~ by Katsucurry 

Eccezioni

Definizione Una procedura/metodo può terminare normalmente o sollevare un' eccezione. Vengono segnalate al chiamante che può gestirle.

Le eccezioni hanno un tipo e dei dati associati che danno indicazione sul problema riscontrato. Possono essere definite dall'utente.

Keyword → throw
Seguito da un reference a un oggetto eccezione.

Definizione Un oggetto eccezione ha
- metodi e dati
- un tipo
- informazioni sul problema riscontrato.

Viene creato quando viene sollevata un'eccezione.
Cosa fa: termina l'esecuzione del blocco di codice;
propaga l'eccezione al chiamante.

Costrutto

```
try {
   // blocco di codice che potrebbe sollevare l'eccezione
} catch(TipoEccezione e){
   // codice di gestione: da eseguire quando si verifica l'eccezione
} catch (UnAltraExeception e){
   // gestione eccezione
}
```

OSS Meglio mettere nelle catch superiori i tipi di eccezioni più specifici.

Es

```
Rettangolo r = new Rettangolo (...);
n = ...; // numero da utente
try {
   r. setBase(n);
} catch (NonPositiveBaseException e){
  // codice per gestire l'eccezione
}
```

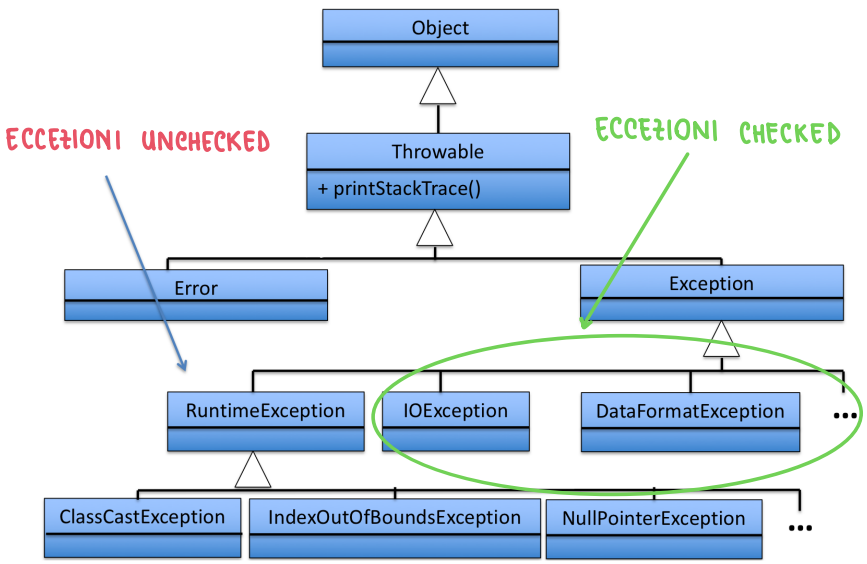
Propagazione delle eccezioni

Se viene sollevata un'eccezione:

I Termina exe blocco in cui si è verificata l'eccezione

II Passa il controllo al primo dei rami catch in grado di gestire l'eccezione
III Altrimenti, si risalgono blocchi di codice più esterni fino a trovare un blocco try/catch che contenga un ramo catch in grado di gestire l'eccezione
IV Altrimenti, il programma termina

Tipologie di eccezioni



Definizione

Throwable : classe generale di tutte le eccezioni
Ha un metodo public printStackTrace() il quale stampa il contenuto dello stack (i metodi contenuti) nel momento in cui viene sollevata un eccezione

Error : rappresenta errori che non dovrebbero mai accadere. Sono generati dalla JVM durante l'esecuzione del programma. Provocano sempre il crash.

Exception : rappresenta condizioni eccezionali che possono essere gestite dall'applicazione.

Eccezioni checked : lanciati dal programma e accade normalmente con utente. Il compilatore ci obbliga a gestirle, altrimenti non compilerebbe.

Eccezioni unchecked : possono essere lanciati dalla JVM e possono essere gestite dal programma, ma non necessariamente. Se il codice è corretto, non esistono queste eccezioni.

eccezioni checked

È importante che queste eccezioni vengano catturate dal programma.
Ci sono 2 opzioni per gestire un eccezione checked :
I Catturiamo l'eccezione e la gestiamo.

esempio

```
public void setBase(int base){
    try{
        if (base <= 0)
            throw new NonPositiveBaseException();
        this.base = base;
    } catch(NonPositiveBaseException e){
        // codice per risolvere
    }
}
```

Osservazione Questa opzione ha senso se sappiamo come gestire questo caso.

II Informiamo il chiamante che c'é una eccezione che deve essere catturata e gestita.

esempio

```
public void setBase(int base) throws new NonPositiveBaseException{
    if (base <= 0)
        throw new NonPositiveBaseException();
    this.base = base;
}

public void creaRettangolo(){
    Rettangolo r = new Rettangolo();
    boolean successo = false;
    double base = // leggi il valore da utente
    while(!successo){
        try{
            r.setBase(base);
            successo = true;
        } catch(NonPositiveBaseExeception e){
            base = ... // leggi di nuovo valore da utente
        }
    }
}
```

Attenzione Uno dei metodi nella catena di invocazioni DEVE gestire l'eccezione.

eccezioni unchecked

È importante che vengano gestiti in quanto potrebbe dare errore a runtime.

esempio metodo che ritorna il minimo di un array di numeri positivi, -1 se array vuoto.

```
public int min(int[] a){
    int minValue;
    try{
```

```
            m = a[0];
        } catch (IndexOutOfBoundsException e) {
            return -1;
        }
        for (int i = 1; i < a.length; i++)
            if (a[i] < m) m = a[i];
        return m;
    }
```

Sintassi

implementazione

```
public class ExceptionName extends Exception{
    public ExceptionName () { super(); }
    public ExceptionName (String s){
        super(s);
    }
}
```

uso

```
throw new ExceptionName ();
```

gestione

```
try {
    ...
} catch (ExceptionName e){
    ...
}
```

ramo Finally

Definizione

Il ramo finally è un ramo che viene eseguito all'interno del blocco Try sia quando non viene sollevata un'eccezione o alla fine della gestione dell'eccezione.

Sintassi

```
try {
    ...
} catch (Exception1 e) {
    ...
} catch (Exception2 e) {
    ...
} finally {
    ...
}
```

Strategie di gestione delle eccezioni

masking → gestione dell'eccezione + ripresa dell'esecuzione

forwarding → l'eccezione non può essere gestita + ritorna l'eccezione

re-throwing → l'eccezione viene catturata ma non gestita + genera eccezione di tipo differente