

Programmazione II (B)

a.a 2012/2013

Trattamento delle eccezioni

Matteo Baldoni

Dipartimento di Informatica
Università degli Studi di Torino
<http://www.di.unito.it/~baldoni>



Lezione di Programmazione II (B), a.a. 2012/2013, Corso di Studi in Informatica,
Università degli Studi di Torino by [Matteo Baldoni](#) is licensed under a
[Creative Commons Attribution-Non commerciale-Condividi allo stesso modo 2.5 Italia License](#).

Cos'è un'eccezione?

- Durante l'esecuzione di un programma possono verificarsi degli errori:
 - errori di programmazione
 - dati errati in ingresso
- *Interruzione dell'esecuzione!*

```
class ErroreProgrammazione {  
    static int[] a = new int[2];  
    static void p(int i, int val) {  
        a[i] = val;  
    }  
    public static void main (String[] args) {  
        p(0, 5);  
        p(2, 4);  
        p(1, 3);  
    }  
}
```

```
class ErroreIngresso {  
    public static void main (String[] args) {  
        String stringa;  
        int num;  
        stringa = Console.readLine();  
        num = Integer.parseInt(stringa);  
        System.out.println("Inserito: " + num);  
    }  
}
```

Eccezioni al "normale" flusso di esecuzione

quando l'utente inserisce ad es. "5a" invece di "5"

Eccezione: cosa fare?

- Una possibile soluzione:
il metodo potrebbe **restituire un valore** che indica il **successo** o il **fallimento**
- Tale valore potrebbe essere eventualmente utilizzato dal *chiamante* per porre rimedio alla situazione e ripristinare l'esecuzione

```
class ErroreProgrammazione {  
    static int[] a = new int[2];  
    static bool p(int i, int val) {  
        if (a.lenght > i) {  
            a[i] = val;  
            return true;  
        }  
        else  
            return false;  
    }  
    public static void main (String[] args) {  
        ....  
        if (!p(2, 4))  
            System.out.println("Errore di indice!");  
        ...  
    }  
}
```

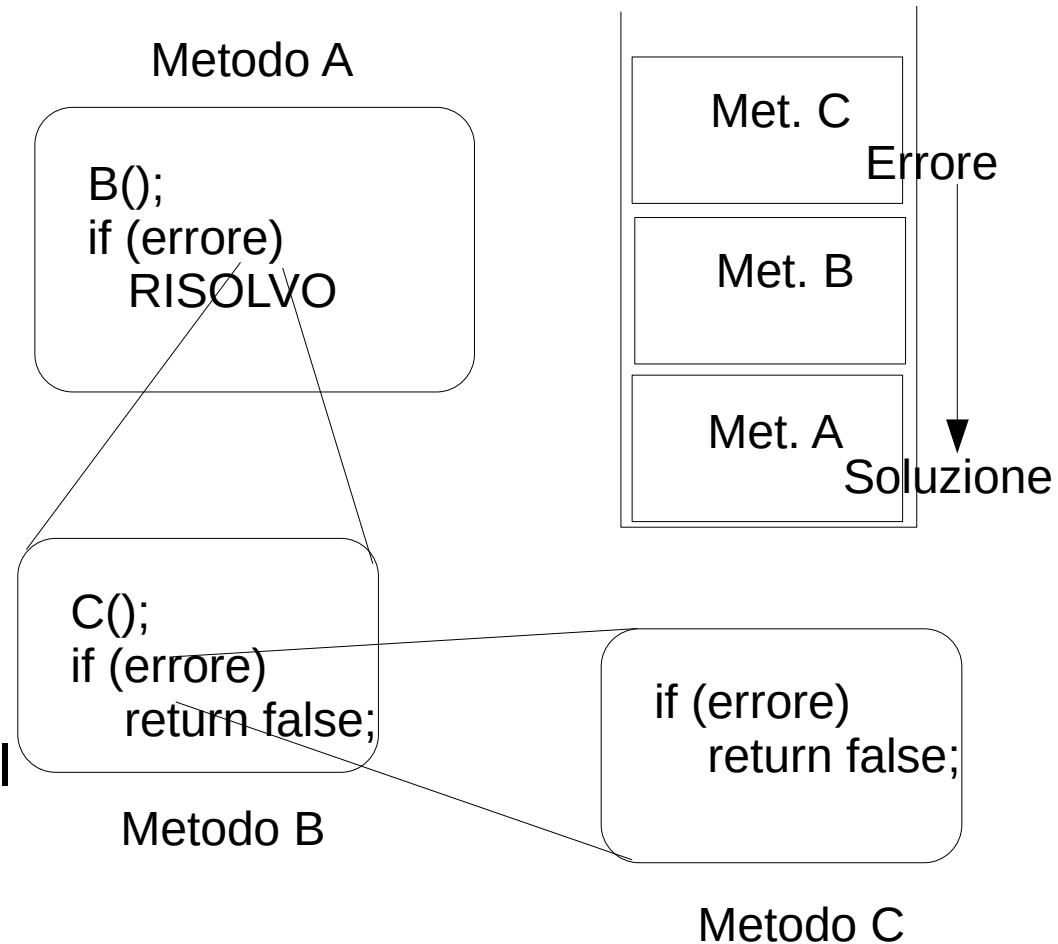
Problemi (#1, #2)

- È necessario controllare sempre il valore ricevuto: **il codice risulta meno leggibile**
- Non sempre è possibile restituire un valore di successo o fallimento, ad esempio quando **un metodo deve già restituire un valore di altro tipo**

```
class ErroreProgrammazione {  
    static int[] a = new int[2];  
    static bool p(int i, int val) {  
        if (a.lenght > i) {  
            a[i] = val;  
            return true;  
        }  
        else  
            return false;  
    }  
    public static void main (String[] args) {  
        if (!p(0, 5))  
            System.out.println("Errore di indice!");  
        if (!p(2, 4))  
            System.out.println("Errore di indice!");  
        if (!p(1, 3))  
            System.out.println("Errore di indice!");  
    }  
}
```

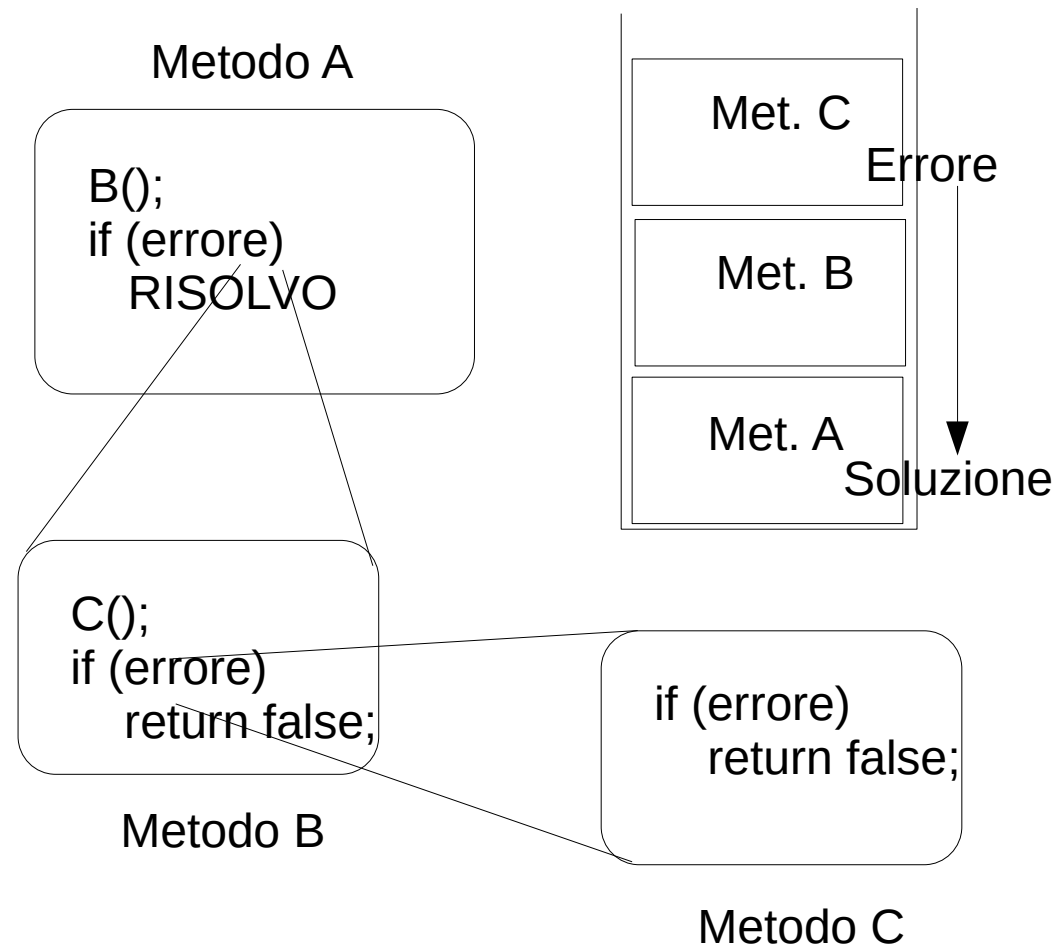
Problemi (#3)

