma, questo terminerà producendo un messaggio opportuno
- Il messaggio mostra la traccia dello ***stack delle chiamate dei metodi*** con l'indicazione:
  - dell'errore
  - della linea in cui l'eccezione si è verificata
  - dellle chiamate di metodi che hanno portato all'eccezione

Esempi

- `Zero.java` può causare un'eccezione

```
java.lang.ArithmeticException: / by zero
  at Zero.calcolaQuoziente(Zero.java:27)
  at Zero.main(Zero.java:21)
Exception in thread "main" Process Exit...
```

- `BasicArray_eccezione.java` causa l'eccezione

```
java.lang.ArrayIndexOutOfBoundsException
  at
  BasicArray_eccezione.main(BasicArray_eccezione.
  java:30)
Exception in thread "main" Process Exit...
```

Esempi

- Postfissa.java può causare eccezioni

```
java.util.EmptyStackException
   at java.util.Stack.peek(Stack.java:82)
   at java.util.Stack.pop(Stack.java:64)
   at Postfissa.elabora(Postfissa.java:37)
   at Postfissa.main(Postfissa.java:21)
Exception in thread "main" Process

java.lang.NumberFormatException: 1=
   at java.lang.Integer.parseInt(Integer.java:423)
   at java.lang.Integer.<init>(Integer.java:549)
   at Postfissa.elabora(Postfissa.java:42)
   at Postfissa.main(Postfissa.java:21)
Exception in thread "main" Process Exit...
```


Gestire l'eccezioni

- Occorre processare l'eccezione quando accade, la linea di codice che lancia l'eccezione deve essere eseguita in un blocco `try`.
- Un blocco `try` è seguito da 1 o più clausole `catch`, che contengono il codice per gestire l'eccezione
- Ogni clausola `catch` è associata ad un tipo d'eccezione e viene chiamata ***exception handler***
- Quando si solleva un'eccezione, la computazione prosegue fino alla prima clausola `catch` che corrisponde al tipo d'eccezione sollevata

L'istruzione `try`

- Si tenta di eseguire il codice e se si intercetta un'eccezione si cerca di porre rimedio

```
try
{
    blocco_1
}
catch (tipo_eccezione identificatore)
{
    blocco_2
}
```

- L'istruzione `try` identifica un blocco d'istruzioni in cui può verificarsi un'eccezione

La clausola `catch`

- Un blocco `try` è seguito da una o più clausole `catch`, che specificano quali eccezioni vengono gestite
  - Ogni clausola `catch` corrisponde a un tipo di eccezione sollevata
- Quando si verifica un'eccezione, la computazione continua con la prima clausola che corrisponde all'eccezione sollevata

- Divisione.java modifica `Zero.java`
- Postfissa_e.java
- ProductCodes.java

La clausola `finally`

- Un'istruzione `try` può essere seguita da una clausola `finally` opzionale
- Le istruzioni della clausola `finally` vengono sempre eseguite:
  - Se non viene sollevata nessuna eccezione, vengono eseguite dopo che si è concluso il blocco `try`
  - Se si verifica un'eccezione, vengono eseguite dopo le istruzioni della clausola `catch` appropriata

Rientro dai metodi in caso d'eccezioni

- Normalmente se un metodo `main` richiama il `metodo_1` che richiama il `metodo_2` che richiama il `metodo_3`, il controllo passa dal `main` al `metodo_1` al `metodo_2` al `metodo_3` e quando questo si conclude ritorna al `metodo_2` che a conclusione lo passa al `metodo_1` e quindi al `main`
- Se si verifica un'eccezione durante l'esecuzione del `metodo_3` il controllo viene passato diversamente

Propagazione dell'eccezioni

- Se l'eccezione non viene intercettata e gestita dove si verifica, può ancora essere trattata a un livello più alto
- L'eccezioni si ***propagano*** attraverso la gerarchia delle chiamate di metodi finché non vengono intercettate e gestite
- `Propagation.java` con la classe `ExceptionScope.java`

L'istruzione `throw`

- Un programmatore può definire un'eccezione estendendo una classe
  - La classe `Exception` o una sua sottoclasse
- L'eccezioni vengono sollevate con l'istruzione `throw`
- Solitamente un'istruzione `throw` è inclusa in un'istruzione `if` che valuta una condizione per verificare se deve essere sollevata l'eccezione
- `CreatingExceptions.java` con l'eccezione `EccezioneFuoriIntervallo.java`
- `CreatingExceptions_2.java`
- `Postfissa_2.java` con l'eccezione `PostfissaException.java`

Classificazione dell'eccezioni

- Le eccezioni possono essere **controllate**
  - Dovute a eventi esterni al programma
    - Cercare di accedere a una pagina web inesistente
    - Cercare una funzione di libreria che manca
  - Si chiamano controllate perché il compilatore controlla che vengano esplicitamente indicate e intercettate
- O **non controllate**
  - Dovute al programma e che potrebbero essere evitate

Eccezioni controllate

- Un'eccezione controllata deve essere raccolta da un metodo in una clausola `catch` e deve essere nella lista delle clausole `throws` di ciascun metodo che possa lanciare l'eccezione o propagarla
- La clausola `throws` deve essere dichiarata nell'intestazione del metodo
- Il compilatore segnala se un'eccezione controllata non viene gestita propriamente

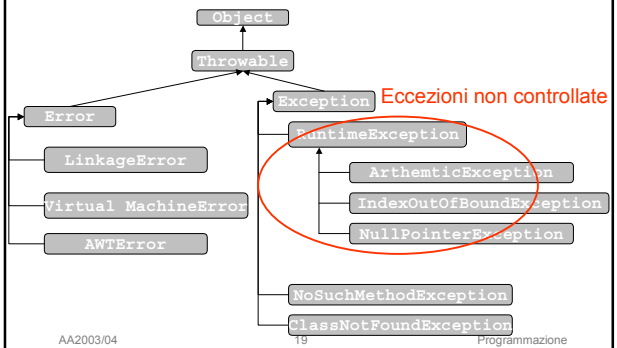
Eccezioni non controllate

- Non richiedono una gestione esplicita con la clausola `throws`
- L'eccezioni non controllate in Java sono quelle che si verificano a run time
- Discendono da `RuntimeException` o da una sua classe discendente
- Tutte le altre sono controllate

Errori

- Gli errori sono simili alle eccezioni `RuntimeException` o ai suoi discendenti
  - Gli errori non devono essere controllati
  - Gli errori non richiedono una clausola `throws`

La gerarchia di classe delle eccezioni



Nuove definizioni d'eccezione

- Tutte le nuove classi che estendono la gerarchia precedente o
  - Discendono da `RuntimeException` e quindi non sono controllate o
  - Disecendono da `Exception` e quindi sono controllate
    - I metodi che le lanciano dovranno dichiararlo nell'intestazione con la clausola `throws`
    - Un metodo che può lanciare un'eccezione controllata dovrà dichiararlo

Propagazione e gestione dell'eccezioni controllate

- Un metodo che può sollevare un eccezione controllata deve dichiararlo con la clausola `throws`
- A sua volta un metodo che lo richiama deve intercettarla o dichiararla, cioè deve:
  - Gestire l'eccezione con la coppia `try-catch` o
  - Dichiarare a sua volta che potrà sollevare l'eccezione nella clausola `throws`