Gestione delle Eccezioni

Condizioni di Errore

Una condizione di errore in un programma può avere molte cause

- Errori di programmazione
  - Divisione per zero, cast non permesso, accesso oltre i limiti di un array, …
- Errori di sistema
  - Disco rotto, connessione remota chiusa, memoria non disponibile, …
- Errori di utilizzo
  - Input non corretti, tentativo di lavorare su file inesistente, …

Condizioni di Errore in java

- Java ha una gerarchia di classi per rappresentare le varie tipologie di errore
  - dislocate in package diversi a seconda del tipo di errore.
- La superclasse di tutti gli errori è la classe `Throwable` nel package `java.lang`.
- Qualsiasi nuovo tipo di errore deve essere inserito nella discendenza di `Throwable`
  - solo su gli oggetti di questa classe si possono usare le parole chiave di java per la gestione degli errori.

La Superclasse Throwable

- La superclasse `Throwable` ha due sottoclassi dirette, sempre in `java.lang`
  - `Error`
    - Errori fatali, dovuti a condizioni incontrollabili
      - Esaurimento delle risorse di sistema necessarie alla JVM, incompatibilità di versioni,…
    - In genere i programmi non gestiscono questi errori
  - `Exception`
    - Tutti gli errori che non rientrano in `Error`
    - I programmi possono gestire o no questi errori a seconda dei casi

Eccezioni

- Una eccezione è un evento che interrompe la normale esecuzione del programma
- Se si verifica un'eccezione il metodo trasferisce il controllo ad un gestore delle eccezioni
  - Il suo compito è quello di uscire dal frammento di codice che ha generato l'eccezione e decidere cosa fare

Eccezioni

- Java mette a disposizione varie classi per gestire le eccezioni, nei package
  - `java.lang`
  - `java.io`
- Tutte le classi che gestiscono le eccezioni sono ereditate dalla classe `Exception`

Gerarchia delle classi di eccezioni

Tipi di Eccezioni

- Due categorie:
  - eccezioni controllate
    - dovute a circostanze esterne che il programmatore non può evitare
    - il compilatore vuole sapere cosa fare nel caso si verifichi l'eccezione
  - eccezioni non controllate
    - dovute a circostanze che il programmatore può evitare, correggendo il programma

Tipi di eccezioni

- Esempio di eccezione controllata
  - `EOFException`: terminazione inaspettata del flusso di dati in ingresso
  - Può essere provocata da eventi esterni
    - errore del disco
    - interruzione del collegamento di rete
  - Il gestore dell'eccezione si occupa del problema

Tipi di eccezioni

- Esempi di eccezione non controllata
  - `NullPointerException`: uso di un riferimento `null`
  - `IndexOutofBoundException`: accesso ad elementi esterni ai limiti di un array
- Non bisogna installare un gestore per questo tipo di eccezione
  - Il programmatore può prevenire queste anomalie, correggendo il codice