- Non sempre si hanno sufficienti informazioni per “ripristinare” l'esecuzione lì dove l'eccezionalità si è verificata
- Potrebbe essere il caso che solo il chiamante del chiamante sia in grado di risolvere il problema
- o ancora peggio...
- Ad ogni livello dovremmo preoccuparci di restituire un valore di successo o fallimento al livello soprastante



Il meccanismo delle eccezioni

- Il *trattamento delle eccezioni* in Java offre una meccanismo **non affetto dai precedenti problemi** per affrontare condizioni “eccezionali”



throw, try e catch

- Il costrutto nel linguaggio Java per *lanciare* le eccezioni è **throw**
- Il costrutto per eseguire istruzioni che potrebbero *lanciare/sollevare* eccezioni e *catturarle* è **try ... catch**

```
class ErroreProgrammazione {  
    static int[] a = new int[2];  
    static void p(int i, int val) {  
        if (a.length > i)  
            a[i] = val;  
        else  
            throw new  
                ArrayIndexOutOfBoundsException();  
    }  
    public static void main (String[] args) {  
        try{  
            p(0, 5);  
            p(2, 4);  
            p(1, 3);  
        }  
        catch (ArrayIndexOutOfBoundsException err) {  
            System.out.println("Errore di indice!");  
        }  
    }  
}
```

lancia un'eccezione

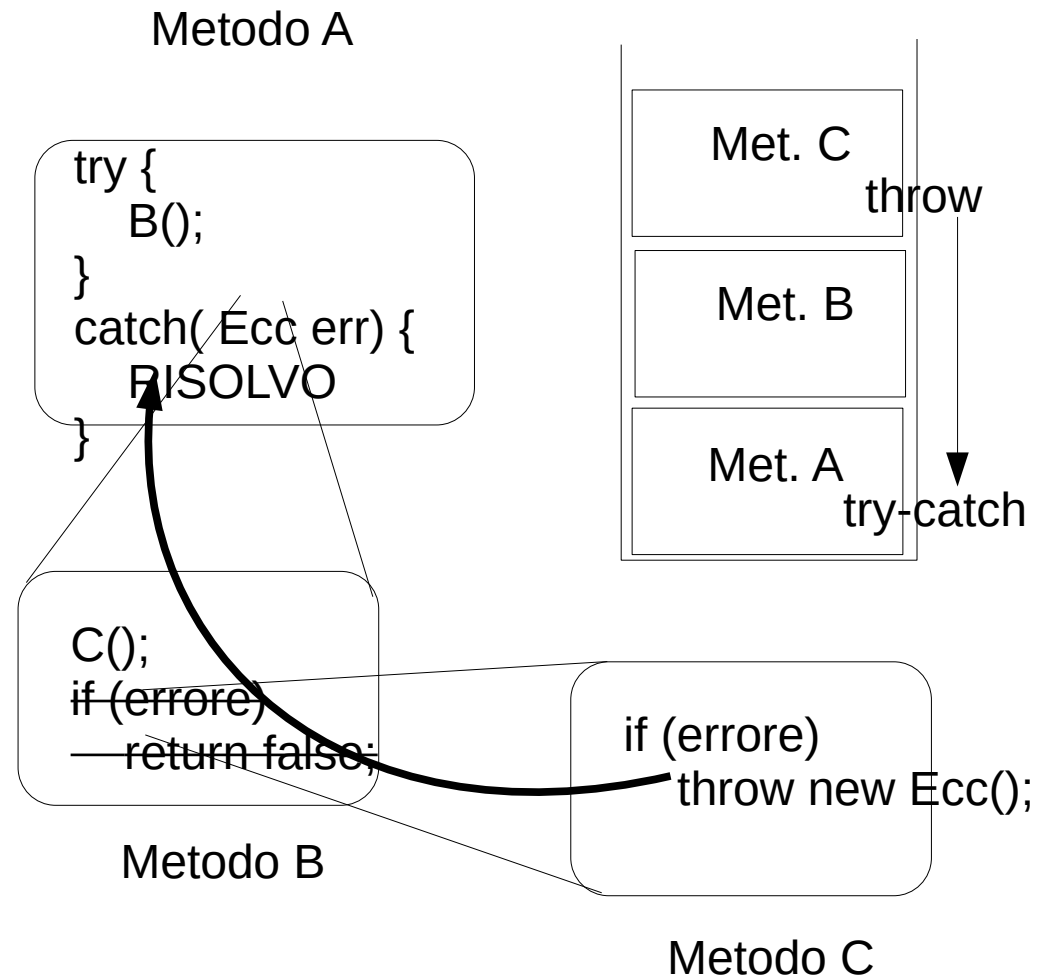
cattura questa eccezione

cosa fare se occorre un'eccezione

potrebbero lanciare un'eccezione

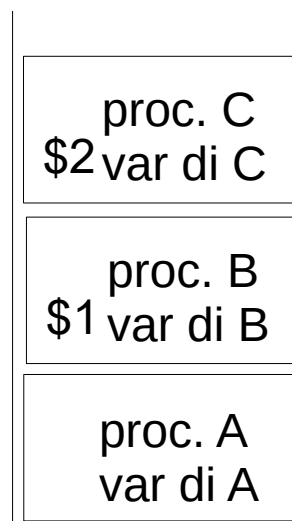
Il meccanismo delle eccezioni

- In caso di errore viene *lanciata una eccezione* (un oggetto) interrompendo il normale flusso di esecuzione
- Le eccezioni sono passate all'indietro da un metodo al suo chiamante fino a quando si trova *exception handler* che la *cattura* e la *gestisce* riprendendo l'esecuzione
- *Chi mi gestisce?* In una libreria i metodi sono utilizzati nei programmi più diversi non è possibile pensare ad una gestione locale



Eccezioni e stack dei record di attivazione

```
...  
void A() {  
    try {...  
        B(); /$1/  
    } catch { ... }  
    ...  
    B();  
    ...  
}
```



