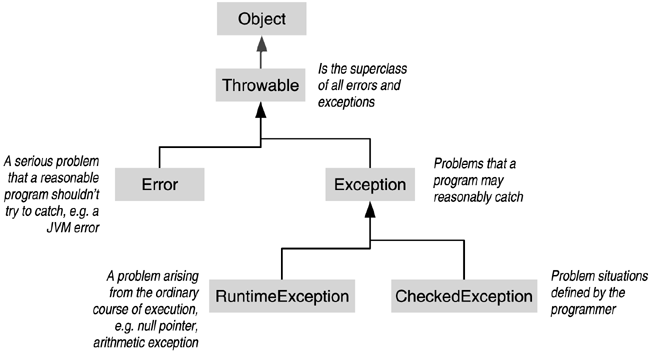
**Exception:**

* **An *exception* is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions.**
* In java, exception handling is managed by the use of five keywords: **try, catch, throw, throws,** and **finally**.
* System generated exceptions are automatically thrown by the Java run-time system. To manually throw an exception, use the keyword **throw**
* When an error occurs within a method, the method creates an object and hands it off to the runtime system. The object, called an exception object, contains information about the error, including its type and the state of the program when the error occurred.
* Creating an exception object and handing it to the runtime system is called throwing an exception.
* After a method throws an exception, the runtime system attempts to find something to handle it.
* A try statement that catches the exception. The try must provide a handler for the exception
* A method that specifies that it can throw the exception. The method must provide a throws clause that lists the exception
* **Three kinds of Exception:**
  1. Checked Exception
  2. Error
  3. Runtime Exception



* **Syntax:**

try {

} catch (FileNotFoundException e) {

System.err.println("FileNotFoundException: " + e.getMessage());

throw new SampleException(e);

} catch (IOException e) {

System.err.println("Caught IOException: " + e.getMessage());

}finally{//Optional

System.out.println(“Finally Executed All The Time”);

}

**Nested:**

class NestedTry{

public static void main(String []args) {

try {

int argnum = args.length;

System.out.println("Number of arguments = "+ argnum);

int quot = 33/argnum;

try {

if (argnum == 1)

argnum = 34/(argnum-argnum);

if (argnum == 2) {

int c[] = {1, 2, 3};

c[22] = 34;

}

} catch(ArithmeticException e) {

System.err.println("Divide by zero: "+ e);

}

} catch(ArrayIndexOutOfBoundsException e) {

System.out.println("Exception: " + e);

}

}

}

**Throwable:**

throw someThrowableInstance;

Some Exceptions

* **Unchecked Exceptions**
  + **ArithmeticException:** Arithmetic error like divide by zero.
  + **ArrayIndexOutOfBoundsException:** Array index is out of bounds.
  + **ArrayStoreException:** Assignment to an array element of an incompatible type.
  + **ClassCastException:** Invalid cast.
  + **IllegalArgumentException:** Illegal argument used to invoke a method.
  + **IndexOutOfBoundsException:** Some types of index is out of bounds.
  + **NegativeArraySizeException:** Array created with negative size.
  + **NullPointerException:** Invalid use of null reference.
  + **NumberFormatException:** Invalid conversion of a string to a numeric format.
  + **StringIndexOutOfBoundsException:** Attempt to index outside the bounds of a string.
* **Checked Exceptions**
  + **ClassNotFoundExce0ption**: Class not found.
  + **CloneNotSupportedException:** Attempt to clone an object that does not implement the **Cloneable** interface.
  + **IllegalAccessException:** Access to a class is denied.
  + **InstantiationException:** Attempt to create an object of an abstract class or interface.
  + **InterruptedException:** One thread has been interrupted by another thread.
  + **NoSuchFieldException:** A requested field does not exist.
  + **NoSuchMethodException:** A requested method does not exist.
* Important: The finally block is a key tool for preventing resource leaks. When closing a file or otherwise recovering resources, place the code in a finally block to insure that resource is *always* recovered.
* A stack trace provides information on the execution history of the current thread and lists the names of the classes and methods that were called at the point when the exception occurred. A stack trace is a useful debugging tool that you'll normally take advantage of when an exception has been thrown.

### Logging API