ANALISI METODI

METODO: wrongIndexSearch

1. Scopo del metodo:

- Il metodo `wrongIndexSearch` è progettato per simulare un malfunzionamento durante la ricerca di un indice in un array di oggetti `Contatto`. L'obiettivo è quello di restituire un indice errato, che è diverso dall'indice corretto, per un certo numero di tentativi specificati casualmente.

2. Parametri:

- Il metodo accetta tre parametri:

- `gestore`: un array di oggetti `Contatto` che rappresenta il database dei contatti.

- `contrattiVenduti`: un intero che indica il numero attuale di contratti venduti, utile per limitare la ricerca all'interno di una porzione dell'array.

- `contatto`: un intero che rappresenta l'indice del contatto da cercare.

3. Generazione del numero di tentativi:

- Viene generato casualmente un numero tra 3 e 5 per determinare il numero di tentativi che il metodo eseguirà per trovare un indice errato.

4. Inizializzazione dell'indice sbagliato:

- L'indice sbagliato viene inizializzato a -1. Questo valore viene utilizzato per tracciare l'indice errato trovato durante l'ultimo tentativo.

5. Esecuzione dei tentativi:

- Il metodo esegue un loop per il numero di tentativi generato casualmente.

- Ad ogni iterazione, viene chiamato il metodo `RicercaIndex` per cercare l'indice corretto del contatto all'interno dell'array.

- Se l'indice trovato è lo stesso dell'indice errato dell'ultimo tentativo, viene generato un nuovo indice sbagliato casuale, diverso dall'ultimo.

- Se l'indice trovato è diverso dall'indice sbagliato dell'ultimo tentativo, viene restituito come indice corretto.

- Se non viene trovato alcun indice errato dopo tutti i tentativi, viene restituito l'indice corretto trovato dall'ultimo tentativo.

6. Gestione dei casi limite:

- Il metodo gestisce correttamente il caso in cui il numero di contratti venduti sia 0 o l'array dei contatti sia vuoto, restituendo correttamente -1 come indice.

7. Gestione dell'errore:

- Durante ogni tentativo, se viene trovato un indice sbagliato, viene stampato un messaggio di errore che indica l'indice sbagliato trovato.

8. Restituzione dell'indice corretto:

- Se non viene trovato alcun indice errato, il metodo restituisce l'indice corretto trovato dall'ultimo tentativo.

METODO evenOddSearch

1. Scopo del metodo:

- Il metodo `EvenOddSearch` è progettato per acquisire input numerici dall'utente, ordinare i numeri pari in ordine crescente e i numeri dispari in ordine decrescente e stampare l'array risultante.

2. Parametri:

- Il metodo accetta un parametro:

- `keyboard`: un oggetto `Scanner` utilizzato per l'input da tastiera.

3. Inizializzazione delle variabili:

- Viene inizializzata una variabile `continuare` a 1 per consentire all'utente di decidere se continuare ad inserire numeri.

- Viene creato un array di interi `numeri` con dimensione massima di 100.

- Viene inizializzata una variabile `count` a 0 per tenere traccia del numero di elementi inseriti nell'array.

4. Input dei numeri:

- Viene eseguito un ciclo `do-while` per consentire all'utente di inserire numeri finché l'utente desidera continuare o fino a quando non vengono inseriti 100 numeri.

- Durante ogni iterazione, l'utente inserisce un numero e viene chiesto se desidera continuare. Il numero inserito viene memorizzato nell'array `numeri` e `count` viene incrementato.

5. Ordinamento dei numeri pari e dispari:

- Dopo aver acquisito tutti i numeri, viene utilizzato il metodo `Arrays.sort` per ordinare gli elementi dell'array.

- Viene utilizzata una lambda expression per definire il criterio di ordinamento. I numeri pari vengono ordinati in modo crescente, mentre i numeri dispari vengono ordinati in modo decrescente.

- La lambda expression confronta due numeri e restituisce un valore negativo se il primo numero è minore del secondo (per ordinamento crescente), un valore positivo se il primo numero è maggiore del secondo (per ordinamento decrescente) e zero se i numeri sono uguali.

6. Stampa dell'array ordinato:

- Dopo l'ordinamento, viene stampato l'array `numeri` per mostrare i numeri pari in ordine crescente seguiti dai numeri dispari in ordine decrescente.

METODO biBinarySearch**:**

1. Scopo del metodo:

- Il metodo `biBinarySearch` è progettato per eseguire una ricerca binaria all'interno di un array ordinato, al fine di trovare la posizione di un dato valore. È chiamato dopo aver eseguito `EvenOddSearch` per cercare un valore nell'array risultante.

2. Parametri:

- Il metodo accetta due parametri:

- `array`: un array di interi su cui eseguire la ricerca.

- `target`: il valore da cercare nell'array.

3. Implementazione della ricerca binaria:

- Il metodo utilizza l'algoritmo di ricerca binaria, che è efficiente per la ricerca in array ordinati.

- Inizia definendo due variabili `low` e `high` per indicare gli estremi dell'array.

- Il metodo itera finché `low` è minore o uguale a `high`.

- Ad ogni iterazione, calcola l'indice medio dell'array.

- Se il valore nella posizione media è uguale al valore cercato (`target`), restituisce l'indice medio.

- Se il valore nella posizione media è maggiore del valore cercato, imposta `high` al valore medio meno uno.

- Se il valore nella posizione media è minore del valore cercato, imposta `low` al valore medio più uno.

4. Restituzione del risultato:

- Se il valore è trovato nell'array, viene restituito l'indice medio.

- Se il valore non è presente nell'array, il metodo restituisce -1.

5. Gestione del caso di array vuoto:

- Se l'array è vuoto, viene restituito immediatamente -1.

6. Riutilizzo dell'array risultante da `EvenOddSearch`:

- Dato che `EvenOddSearch` ordina l'array, è garantito che sia ordinato quando viene passato a `biBinarySearch`.

- Questo permette di utilizzare efficacemente l'algoritmo di ricerca binaria.