- Le eccezioni sono passate all'indietro da un metodo al suo chiamante fino a quando si trova *exception handler* che la *cattura* e la *gestisce* riprendendo l'esecuzione

```
...  
void B() {  
    ...  
    C(); /$2/  
    ...  
}
```

```
...  
void C() {  
    ...  
}
```

lanciata eccezione

Eccezioni e stack dei record di attivazione

```
...  
void A() {  
    try {...  
        B(); /$1/  
    } catch { ... }  
    ...  
    B();  
    ...  
}
```

proc. B
\$1 var di B

proc. A
var di A

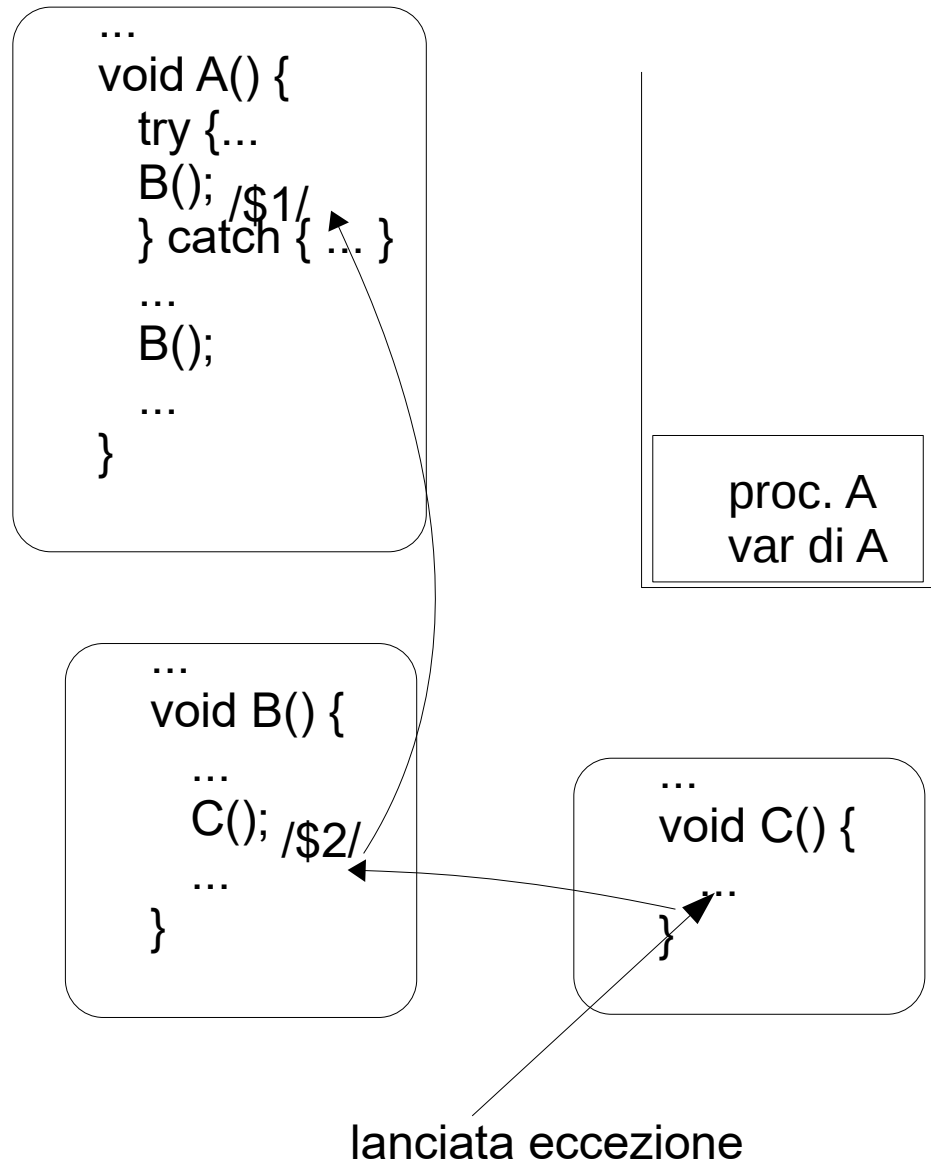
```
...  
void B() {  
    ...  
    C(); /$2/  
    ...  
}
```

```
...  
void C() {  
    ...  
}
```

lanciata eccezione

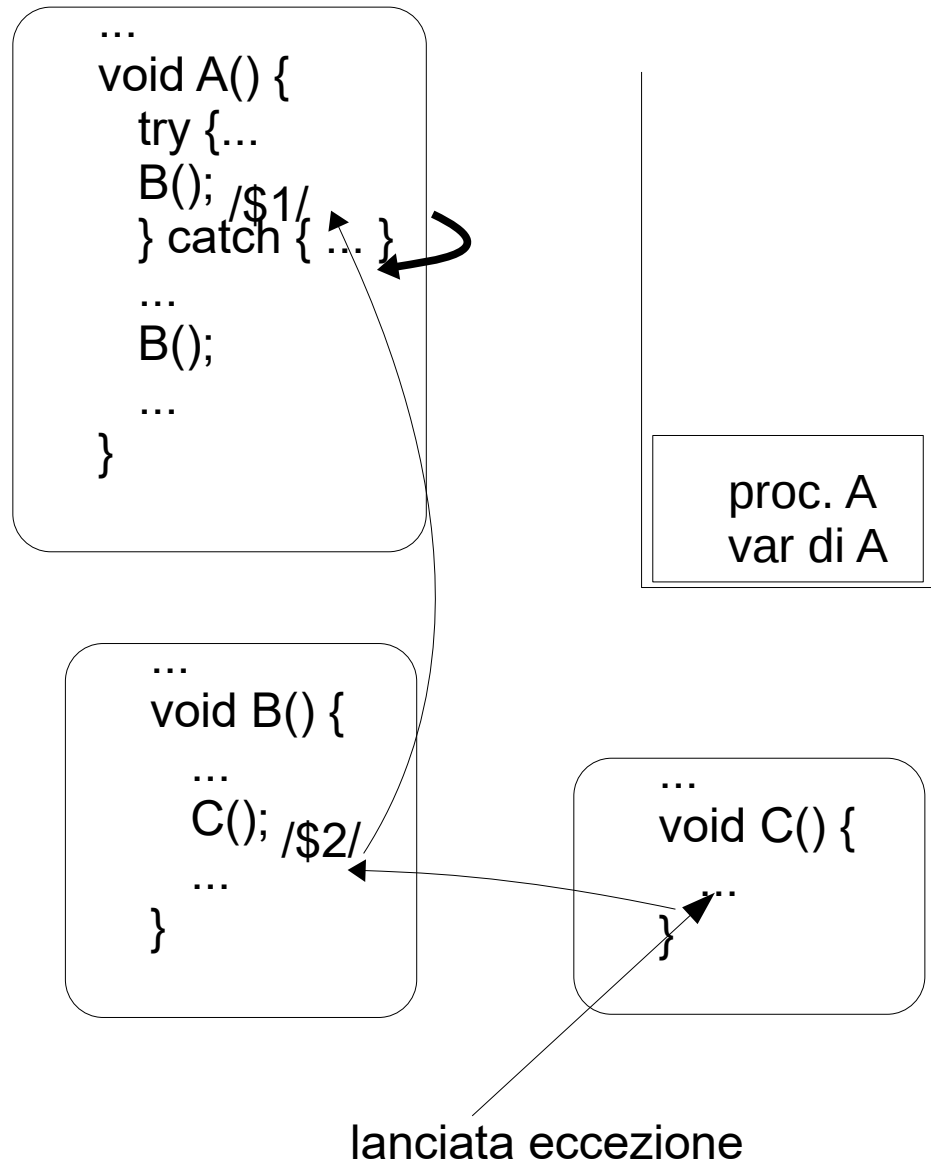
- Le eccezioni sono passate all'indietro da un metodo al suo chiamante fino a quando si trova *exception handler* che la *cattura* e la *gestisce* riprendendo l'esecuzione

