

ใบงานการทดลองที่ 12

เรื่อง การใช้งานคำสั่ง try catch และ throw exception

1. จุดประสงค์ทั่วไป

- 1.1. รู้และเข้าใจการใช้วัตถุ การทำหลายงานพร้อมกัน และการติดต่อระหว่างงาน
- 1.2. รู้และเข้าใจการจัดการกับความผิดปกติในการเขียนโปรแกรมเชิงวัตถุ

2. เครื่องมือและอุปกรณ์

เครื่องคอมพิวเตอร์ 1 เครื่อง ที่ติดตั้งโปรแกรม Eclipse

3. ทฤษฎีการทดลอง

- 3.1. Java Exception คืออะไร? อธิบายพร้อมยกตัวอย่างประกอบ

Exception คือการที่โปรแกรมพยายามจะทำงานบางอย่าง แต่เกิดข้อผิดพลาดขึ้น แล้วโปรแกรมไม่สามารถจัดการข้อผิดพลาดนั้นได้ ซึ่งทำให้เกิด exception ขึ้น และส่งผลทำให้โปรแกรมหยุดทำงาน

Exception เกิดขึ้นในขณะที่โปรแกรมทำงาน ยกตัวอย่างเช่น โปรแกรมกำลังจะเปิดไฟล์ขึ้นมา แต่ไฟล์ที่ต้องการไม่มีอยู่ เป็นต้น รูปแบบการใช้ exception ในภาษา Java

```
try {
    // try to do something
} catch (Exception1 ex1) {
    // handle for exception 1
}

...

} catch (ExceptionN exN) {
    // handle for exception N
} finally {
    // Always proceed this block whether
    // an exception is thrown or not
}
```

- 3.2. คำสั่ง try มีลักษณะการทำงานอย่างไร?

เป็นส่วนของโปรแกรมที่อาจจะทำให้เกิด exception ขึ้น

- 3.3. คำสั่ง catch มีลักษณะการทำงานอย่างไร?

ในแต่ละ catch บล็อกเป็นการจัดการกับ exception แต่ละแบบ

- 3.4. คำสั่ง finally มีลักษณะการทำงานอย่างไร?

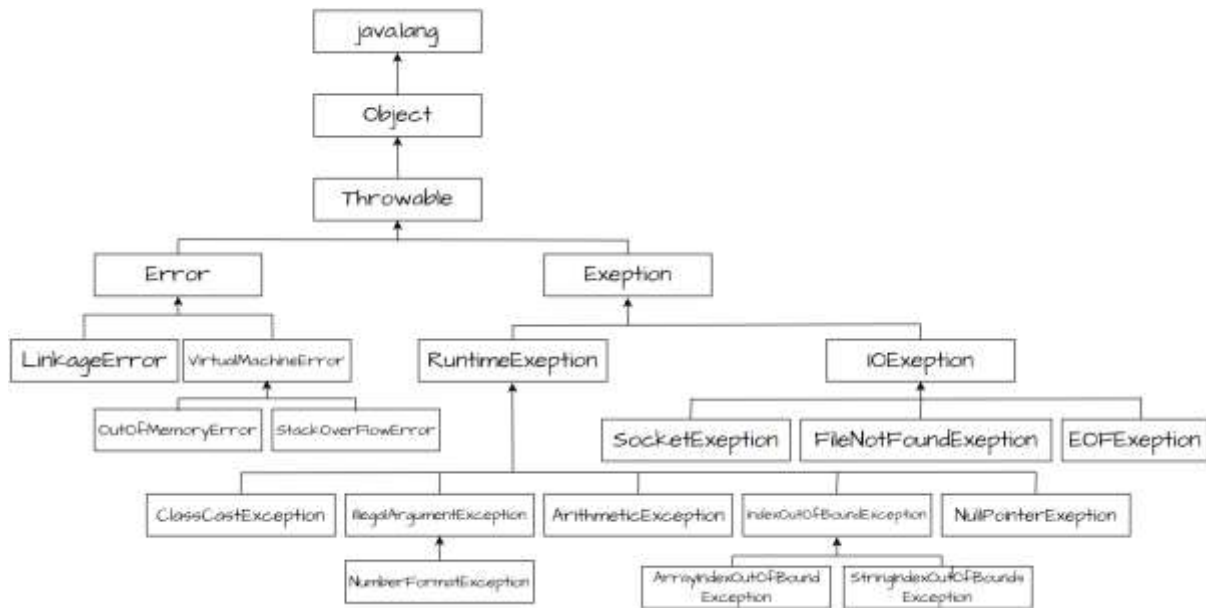
โปรแกรมจะเข้ามาทำงานเสมอไม่ว่าจะเกิด exception ในบล็อกของคำสั่ง try หรือไม่ก็ตาม

