# Raising Exceptions in Java

To throw an exception from an operation implementation, you simply instantiate the exception, initialize it, and throw it. For example:

**Java**

```
// ...

public void
write(String[] text, Ice.Current current)
    throws GenericError
{
    try
    {
        // Try to write file contents here...
    }
    catch(Exception ex)
    {
        throw new GenericError("Exception during write operation", ex);
    }
}
```

Note that, for this example, we have supplied the optional second parameter to the `GenericError` constructor. This parameter sets the inner exception and preserves the original cause of the error for later diagnosis.

If you throw an arbitrary Java run-time exception (such as a `ClassCastException`), the Ice run time catches the exception and then returns an `UnknownException` to the client. Similarly, if you throw an "impossible" user exception (a user exception that is not listed in the exception specification of the operation), the client receives an `UnknownUserException`.

If you throw an Ice run-time exception, such as `MemoryLimitException`, the client receives an `UnknownLocalException`. For that reason, you should never throw system exceptions from operation implementations. If you do, all the client will see is an `UnknownLocalException`, which does not tell the client anything useful.

Three run-time exceptions are treated specially and not changed to `UnknownLocalException` when returned to the client: `ObjectNotExistException`, `OperationNotExistException`, and `FacetNotExistException`.

## See Also

- Run-Time Exceptions
- Java Mapping for Exceptions
- Server-Side Java Mapping for Interfaces
- Parameter Passing in Java
- Tie Classes in Java