Eccezioni e stack dei record di attivazione



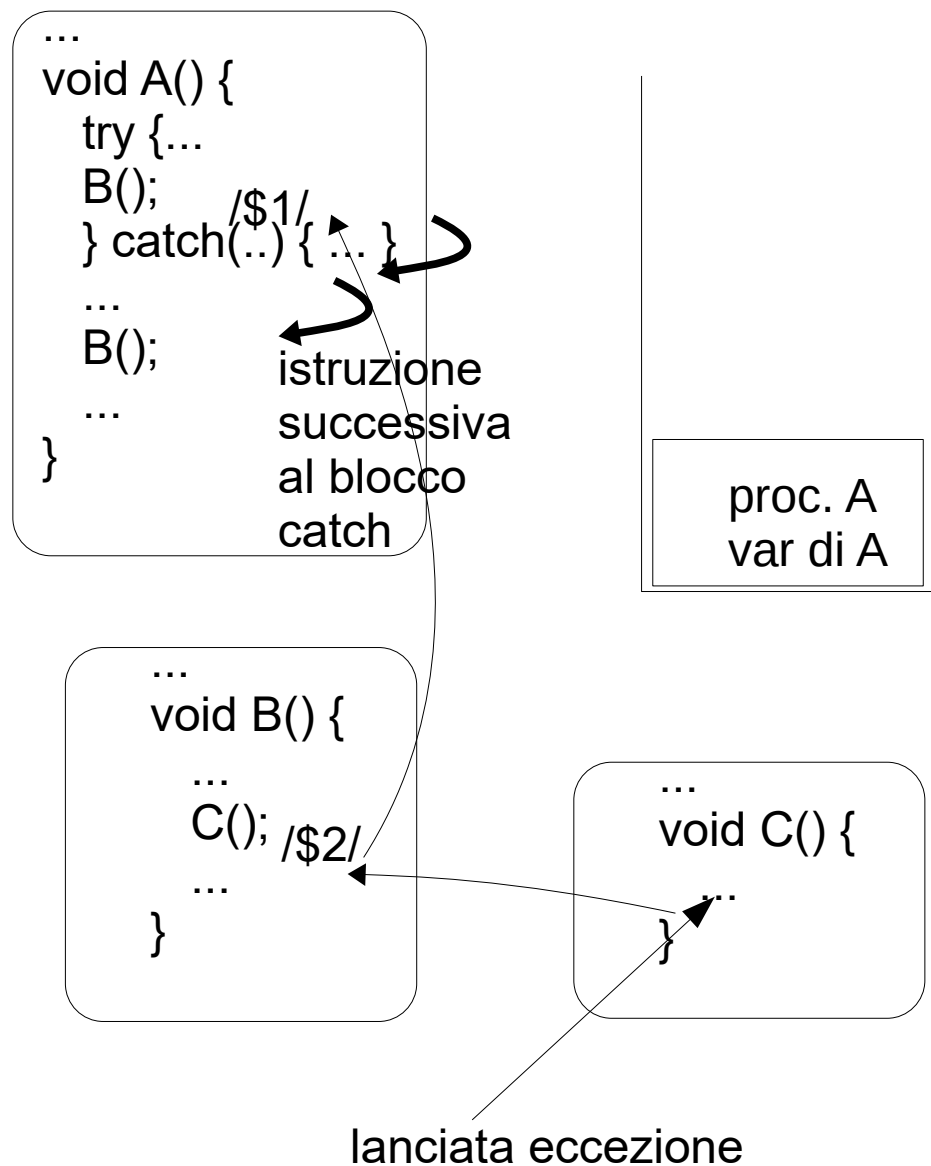
- Le eccezioni sono passate all'indietro da un metodo al suo chiamante fino a quando si trova *exception handler* che la *cattura* e la *gestisce* riprendendo l'esecuzione

Eccezioni e stack dei record di attivazione

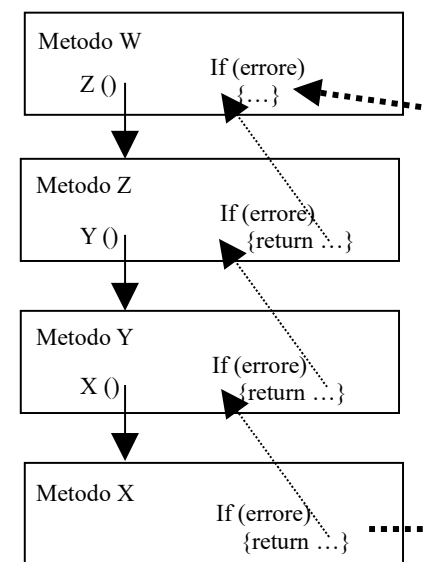


- Le eccezioni sono passate all'indietro da un metodo al suo chiamante fino a quando si trova *exception handler* che la *cattura* e la *gestisce* riprendendo l'esecuzione

Eccezioni e stack dei record di attivazione



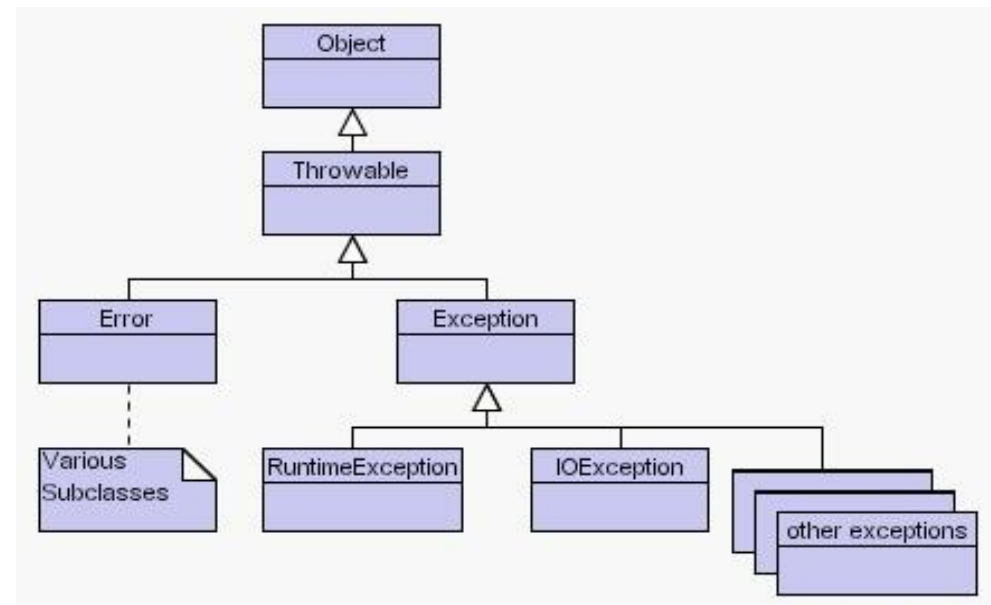
- Le eccezioni sono passate all'indietro da un metodo al suo chiamante fino a quando si trova *exception handler* che la *cattura* e la *gestisce* riprendendo l'esecuzione



Lanciare eccezioni

- Il costrutto **throw** permette di lanciare eccezioni
- Le eccezioni sono veri e propri oggetti il cui stato riporta le informazioni riguardanti la situazione in cui si è verificato errore (traccia dello stack di esecuzione)
- Gerarchia di eccezioni (il concetto di gerarchia di classi lo vedremo più avanti nel corso)

```
...  
throw new xxxException();  
...
```



Lanciare eccezioni

- Esempi di eccezioni presenti in Java:

- *IOException*
- *IllegalArgumentException*
- *NullPointerException*
- *NumberFormatException*
- *ArrayIndexOutOfBoundsException*

“ArrayIndexOutOfBoundsException”

```
class ErroreProgrammazione {  
    static int[] a = new int[2];  
    static void p(int i, int val) {  
        a[i] = val;  
    }  
    public static void main (String[] args) {  
        p(0, 5);  
        p(1, 3);  
        p(2, 4);  
    }  
}
```

```
class ErroreIngresso {  
    public static void main (String[] args) {  
        String stringa;  
        int num;  
        stringa = Console.readLine();  
        num = Integer.parseInt(stringa);  
        System.out.println("Inserito: " + num);  
    }  
}
```

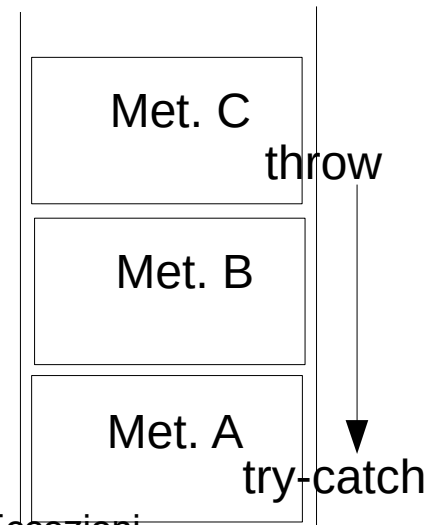
“NumberFormatException”

quando l'utente inserisce ad es. “5a” invece di “5”