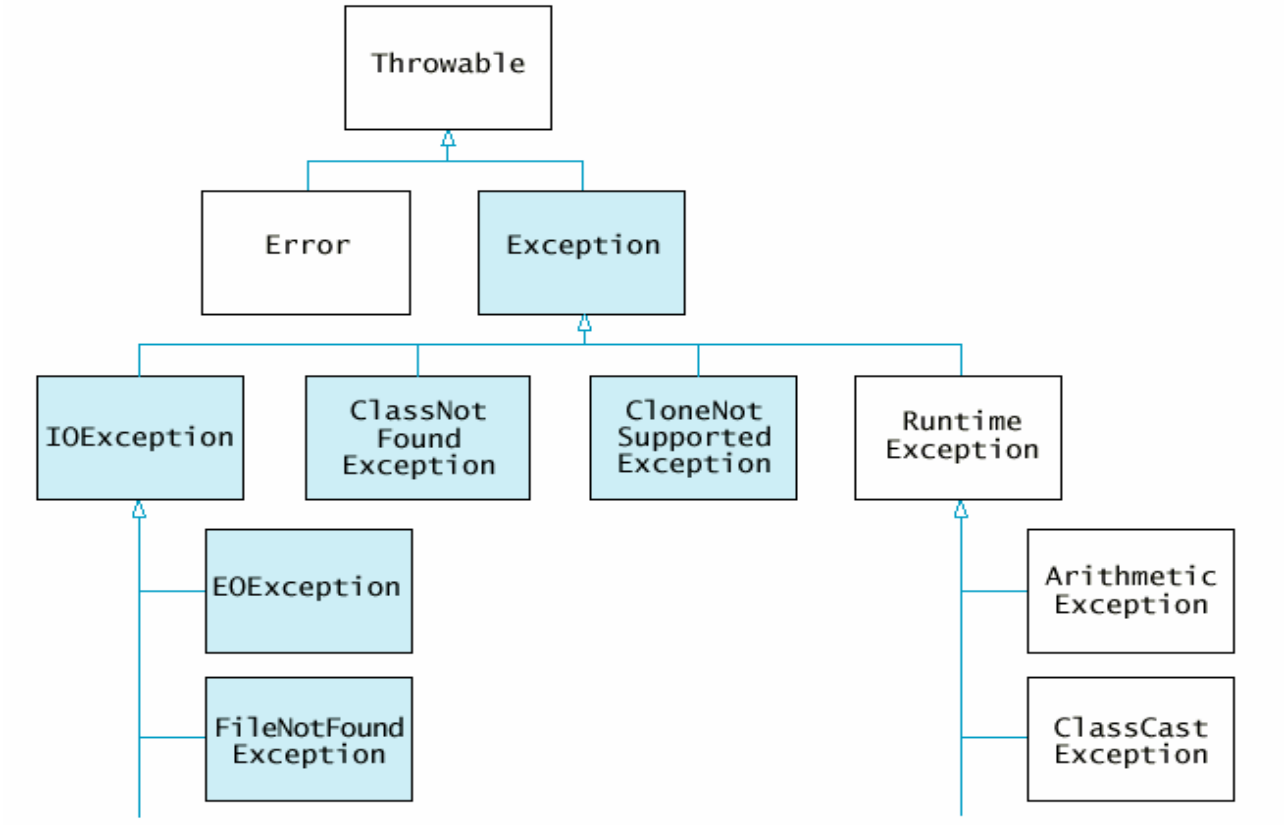
Eccezioni controllate

- Tutte le sottoclassi di `IOException`
  - `EOFException`
  - `FileNotFoundException`
  - `MalformedURLException`
  - `UnknownHostException`
- `ClassNotFoundException`
- `CloneNotSupportedException`

Eccezioni non controllate

- Tutte le sottoclassi di `RunTimeException`
  - `ArithmeticException`
  - `ClassCastException`
  - `IllegalArgumentException`
  - `IllegalStateException`
  - `IndexOutOfBoundsException`
  - `NoSuchElementException`
  - `NullPointerException`

Eccezioni controllate e non controllate



Eccezioni

- Per lanciare un'eccezione, usiamo la parola chiave throw (lancia), seguita da un oggetto di tipo eccezione

```
throw exceptionObject;
```

- Il metodo termina immediatamente e passa il controllo al gestore delle eccezioni
  - Le istruzioni successive non vengono eseguite

Lanciare eccezioni: Esempio

```
public class BankAccount
{
  public void withdraw(double amount)
   {
        if (amount > balance)
           throw new IllegalArgumentException(“Saldo
                  insufficiente");
        balance = balance - amount;
   }
   ...
}
```

La stringa in input al costruttore di `IllegalArgumentException` rappresenta il messaggio d’errore da associare all’eccezione

Segnalare eccezioni

- `BufferedReader.readLine()` può lanciare una `IOException`
- Un metodo che chiama `readLine()` può
  - gestire l’eccezione, cioè dire al compilatore cosa fare
  - non gestire l’eccezione, ma dichiarare di poterla lanciare
    - In tal caso, se l’eccezione viene lanciata, il programma termina visualizzando un messaggio di errore

Segnalare eccezioni

- Per segnalare le eccezioni controllate che il metodo può lanciare usiamo la parola chiave `throws`
- Esempio:

```
public void read(BufferedReader in) throws
IOException
```


