Exception Handling

Chapter 11

December 8, 2010

Exceptions

Compiling

```
File file = new File("test.txt");
PrintStream fileOutput = new PrintStream(file);
```

gives rise to the error

```
Client.java:13: unreported exception java.io.
FileNotFoundException; must be caught or declared
to be thrown
   PrintStream fileOutput = new PrintStream(file);
                            ^
1 error
```

Why?

Exceptions

Answer

Because the constructor `PrintStream(File)` throws a `FileNotFoundException` if the `file` object does not denote an existing, writable regular file and a new regular file of that name cannot be created, or if some other error occurs while opening or creating the file (see API).

Exceptions

Answer

Because the constructor `PrintStream(File)` throws a `FileNotFoundException` if the `file` object does not denote an existing, writable regular file and a new regular file of that name cannot be created, or if some other error occurs while opening or creating the file (see API).

Question

How does a client fix a "must be caught or declared to be thrown" error?

Exceptions

Answer

Because the constructor `PrintStream(File)` throws a `FileNotFoundException` if the `file` object does not denote an existing, writable regular file and a new regular file of that name cannot be created, or if some other error occurs while opening or creating the file (see API).

Question

How does a client fix a "must be caught or declared to be thrown" error?

Answer

Catch the exception. (An implementer may also decide the declare the exception to be thrown.)

How to Catch Exceptions?

Step 1

Place a try block around the statement(s) that may throw the exception.

```
try
{
    ...
}
```

Step 2

Place one or more catch blocks right after the try block.

```
catch (...Exception e)
{
    ...
}
```

How to Catch Exceptions?

```
import java.io.FileNotFoundException;
...
try
{
   File file = new File("test.txt");
   PrintStream fileOutput = new PrintStream(file);
}
catch (FileNotFoundException e)
{
   output.println("Failed to write to file: "
      + e.getMessage())
}
```

Exceptions

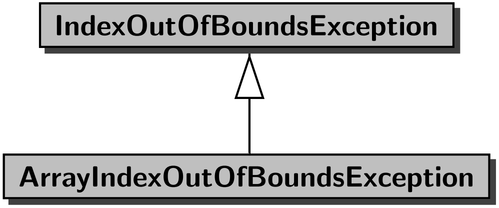
```
try
{
   output.println(arguments[0]);
}
catch (IndexOutOfBoundsException e)
{  output.println(e.getMessage()); }
catch (ArrayIndexOutOfBoundsException e)
{  e.printStackTrace(); }
```

gives rise to the compile-time error

```
Client.java:19: exception java.lang.
ArrayIndexOutOfBoundsException has already been
caught
   catch (ArrayIndexOutOfBoundsException e)
   ^
1 error
```

Why?

Exceptions



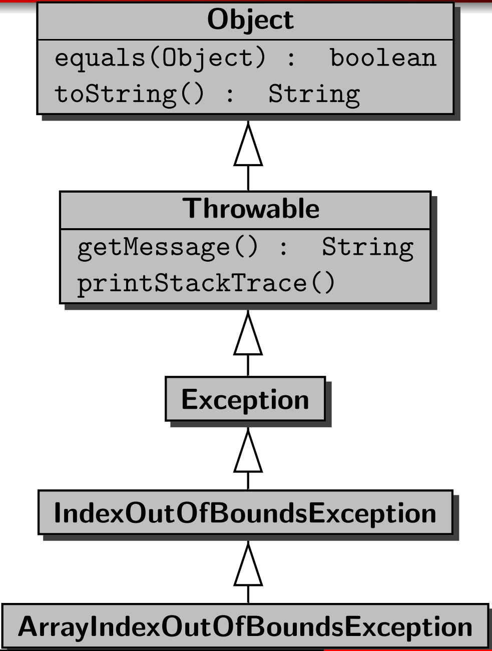
An `ArrayIndexOutOfBoundsException` is-an `IndexOutOfBoundsException`.

Exceptions

```
try
{
   output.println(arguments[0]);
}
catch (IndexOutOfBoundsException e)
{
   output.println(e.getMessage());
}
catch (ArrayIndexOutOfBoundsException e)
{
   e.printStackTrace();
}
```

The second catch block is redundant, because an `ArrayIndexOutOfBoundsException` is-an `IndexOutOfBoundsException`.

Inheritance Hierarchy



Exceptions

Question

May the method `charAt(int)` of the class `String` throw an exception?

Exceptions

Question

May the method `charAt(int)` of the class `String` throw an exception?

Answer

Yes.

Exceptions

Question

May the method `charAt(int)` of the class `String` throw an exception?

Answer

Yes.

Question

Which type of exception?

Exceptions

Question

May the method `charAt(int)` of the class `String` throw an exception?

Answer

Yes.

Question

Which type of exception?

Answer

An `IndexOutOfBoundsException`.