Il gestore dell'eccezione

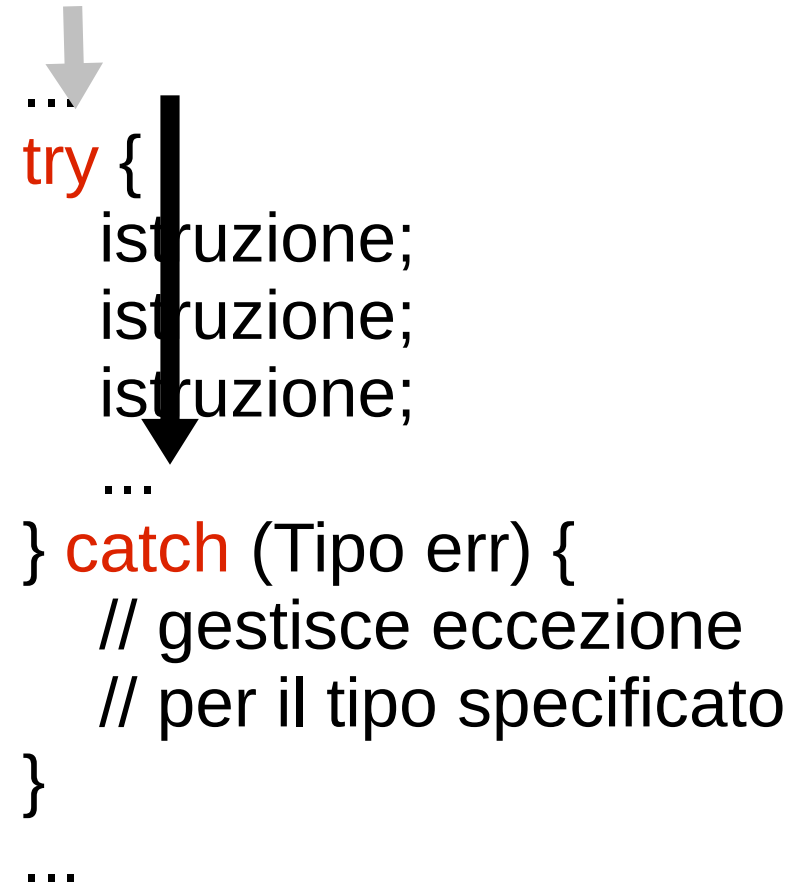
- Quando viene lanciata una eccezione, si interrompe l'esecuzione e *si cerca dinamicamente*, percorrendo all'indietro lo stack delle chiamate, *il primo gestore di eccezioni* che ha *una catch per il tipo dell'eccezione lanciata*, e l'esecuzione riprende da quel punto
- Se l'eccezione non viene catturata, l'interprete blocca l'esecuzione

```
...  
try {  
    istruzione;  
    istruzione;  
    istruzione;  
    ...  
} catch (Tipo err) {  
    // gestisce eccezione  
    // per il tipo specificato  
}  
...
```



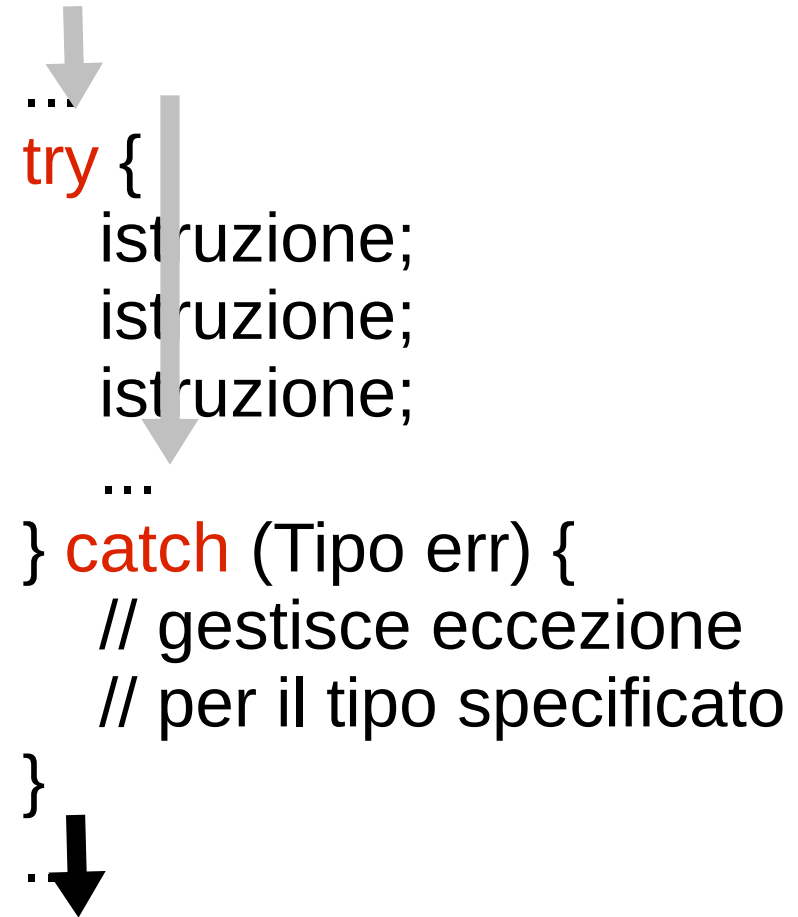
try...catch: flusso di esecuzione

- Se all'interno del blocco try non si verifica alcuna eccezione



try...catch: flusso di esecuzione

- Se all'interno del blocco try non si verifica alcuna eccezione
- L'esecuzione prosegue normalmente con l'istruzione che segue la try ... catch, **senza eseguire il codice della catch**



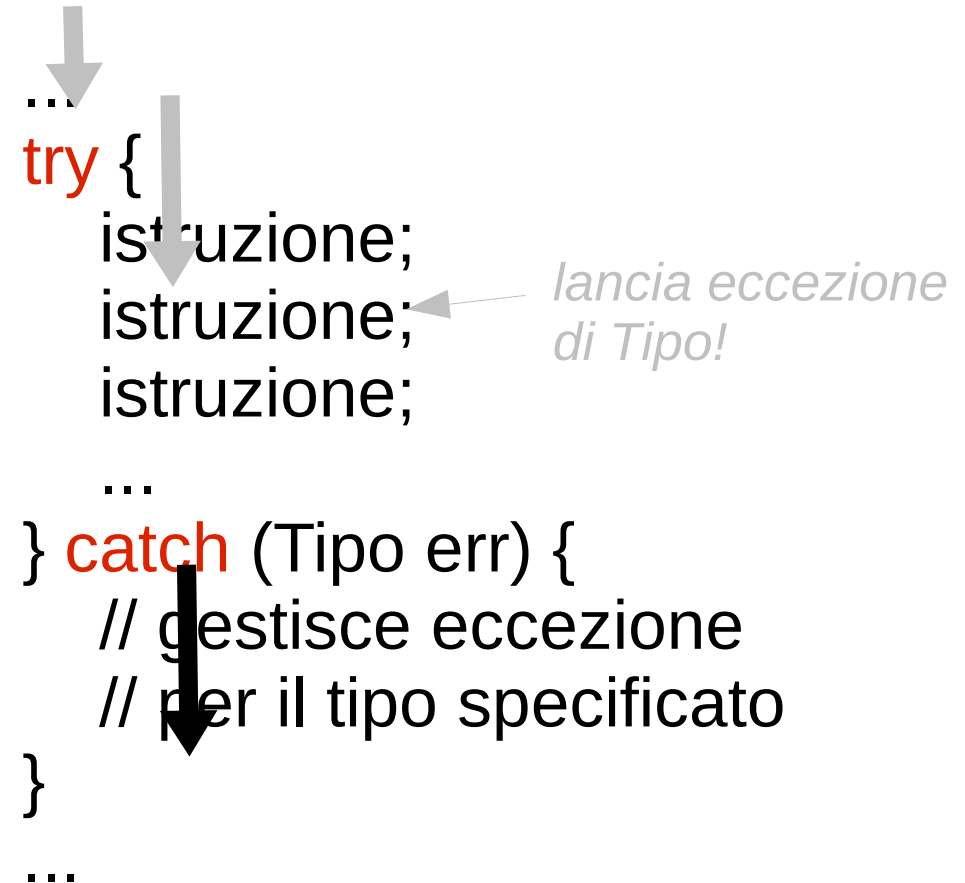
try...catch: flusso di esecuzione

- Se all'interno del blocco try si verifica una eccezione
- ***non viene eseguito il rimanente codice del blocco try***



