Esercitazione 1: 20240913.1

L'obiettivo di queste esercitazioni è di accumulare esperienza con il codice sorgente. Carica questi esercizi nella tua repository GitHub e condividi il codice con Giovanni inviando l'elenco dei link delle repository alla mail: g.pace@braintechinnovations.it

---

Esercizio 1: Calcolatrice Base

Obiettivo:

Creare una semplice calcolatrice che esegua operazioni matematiche di base: somma, sottrazione, moltiplicazione e divisione.

Requisiti:

1. L'applicazione deve permettere all'utente di:
    - Inserire due numeri.
    - Scegliere l'operazione da eseguire (somma, sottrazione, moltiplicazione, divisione).
    - Visualizzare il risultato dell'operazione.
2. **Gestione degli errori**:
    - Se l'utente tenta di eseguire una divisione per zero, il programma deve mostrare un messaggio d'errore senza andare in crash.
    - L'applicazione deve gestire eventuali inserimenti non validi (es. caratteri anziché numeri).
3. **Operazioni**:
    - Implementare un ciclo che permetta all'utente di eseguire più operazioni senza dover riavviare il programma.
    - Il programma deve terminare solo quando l'utente lo desidera (es. inserendo 'Q').

Requisiti HARD:

- **H1**: Aggiungere la possibilità di eseguire operazioni aggiuntive come potenza e radice quadrata.
- **H2**: Implementare il calcolo su più numeri, ad esempio utilizzando un array.
- **H3**: Aggiungere il supporto alle operazioni con numeri decimali.
- **H4**: Implementare la gestione delle eccezioni per tutti gli input errati.

Esercizio 2: Gestione di una lista di studenti

Obiettivo:

Creare un'applicazione console che permetta la gestione di una lista di studenti, con la possibilità di inserire nuovi studenti, modificarne i dati, visualizzarli e filtrare la lista in base ai voti.

Requisiti:

1. L'applicazione deve consentire all'utente di:
    - Inserire nuovi studenti (nome, cognome, voto).
    - Visualizzare tutti gli studenti inseriti.
    - Modificare i dati di uno studente esistente.
    - Filtrare e visualizzare gli studenti in base a un intervallo di voti.
    - Eliminare uno studente dalla lista.
2. **Dettagli del modello Studente**:
    - Ogni studente deve essere rappresentato da una classe Studente con i seguenti attributi:
        - `string Nome`
        - `string Cognome`
        - `double Voto` (da 0 a 10).
3. **Funzionalità principali**:
    - Aggiungere uno studente alla lista.
    - Modificare i dati di uno studente esistente.
    - Visualizzare l'elenco completo degli studenti.
    - Filtrare gli studenti in base a un voto minimo e massimo (chiesto in due tempi).
    - Eliminare uno studente dalla lista in base al nome.
4. **Gestione degli errori**:
    - Validare che i voti siano compresi tra 0 e 10.
    - Gestire input non validi o vuoti.
    - Fornire feedback appropriato in caso di errore.

Requisiti HARD:

- **H1**: Implementare un sistema di ricerca per nome o cognome (attenzione, la funzione è la stessa).
- **H2**: Visualizzare statistiche, come il voto medio, il voto massimo e il minimo.

Esercizio 3: Gestione di eccezioni

Obiettivo:

Creare un'applicazione console in C# che legga numeri da un file di testo, calcoli la somma di questi numeri e gestisca le eccezioni che potrebbero verificarsi durante l'esecuzione del programma.

Requisiti:

1. L'applicazione deve leggere un file di testo fornito dall'utente. Il file conterrà una serie di numeri, uno per riga.
2. **Operazioni principali**:
    - Leggere il contenuto del file.
    - Calcolare e visualizzare la somma di tutti i numeri presenti nel file.
    - Quali sono le eccezioni riscontrabili? Gestiscile tutte con un messaggio di errore personalizzato a seconda di quello che accade.
3. **Gestione degli errori**:
    - Ogni volta che si verifica un'eccezione, il programma deve continuare a funzionare senza terminare inaspettatamente, mostrando un messaggio d'errore all'utente.