WSA | Forward looking IT finishing school

Java Programming Language SE – 6

Module 8 : Exceptions and Assertions

ORACLE

Certified Professional

Java SE 6 Programmer

www.webstackacademy.com

Objectives

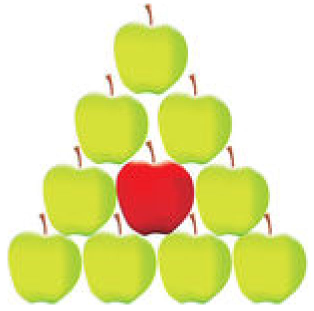
- Define exceptions
- Use try, catch, and finally statements
- Describe exception categories
- Identify common exceptions
- Develop programs to handle your own exceptions
- Use assertions
- Distinguish appropriate and inappropriate uses of assertions
- Enable assertions at run time

Relevance

- In most programming languages, how do you resolve runtime errors?
- If you make assumptions about the way your code works, and those assumptions are wrong, what might happen?
- Is it always necessary or desirable to expend CPU power testing assertions in production programs?

Exceptions and Assertions

- Exceptions handle unexpected situations – Illegal argument, network failure, or file not found
- Assertions document and test programming assumptions – This can never be negative here
- Assertion tests can be removed entirely from code at runtime, so the code is not slowed down at all.



Exceptions

- Conditions that can readily occur in a correct program are checked exceptions. These are represented by the Exception class.
- Severe problems that normally are treated as fatal or situations that probably reflect program bugs are unchecked exceptions.

  Fatal situations are represented by the Error class. Probable bugs are represented by the RuntimeException class.
- The API documentation shows checked exceptions that

  can be thrown from a method.

Exception Example

```
public class AddArguments {
public static void main(String args[]) {
int sum = 0;
for ( String arg : args ) {
sum += Integer.parseInt(arg);
}
System.out.println("Sum = " + sum);
}
}
```

Exception Example

- java AddArguments 1 2 3 4

  Sum = 10
- java AddArguments 1 two 3.0 4

- Exception in thread "main" java.lang.NumberFormatException: For input string: "two"

  at java.lang.NumberFormatException.forInputString(NumberFormatException.java:48)

  at java.lang.Integer.parseInt(Integer.java:447)

  at java.lang.Integer.parseInt(Integer.java:497)

  at AddArguments.main(AddArguments.java:5)

The try-catch Statement

```
public class AddArguments2 {
public static void main(String args[]) {
try {
int sum = 0;
for ( String arg : args ) {
sum += Integer.parseInt(arg);
}
System.out.println("Sum = " + sum);
} catch (NumberFormatException nfe) {
System.err.println("One of the command-line "
+ "arguments is not an integer.");
}}}
```

The try-catch Statement

```
java AddArguments2 1 two 3.0 4
One of the command-line arguments is not an integer.
```

The try-catch Statement

```
public class AddArguments3 {
public static void main(String args[]) {
int sum = 0;
for ( String arg : args ) {
try {
sum += Integer.parseInt(arg);
} catch (NumberFormatException nfe) {
System.err.println("[" + arg + "] is not an integer"
+ " and will not be included in the sum.");
}
}
System.out.println("Sum = " + sum);
}}
```

The try-catch Statement

```
java AddArguments3 1 two 3.0 4
```

```
[two] is not an integer and will not be included in the sum.
[3.0] is not an integer and will not be included in the sum.
Sum = 5
```

The try-catch Statement

- *A try-catch statement can use multiple catch clauses:*

```
try {
// code that might throw one or more exceptions
} catch (MyException e1) {
// code to execute if a MyException exception is thrown
} catch (MyOtherException e2) {
// code to execute if a MyOtherException exception is thrown
} catch (Exception e3) {
// code to execute if any other exception is thrown
}
```