Segnalare eccezioni

```
public class Coin
{
    public void read(BufferedReader in) throws
    IOException
    {
       value = Double.parseDouble(in.readLine());
        name =in.readLine();
    }
  ...
}
```

La clausola throws segnala al chiamante di `Coin.read` che esso può generare un'eccezione di tipo `IOException`

Segnalare eccezioni

- Qualunque metodo che chiama `Coin.read` deve decidere se gestire l'eccezione o dichiarare se poterla lanciare

```
public class Purse
   {
        public void read(BufferedReader in) throws
        IOException
        {
          while (…)
          { Coin c = new Coin();
            c.read(in);
            add(c);
          }
        }
    ...
   }
```


Segnalare eccezioni

- Un metodo può lanciare più eccezioni controllate, di tipo diverso

```
public void read(BufferedReader in)
   throws IOException, ClassNotFoundException
```

Usare le Eccezioni di Run Time

- Le eccezioni di runtime (`RuntimeException`) possono essere utilizzate per segnalare problemi dovuti ad input errati.

- Esempi:
  - Un metodo che preleva soldi da un conto corrente non può prelevare una quantità maggiore del saldo
  - Un metodo che effettua una divisione non può dividere un numero per zero

Progettare Nuove Eccezioni

- Se nessuna delle eccezioni di runtime ci sembra adeguata al nostro caso, possiamo progettarne una nuova.

- I nuovi tipi di eccezioni devono essere inseriti nella discendenza di `Throwable`, e in genere sono sottoclassi di `RuntimeException`.

- Un tipo di eccezione che sia sottoclasse di `RuntimeException` sarà a controllo non obbligatorio.

Progettare Nuove Eccezioni

Introduciamo un nuovo tipo di eccezione per controllare che il denominatore sia diverso da zero, prima di eseguire una divisione:

```
public class DivisionePerZeroException extends
RuntimeException{

  public DivisionePerZeroException() {
  super("Divisione per zero!");
  }

  public DivisionePerZeroException(String msg){
  super(msg);
  }
}
```


Usare Nuove Eccezioni

```
public class Divisione {
  public Divisione(int n, int d) {
  num=n;
  den=d;
  }
  public double dividi(){
  if (den==0)
    throw new DivisionePerZeroException();
  return num/den;
  }
  private int num;
  private int den;
}
```

Usare Nuove Eccezioni: Esempio

```
public class Test {


  public static void main(String[] args) throws IOException{

  double res;
  InputStreamReader isr=new   InputStreamReader(System.in);
  BufferedReader br=new BufferedReader(isr);

  System.out.print("Inserisci il numeratore:");
  int n= Integer.parseInt(br.readLine());
  System.out.print("Inserisci il denominatore:");
  int d= Integer.parseInt(br.readLine());

  Divisione div = new Divisione(n,d);
  res = div.dividi();
  }
}
```


Usare Nuove Eccezioni: Esempio

```
Inserisci il numeratore: 5
Inserisci il denominatore: 0
DivisionePerZeroException: Divisione per zero!
at Divisione.dividi(Divisione.java:12)
at divisioneperzero.Test.main(Test.java:22)
Exception in thread "main"
```

- Il **`main`** invoca il metodo **`dividi`** della classe **`Divisione`** alla linea 22
- Il metodo **`dividi`** genera una eccezione alla linea 12

Catturare eccezioni

- Ogni eccezione deve essere gestita, altrimenti causa l'arresto del programma

- Per installare un gestore si usa l'enunciato `try`, seguito da tante clausole `catch` quante sono le eccezioni da gestire

Catturare eccezioni

```
try
{
    enunciato
    enunciato
        ...
}
catch (ClasseEccezione oggettoEccezione)
{
    enunciato
    enunciato
    ...
}
catch (ClasseEccezione oggettoEccezione)
{
    enunciato
    enunciato
...
}
...
```

Catturare eccezioni

- Vengono eseguite le istruzioni all'interno del blocco `try`
- Se nessuna eccezione viene lanciata, le clausole `catch` sono ignorate
- Se viene lanciata una eccezione viene eseguita la corrispondente clausola `catch`

