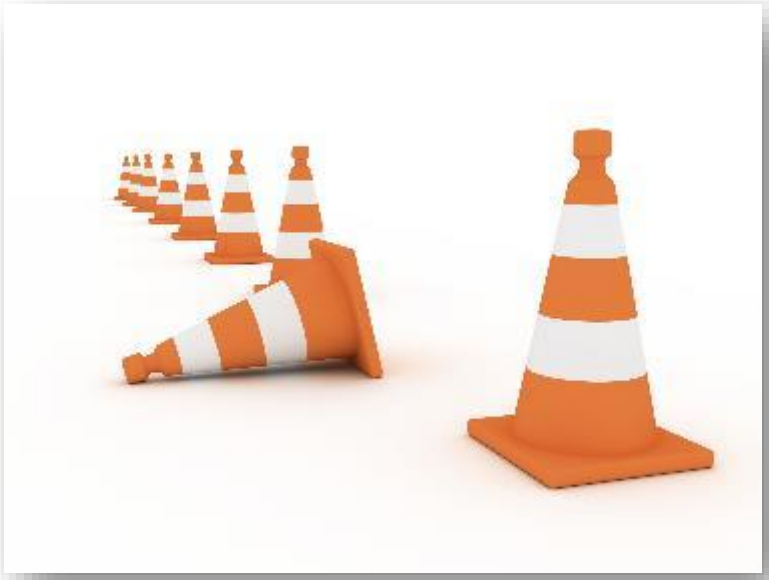


eccezioni

Java

- un'***eccezione*** è un oggetto che descrive una situazione ***anomala*** o una situazione di ***errore***
- le eccezioni vengono «lanciate» (***throw***) in un punto del programma e possono essere catturate (***catch***) e gestite da altre parti del programma
- anche un errore è rappresentato come un oggetto Java, ma solitamente rappresenta una situazione non recuperabile e non gestibile

- al verificarsi di una eccezione è possibile:
  - ***ignorarla***
  - ***gestirla*** direttamente
  - ***propagare*** l'eccezione per gestirla in un'altra parte del programma

ignorare le eccezioni

- se un'eccezione è ignorata il programma ***termina*** producendo un messaggio d'errore
- 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
  - delle chiamate di metodi che hanno portato all'eccezione

```
int num,den;
double val;
num = 3;
den = 0;
val = num / den;
```

- Il risultato è:
  - Exception in thread "main" java.lang.ArithmeticException: / by zero at ProvaEccezioni.main(ProvaEccezioni.java:--)

- la ***gestione*** dell'eccezione avviene nel momento stesso in cui questa può accadere
- le operazioni che possono causare eccezioni vanno inserite in un blocco ***try*** seguito da una o più clausole ***catch***, che contengono il codice per gestire l'eccezione
- ogni clausola catch è associata ad un tipo di eccezione e viene chiamata ***exception handler***
- quando si solleva un'eccezione, l'esecuzione prosegue dalla prima clausola catch che corrisponde al tipo d'eccezione sollevata

- si ***tenta*** (try) di eseguire il codice e se si verifica un'eccezione si ***intercetta*** (catch) per gestirla

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

```
int num,den;
double val;
num = 3;
den = 0;
try {
  val = num / den;
}
catch (ArithmeticException e){
  System.out.println("Si è verificato un errore: ");
  System.out.println (e.toString());
}
```

- il risultato è il seguente:
  Si è verificato un errore: java.lang.ArithmeticException: / by zero
- il programma però ***non si blocca*** e prosegue la sua esecuzione

finally

- l'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

propagare le eccezioni

- se l'eccezione non viene intercettata e gestita nel metodo in cui si verifica, può ancora essere ***trattata a un livello più alto***
- le eccezioni si propagano attraverso la gerarchia delle chiamate di metodi finché non vengono intercettate e gestite

- un'eccezione controllata deve essere raccolta da un metodo in una clausola catch o 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

gestione delle 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

esempio (1)


```
public class Frazione {
  private int num;
  private int den;
  public Frazione(int num, int den) {
    this.num = num;
    this.den = den;
  }
  public double valore() throws ArithmeticException
  {
    return num/den);
  }
}
```

- il metodo valore() della classe Frazione può ***sollevare un'eccezione***

Esempio (2)

- chi chiama il metodo può controllare l'eventuale eccezione

```
try {
      val = f.valore();
}
catch (ArithmeticException e){
      System.out.println ("Errore "+e);
}
```

lanciare un'eccezione

- è possibile definire un'eccezione ***estendendo la classe Exception*** o una sua sottoclasse
- le 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

esempio (1)


```
public class Frazione {
    private int num;
    private int den;
    public Frazione(int num, int den) {
      this.num = num;
      this.den = den;
    }
    public double valore() throws Exception {
      if (den==0)
        throw new Exception("divisione per 0");
      return (num/den);
    }

}
```