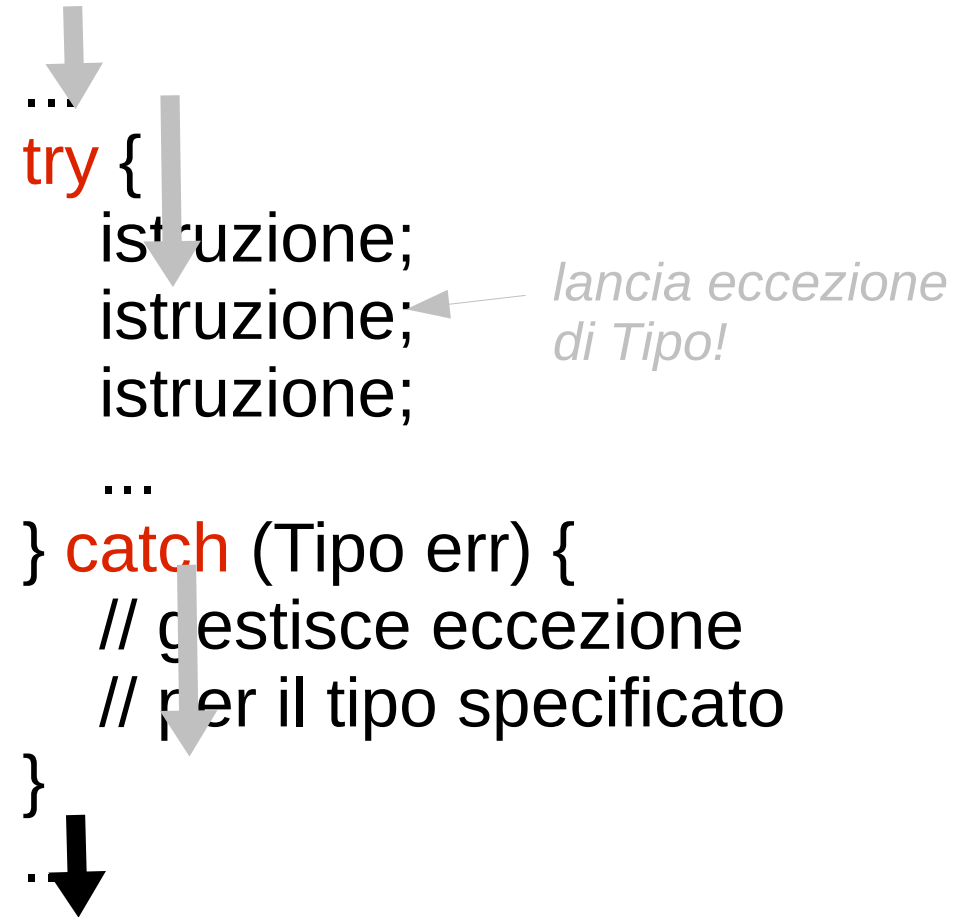
try...catch: flusso di esecuzione

- Se all'interno del blocco try si verifica una eccezione
- non viene eseguito il rimanente codice del blocco try
- Se l'eccezione sollevata è di tipo Tipo, si ***esegue il codice del blocco catch***



try...catch: flusso di esecuzione

- Se all'interno del blocco try si verifica una eccezione
- Non viene eseguito il rimanente codice del blocco try
- Se l'eccezione sollevata è di tipo Tipo, si esegue il codice del blocco catch
- e si **riprende l'esecuzione dall'istruzione che segue la try...catch**



try...catch: flusso di esecuzione

- Se all'interno del blocco try si verifica una eccezione
- Non viene eseguito il rimanente codice del blocco try
- Se l'eccezione sollevata non è di tipo Tipo, il metodo che contiene la try...catch termina immediatamente, il suo frame viene disallocato e ***l'esecuzione viene passata indietro al suo chiamante***



Un esempio

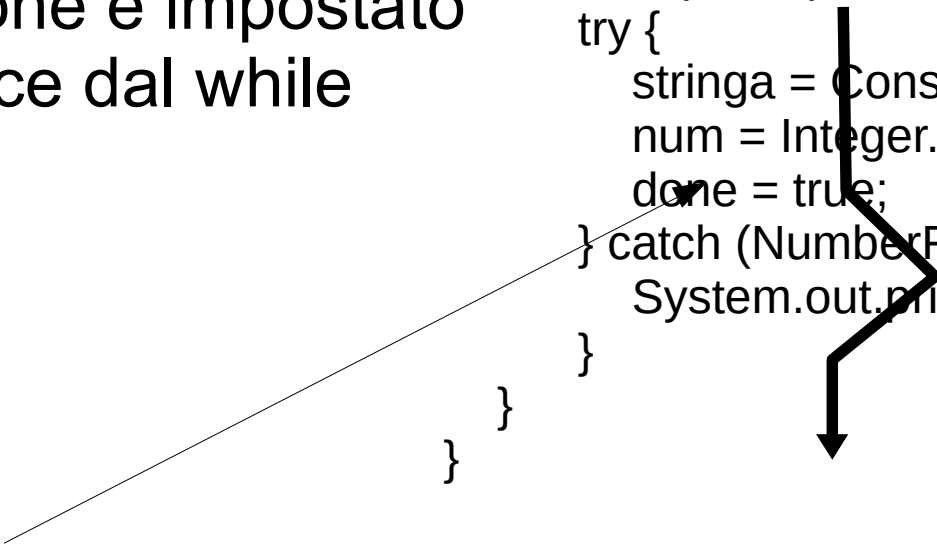
- L'esempio della lettura di un valore numerico
- Il metodo `Integer.parseInt` può lanciare l'eccezione `NumberFormatException` nel caso che la stringa passata come parametro non rappresenti un valore intero
- In caso di eccezione si desidera domandare all'utente di introdurre una nuova stringa

```
class ErroreIngresso {  
    public static void main (String[] args) {  
        String stringa;  
        int num = 0;  
        boolean done = false;  
  
        while (!done)  
            try {  
                stringa = Console.readLine();  
                num = Integer.parseInt(stringa);  
                done = true;  
            } catch (NumberFormatException err) {  
                System.out.println("Errore di immissione");  
            }  
    }  
}
```

Un esempio

- Se la stringa inserita **rappresenta un numero intero**, il blocco catch viene saltato
- e poiché done è impostato a true si esce dal while

```
class ErroreIngresso {  
    public static void main (String[] args) {  
        String stringa;  
        int num = 0;  
        boolean done = false;  
  
        while (!done)  
            try {  
                stringa = Console.readLine();  
                num = Integer.parseInt(stringa);  
                done = true;  
            } catch (NumberFormatException err) {  
                System.out.println("Errore di immissione");  
            }  
        }  
    }  
}
```



Viene eseguita solo se l'istruzione precedente non ha sollevato un'eccezione

Un esempio

- Se, invece, la stringa inserita **non rappresenta un numero intero**, l'esecuzione è interrotta in corrispondenza all'invocazione del metodo `Integer.parseInt`
- e l'esecuzione prosegue all'interno del blocco della `catch`, lasciando a `false` la variabile `done`
- e quindi non si esce dal ciclo `while`

```
class ErroreIngresso {  
    public static void main (String[] args) {  
        String stringa;  
        int num = 0;  
        boolean done = false;  
  
        while (!done)  
        {  
            try {  
                stringa = Console.readLine();  
                num = Integer.parseInt(stringa);  
                done = true;  
            } catch (NumberFormatException err) {  
                System.out.println("Errore di immissione");  
            }  
        }  
    }  
}
```

