11.Error handling and miscellaneous

Sakib Abrar

CSE

Bangladesh University of Engineering & Technology

sakib.cghs@gmail.com

September 10, 2020

Overview

Why errors?

Exception

Uncaught Exceptions

Caught Exceptions

Throws

Custom Exceptions

Switch

Interface

Abstract Class

Errors are tend to happen

- You cant' always write error free code. Your program may go through some sort of error.
- Whenever your code gets an error it crashes unless you handle that.
- That's why you need to predict and handle errors.

Errors are tend to happen

Terms you need to know.

- Uncaught exception.
- Caught exception.
- try
- catch
- finally
- throw
- throws
- Creating Custom exceptions

Uncaught Exceptions

```
3   public class TestException1
4   {
5       public static void main(String args[]) {
6           int a = 10, b = 0;
7           int c = a/b; // ArithmeticException: / by zero
8           System.out.println(a);
9           System.out.println(b);
10          System.out.println(c);
11          String s = null;
12          System.out.println(s.length()); // NullPointerException
13      }
14  }
```

Caught Exceptions

```
3  public class TestException2
4  {
5      public static void main(String args[])
6      {
7          int a = 10, b = 0, c = 0;
8          try {
9              c = a/b;
10         } catch(Exception e) {
11             System.out.println (e);
12         } finally {
13             // finally block will always execute
14             System.out.println ("In Finally");
15         }
16         System.out.println(a);
17         System.out.println(b);
18         System.out.println(c);
19     }
20 }
```

Throws

```
3  public class TestException3
4  {
5      public static void f() throws Exception {
6          int a = 10;
7          int b = 0;
8          int c = a/b;
9      }
10
11     public static void main(String args[])
12     {
13         try {
14             f();
15         } catch(Exception e) {
16             System.out.println (e);
17             e.printStackTrace();
18         }
19         System.out.println("Hello World");
20     }
21 }
```

Custom Exceptions

```
3  class MyException extends Exception {
4      private int detail;
5
6      MyException(int a) {
7          detail = a;
8      }
9
10     public String toString() {
11         return "My Exception :  "  + detail;
12     }
13 }
14
15 public class TestException4 {
16     static void compute(int a) throws MyException {
17         if(a > 10) {
18             throw new MyException(a);
19         }
20         System.out.println(a);
21     }
22
23     public static void main(String args[]) {
24         try {
25             compute(10);
26             compute(20);
27         } catch(MyException e) {
28             System.out.println(e);
29         }
30     }
31 }
```

Switch

Interface

Abstract Class

THE END