- 3.5. ลักษณะโครงสร้างของคำสั่ง try catch เป็นอย่างไร? อธิบายพร้อมยกตัวอย่างประกอบ

```
try {
    // try to do something
} catch (Exception1 ex1) {
    // handle for exception 1
}
```

4. ลำดับชั้นการปฏิบัติการ

- 4.1. จากผังงานต่อไปนี้ จงเขียนโค้ดโปรแกรมเพื่อแสดงตัวอย่างการจัดการความผิดปกติของคลาสการจัดการสิ่งผิดปกติจนครบทุกคลาส (เน้นเฉพาะส่วนของ Error และ Exception)



ตัวอย่างโค้ดโปรแกรมการจัดการสิ่งผิดปกติในส่วนของ **Error**

```
162         try {
163             check(5);
164         } catch (Error e) {
165             System.out.println("Error");
166         } // End try..catch ---| Error
167
168
169         System.out.println(" This is LinkageError");
170
171
172         try {
173             check(5);
174         } catch (VirtualMachineError e) {
175             System.out.println(" This is VirtualMachineError");
176         } // End try..catch ---| VirtualMachineError
177
178         try {
179             int arrSize = 15;
180             long memoryConsumed = 0;
181
182             long[] memoryAllocated = null;
183             for (int loop = 0; loop < Integer.MAX_VALUE; loop++) {
184                 memoryAllocated = new long[arrSize];
185                 memoryAllocated[0] = 0;
186                 memoryConsumed += arrSize * Long.SIZE;
187                 arrSize *= arrSize * 2;
188                 Thread.sleep(100);
189             }
190         } catch (OutOfMemoryError e) {
191             System.out.println(" ----| OutOfMemoryError");
192         } // End try..catch ---| OutOfMemoryError
193
194         try {
195             check(5);
196         } catch (StackOverflowError e) {
197             System.out.println(" ----| StackOverflowError");
198         } // End try..catch ---| StackOverflowError
199
200     }
```