Exceptions

```
String word = ...;
output.println(word.charAt(2));
```

Question

Why does the above snippet not give rise to a “must be caught or declared to be thrown” error?

Exceptions

```
String word = ...;
output.println(word.charAt(2));
```

Question

Why does the above snippet not give rise to a “must be caught or declared to be thrown” error?

Answer

The “must be caught or declared to be thrown” rule is only applicable to checked exceptions and an `IndexOutOfBoundsException` is not checked.

What are Checked Exceptions?

Definition

An exception is checked if

- it is `Exception` or any of its subclasses, and
- it is not `RuntimeException` or any of its subclasses.

Question

Is `NullPointerException` checked?

What are Checked Exceptions?

Definition

An exception is checked if

- it is `Exception` or any of its subclasses, and
- it is not `RuntimeException` or any of its subclasses.

Question

Is `NullPointerException` checked?

Answer

No.

What are Checked Exceptions?

Definition

An exception is checked if

- it is `Exception` or any of its subclasses, and
- it is not `RuntimeException` or any of its subclasses.

Question

Is `NullPointerException` checked?

Answer

No.

Question

Is `InvalidPropertiesFormatException` checked?

What are Checked Exceptions?

Definition

An exception is checked if

- it is `Exception` or any of its subclasses, and
- it is not `RuntimeException` or any of its subclasses.

Question

Is `NullPointerException` checked?

Answer

No.

Question

Is `InvalidPropertiesFormatException` checked?

Answer

Yes.

What are Checked Exceptions?

Definition

An exception is checked if

- it is `Exception` or any of its subclasses, and
- it is not `RuntimeException` or any of its subclasses.

Question

Is `Exception` checked?

What are Checked Exceptions?

Definition

An exception is checked if

- it is `Exception` or any of its subclasses, and
- it is not `RuntimeException` or any of its subclasses.

Question

Is `Exception` checked?

Answer

Yes.

What are Checked Exceptions?

Definition

An exception is checked if

- it is `Exception` or any of its subclasses, and
- it is not `RuntimeException` or any of its subclasses.

Question

Is `Exception` checked?

Answer

Yes.

Question

Is `RuntimeException` checked?

What are Checked Exceptions?

Definition

An exception is checked if

- it is `Exception` or any of its subclasses, and
- it is not `RuntimeException` or any of its subclasses.

Question

Is `Exception` checked?

Answer

Yes.

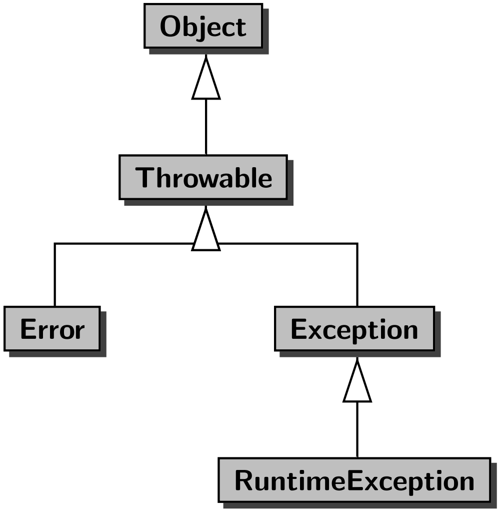
Question

Is `RuntimeException` checked?

Answer

No.

Inheritance Hierarchy



Errors

Question

Why are `Errors` exempt from the “must be caught or declared to be thrown” rule?

Errors

Question

Why are `Errors` exempt from the “must be caught or declared to be thrown” rule?

Answer

`Errors` represent conditions that are so abnormal the reliability of the whole environment is suspect and, hence, the code in the catch block may not run properly either.

Errors

Question

Why are `Errors` exempt from the “must be caught or declared to be thrown” rule?

Answer

`Errors` represent conditions that are so abnormal the reliability of the whole environment is suspect and, hence, the code in the catch block may not run properly either.

Question

Why are `RuntimeExceptions` exempt from the “must be caught or declared to be thrown” rule?

Errors

Question

Why are `Errors` exempt from the "must be caught or declared to be thrown" rule?

Answer

`Errors` represent conditions that are so abnormal the reliability of the whole environment is suspect and, hence, the code in the catch block may not run properly either.

Question

Why are `RuntimeExceptions` exempt from the "must be caught or declared to be thrown" rule?

Answer

`RuntimeExceptions` represent conditions that can be validated by the client.

Throwing Exceptions

Question

How can the client throw an exception?

Throwing Exceptions

Question

How can the client throw an exception?

Answer

```
throw new ...Exception(...);
```

Throwing Exceptions

Question

How can the client throw an exception?

Answer

```
throw new ...Exception(...);
```

Question

Why would a client ever throw an exception?

Throwing Exceptions

Question

How can the client throw an exception?

Answer

```
throw new ...Exception(...);
```

Question

Why would a client ever throw an exception?

Answer

For example, the client may want to separate the error handling code from the rest.

