Analisa Penggunaan try dan catch

---

1. **Blok try**:
    - Blok ini digunakan untuk membungkus kode yang mungkin menghasilkan kesalahan. Dalam hal ini, kode yang mencoba membagi **bil** dengan 0 (**bil / 0**).
    - Jika kesalahan terjadi dalam blok **try**, eksekusi segera beralih ke blok **catch** yang sesuai.
2. **Blok catch (ArithmeticException e)**:
    - Blok ini menangani kesalahan spesifik **ArithmeticException** yang terjadi ketika mencoba membagi sebuah bilangan dengan 0.
    - Di dalam blok ini:
        - **System.out.println("Pesan error: ");** mencetak pesan awal yang menunjukkan adanya error.
        - **System.out.println(e.getMessage());** mencetak pesan kesalahan yang ditangkap oleh exception.
        - **System.out.println("Info stack erase");** mencetak pesan yang menunjukkan bahwa stack trace akan ditampilkan.
        - **e.printStackTrace();** dan **e.printStackTrace(System.out);** mencetak stack trace error ke konsol dan output standar, yang membantu dalam mendiagnosis dan memahami di mana dan mengapa kesalahan terjadi.
3. **Blok catch (Exception e)**:
    - Blok ini adalah penanganan umum untuk semua jenis exception yang tidak secara spesifik ditangkap oleh blok **catch** sebelumnya.
    - Di dalam blok ini, pesan **System.out.println("Ini menghandle error yang terjadi");** dicetak jika ada kesalahan selain **ArithmeticException**.

Kesimpulan

- Program ini secara efektif menggunakan **try** dan **catch** untuk menangani kesalahan secara spesifik (**ArithmeticException**) dan umum (**Exception**).
- Dengan menggunakan **e.getMessage()** dan **e.printStackTrace()**, program memberikan informasi rinci tentang kesalahan yang terjadi, yang sangat berguna untuk debugging.
- Urutan penanganan exception memastikan bahwa kesalahan aritmatika ditangani terlebih dahulu, sementara kesalahan lainnya ditangani oleh blok catch umum.