**Bab 9**

**Exception Handling**

**Tujuan**

1. Praktikan mampu mengetahui konsep exception handling
2. Praktikan mampu melakukan exception handling pada kode program

**Ringkasan Materi**

1. **Exception**

Exception adalah kondisi atau event yang terjadi ketika runtime yang mengganggu alur instruksi pada kode program yang sebenarnya. Exception biasa terjadi jika ada runtime error yang disebabkan oleh: null pointer exception, I/O error, index out of bounds, division by zero dan sebagainya. Ketika exception tidak dihandle dengan baik maka program akan seketika crash.

1. **Checked Exception & Unchecked Exception**

Exception memiliki 2 tipe exception, checked dan unchecked. Checked exception adalah exception yang dapat dicek ketika proses compile kode program. Compiler akan menampilkan pesan error jika ada kode program yang berpeluang melemparkan checked exception jika tidak dihandle dengan baik. Perhatikan potongan kode dibawah

|  |
| --- |
| import java.io.\*;    class Main {      public static void main(String[] args) {          FileReader file = new FileReader("/home/box/a.txt");          BufferedReader fileInput = new BufferedReader(file);            // Print first 3 lines of file "/home/box/a.txt"          for (int counter = 0; counter < 3; counter++)              System.out.println(fileInput.readLine());            fileInput.close();      }  } |

Ketika dicompile, maka akan muncul error

|  |
| --- |
| Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - unreported exception java.io.FileNotFoundException; must be caught or declared to be thrown at Main.main(Main.java:5) |

Hal ini dikarenakan method main() menggunakan FileReader() dan BufferedReader() yang dapat melemparkan checked exception FileNotFoundException, sedangkan readLine() dan close() dapat melemparkan checked exception IOException. Untuk membenahinya, maka ada 2 cara, mendefinisikan IOException didalam throws list dalam sebuah method atau menggunakan blok try catch.

|  |
| --- |
| import java.io.\*;    class Main {      public static void main(String[] args) **throws IOException** {          FileReader file = new FileReader("/home/box/a.txt");          BufferedReader fileInput = new BufferedReader(file);            // Print first 3 lines of file "/home/box/a.txt"          for (int counter = 0; counter < 3; counter++)              System.out.println(fileInput.readLine());            fileInput.close();      }  } |

Atau

|  |
| --- |
| import java.io.\*;    class Main {      public static void main(String[] args) {  try {           FileReader file = new FileReader("/home/box/a.txt");           BufferedReader fileInput = new BufferedReader(file);             // Print first 3 lines of file "/home/box/a.txt"           for (int counter = 0; counter < 3; counter++)               System.out.println(fileInput.readLine());             fileInput.close();  } catch (IOException e) {  e.printStackTrace();  }      }  } |

Sedangkan unchecked exception adalah jenis exception yang tidak dicek ketika waktu compile. Runtime exception seperti ArithmeticException merupakan unchecked exception yang mana program akan crash ketika pada runtime melemparkan exception ini.

|  |
| --- |
| class Main {  public static void main(String args[]) {  int x = 0;  int y = 10;  int z = y/x;  }  } |

Java memiliki kedua jenis exception ini, sedangkan C++ semua exception merupakan unchecked exception.

1. **Catch block**

Dalam Java dan C++, untuk melakukan exception catch, maka digunakanlah blok catch.

|  |
| --- |
| try {  // Code that will possible to throw an exception  } catch (// exception type) {  // do something when exception has been caught  } |

1. **Throw keyword**

Developer bisa melemparkan exception sendiri jika diperlukan dengan menggunakan keyword throw

Java

|  |
| --- |
| static double division(int a, int b) throws Exception {  if (b == 0) {  throw new Exception("division by zero!");  }    return a / b;  }    public static void main (String[] args) {  try {  double x = division(9, 0);  } catch (Exception e) {  System.out.println(e.getMessage());  }  } |

C++

|  |
| --- |
| double division(int a, int b) {  if (b == 0) {  throw "division by zero!";  }    return a / b;  }  int main() {  try {  double x = division(9, 0);  } catch (const char msg) {  cout << msg << endl;  }  return 0;  } |

1. **Finally block (Java only)**

Kode program dalam blok finally akan selalu dijalankan ketika kode program yang berada dalam blok try maupun catch telah dijalankan.

|  |
| --- |
| static double division(int a, int b) throws Exception {  if (b == 0) {  throw new Exception("division by zero!");  }    return a / b;  }    public static void main (String[] args) {  try {  double x = division(9, 0);  } catch (Exception e) {  System.out.println(e.getMessage());  } finally {  System.out.println(“Horee”);  }  } |

**Pelaksanaan Percobaan**

Ketikkan snippet code dibawah (pilih salah satu bahasa yang digunakan dalam praktikum) dan jawablah pertanyaan dibawah!

Java

|  |
| --- |
| public class ExcTest {  static double division(int a, int b) throws Exception {  if (b == 0) {  throw new IllegalArgumentException("division by zero!");  }    return a / b;  }    public static void main (String[] x) {  try {  double x = division(9, 0);  } catch (Exception e) {  System.out.println(e.getMessage());  } finally {  System.out.println(“Horee”);  }  String yy;  String sub = yy.substring(0, yy.length() – 3);  }} |

C++

|  |
| --- |
| #include <iostream>  using namespace std;  double division(int a, int b) {  if (b == 0) {  throw "division by zero!";  }    return a / b;  }  int main(int argc, char\*\* argv) {  try {  double x = division(9, 0);  } catch (const char msg) {  cout << msg << endl;  }  string yy = NULL;  string sub = yy.substr(0, yy.length() – 3);  return 0;  } |

1. Jalankan kode program dan benahi jika ada kesalahan!
2. Jelaskan secara lengkap bagaimana pembenahan kesalahan yang telah dilakukan di langkah pertama!
3. (C++) jika parameter method main diganti dengan void, dan return value method main diganti dengan void, apa yang terjadi? Jelaskan secara lengkap!
4. (Java) jika pada method division(int, int) statement throws Exception dihapus, apa yang terjadi? Jelaskan secara lengkap!
5. Jelaskan secara lengkap menurut pendapat Anda mengenai null pointer exception, bagaimana bisa terjadi, bagaimana mereproduksi exception tersebut, dan penanggulannya!

**Tugas Praktikum**

1. Buatlah sebuah class turunan dari Exception baik di Java maupun C++ (pilih salah satu bahasa sesuai apa yang digunakan di praktikum) yang digunakan sebagai tipe exception yang dapat dilemparkan ketika menghitung n faktorial dimana n < 0. Buat method untuk menghitung faktorial terlebih dahulu!
2. Lakukan reproduksi exception dan handlingnya dalam beberapa kasus dibawah (pilih minimal 2)
   1. I/O Exception
   2. Stack Overflow Exception
   3. Array index out of bound exception