Un esempio

- È fondamentale che l'istruzione “done = true” sia all'interno del blocco try
- Nell'esempio qui a fianco si esce dall'iterazione while in ogni caso, stringa che rappresenta un valore intero oppure no!
- ... con sicuri problemi in seguito

```
class ErroreIngresso {  
    public static void main (String[] args) {  
        String stringa;  
        int num = 0;  
        boolean done = false;  
  
        while (!done) {  
            try {  
                stringa = Console.readLine();  
                num = Integer.parseInt(stringa);  
            } catch (NumberFormatException err) {  
                System.out.println("Errore di immissione");  
            }  
            done = true;  
        }  
    }  
}
```

Exception handler generale

- Il blocco di finally è eseguito sempre, anche se non sono sollevate eccezioni
- Possono essere catturate eccezioni di più tipi diversi

```
try {  
    codice che può generare una eccezione  
} catch(Tipo1 id1) {  
    gestisce eccez. di Tipo1  
} catch(Tipo2 id2) {  
    gestisce eccez. di Tipo2  
}.....  
} finally {....}
```

Eccezioni controllate e non

- Eccezioni controllate (e.g. tutte le IOException)
 - associate a problemi non prevedibili, che capitano frequentemente durante esecuzione di programma bisogna specificare come gestirle (se no compilatore Java dà errore)
- Eccezioni non controllate (e.g. le RuntimeException, come ArrayIndexOutOfBoundsException)
 - associate a problemi dovuti a errori di programmazione, che dovrebbero essere prevenuti da programmatore non è necessario specificare come gestirle mediante gestore di eccezioni

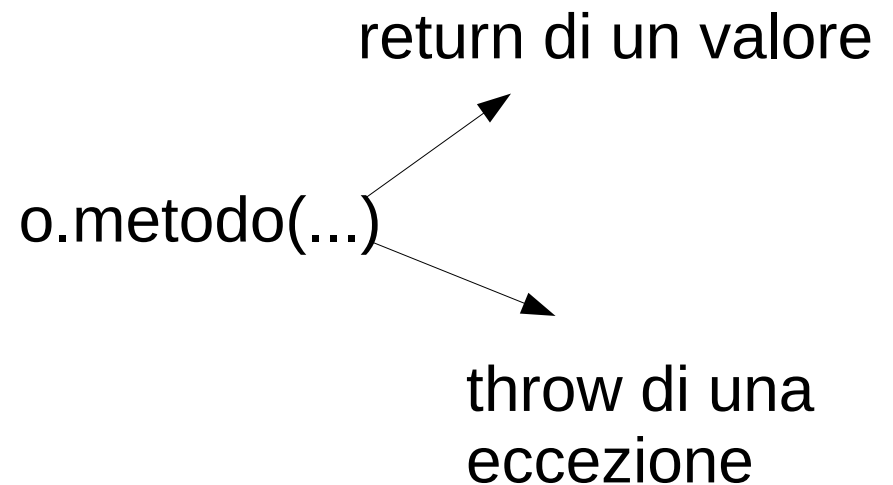
Throws

- Se un metodo può lanciare un'eccezione e non la cattura, deve segnalarlo con *throws* (tranne che per `RuntimeException`).

```
class Generica {  
    void m(String s) throws ClassNotFoundException {  
        Class c = Class.forName(s);  
        .....  
    }  
}
```

Throws

- Se lo si desidera è possibile vedere l'eccezione come un valore alternativo a quello usuale restituito dal metodo quando occorre una qualche situazione eccezionale



Suggerimenti di uso

- Suggerimenti per l'uso delle eccezioni (da Core Java)
- La gestione delle eccezioni non è concepita per sostituire un semplice test
- Le eccezioni non devono essere gestite in modo troppo frammentario
- Si faccia buon uso della gerarchia delle eccezioni: meglio individuare una sottoclasse appropriata di RuntimeException o crearne una nuova
- Non si devono ridurre al silenzio le eccezioni
~~try{...} catch(Exception e) {}~~
- Quando si cattura un errore, è preferibile essere severi piuttosto che indulgenti
- Propagare le eccezioni non è da biasimare: spesso conviene passare una eccezione piuttosto che gestirla subito

Definizione di eccezioni

- È possibile definirsi una propria eccezione: questa potrà essere utilizzata mediante `new MiaEccezione` come una qualsiasi altra eccezione nota a Java e descritta nella sua documentazione

```
class MiaEccezione extends Exception  
{ ...}
```

```
class MiaEccezione extends RuntimeException  
{ ...}s
```


Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
    [...]
```

```
    [...]  
    public static void main(String[] args) throws Ecc3 {  
        int i = UtilLeggiTastiera.leggiInteroPositivo  
            DaTastiera("Immetti un valore intero [0..]: ");  
        System.out.println("istr6");  
        try {  
            p(i);  
        } catch (Ecc1 e) {  
            System.out.println("istr7");  
        } catch (Ecc2 e) {  
            System.out.println("istr8");  
        }  
        System.out.println("istr9");  
    }  
}
```

Cosa succede per gli input 0, 1, 2 e 3?

Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
    [...]  
}
```

```
[...]  
public static void main(String[] args) throws Ecc3 {  
    int i = UtilLeggiTastiera.leggiInteroPositivo  
    DaTastiera("Immetti un valore intero [0..]: ");  
    System.out.println("istr6");  
    try {  
        p(i);  
    } catch (Ecc1 e) {  
        System.out.println("istr7");  
    } catch (Ecc2 e) {  
        System.out.println("istr8");  
    }  
    System.out.println("istr9");  
}
```

Input: 0

Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
    [...]
```

```
[...]  
public static void main(String[] args) throws Ecc3 {  
    int i = UtilLeggiTastiera.leggiInteroPositivo  
    DaTastiera("Immetti un valore intero [0..]: ");  
    System.out.println("istr6");  
    try {  
        p(i);  
    } catch (Ecc1 e) {  
        System.out.println("istr7");  
    } catch (Ecc2 e) {  
        System.out.println("istr8");  
    }  
    System.out.println("istr9");  
}
```

i: 0

p(0)

0 == 0

Input: 0

Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
}
```

```
[...]  
public static void main(String[] args) throws Ecc3 {  
    int i = UtilLeggiTastiera.leggiInteroPositivo  
    DaTastiera("Immetti un valore intero [0..]: ");  
    System.out.println("istr6");  
    try {  
        p(i);  
    } catch (Ecc1 e) {  
        System.out.println("istr7");  
    } catch (Ecc2 e) {  
        System.out.println("istr8");  
    }  
    System.out.println("istr9");  
}
```

i: 0

p(0)

0 == 0

Input: 0

Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
}
```

```
[...]  
public static void main(String[] args) throws Ecc3 {  
    int i = UtilLeggiTastiera.leggiInteroPositivo  
    DaTastiera("Immetti un valore intero [0..]: ");  
    System.out.println("istr6");  
    try {  
        p(i);  
    } catch (Ecc1 e) {  
        System.out.println("istr7");  
    } catch (Ecc2 e) {  
        System.out.println("istr8");  
    }  
    System.out.println("istr9");  
}
```

i: 0

p(0)

0 == 0

Input: 0

Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
    [...]
```

```
    [...]  
    public static void main(String[] args) throws Ecc3 {  
        int i = UtilLeggiTastiera.leggiInteroPositivo  
        DaTastiera("Immetti un valore intero [0..]: ");  
        System.out.println("istr6");  
        try {  
            p(i);  
        } catch (Ecc1 e) {  
            System.out.println("istr7");  
        } catch (Ecc2 e) {  
            System.out.println("istr8");  
        }  
        System.out.println("istr9");  
    }  
}
```

Input: 1

Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
    [...]
```

i: 0

1 == 0

0 == 0

[...]