Call Stack Mechanism

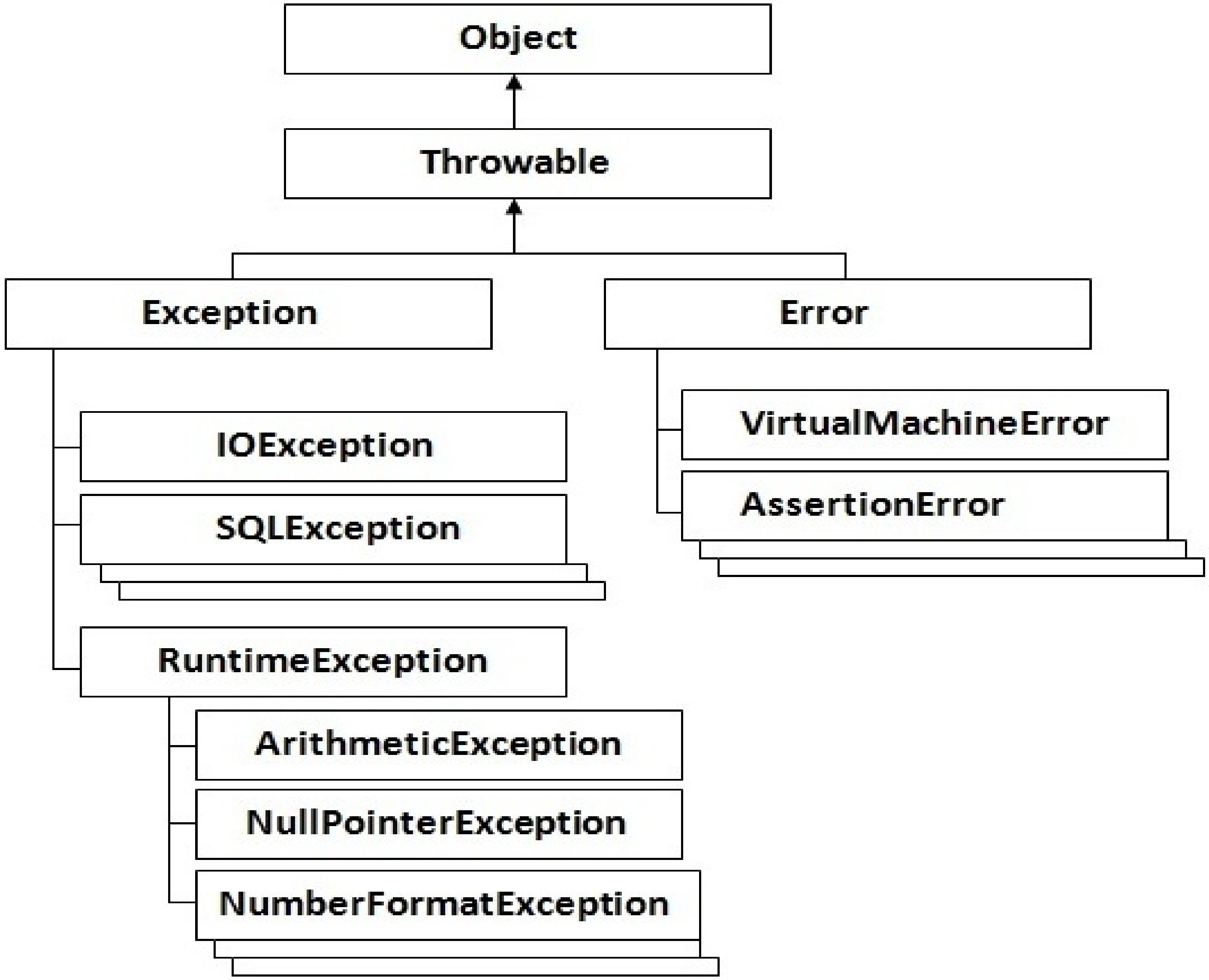
- If an exception is not handled in the current try-catch block, it is thrown to the caller of that method.
- If the exception gets back to the main method and is not handled there, the program is terminated abnormally.

