Nama : Restu Wibisono

NPM : 2340506061

1. Jelaskan konsep dasar dari Exception Handling dan mengapa penting dalam pemrograman berorientasi objek?

   Exception Handling adalah mekanisme pemrograman untuk menangani kesalahan yang terjadi pada eksekusi program. Dalam PBO, exception handling ini penting untuk membantu integrasi aplikasi dengan menangkap dan mengelola kesalahan yang tidak terduga, Dengan ini pengembang dapat menghindari crash program dan memberikan pengalaman yang lebih baik.

2. Apa perbedaan antara Checked Exception dan Unchecked Exception? Berikan contoh masing-masing, dan jelaskan situasi kapan masing-masing jenis exception ini digunakan.

   Checked Exception adalah jenis exception yang harus ditangani pengembang, dengan menggunakan blok try-catch. dalam signature metode menggunakan kata kunci throws. Contohnya adalah IOException yang mana digunakan adalah untuk berinteraksi dengan sumber ekxternal, seperti file atau jaringan, dan dimana kesalahan itu terjadi.

   Contoh:

   ```
   public void readFile(String filePath) throws IOException {
      FileReader fileReader = new FileReader(filePath);
   }
   ```

3. Bagaimana mekanisme kerja blok try-catch dalam Java? Jelaskan fungsi setiap bagian dari blok ini dan mengapa penggunaan blok ini disarankan.

   Blok try-catch berfungsi untuk menangkap exception yang muyngkiun terjadi saat dalam eksekusi kode dalam blok try. Penggunaan blok ini memungkinkan program untuk terus berjalan meskipun ada kesalahan, yang akan memberikan kesempatan unruk menangani kesalahan dengan cara yang tepat.

   Contoh:

   ```
   try {
      int result = 10 / 0;
   ```

```
} catch (ArithmeticException e) {
    System.out.println("Tidak dapat membagi dengan nol: " + e.getMessage());
}
```

4. Apa tujuan dari blok finally dalam Exception Handling? Jelaskan situasi di mana blok finally akan selalu dijalankan, dan berikan contoh penggunaannya dalam konteks membuka dan menutup file.

   Blok finally selalu dijalankan setelah eksekusi blok try dan catch, terlepas dari apakah exception terjadi atau tidak. Blok ini biasanya digunakan untuk mengeksekusi kode pembersihan, seperti menutup file atau melepaskan sumber daya.

   Contoh:

```
try {
    fileReader = new FileReader("file.txt");
} catch (IOException e) {
    System.out.println("Error: " + e.getMessage());
} finally {
    if (fileReader != null) {
        try {
            fileReader.close();
        } catch (IOException e) {
            System.out.println("Error closing file: " + e.getMessage());
        }
    }
}
```

5. Jelaskan bagaimana penggunaan kata kunci throw dan throws dalam Java untuk Exception Handling. Apa perbedaan keduanya, dan kapan sebaiknya masing-masing digunakan?

   Throw digunakan untuk membuat exception secara explisit dari dalam metode, sementara throws digunakan dalam dignature yang mendeklarasikan bahwa metode itu dapat melempet exception tertentu.

   Contoh:

```
public void validate(int age) {
    if (age < 18) {
        throw new IllegalArgumentException("Usia harus lebih dari 18.");
    }
}
```

6. Mengapa diperlukan custom exception? Jelaskan bagaimana cara membuat custom exception dalam Java, dan berikan contoh situasi di mana custom exception akan sangat membantu dalam penanganan error.

   Custom exception diperlukan utnuk situasi yang spesifik dan sudah dijelaskan dengan esception yang biasanya (sudah ada). Dengan membuat custom ini pengembang dapat memberikan informasi lebih lanjut mengenai kesalahan yang ada.

   Contoh:

```
public class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
}

public void setAge(int age) throws InvalidAgeException {
    if (age < 0) {
        throw new InvalidAgeException("Usia tidak boleh negatif.");
    }
}
```

7. Sebutkan tiga jenis exception yang sering terjadi pada aplikasi Java dan jelaskan penyebab terjadinya masing-masing exception tersebut.
   a. NullPointerException: Terjadi ketika mencoba mengakses objek yang belum diinisialisasi (null).

b. ArrayIndexOutOfBoundsException: Terjadi saat mencoba mengakses indeks yang di luar batas array.
c. IOException: Terjadi saat ada masalah dalam input/output, seperti kesalahan saat membaca atau menulis file.

8. Bagaimana penerapan Exception Handling dapat meningkatkan keamanan dan stabilitas aplikasi? Jelaskan dengan contoh atau skenario.

   Mneggunakan exception handling, aplikasi bisa menangani kesalahan dnegan cara yang terkontrol. Seperti, aplikasi web ada kesalahan untuk mengakses database, exception ini memungkinkan untuk memberi pesan kepada mengguna tanpa menunjukkan informasi yang sensitif dan tidak crash.

9. Jelaskan apa yang akan terjadi jika sebuah exception tidak ditangani dalam program (tidak ada blok try-catch). Bagaimana cara sistem menangani hal tersebut, dan bagaimana hal ini dapat memengaruhi user experience?

   Jika sebuah exception tidak ditangani, Java akan melempar exception tersebut ke level atas, yang dapat menyebabkan program crash. Pengguna akan melihat pesan kesalahan yang mungkin tidak informatif, yang dapat mengganggu pengalaman pengguna.

10. Diskusikan kelebihan dan kekurangan dalam menggunakan Exception Handling. Kapan sebaiknya kita tidak menggunakan Exception Handling dalam suatu blok kode?

    Kelebihan:

    a. Meningkatkan keandalan dan stabilitas aplikasi.
    b. Memungkinkan penanganan kesalahan yang terkontrol.

    Kekurangan:

    a. Dapat menambah kompleksitas kode.
    b. Penggunaan berlebihan dapat menyebabkan penurunan kinerja.
    c. Sebaiknya hindari menggunakan exception handling dalam blok kode yang sangat cepat dieksekusi, karena dapat mempengaruhi kinerja aplikasi.