```
public static void main(String[] args) throws Ecc3 {  
    int i = UtilLeggiTastiera.leggiInteroPositivo  
    DaTastiera("Immetti un valore intero [0..]: ");  
    System.out.println("istr6");  
    try {  
        p(i);  
    } catch (Ecc1 e) {  
        System.out.println("istr7");  
    } catch (Ecc2 e) {  
        System.out.println("istr8");  
    }  
    System.out.println("istr9");  
}
```

i: 1

p(1)

Input: 1

Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
    [...]
```

i: 0

1 == 0

0 == 0

[...]

```
public static void main(String[] args) throws Ecc3 {  
    int i = UtilLeggiTastiera.leggiInteroPositivo  
    DaTastiera("Immetti un valore intero [0..]: ");  
    System.out.println("istr6");  
    try {  
        p(i);  
    } catch (Ecc1 e) {  
        System.out.println("istr7");  
    } catch (Ecc2 e) {  
        System.out.println("istr8");  
    }  
    System.out.println("istr9");  
}
```

i: 1

p(1)

Input: 1

Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
}
```

i: 0

1 == 0

0 == 0

[...]

```
public static void main(String[] args) throws Ecc3 {  
    int i = UtilLeggiTastiera.leggiInteroPositivo  
    DaTastiera("Immetti un valore intero [0..]: ");  
    System.out.println("istr6");  
    try {  
        p(i);  
    } catch (Ecc1 e) {  
        System.out.println("istr7");  
    } catch (Ecc2 e) {  
        System.out.println("istr8");  
    }  
    System.out.println("istr9");  
}
```

i: 1

p(1)

Input: 1



Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
}
```

i: 0

1 == 0

0 == 0

[...]

```
public static void main(String[] args) throws Ecc3 {  
    int i = UtilLeggiTastiera.leggiInteroPositivo  
    DaTastiera("Immetti un valore intero [0..]: ");  
    System.out.println("istr6");  
    try {  
        p(i);  
    } catch (Ecc1 e) {  
        System.out.println("istr7");  
    } catch (Ecc2 e) {  
        System.out.println("istr8");  
    }  
    System.out.println("istr9");  
}
```

i: 1

p(1)

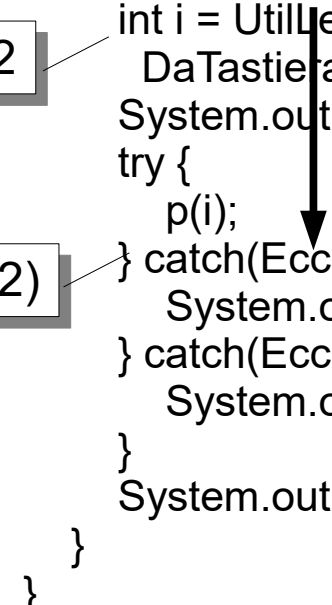
Input: 1

Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
    [...]
```

```
[...]  
public static void main(String[] args) throws Ecc3 {  
    int i = UtilLeggiTastiera.leggiInteroPositivo  
    DaTastiera("Immetti un valore intero [0..]: ");  
    System.out.println("istr6");  
    try {  
        p(i);  
    } catch (Ecc1 e) {  
        System.out.println("istr7");  
    } catch (Ecc2 e) {  
        System.out.println("istr8");  
    }  
    System.out.println("istr9");  
}
```



Input: 2

Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
    [...]
```

i: 1

i: 0

2 == 0

1 == 0

0 == 0

[...]

```
public static void main(String[] args) throws Ecc3 {  
    int i = UtilLeggiTastiera.leggiInteroPositivo  
    DaTastiera("Immetti un valore intero [0..]: ");  
    System.out.println("istr6");  
    try {  
        p(i);  
    } catch (Ecc1 e) {  
        System.out.println("istr7");  
    } catch (Ecc2 e) {  
        System.out.println("istr8");  
    }  
    System.out.println("istr9");  
}
```

i: 2

p(2)

Input: 2

Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
    [...]
```

i: 1

i: 0

2 == 0

1 == 0

0 == 0

[...]

```
public static void main(String[] args) throws Ecc3 {  
    int i = UtilLeggiTastiera.leggiInteroPositivo  
    DaTastiera("Immetti un valore intero [0..]: ");  
    System.out.println("istr6");  
    try {  
        p(i);  
    } catch (Ecc1 e) {  
        System.out.println("istr7");  
    } catch (Ecc2 e) {  
        System.out.println("istr8");  
    }  
    System.out.println("istr9");  
}
```

i: 2

p(2)

Input: 2

Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
    [...]  
}
```

```
[...]  
public static void main(String[] args) throws Ecc3 {  
    int i = UtilLeggiTastiera.leggiInteroPositivo(  
        DaTastiera("Immetti un valore intero [0..]. ");  
        System.out.println("istr6");  
        try {  
            p(i);  
        } catch (Ecc1 e) {  
            System.out.println("istr7");  
        } catch (Ecc2 e) {  
            System.out.println("istr8");  
        }  
        System.out.println("istr9");  
    }  
}
```

Input: 2

2 == 0

1 == 0

0 == 0

i: 2

p(2)

i: 1

i: 0

Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
    [...]
```

```
[...]  
public static void main(String[] args) throws Ecc3 {  
    int i = UtilLeggiTastiera.leggiInteroPositivo  
    DaTastiera("Immetti un valore intero [0..]: ");  
    System.out.println("istr6");  
    try {  
        p(i);  
    } catch (Ecc1 e) {  
        System.out.println("istr7");  
    } catch (Ecc2 e) {  
        System.out.println("istr8");  
    }  
    System.out.println("istr9");  
}
```

i: 3

p(3)

Input: 3

Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
    [...]  
}
```

```
[...]  
public static void main(String[] args) throws Ecc3 {  
    int i = UtilLeggiTastiera.leggiInteroPositivo  
    DaTastiera("Immetti un valore intero [0..]: ");  
    System.out.println("istr6");  
    try {  
        p(i);  
    } catch (Ecc1 e) {  
        System.out.println("istr7");  
    } catch (Ecc2 e) {  
        System.out.println("istr8");  
    }  
    System.out.println("istr9");  
}
```

Input: 3

3 == 0

i: 3

p(3)

2 == 0

1 == 0

i: 2

i: 1

i: 0

Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
    [...]  
}
```

i: 2

i: 1

i: 0

3 == 0

2 == 0

1 == 0

[...]

```
public static void main(String[] args) throws Ecc3 {  
    int i = UtilLeggiTastiera.leggiInteroPositivo  
    DaTastiera("Immetti un valore intero [0..]: ");  
    System.out.println("istr6");  
    try {  
        p(i);  
    } catch (Ecc1 e) {  
        System.out.println("istr7");  
    } catch (Ecc2 e) {  
        System.out.println("istr8");  
    }  
    System.out.println("istr9");  
}
```

i: 3

p(3)

Input: 3



Un esempio

```
class Ecc1 extends RuntimeException {}  
class Ecc2 extends RuntimeException {}  
class Ecc3 extends Exception {}
```

```
public class EsempioEccezioni {  
    public static void p(int i) throws Ecc3 {  
        System.out.println("istr0");  
        try {  
            if (i == 0) throw new Ecc1();  
            i--;  
            System.out.println("istr1");  
            if (i == 0) throw new Ecc2();  
            i--;  
            System.out.println("istr2");  
            if (i == 0) throw new Ecc3();  
            i--;  
            System.out.println("istr3");  
        } catch (Ecc2 e) {  
            System.out.println("istr4");  
        }  
        System.out.println("istr5");  
    }  
    [...]  
}
```

i: 2

i: 1

i: 0

3 == 0

2 == 0

1 == 0

[...]

```
public static void main(String[] args) throws Ecc3 {  
    int i = UtilLeggiTastiera.leggiInteroPositivo  
    DaTastiera("Immetti un valore intero [0..]: ");  
    System.out.println("istr6");  
    try {  
        p(i);  
    } catch (Ecc1 e) {  
        System.out.println("istr7");  
    } catch (Ecc2 e) {  
        System.out.println("istr8");  
    }  
    System.out.println("istr9");  
}
```

i: 3

p(3)

Input: 3