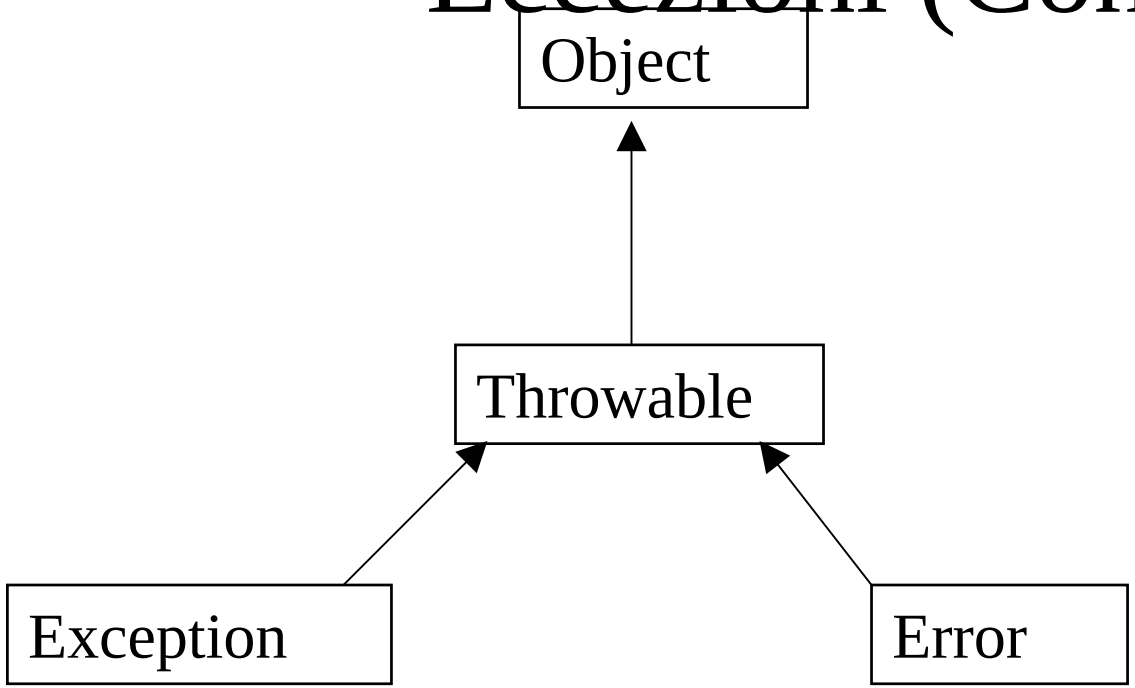
Eccezioni

- la generazione di un’eccezione segnala un evento eccezionale (!)
- separa il cammino logico standard dalla gestione delle situazioni inattese
- Java tramite le eccezioni permette di concentrare il codice che gestisce le condizioni di errore o comunque anormali in modo centralizzato
- limita la complessita’ della struttura e del flusso di programma
- ATTENZIONE: richiedono un grosso overhead computazionale

Eccezioni (Cont.)



Possono essere gestite

Non dovrebbero essere gestite

Blocco Try Catch

```
try
{
	<azione che potrebbe causare un eccezione>
}
catch(Exception e)
{
	<azione correttiva o semplice segnalazione>
}
finally
{
	<azione eseguita comunque>
}
```

Metodi ed Eccezioni

- si possono specificare le eccezioni lanciate da un metodo
- si forza il codice che invoca il metodo ad essere in grado di gestire le eccezioni elencate (controllo al tempo della compilazione)
- il compilatore controlla anche che le eccezioni elencate possono effettivamente essere generate

Rilanciare le eccezioni

- Le eccezioni catturate con un blocco catch possono essre rilanciate
  - permette di effettuare clean-up locale e propagare comunque l'avvenimento eccezionale
  - permette di specificare ulteriormente cosa e' accaduto

```
try
{
        <azione>
}
catch(Exception e)
{
        Exception newE = new Exception(e.toString() + " " + (new java.util.Date()).toString());
        throw newE;
}
```

Errori

- Segnalano situazioni che di solito sono o irrecuperabili a livello di macchina virtuale
  - errori di link (problemi di caricamento di un file di classe)
  - errori interni alla macchina virtuale
- o servono ad effettuare il clean-up della macchina virtuale
  - morte di Thread (segnala la terminazione di un Thread)

Abstract Data Type

- **Definition:** A mathematically specified collection of data-storing entities with operations to create, access, change, etc. instances.
- *Note: Since the collection is defined mathematically, rather than as an implementation in a computer language, we may reason about effects of the operations, relations to other abstract data types, whether a program implements the data type, etc.*

Esempio: Stack

- *One of the simplest abstract data types is the stack. The operations new(), push(v, S), top(S), and pop(S) may be defined with the following.*
- *new() returns a stack*
  pop(push(v, S)) = S
  top(push(v, S)) = v
  where S is a stack and v is a value.

Stack

- *From the previous slide axioms, one may define equality between stacks, define a pop function which returns the top value in a non-empty stack, etc. For instance, the predicate isEmpty(S) may be added and defined with the following two axioms.*
  isEmpty(new()) = true
  isEmpty(push(v,S)) = false