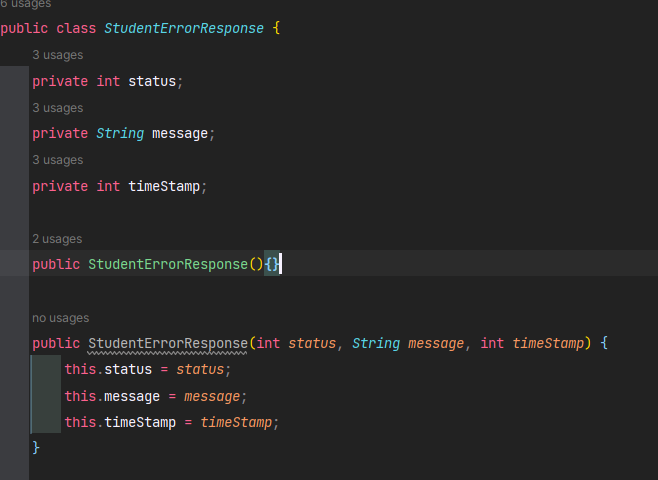
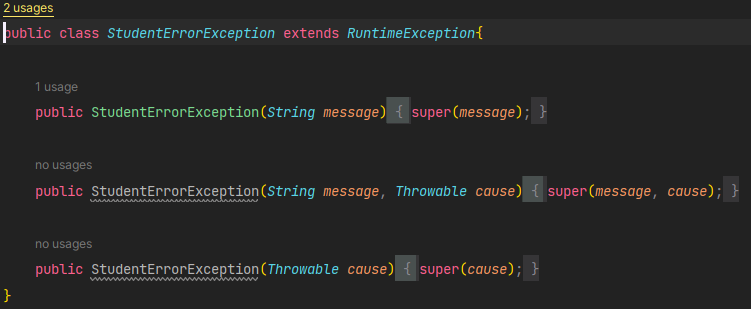
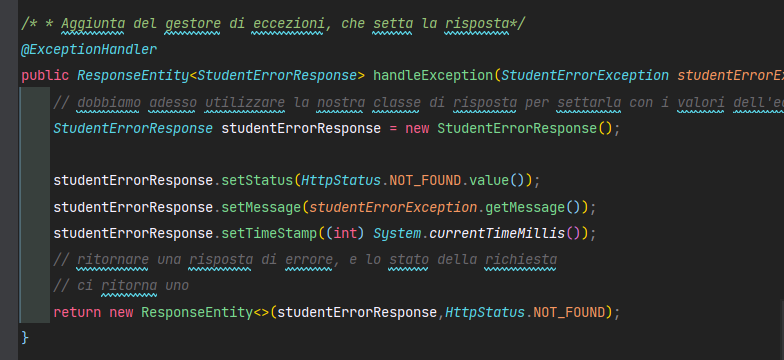
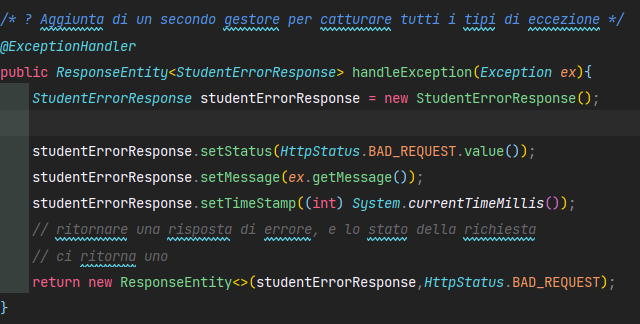
1) **personalizzare un messaggio di risposta qunado si effettua una richiesta non valida:**   
 Possiamo gestire un messaggio personalizzato qunado magari viene lanciata una richiesta per la quale non vogliamo fornire informaizoni o renderizzare direttamente il messaggio di errore.  
   
I passaggi per creare un messaggio personalizzato e quindi creare un eccezione sono:

1. creare una classe java che lancerà il messaggio, quindi una risposta personalizzata, questa classe avrà 3 campi (-(int)status -(String)message -(int)timeStamp).  
     
   
2. creare una classe che estenderà RunTimeException, aggiungere dal 2 al penultimo costruttore  
   

fin’ora abbiamo il codice per lanciare l’eccezione ma nessun componente che la gestisca, quindi viene aggiunto un gestore di eccezioni @ExceptionHandler

1. creare all’interno di metodo un istruzione che ti permetta di lanciare l’eccezione, dentro la classe che utilizzerà l’eccezione



1. creare un secondo gestore per catturare tutti i tipi di eccezioni  
   

\* Aggiunta dell'annotazione @ExceptionHandler per dichiarare quel metodo, devo dire a spring che questo metodo è un gestore di eccezioni  
 {spiegazione del metodo @ExceptionHandler:  
 - passare come parametro la classe exception creata, affinche venga catturata l'eccezione  
 - creare uno studentErrorResponse, quindi un messaggio di risposta con la classe che hai creato  
 - ritornare l'entità di risposta  
  
ma non abbiamo finito inserendo una letterà comparirà badRequest  
  
Per risolvere quest'altro tipo di eccezione, aggiungiamo un metodo che catturi qualsiasi tipo di eccezione, Sarà renderizzato il messaggio di errore.  
  
-BAD REQUEST quando viene fatta una richiesta totalmente sbagliata per la logica del nostro server  
-NOT FOUND qunado viene fatta una richiesta per la quale ci sia un senso, richiesta non trovata  
  
quindi ciò che abbiamo fatto è aggiungere un gestore per alcune eccezioni, e un gestore di tutte le eccezioni  
i quali ci restituiscono un JSON, che informa sullo stato, erroreMessagge, timeStamp.  
  
Ma in questo modo, non abbiamo gestori globali ma legati alla singola classe, al singolo oggetto, non permettendo il riuso  
  
Spring a proprosito di ciò ci suggerisci di aggiungere un **@ControllerAdvice**, ossia un controller simile ad un intercettore, filtro. Quindi possiamo usarlo per **pre-processare le richieste dei controllori**, ma anche per post-processare le risposte e gestire le eccezioni  
  
In questo modo avermo un gestore globale delle eccezioni e non bloccato in un singolo controller  
  
quindi prima che la richiesta raggiunga il service-rest (controller) questa viene processata raggiungendo in segutio il servizio, e se c’è qualcosa di sbagliato questo lancierà un’eccezione  
  
   
  
rientrando nell’utilizzo di AspectOrientedProgramming

quindi possiamo organizzare le classi per gestire le eccezioni in un’unico package chiamato excpetion.