Catturare Eccezioni: Esempio

```
public class Test {
  public static void main(String[] args) throws
IOException{
  double res;
 InputStreamReader isr=new
   InputStreamReader(System.in);
  BufferedReader br=new BufferedReader(isr);

  System.out.print("Inserisci il numeratore:");
  int n= Integer.parseInt(br.readLine());

  System.out.print("Inserisci il denominatore:");
  int d= Integer.parseInt(br.readLine());
```

Catturare Eccezioni: Esempio

```
try
  {
  Divisione div = new Divisione(n,d);
  res = div.dividi();
  System.out.print(res);
  }

  catch(DivisionePerZeroException exception)
  {
  System.out.println(exception);
  }
  }
}
```


Catturare eccezioni

- Cosa fa l'istruzione **`System.out.println(exception);`**?
- Invoca il metodo **`toString()`** della classe **`DivisioneperZeroException`**
  - Ereditato dalla classe **`RuntimeException`**
  - Restituisce una stringa che descrive l'oggetto **`exception`** costituita da
    - Il nome della classe a cui l'oggetto appartiene seguito da : e dal messaggio di errore associato all'oggetto

Catturare Eccezioni: Esempio

```
Inserisci il numeratore:5
Inserisci il denominatore:0

DivisionePerZeroException: Divisione per zero!
```

- **`DivisionePerZeroException`**
  - è la classe a cui l'oggetto **`exception`** appartiene
- **`Divisione per zero!`**
  - È il messaggio di errore associato all'oggetto **`exception`** (dal costruttore)

Catturare eccezioni

- Per avere un messaggio di errore che stampa lo stack delle chiamate ai metodi in cui si è verificata l'eccezione usiamo il metodo **`printStackTrace()`**

```
catch(DivisionePerZeroException exception)
  {
   exception.printStackTrace();
  }
```

Catturare Eccezioni: Esempio

```
Inserisci il numeratore: 5
Inserisci il denominatore: 0
DivisionePerZeroException: Divisione per zero!
at Divisione.dividi(Divisione.java:12)
at divisioneperzero.Test.main(Test.java:22)
```


Catturare eccezioni

- Scriviamo un programma che chiede all'utente il nome di un file
- Se il file esiste, il suo contenuto viene stampato a video
- Se il file non esiste viene generata un'eccezione
- Il gestore delle eccezioni avvisa l'utente del problema e gli chiede un nuovo file

Catturare eccezioni: Esempio

```
import java.io.*;
public class TestTry {

   public static void main(String[ ] arg)
          throws IOException  {

     InputStreamReader isr=
           new InputStreamReader(System.in);

     BufferedReader br=new BufferedReader(isr);

     boolean ok=false;

     String s;

     System.out.println("Nome del file?");
```


Catturare eccezioni: Esempio

```
while(!ok) {

          try {

          s=br.readLine();
          FileReader fr=new FileReader(s);
          br=new BufferedReader(fr);
          ok=true;
          while((s=br.readLine())!=null)
            System.out.println(s);
         }

        catch(FileNotFoundException e) {
          System.out.println("File
               inesistente, nome?");
       }
     }
   }
}
```

La clausola finally

- Il lancio di un'eccezione arresta il metodo corrente
- A volte vogliamo eseguire altre istruzioni prima dell'arresto
- La clausola `finally` viene usata per indicare un'istruzione che va eseguita sempre
  - Ad, esempio, se stiamo leggendo un file e si verifica un'eccezione, vogliamo comunque chiudere il file

La clausola finally

```
try
{
    enunciato
    enunciato
    ...
}
finally
{
    enunciato
    enunciato
    ...
}
```

La clausola finally

- Viene eseguita al termine del blocco `try`
- Viene comunque eseguita se un'istruzione del blocco `try` lancia un'eccezione
- Può anche essere combinata con clausole `catch`

La clausola finally

```
BufferedReader in;
try
{
      in = new BufferedReader(
         new FileReader(filename));
      purse.read(in);
}
finally
{
      if (in !=null) in.close();
}
```