BITS Pilani
Pilani Campus

CS F213 - Object Oriented Programming

J. Jennifer Ranjani
email: jennifer.ranjani@pilani.bits-pilani.ac.in
Chamber: 6121 P, NAB
Consultation: Appointment by e-mail

https://github.com/JenniferRanjani/Object-Oriented-Programming-with-Java

Uncaught Exceptions

- When the Java run time detects an exception, it constructs an object and throws the exception.
- If there are no exception handlers in the program, the exception is caught by the default exception handler.
- The default handler display a string describing the exception and prints a stack trace from the point at which the exception occurred and terminates the program.

Example

```
package class1;

public class test  {
public static void main(String args[]) {
int b = 30,c = 0;
System.out.println(b/c);
}
}
```

Console:

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
at class1.test.main(test.java:5)
```

Using try and catch

- Enclose the code you want to monitor inside the try block and include a catch block immediately after the try block
- Once an exception is thrown, the program control transfers out of the try block into the catch and it never returns to the try block
- A try and its catch is a unit. The scope of the catch is restricted to the statements specified by the immediately preceding try statement.
- A catch statement cannot catch an exception thrown by another try statement.

Key Concepts

- Displaying an exception
  - Throwable overrides the toString(), and the exception object can be passed to println() and the description of the exception gets printed.
- Multiple catch clauses
  - A single try block can be accompanied with multiple catch blocks, but the exception subclass should come before any of their super classes.
- Nested try
  - If the inner try statement does not have a catch handler for a particular exception the next try statement's catch handler are inspected for a match.
  - Nested try can occur less obvious ways when method calls are involved.

Key Concepts

- ‘throw’ keyword is used to throw an exception explicitly.
- But object thrown should be of type Throwable or subclass of throwable. Primitive types such as int, char … and non throwable classes like String can not be used.
- Many built in run-time exceptions have atleast two constructors, one that takes a string parameter describing the exception and one no argument constructor.
  - e.getMessage() can also be used.

Key Concepts

- ‘throws’ keyword is used to specify the callers of the method, when it is capable of causing an exception and it does not handle.

- ‘finally’ is executed after try/catch block and before the code following the try/catch block whether or not an exception is thrown.
  - finally clause is optional
  - But every try block should have atleast one catch or finally block.
  - Even if there is a return statement in the try block, the method returns after the finally block is executed.

Extra Topic (only if you are interested) Difference between C++ and JAVA

- In C++, all types (including primitive and pointer) can be thrown as exception. But in Java only throwable objects (Throwable objects are instances of any subclass of the Throwable class) can be thrown as exception.
- In C++, there is a special catch called “catch all” that can catch all kind of exceptions. In Java, we can catch Exception object to catch all kind of exceptions.
- In Java, there is a block called finally that is always executed after the try-catch block. This block can be used to do cleanup work.
- In C++, all exceptions are unchecked. In Java, there are two types of exceptions – checked and unchecked.
- In Java, a new keyword *throws* is used to list exceptions that can be thrown by a function. In C++, there is no *throws* keyword, the same keyword *throw* is used for this purpose also.