Exception handling facts for kids

Kids Encyclopedia Facts

In computing, an **exception** is a special situation where the program cannot do things the way it usually would and is forced to do something else instead. One layer of the system uses an exception to give another layer information about special states the system is currently in. The different layers of software or hardware have contracts, that tell what can be expected; this is generally known as Programming by Contract. In the context of exception handling, a program is said to be exception-safe, if exceptions that occur will not produce side-effects (such as memory leaks), will not change stored data so that it becomes unreadable, or generate output that is invalid. There are different levels of exception safety:

1. **Failure transparency** or **no throw guarantee**: No matter what happens, no exceptions will be thrown. This is the best level of exception safety, but also the most difficult to implement.

2. **Commit or rollback semantics, Strong exception safety, no change guarantee**: Operations can fail, and exceptions will be thrown. However, a failed operation is guaranteed to not have side-effects or change the data.

3. **Basic exception safety**: A part of the failed operation may have been executed, and can have side-effects. The state of the data may be different before and after the execution, but in both cases, the data will be in a valid state.

4. **Minimal exception safety,No-leak guarantee**: As only a part of the operation was executed, the stored data may be invalid, afterwards. The system will keep running, and no resources get leaked, however.

5. **No exception safety**: No guarantees can be made. This is the worst level of exception safety.

Usually, a programmer will try to *catch* the exception early so that problems don't get worse over time.

Example

Suppose a program tries to add something to an *array*, or group of *objects* that doesn't exist. This is called a *null reference*. Look at the following code from the Java programming language:

```
class SomeProgram {
  int[] SomeArray = null; // This
array of numbers doesn't exist
  public static void main(String[]
args) {
    System.out.println("The 1st
number in the array is " +
SomeArray[0] + "."); // This will
throw an exception because it refers
to an imaginary array
  }
}
```

This code throws what programmers call a *null-pointer exception*. This is fixed by adding "`try`" in front of the code that might throw the exception, like is done with the code shown below:

This code throws what programmers call a *null-pointer exception*. This is fixed by adding "`try`" in front of the code that might throw the exception, like is done with the code shown below:

```
class AnotherProgram {
  int[] SomeArray = null; // This
array of numbers doesn't exist
  public static void main(String[]
args) {
    try {
      System.out.println("The 1st
number in the array is " +
SomeArray[0] + "."); // This will
throw an exception because it

// refers to an imaginary array
    }
    catch (NullPointerException e) {
// This is how you catch a null-
pointer exception
      System.err.println("Sorry.  I
could not find the first number in
the array."); // This creates an error
message
      e.printStackTrace(); // This
tells you where to look for bugs in
your program
    }
  }
}
```