Programmazione in Java
e
gestione della grafica

Lezione 23

Eccezioni

- Eccezioni – indicazione che si verifica un problema durante l'esecuzione di un programma
- Gestione delle eccezioni – risolve le eccezioni che si verificano in modo che il programma può continuare o terminare in modo regolare
- Gestione delle eccezioni consente ai programmatori di creare programmi più robusti e fault-tolerant

Esempi

- `ArrayIndexOutOfBoundsException` – tentativo di accedere ad un elemento oltre l'ultimo di un array.
- `ClassCastException` – tentativo di cast di un oggetto che non ha relazione *è-un* con il tipo specificato nell'operatore cast
- `NullPointerException` – quando una referenza `null` è usata quando si aspetta un oggetto.

Gestione delle eccezioni

- Mescolare la logica del programma con la logica della gestione delle eccezioni rende i programmi difficili da leggere , modificare e mantenere
- La gestione delle eccezioni permette di eliminare la gestione degli errori dal “flusso principale” della esecuzione del programma.
- Aumenta la chiarezza
- Migliora la modificabilità

Esempio:
Divide By Zero senza Gestione delle eccezioni

- Sollevare una eccezione (Thrown exception)
  - si verifica una eccezione
- Traccia dello stack (Stack trace)
  - Nome della eccezione in un messaggio descrittivo che indica il problema
  - Stack completo di tute le chiamate ai metodi
- `ArithmeticException` – si verificano per vari problemi aritmetici
- Throw point – punto in cui si verifica l'eccezione,
- `InputMismatchException` – si verifica quando il metodo `nextInt` di `Scanner` riceve una stringa che non è un valore intero valido

Vediamo il programma

DivideByZeroNoExceptionHandling.java

Esempio:
Gestire ArithmeticExceptions e InputMismatchExceptions

- Con la gestione delle eccezioni, il programma rileva e gestisce (“catches” e “handles”) le eccezioni

- Rivediamo l’esempio in cui l’utente può inserire altri valori se ha inserito un input non valido (zero nel denominatore o un input non integer) input)

Inserire il codice in un blocco `try`

- blocco `try`– include il codice che può generare (throw) una eccezione e il codice che non deve essere eseguito se l'eccezione si verifica
- keyword `try` seguita da codice tra { }

Attenzione al fatto che ….

- Le eccezioni possono venire fuori dal codice scritto dentro il blocco `try` oppure attraverso chiamate ad altri metodi innestati chiamati dal blocco `try` o dalla Java Virtual Machine

…blocco catch

- – catches (i.e., riceve) e gestisce una eccezione
  - keyword `catch`
  - Parametro di tipo eccezione che identifica il tipo di eccezione e abilita il blocco `catch` ad interagire con l'eccezione.
  - Blocco di codice tra le parentesi { } viene eseguito quando si verifica proprio quella eccezione.
- Blocco `catch` corrisponde – se il tipo del paramentro eccezione corrisponde esattamente al tipo di eccezione sollevata o è una sua superclasse.
- Se nessun blocco `catch` corrisponde – l'eccezione non viene gestita
  - Il programma termina (se ha un solo thread)

- I blocchi `catch` devono seguire immediatamente il blocco `try` (no altro codice in mezzo)

Modello di terminazione per la gestione delle eccezioni

- Quando si verifica una eccezione:
  - blocco `try` termina immediatamente
  - Il controllo del programma si trasferisce al primo blocco `catch`
- Dopo la gestione della eccezione:
  - Il controllo del programma ritorna alla prima istruzione dopo l'ultimo blocco `catch` . ("termination model" per la gestione delle eccezioni)
  - Esistono in altri linguaggi altri modi di gestione delle eccezioni per esempio il "Resumption model" in cui il controllo ritorna proprio al throw point

- Con la gestione delle eccezioni un programma continua l’esecuzione (invece che terminare) dopo aver gestito il problema.

Attenzione!

- Dopo la gestione di una eccezione il controllo NON ritorna alla prima istruzione dopo throw point.

La clausola `throws`

- specifica le eccezioni che un metodo può sollevare
  - Si trova dopo la lista dei parametri di un metodo e prima del corpo del metodo.
  - Contiene una lista di eccezioni separate da virgola che il metodo solleva se si verifica un problema
  - Le eccezioni possono essere sollevate da istruzioni nel metodo o da metodi chiamati dal metodo
  - Le eccezioni possono essere del tipo elencato nella clausola `throws` o di sottoclassi.

- Nella documentazione dei metodi nelle API, sono specificate le eccezioni sollevate dai metodi e i motivi del perchè queste eccezioni sono sollevate in modo che poi possono essere gestite nei programmi.

Vediamo il programma

DivideByZeroWithExceptionHandling.java

Gerarchia Eccezioni Java

- Tutte le eccezioni ereditano direttamente o indirettamente dalla classe `Exception`
- Le classi Exception formano una gerarchia di ereditarietà che può essere estesa
- Classe `Throwable` è una superclasse di `Exception`
  - Soltanto oggetti `Throwable` possono essere inclusi nel meccanismo di gestione delle eccezioni
  - Ha due subclassi: `Exception` e `Error`
    - Classe `Exception` e le sottoclassi rappresentano le situazioni di eccezioni che si verificano nei programmi Java
    - Classe `Error` e le sottoclassi rappresentano le situazioni anormali che si verificano nei programmi nella JVM – di solito non è possibile recuperare il programma da una situazione di `Error`

blocco finally

- Se non si verificano eccezioni il blocchi catch vengono saltati e il controllo va sul blocco finally opzionale dopo I blocchi catch
- Dopo l'esecuzione del blocco finally il controllo si sposta sulla prima istruzione dopo il blocco finally
- Se si verifica una eccezione in un blocco, il programma salta tutto il resto del try. Cerca il blocco catch che corrisponde al tipo di eccezione e poi esegue il blocco finally.
- Se si verifica una eccezione in blocco try e non ci sono blocchi catch adatti a quella eccezione, il controllo va comunque al blocco finally block. After the finally
- Anche se si verifica una eccezione nel blocco catch il blocco finally viene comuqnue eseguito.

- Due categorie di eccezioni: checked eunchecked
- Eccezioni Checked (controllate)
  - Exceptions derivate da classe `Exception` ma NON da `RuntimeException`
  - Il compilatore impone che siano ”gestite”
  - Il compilatore controlla che ogni metodo gestisca le eccezioni controllate ed eventualmente da errore di compilazione
- Eccezioni Unchecked (non controllate)
  - Derivate dalla classe `RuntimeException` o da `Error`
  - Il compilatore non controlla se l’eccezione è sollevata o dichiarata
  - Se si verifica e non è sollevata il programma termina con errore
  - Potrebbero essere evitate con codice ad hoc