ตัวอย่างโค้ดโปรแกรมการจัดการสิ่งผิดปกติในส่วนของ Exeption

```
1 package lap12try;
2 import java.io.DataInputStream;
15 public class lap12 {
16     public static BigDecimal addOne(Object obj){
17         BigDecimal a = (BigDecimal)obj;
18         return a.add(BigDecimal.ONE);
19     }
20
21     public static void a() { b(); }
22     public static void b() { c(); }
23     public static void c() {
24         throw new IllegalStateException("test");
25     }
26
27
28     private static void createConnection() throws Exception, IOException {
29
30         String host = null;
31         int port = 0;
32         Socket socket = new Socket(host, port);
33
34     }
35
36     private static void initiateIO() throws IOException {
37
38         Socket socket = null;
39         PrintWriter outbound = new PrintWriter(socket.getOutputStream(), true);
40
41     }
42
43     public static void check(int i){
44
45         if (i == 0)
46             return;
47         else {
48             check(i++);
49         }
50     }
51
52
53     public static void main( String[] args ) throws Exception {
54
55
56         try {
57             String a = "1234 ";
58             Integer.parseInt(a);
59         } catch (Exception e) {
```

```
60      System.out.println("Exception");
61  } // End try..catch ---| Exception
62
63      try {
64          int[] arrayin = {1,2,3};
65          System.out.println(arrayin[10]);
66      } catch (RuntimeException e) {
67          System.out.println(" This is RuntimeException");
68      } // End try..catch ---| RuntimeException
69
70      try {
71          String objStr = "567654";
72          BigDecimal result = addOne(objStr);
73          System.out.println(result);
74      } catch ( ClassCastException e ) {
75          System.out.println(" ----| This is ClassCastException");
76      } // End try..catch ---| ClassCastException
77
78      try {
79          a();
80      } catch (IllegalStateException e) {
81          System.out.println(" ----|_ This is IllegalStateException");
82      } // End try..catch ---| IllegalStateException
83
84      try {
85          String a = "1234 ";
86          Integer.parseInt(a);
87      } catch (NumberFormatException e) {
88          System.out.println(" |----> This is NumberFormatException");
89      } // End try..catch ---| NumberFormatException
90
91      try {
92          int a = 5;
93          int b = 0;
94          int ans = a / b ;
95      } catch (ArithmeticException e) {
96          System.out.println(" ----| This is ArithmeticException");
97      } // End try..catch ---| ArithmeticException
98
99      try {
100          int[] arrayin = {1,2,3};
101          System.out.println(arrayin[10]);
102      } catch (IndexOutOfBoundsException e) {
103          System.out.println(" ----|_ This is IndexOutOfBoundsException");
104      } // End try..catch ---| IndexOutOfBoundsException
```

```

106         try {
107             int[] arrayin = {1,2,3};
108             System.out.println(arrayin[10]);
109         } catch (ArrayIndexOutOfBoundsException e) {
110             System.out.println("        |----> ArrayIndexOutOfBoundsException");
111         } // End try..catch ----| ArrayIndexOutOfBoundsException
112
113         try {
114             String st = "aun";
115             System.out.println(st.charAt(4));
116         } catch (StringIndexOutOfBoundsException e) {
117             System.out.println("        |----> StringIndexOutOfBoundsException");
118         } // End try..catch ----| StringIndexOutOfBoundsException
119
120         try {
121             Path file = null;
122             Files.delete(file);
123         } catch (NullPointerException e) {
124             System.out.println("        |----| NullPointerException");
125         } // End try..catch ----| NullPointerException
126
127
128     try {
129         FileInputStream f = new FileInputStream("code.txt");
130     } catch (IOException e) {
131         System.out.println("        |----| IOException");
132     } // End try..catch ----| IOException
133
134         try {
135             createConnection();
136             System.out.println("two test");
137             initiateIO();
138         } catch (SocketException e) {
139             System.out.println("        |----| SocketException");
140         } // End try..catch ----| SocketException
141
142
143         try {
144             FileInputStream f = new FileInputStream("code.txt");
145         } catch (FileNotFoundException e) {
146             System.out.println("        |----| FileNotFoundException");
147         } // End try..catch ----| FileNotFoundException
148
149         try {
150             DataInputStream dis = new DataInputStream(new FileInputStream("/Users/baconinhell/Desktop/Pawit's Files 🍌/Data.txt"));
151             while (true) {
152                 char ch ;
153                 ch = dis.readChar();
154                 System.out.println(ch);
155             }
156         } catch (EOFException e) {
157             System.out.println("        |----| EOFException");
158         } // End try..catch ----| EOFException
159         System.out.println("_____");

```

5. สรุปผลการปฏิบัติการ

Error แต่ละอย่างสามารถบ่งบอกได้ถึง error ของข้อมูลนั้นๆได้

6. คำถามท้ายการทดลอง

- 6.1. เพราะเหตุใดการใช้ `catch(Exception e)` ; จึงไม่เหมาะสมกับการจัดการสิ่งผิดปกติที่ดีที่สุด เพราะ `catch(Exception e)` ; มันกว้างเกินไป ไม่สามารถจะรู้ error ได้
- 6.2. การจัดการสิ่งผิดปกติจากการตัวเลขต่างๆ ด้วยเลขศูนย์ ควรเลือกใช้วิธีใด?
ใช้ `ArithmeticException`
- 6.3. การจัดการสิ่งผิดปกติจากการเรียกใช้งาน Element เกินขนาดของอาร์เรย์ ควรเลือกใช้วิธีใด?
ใช้ `indexoutofboundsException`