The finally Clause

- *The finally clause defines a block of code that always executes.*

```
try {
startFaucet();
waterLawn();
} catch (BrokenPipeException e) {
logProblem(e);
} finally {
stopFaucet();
}
```

Exception Categories



Common Exceptions

- NullPointerException
- FileNotFoundException
- NumberFormatException
- ArithmeticException
- SecurityException

The Handle or Declare Rule

Use the handle or declare rule as follows:

- Handle the exception by using the try-catch-finally block.
- Declare that the code causes an exception by using the throws clause.
  - void trouble() throws IOException { ... }
  - void trouble() throws IOException, MyException { ... }

The Handle or Declare Rule

Other Principles

- You do not need to declare runtime exceptions or errors.
- You can choose to handle runtime exceptions.

Method Overriding and Exceptions

The overriding method can throw:

- No exceptions
- One or more of the exceptions thrown by the overridden method
- One or more subclasses of the exceptions thrown by the overridden method

Method Overriding and Exceptions

The overriding method cannot throw:

- Additional exceptions not thrown by the overridden method
- Superclasses of the exceptions thrown by the overridden method

Method Overriding and Exceptions

```
public class TestA {
public void methodA() throws IOException {
// do some file manipulation
}}
public class TestB1 extends TestA {
public void methodA() throws EOFException {
// do some file manipulation
}}
public class TestB2 extends TestA {
public void methodA() throws Exception { // WRONG
// do some file manipulation
}}
```

Creating Your Own Exceptions

1. Extends Exception class
2. Override toString() method
3. Define constructor.

Example of Override Exception

```
class NegativeAgeException extends Exception{
int age;
NegativeAgeException(int age){
This.age=age;
}
Public String toString(){
Return “negative age exception:”+ age;
}
Public static void main(String[] args){
Int age =-25;
if(age<=0){throw new NegativeAgeException(age)}
else{System.ou.println(“your age is”+age)}
}}
```

Assertions

- Syntax of an assertion is:

```
assert <boolean_expression> ;
assert <boolean_expression> : <detail_expression> ;
```

- If <boolean_expression> evaluates false, then an AssertionError is thrown.
- The second argument is converted to a string and used as descriptive text in the AssertionError message.

Recommended Uses of Assertions

Use assertions to document and verify the assumptions and internal logic of a single method:

- Internal invariants
- Control flow invariants
- Postconditions and class invariants

Recommended Uses of Assertions

Inappropriate Uses of Assertions:

- Do not use assertions to check the parameters of a public method.
- Do not use methods in the assertion check that can cause side-effects.

Internal Invariants

The problem is:

```
if (x > 0) {
// do this
} else {
// do that
}
```

Internal Invariants

The solution is:

```
if (x > 0) {
// do this
} else {
assert ( x == 0 );
// do that, unless x is negative
}
```

Control Flow Invariants

For example:

```
switch (suit) {
case Suit.CLUBS: // ...
break;
case Suit.DIAMONDS: // ...
break;
case Suit.HEARTS: // ...
break;
case Suit.SPADES: // ...
break;
default: assert false : "Unknown playing card suit";
break;
}
```

Postconditions and Class Invariants

```
public Object pop() {
int size = this.getElementCount();
if (size == 0) {
throw new RuntimeException("Attempt to pop from empty stack");
}
Object result = /* code to retrieve the popped element */ ;
// test the postcondition
assert (this.getElementCount() == size - 1);
return result;
}
```

Controlling Runtime Evaluation of Assertions

- If assertion checking is disabled, the code runs as fast as if the check was never there.
- Assertion checks are disabled by default. Enable assertions with the following commands:

  ```
  java -enableassertions MyProgram
  ```

  or:

  ```
  java -ea MyProgram
  ```

- Assertion checking can be controlled on class, package, and package hierarchy bases, see: docs/guide/language/assert.html

WSA | Forward looking IT finishing school

Thank you

Web Stack Academy (P) Ltd

#83, Farah Towers,

1st floor,MG Road,

Bangalore – 560001

M: +91-80-4128 9576

T: +91-98862 69112

E: info@www.webstackacademy.com

www.webstackacademy.com