Exception Handling

Shirley B. Chu

June 26, 2020

De La Salle University
College of Computer Studies

Exception Handling

Exception

an event that disrupts the normal flow of the program.

Exception Handling

Exception

an event that disrupts the normal flow of the program.

Exception Handling

one of Java's powerful mechanism to handle *runtime errors* so that the normal flow of the application can be maintained.

Types of Exceptions

- **Checked exceptions** These are checked at compile-time, e.g. `IOException`.

Types of Exceptions

- **Checked exceptions** These are checked at compile-time, e.g. `IOException`.
- **Unchecked exceptions** These are checked at run-time, e.g. `ArithmeticException`, `NullPointerException`, `ArrayIndexOutOfBoundExceptions`

Types of Exceptions

- **Checked exceptions** These are checked at compile-time, e.g. `IOException`.
- **Unchecked exceptions** These are checked at run-time, e.g. `ArithmeticException`, `NullPointerException`, `ArrayIndexOutOfBoundExceptions`
- **Error** According to Oracle, this is the third type of exception, which is irrecoverable, e.g. `OutOfMemoryError`, `VirtualMachine`, etc.

Keywords

- `try-catch-finallly`
- `throws`
- `throw`

try-catch-finally

Syntax:

```
try
{
  /* code that may cause an exception */
}
catch (Exception e)
{
  /* what to do when exception occurs */
}
finally
{
  /* code that will always be executed
     whether the exception occurred or not */
}
```

try-catch-finally

Syntax:

```
try
{
  /* code that may cause an exception */
}
catch (Exception e)
{
  /* what to do when exception occurs */
}
finally
{
  /* code that will always be executed
     whether the exception occurred or not */
}
```

- The `try` block must be followed by either a `catch` block or a `finally` block.

`try-catch-finally`

Syntax:

```
try
{
  /* code that may cause an exception */
}
catch (Exception e)
{
  /* what to do when exception occurs */
}
finally
{
  /* code that will always be executed
       whether the exception occurred or not */
}
```

- The `catch` block must be preceded by `try` block.
- Multiple `catch` blocks may be used with a single `try` block.

try-catch-finally

Syntax:

```
try
{
  /* code that may cause an exception */
}
catch (Exception e)
{
  /* what to do when exception occurs */
}
finally
{
  /* code that will always be executed
      whether the exception occurred or not */
}
```

- The `finally` block is used to execute important code. This is executed whether the exception occurred or not, or whether the exception is handled or not (e.g cleanup, closing a file/connection).

Try this!

```
Scanner input = new Scanner (System.in);
System.out.println ("Enter a number:  ");
int nVal = input.nextInt ();
System.out.println ("Input is :  " + nVal);
```

Now, try this!

```
Scanner input = new Scanner (System.in);
System.out.println ("One - outside try");
try
{
    int nVal = input.nextInt ();
    System.out.println ("Two - inside try");
    System.out.println ("Input is :  " + nVal);
}
catch (Exception e)
{
    System.out.println ("Three - inside catch");
    System.out.println (e.toString ());
}
finaly
{
    System.out.println ("Four - inside finally");
}
Sytem.out.println ("Five - outside finally");
```

Now, try this!

```
Scanner input = new Scanner (System.in);
System.out.println ("One - outside try");
try
{
    int nVal = input.nextInt ();
    System.out.println ("Two - inside try");
    System.out.println ("Input is :  " + nVal);
}
catch (Exception e)
{
    System.out.println ("Three - inside catch");
    System.out.println (e.toString ());
}
finaly
{
    System.out.println ("Four - inside finally");
}
Sytem.out.println ("Five - outside finally");
```

What happens when...

- user enters a digit?

Now, try this!

```
Scanner input = new Scanner (System.in);
System.out.println ("One - outside try");
try
{
    int nVal = input.nextInt ();
    System.out.println ("Two - inside try");
    System.out.println ("Input is :  " + nVal);
}
catch (Exception e)
{
    System.out.println ("Three - inside catch");
    System.out.println (e.toString ());
}
finaly
{
    System.out.println ("Four - inside finally");
}
Sytem.out.println ("Five - outside finally");
```

What happens when...

- user enters a digit?
- user enters a character?

throws

- used to declare that an exception may occur

throws

- used to declare that an exception may occur
- Syntax `throws` ⟨*exception/s*⟩

throws

- used to declare that an exception may occur
- Syntax `throws` ⟨*exception/s*⟩
- Example:

```
public void theMethod (int val)
        throws ArithmeticException, NullPointerException
{
  /* code that might throw exception/s */
}
```

throw

- used to explicitly throw an exception, usually custom exceptions.

throw

- used to explicitly throw an exception, usually custom exceptions.
- Syntax: `throw` ⟨*exception_object*⟩`;`

throw

- used to explicitly throw an exception, usually custom exceptions.
- Syntax: `throw` ⟨*exception_object*⟩`;`
- Examples:

throw

- used to explicitly throw an exception, usually custom exceptions.
- Syntax: `throw` ⟨*exception_object*⟩`;`
- Examples:

  -
    ```
    if (nYear < 1900)
      throw new ArithmeticException ("invalid year value");
    ```

throw

- used to explicitly throw an exception, usually custom exceptions.
- Syntax: `throw` ⟨*exception_object*⟩`;`
- Examples:

  -
    ```
    if (nYear < 1900)
      throw new ArithmeticException ("invalid year value");
    ```

  -
    ```
    public class InvalidInputException extends Exception
    {
        public InvalidInputException (String msg)
        {
            super (msg);
        }
    }
    ```

Example

```
public void theMethod (int val) throws ArithmeticException
{
    if (nYear < 1900)
      throw new ArithmeticException ("invalid year value");
}

public void someOtherMethod ()
{
    Scanner kb = new Scanner (System.in);
    int nVal = Integer.parseInt (kb.nextLine ());
    try
    {
      theMethod(nVal);
    }
    catch (Exception e)
    {
      System.out.println ("do something here");
    }
}
```

Checked vs Unchecked Exceptions

Checked exceptions are checked at compile-time. When a method that throws an exception is called, the calling method should either be handled it using `try-catch-finally`, or the calling method should declare that it `throws` that (unhandled) exception.

Unchecked exceptions are not checked at compil-time. These are experienced at run-time when it occurs. Examples:

```
int nOne = 9;
int nTwo = 0;
int nThree = nOne / nTwo;
```

```
int numbers = new int[5];
numbers[5] = 5;
```

☺ **Thank you!** ☺