Check11D

Write a program that reads from the user a mathematical expression that may contain nested precedence characters, which are the following: parentheses (and), brackets [and], and braces { and }.

For example, the input string can be

```
{a-b*[c/(d+e)]+f*[g-h*{i+j*k}]-m/(n+l)/w}
```

The program should ignore all other characters and focus only on the above precedence characters. Specifically, it should determine whether these characters are "OK", "Imbalanced", "Overlapping" or "Too Deeply Nested".

Check11D

```
Enter expression
[a+b*(a-d])
Overlapping

Enter expression
(a+(b-[a/d])
Imbalanced

Enter expression
(a+b)]-a
Imbalanced

Enter expression
{a-[b/(a+d)]}
OK

Enter expression
(((([[a+b]/a]+d)+e)+f)
Too Deeply Nested
```

Check11D

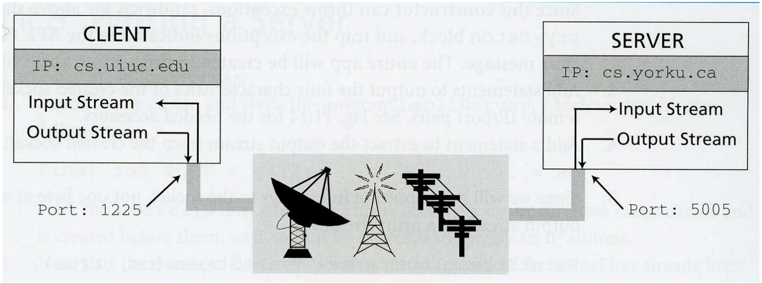
- Create an instance of the `type.lib.CharStack` class, an ordered collection of chars, using its default constructor.
- Scan the input string character by character, left to right.
- Ignore characters different from the precedence characters.
- If an open precedence character is encountered (i.e., (or [or {), store it in the stack.
- If a close precedence character is encountered, remove from the stack the character that was stored last. If last stored character is the corresponding open character (for example, the encountered close precedence character is] and the last stored character is [), then continue scanning. Otherwise, end the program with the "Overlapping" message.

Check11D

- If you need to remove a character from the stack but find it empty, or you complete scanning the string but find the stack not empty, then end the program with the "Imbalanced" message.
- If you need to store something in the stack but find it full, then end the program with the "Too Deeply Nested" message.
- If the program did not end in one of the above cases, print "OK."

Socket Programming

How to make two different Java apps running on different machines communicate?



source: Java By Abstraction

Socket Programming

- IP Address Every computer on a network must have a unique address, called its IP address (IP stands for the Internet protocol). The address is expressed either as four dot-delimited numbers, for example, `130.63.96.21`, or as several dot-delimited names, for example, `red.cse.yorku.ca`.
- Port If an app plans to communicate over a network, it must first register itself with the operating system of the machine on which it is running. Upon registration, the operating system assigns a unique port number (an integer) to the app.
- Socket A socket is a virtual channel that connects an app running on one machine with one running on another. It is uniquely identified by two IP/port pairs: the IP address of the first machine and the port number of the first app, and a similar pair at the other end.

Socket Programming: The Client

- Create an app named `Client` and introduce the constants

  ```
  final String HOST = ...;
  final int PORT = ...;
  ```

- Create a `Socket`.
- Extract the `OutputStream` from the `Socket`.
- From the `OutputStream` create a `PrintStream` with auto-flushing.
- Extract the `InputStream` from the `Socket`.
- From the `InputStream` create a `Scanner`.
- Close the `Socket`.

Socket Programming: The Server

- Create an app named `Server` and introduce the constants

```
final int PORT = ...;
```

- Create a `ServerSocket`.
- Listen for a connection to be made to the `ServerSocket` and accepts it.
- Extract the `OutputStream` from the `Socket`.
- From the `OutputStream` create a `PrintStream` with auto-flushing.
- Extract the `InputStream` from the `Socket`.
- From the `InputStream` create a `Scanner`.
- Close the `Socket`.

Socket Programming: The Client

- Prompt the user `Enter a number:`
- Read the number from the keyboard.
- Write the number to the socket.
- Read the result from the socket.
- Print `The square root of` ... `is` ...

Socket Programming: The Server

- Read the number from the socket.
- Compute its square root.
- Write the square root to the socket.

What if I Miss the Programming Test or the Final Exam?

If you miss either the programming test or the final exam, follow the instructions at this URL.

The deferred exam will take place on Monday, January 10, 2011 from 16:00 until 19:00.

How to Study for the Final Exam?

- read the textbook,
- read the additional material posted on the website and the forum,
- make up exam questions yourself and post them on the forum,
- try to answer exam questions posted on the forum,
- discuss answers with fellow students (preferably in person, but can also be done on the forum).

Questions

My office hours during the exam period will be posted on the forum.

Feel free to email me your questions or post them on the forum.