1 Teoria

- Che cos'è un'eccezione in Java? Qual è il suo scopo?
- Qual è la differenza tra errori e eccezioni?
- Quali sono le principali classi di eccezioni in Java?
- Che cos'è il blocco `try-catch`? Come funziona?
- Cosa significa lanciare (throw) un'eccezione? Come si fa?
- Che cos'è il blocco `finally`? Quando viene eseguito?
- Qual è la differenza tra eccezioni controllate (checked) e non controllate (unchecked)?
- Come si crea una propria eccezione personalizzata?
- Che ruolo ha la parola chiave `throws` nella dichiarazione di un metodo?
- Perché è importante gestire correttamente le eccezioni?

2 Eccezioni e Gestione degli Errori

1. **Divisione per zero**
   Scrivi un programma che chiede due numeri interi e calcola la divisione. Gestisci l'eccezione di divisione per zero con un blocco `try-catch`.
2. **Parsing numerico**
   Scrivi un programma che chiede una stringa e prova a convertirla in un numero intero. Gestisci l'eccezione `NumberFormatException`.
3. **Accesso a array**
   Scrivi un programma che accede a un elemento di un array dato un indice inserito dall'utente. Gestisci l'eccezione `ArrayIndexOutOfBoundsException`.
4. **Lancio di eccezione personalizzata**
   Crea una classe di eccezione personalizzata chiamata `ValoreNonValidoException`. Scrivi un metodo che lancia questa eccezione se un valore passato è negativo.
5. **Blocco finally**
   Scrivi un programma con un blocco `try-catch-finally` che stampa un messaggio sia in caso di eccezione sia alla fine dell'esecuzione.
6. **Propagazione eccezioni**
   Scrivi un metodo che lancia un'eccezione e un metodo chiamante che la gestisce con un blocco `try-catch`.

7. **Multipli catch**
   Scrivi un esempio di codice che gestisce diverse eccezioni (`ArithmeticException`, `NullPointerException`) con più blocchi `catch`.

8. **Eccezioni runtime**
   Spiega e mostra un esempio di eccezione non controllata (`NullPointerException`).

9. **Throws in dichiarazione metodo**
   Scrivi un metodo che dichiara di lanciare un'eccezione con `throws` e mostra come chiamarlo da un altro metodo.

10. **Try-with-resources**
    Spiega cos'è il try-with-resources e scrivi un esempio di utilizzo con un `BufferedReader` per leggere un file (puoi simulare il codice senza file esterno).