

Chapter 15 Exceptions and Assertions

1. See the section "Exceptions and Exception Types." The `Throwable` class is the root of Java exception classes. `Error` and `Exception` are subclasses of `Throwable`. `Error` describes fatal system errors, and `Exception` describes the errors that can be handled by Java programs. The subclasses of `Error` are `LinkageError`, `VirtualMachineError`, and `AWTError`. The subclasses of `Exception` include `RuntimeException`, `IOException`, `AWTException`, and `InstantiationException`.
2. The purpose of claiming exceptions is to tell the Java runtime system what can go wrong. You claim an exception using the `throws` keyword in the method declaration. You can claim multiple exceptions, separated by commas.
3. A checked exception must be explicitly declared in the method declaration, if a method throws it. A checked exception must be caught in a try-catch block. An unchecked exception does not need to be declared and does not need to be caught. In Java, the only unchecked exceptions are `RuntimeException` and `Error` and their subclasses.
4. You use the `throw` statement in the method to throw an exception. You cannot throw multiple exceptions in a single `throw` statement.
5. `throw` is for throwing exceptions and `throws` is for claiming exceptions.
6. When an exception occurs, the Java runtime system creates an object for the exception, and the catch process starts.
7. Use a `try-catch` block to catch exceptions.
8. Will `statement3` be executed?
Answer: No.

If the exception is not caught, will `statement4` be executed?
Answer: No.

If the exception is caught in the `catch` clause, will `statement4` be executed?
Answer: Yes.

If the exception is passed to the caller, will `statement4` be executed?
Answer: No.
9. Rational operation error
10. Rational operation error

11. Rational operation error
After the method call
12. If an exception were not caught in a non-GUI application, the program would terminate. If an exception were not caught in a GUI application, the program would continue and the error message would be reported on the console.
13. To print trace information to the console.
14. No.
15. Will statement5 be executed if the exception is not caught?
Answer: No.

If the exception is of type `Exception3`, will statement4 be executed? Will statement5 be executed?

Answer: This exception is caught by the `catch (Exception3 e3)` clause and statement4 will be executed, but statement5 will not be executed because it is rethrown to its caller.

16. Rational operation error
Rational operation error
17. An *assertion* is a Java statement that enables you to assert an assumption about your program. An assertion contains a Boolean expression that should be true during program execution. To declare an assertion, use the `assert` key word followed by a Boolean expression with an optional value. To compile the code with assertion, use the `-source 1.4` switch in the `javac` command. To run the code with assertion, use the `-ea` switch in the `java` command.
18. An `java.lang.AssertionError` would be thrown since the assertion is violated at Line 7 (i is actually 11).