- il metodo valore() della classe Frazione può sollevare un'eccezione

esempio (2)

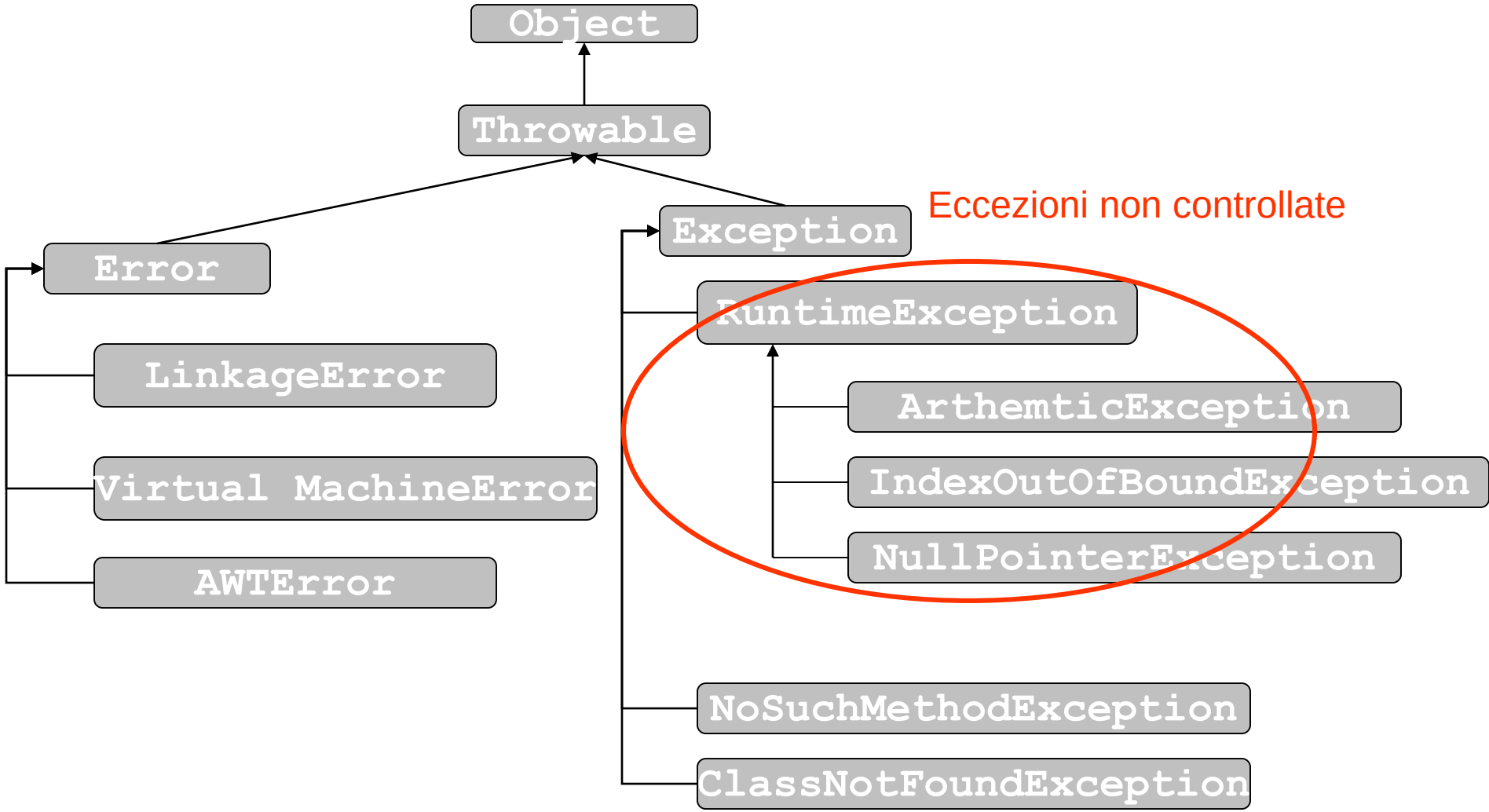
- chi chiama il metodo può controllare l'eventuale eccezione

```
try {
      val = f.valore();
}
catch (Exception e){
      System.out.println ("Errore "+e);
}
```

classificazione delle eccezioni

- eccezioni ***controllate***
  - Il compilatore impone di gestirle
- eccezioni ***non controllate***
  - Non viene imposta la gestione che è lasciata al programmatore

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


Object
Throwable
Error
Exception
Eccezioni non controllate
LinkageError
VirtualMachineError
AWTError
RuntimeException
ArthemticException
IndexOutOfBoundException
NullPointerException
NoSuchMethodException
ClassNotFoundException